699.1

Adversary Emulation for Breach Prevention & Detection

SANS

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

SEC699.1 | Purple Team Tactics

SANS

Adversary Emulation for Breach Prevention & Detection

Welcome to SANS Security SEC699: Advanced Purple Team Tactics – Adversary Emulation for Breach Prevention & Detection.

Erik Van Buggenhout
evanbuggenhout@nviso.eu
www.nviso.eu

Update: G01_02

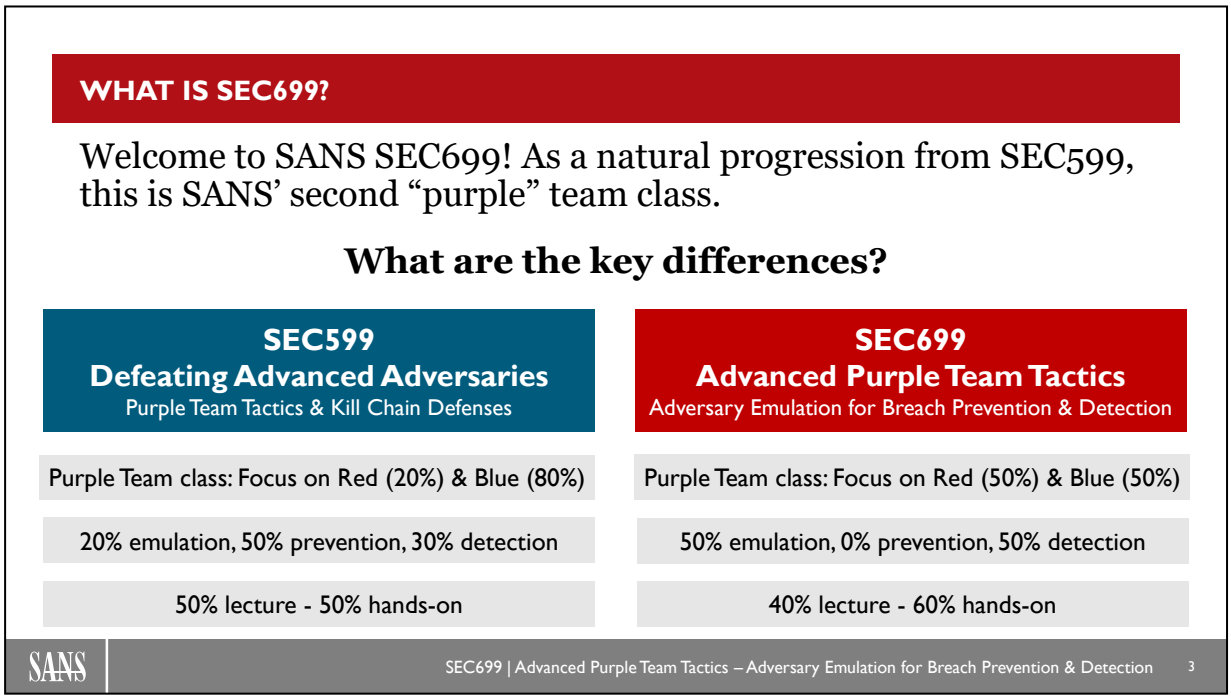
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.



What Is SEC699?

Welcome to SANS SEC699! SEC699 is SANS' advanced Purple Team offering, with a key focus on adversary emulation. Throughout SEC699, students will learn how real-life threat actors can be emulated in a realistic, enterprise, environment. In true purple fashion, the goal of the course is to educate students on how adversarial techniques can be emulated and detected.

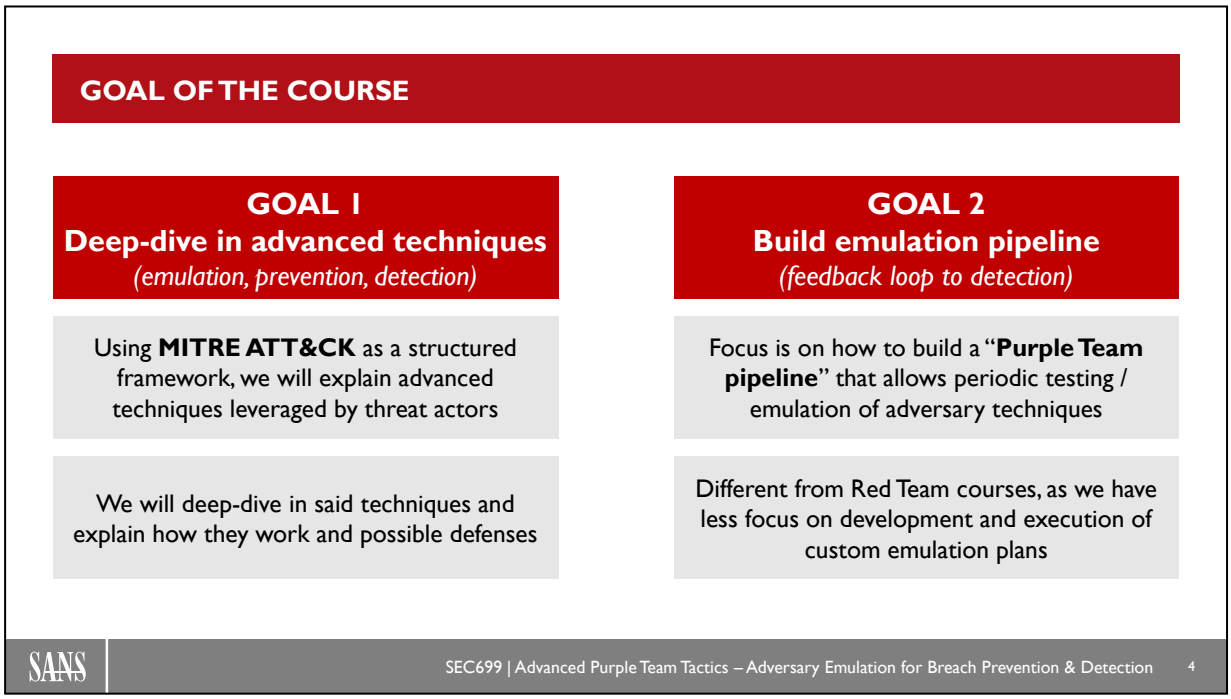
A natural follow-up after SEC599, this is an advanced course offering by SANS, which covers +-40% lecture and 60% hands-on labs! What are the main differences between 599 and 699?

SEC599 – Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

- Purple Team class: Focus on Red (20%) & Blue (80%)
- 20% emulation, 50% prevention, 30% detection
- 50% lecture – 50% hands-on

SEC699 – Advanced Purple Team Tactics – Adversary Emulation for Breach Prevention & Detection

- Purple Team class: Focus on Red (50%) & Blue (50%)
- 50% emulation, 0% prevention, 50% detection
- 40% lecture – 60% hands-on



Goal of the Course

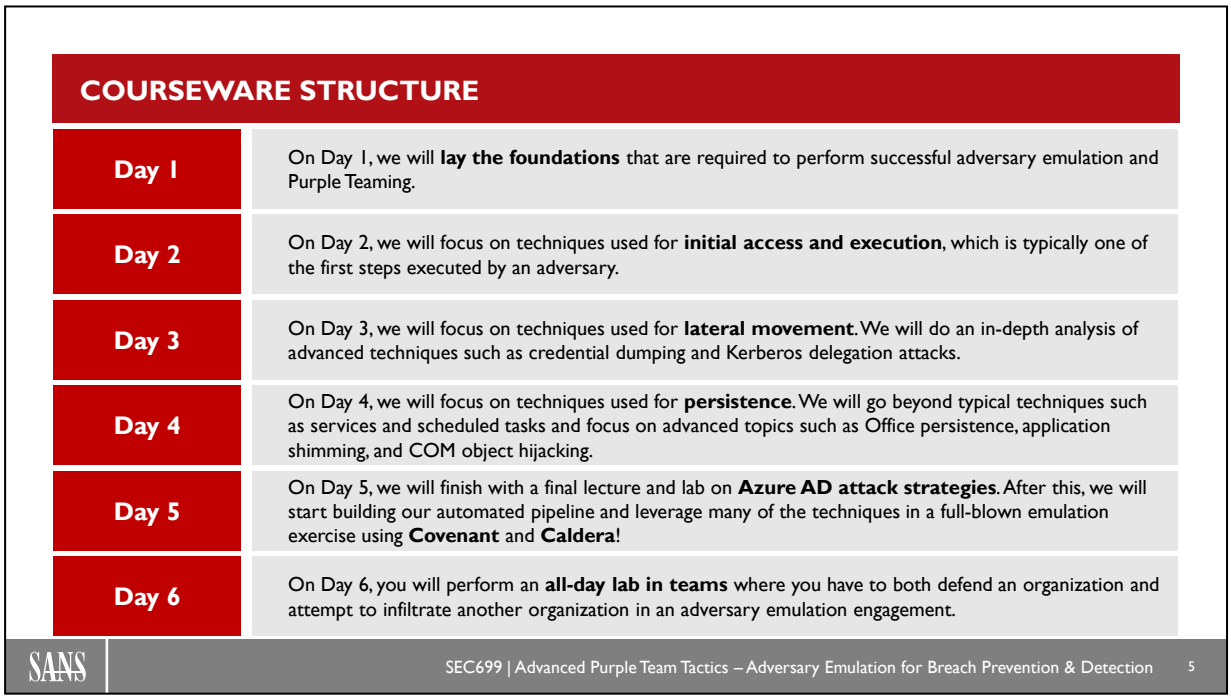
So, what is the goal of the course? When authoring the courseware, we took the following goals into account:

Goal 1: Teach students about advanced techniques used by adversaries

As a first goal, the course will discuss adversarial techniques, following MITRE ATT&CK as a structured framework. As this is a 6-level course, we will primarily focus on less basic techniques that are easy to prevent or detect. Our primary focus will be on more advanced techniques, thereby investigating options for prevention and detection. This will include both theory and a lot of practical, hands-on exercises.

Goal 2: Enable students to build an emulation pipeline

We cannot possibly explain all possible adversarial techniques during this 6-day course (we wouldn't even come close). What we can do is teach students how they can build a Purple Team pipeline that provides a feedback loop toward the cyber defense teams. In such a pipeline, the focus is on (automated) emulation of selected techniques and immediately assessing whether or not they were blocked and / or detected in the target environment.



Courseware Structure

Throughout the course, we have incorporated the following structure:

- On Day 1, we will lay the foundations that are required to perform successful adversary emulation and Purple Teaming.
- On Day 2, we will focus on techniques used for initial access and execution, which is typically one of the first steps executed by an adversary.
- On Day 3, we will focus on techniques used for privilege escalation and lateral movement. We will do an in-depth analysis of advanced techniques such as credential dumping and Kerberos delegation attacks.
- On Day 4, we will focus on techniques used for persistence. We will go beyond typical techniques such as services and scheduled tasks and focus on advanced topics such as Office persistence, application shimming, and COM object hijacking.
- On Day 5, we will finish with a final lecture and lab on Azure AD attack strategies. After this, we will start building our automated pipeline and leverage many of the techniques in a full-blown emulation exercise using Covenant and Caldera!
- On Day 6, you will perform an all-day lab in teams where you have to both defend an organization and attempt to infiltrate another organization in an adversary emulation engagement.

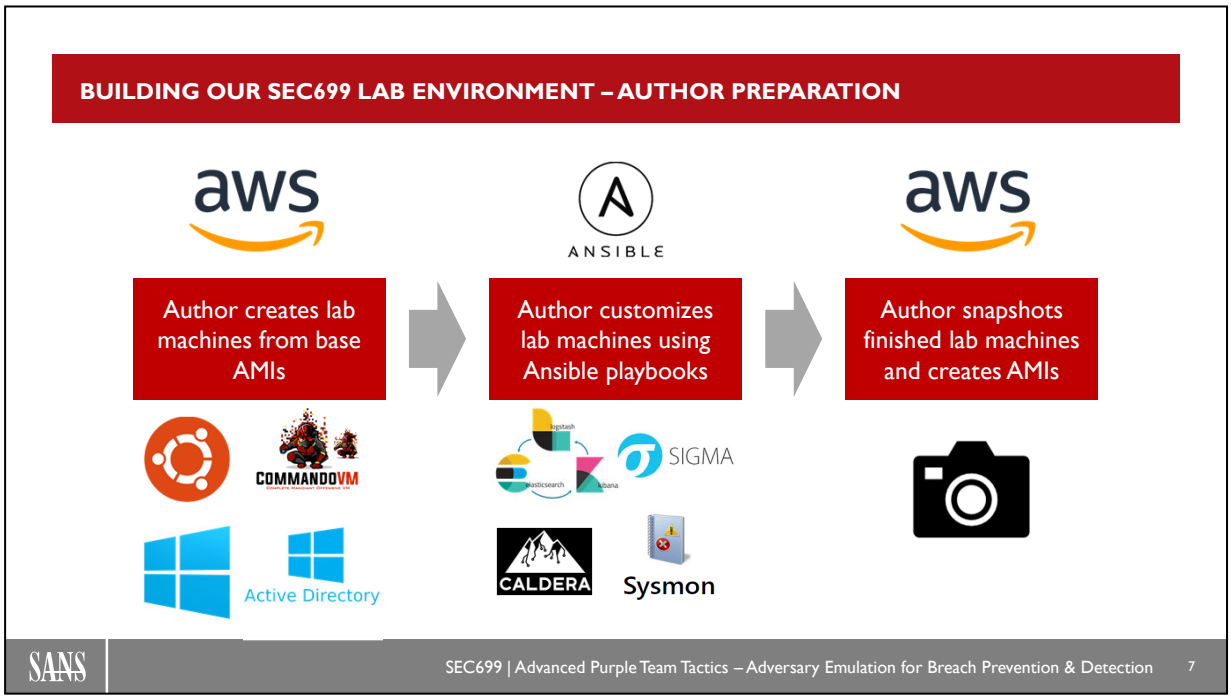
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

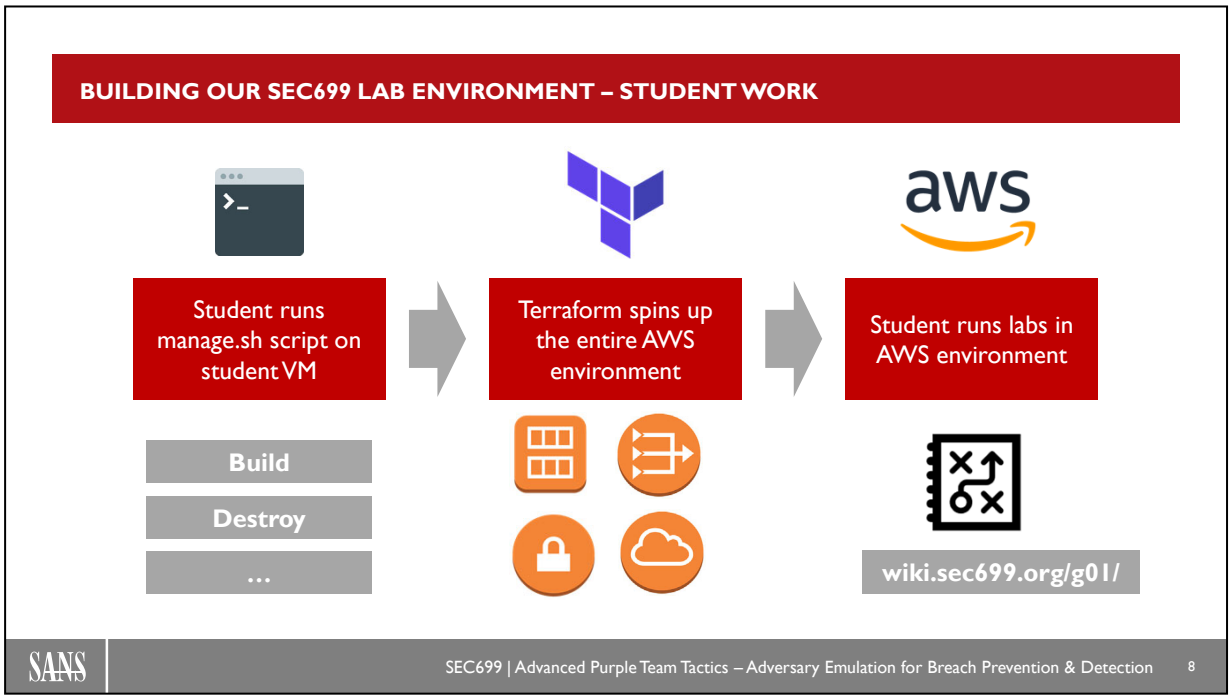


Building Our SEC699 Lab Environment – Author Preparation
When preparing the lab environment for this class, our aim was something that would easily scale across a large student base. Furthermore, we wanted to ensure the labs were easily accessible and that students could enjoy labs long after taking the class at a live event / OnDemand. We wanted to take away the hassle of setting up our own infrastructure and, thus, opted to leverage a cloud-based environment such as AWS.

So how does it all work? The author team has done quite some work to make things run smooth for you. As such, we've taken the following steps:

- We created base lab machines from base AMIs that were already available (e.g. Ubuntu, Windows,…) or AMIs created by us (e.g. CommandoVM)
- We configured the base lab machines using Ansible playbooks (install software, reconfigure settings,…)
- Once finished, the ready-to-go machines were snapshotted and saved as new AMIs

These new AMIs are then leveraged by students using Terraform, which we'll further explain in the next slide. We have added some additional details and guidance on how Ansible works in the upcoming slides. Interested in obtaining the Ansible playbooks we are using to configure our machines? Get in touch with the authors!



Building Our SEC699 Lab Environment – Student Work

Leveraging the preparation done by the author team, students can now take the following steps to run the lab exercises:

- Boot the course VM and run the "manage.sh" script, which will launch Terraform in the background
- Depending on the command executed, Terraform will launch or destroy a lab environment for the student. This lab environment is prepared by the author and primarily consists of:
  - Amazon Machine Images (AMIs)
  - NAT Gateways
  - Internet Gateways
  - Elastic IP addresses
  - VPCs
  - Security Groups
- The student connects to the CommandoVM and starts running the different labs described on https://wiki.sec699.org/g01/

We will add some additional details on the manage.sh script and Terraform in the next slides.

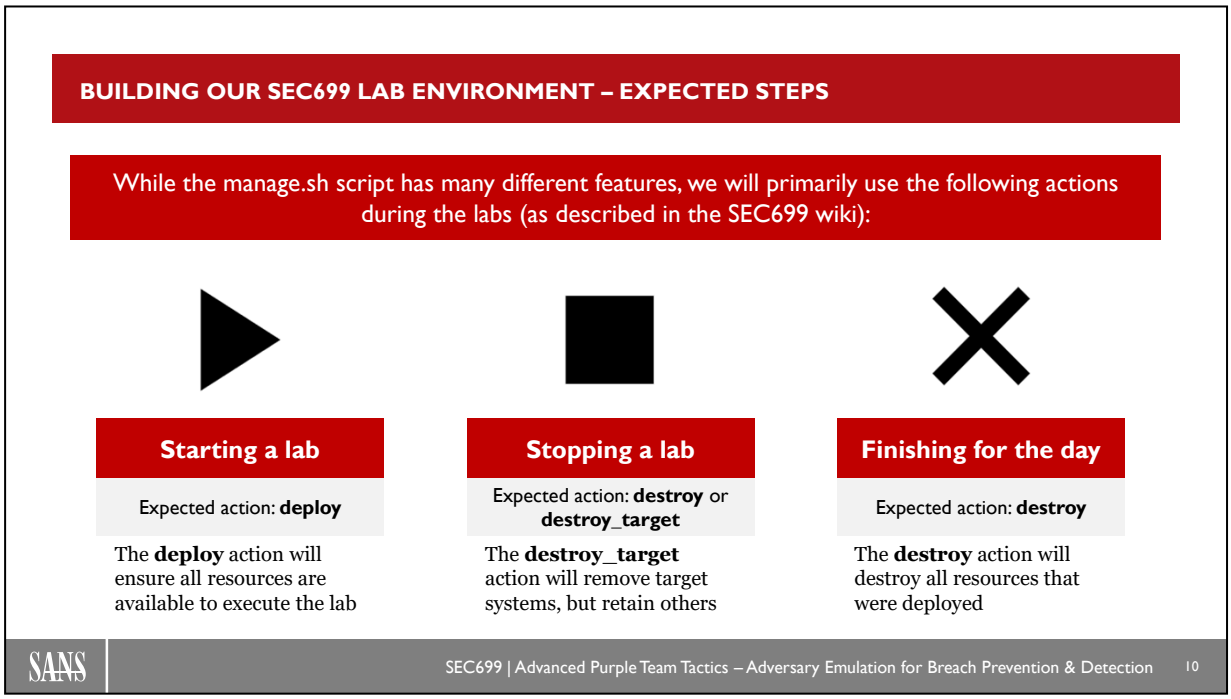
Building Our SEC699 Lab Environment – manage.sh Script

The manage.sh script is essentially a wrapper to allow students to easily manage lab environments.

It supports the following functions:

- deploy: Deploy an entire SANS SEC699 lab environment
- destroy: Destroy an entire SANS SEC699 lab environment
- destroy_target: Destroy the "target" section of a SANS SEC699 lab environment (the Windows workstations and servers)
- pause: Pause the instances deployed in a SANS SEC699 lab environment
- start: Start (resume) the instances deployed in a SANS SEC699 lab environment
- list: List all currently deployed SANS SEC699 lab environments
- configure: Configure AWS CLI credentials that can be used to spin up the environment

The manage.sh script is further documented on the wiki.

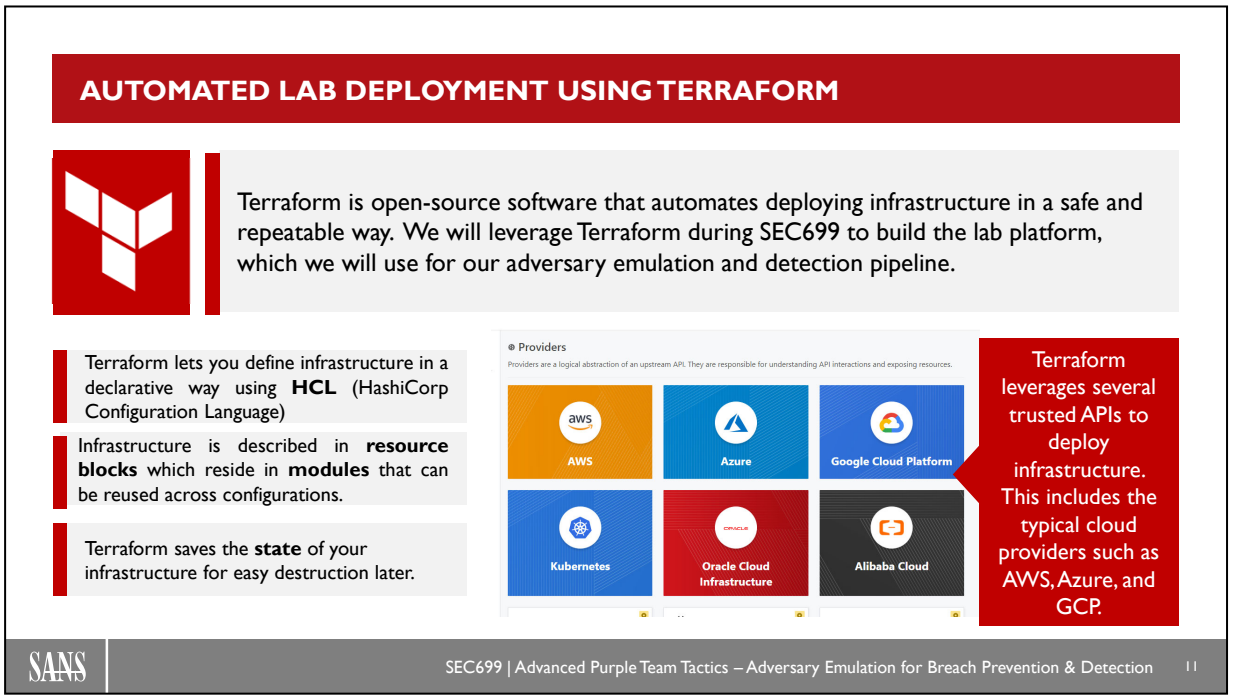


Building Our SEC699 Lab Environment – Expected Steps

While the manage.sh script has many different features, we will primarily use the following actions during the labs (as described in the SEC699 wiki):

- When starting a lab, we will use the "deploy" action to deploy everything that is needed to launch a lab environment
- When stopping a lab, we will either use "destroy" to remove everything or "destroy_target" to remove the target environment. What's the difference?
  - The entire environment includes AWS security groups, NAT gateways, Internet gateways, VPCs and all lab machines (including the targets)
  - The "target" environment only includes the Windows target machines (Windows DC, Workstations and Server). It does not include the SOC, C2 and CommandoVM systems (which we will introduce a bit later).
- When finished for the day, it's best to remove all resources by using the "destroy" command (this will avoid any unnecessary costs).

The wiki will include detailed instructions on when you are expected to run each command. However, if you get stuck or require assistance, please reach out to the instructor.



Automated Lab Deployment Using Terraform

Terraform is open-source software that automates deploying infrastructure in a safe and repeatable way. We will leverage Terraform during SEC699 to build the lab platform, which we will use for our adversary emulation and detection pipeline. What is Terraform?

- Terraform lets you define infrastructure in a declarative way using HCL (HashiCorp Configuration Language). This infrastructure can later be deployed using a variety of technology providers.
- Infrastructure is described in resource blocks which reside in modules that can be reused across configurations.
- Terraform saves the state of your infrastructure for easy destruction later.
- Terraform leverages several trusted APIs to deploy infrastructure. This includes the typical cloud providers such as AWS, Azure, and GCP.

Additional information on Terraform can be found at https://www.terraform.io/.

TERRAFORM STATE FILES

Terraform maintains the state of the configured resources in a **.tfstate** file.

This is very handy if you want to destroy, redeploy or expand on the infrastructure as **Terraform** will make smart decisions based on the current state of the resources.

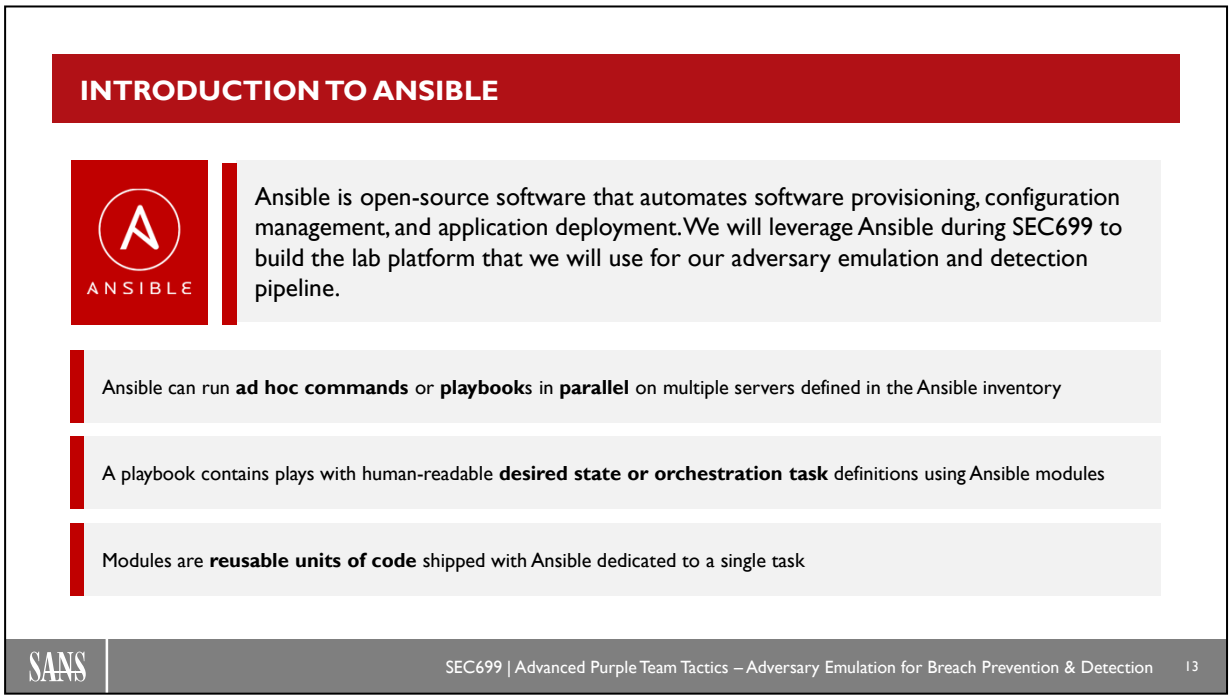
Please be very careful when dealing with the tfstate files. Manually altering these files is not expected in SEC699.

If your Terraform state files are "out of sync" you may need to revert to manually destroying AWS resources from the AWS console!

Terraform State Files
Terraform maintains the state of the configured resources in a .tfstate file. This is very handy if you want to destroy, redeploy or expand on the infrastructure as Terraform will make smart decisions based on the current state of the resources. Please be very careful when dealing with the tfstate files. Manually altering these files is not expected in SEC699.

If your Terraform state files are "out of sync" you may need to revert to manually destroying AWS resources from the AWS console!

If you encounter any issues, please don't hesitate to reach out to an instructor for support.



Introduction to Ansible
Ansible is open-source software written in Python that automates software provisioning, configuration management, and application deployment. It has become one of the most popular orchestration tools in the DevOps community because of the flat learning curve allowing administrators to quickly start automating daily tasks. Ansible has been acquired by Red Hat in 2015 which, in turn, has been acquired by IBM in 2019.

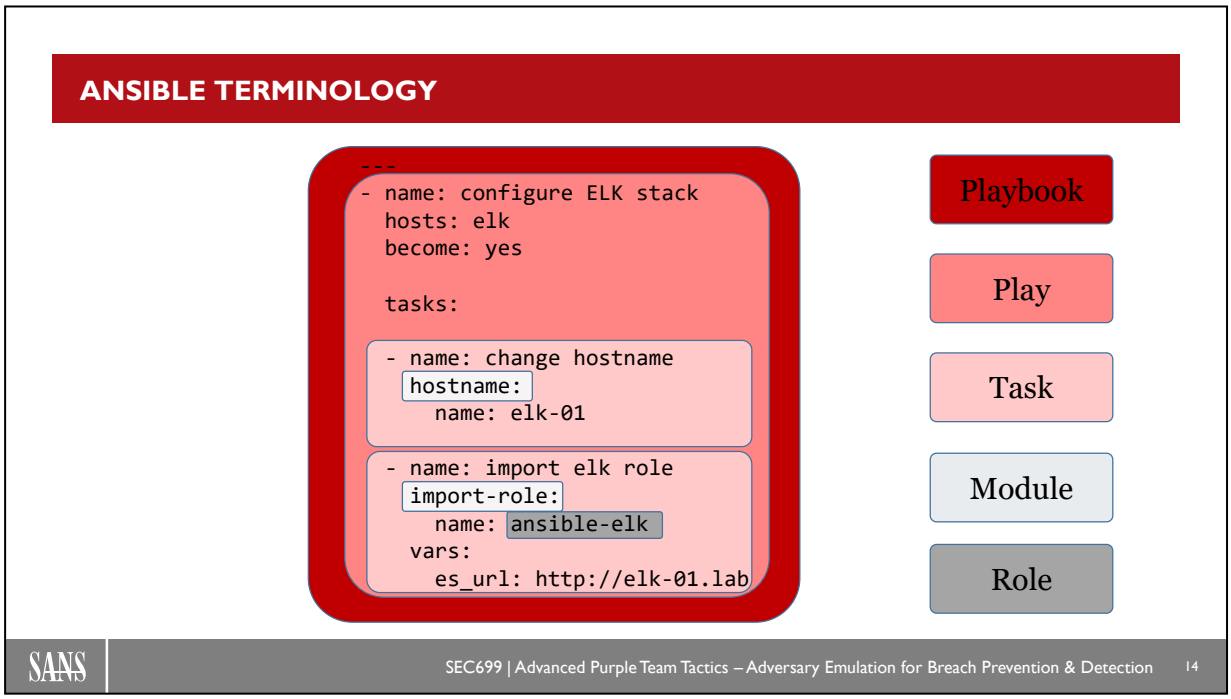
Ansible can idempotently run tasks in parallel on multiple servers defined in the Ansible inventory. In computing, Idempotency means that an operation has no subsequent effect if it is called multiple times with the same input parameters. Because of this, Ansible can be used to periodically audit or enforce a desired state on a large group of systems.

Ansible can run ad-hoc commands to execute a single task from an Ansible module or these tasks can be grouped in a playbook for reusability. Playbooks are written in YAML and can contain one or more plays with desired state configurations or orchestration tasks. The tasks are defined with Ansible modules, which are pieces of code configurable with arguments.

The configuration of Ansible is set in the ansible.cfg file in /etc/ansible, but this file can be different for each repository as a playbook will first look for the ansible.cfg file in their current directory before checking /etc/ansible.

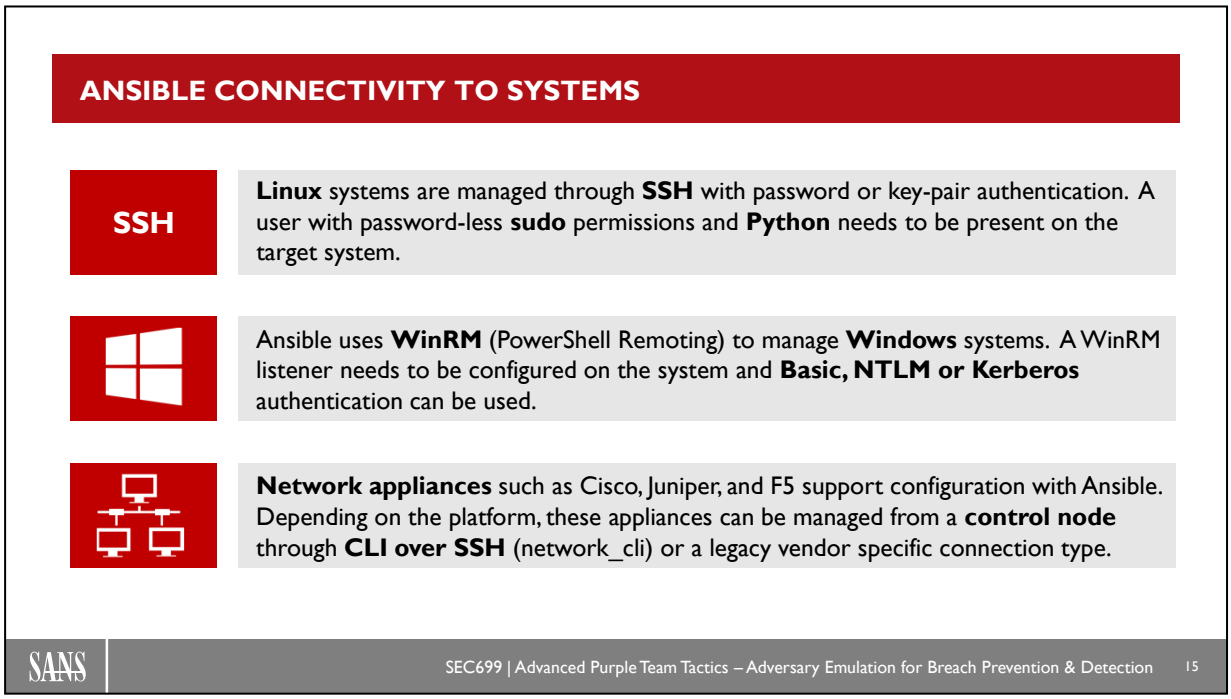
Some good references include:

https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html
https://docs.ansible.com/ansible/latest/installation_guide/intro_configuration.html



Ansible Terminology
Ansible can be a little intimidating, as there are many different terms used. Throughout the labs, you will be introduced to them as we go along. Here's a small intro:

- An Ansible playbook is written in YAML and can contain one or more plays.
- A play targets a single node, or a group of nodes, defined in the Ansible inventory file using the hosts parameter. It consists of one or more desired state configurations or orchestration tasks to configure the targeted node.
- A task uses Ansible modules to execute a single configuration or orchestration item.
- A module is a reusable unit of code build into Ansible dedicated to a single task and customized by parameters. The description of all possible parameters for a module can be found in the Ansible modules documentation. https://docs.ansible.com/ansible/latest/collections/index_module.html
- A role contains a well-defined reusable set of tasks and can be imported into a playbook by using the import-role module and customized by overloading role variables.



Ansible Connectivity to Systems
Ansible supports a wide range of platforms it can manage:

- Red Hat Enterprise Linux (6.3 and later, 7.2 and later)
- Red Hat Enterprise Linux Server 8
- Ubuntu (14.04 LTS, 16.04 LTS, 18.04 LTS (all x86_64 only))
- Windows 7, 8.1, 10
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016, 2019
- Network Devices:
  * Arista EOS
  * Cisco IOS, IOS-XE, IOS-XR, NX-OS
  * Juniper Junos OS
  * VyOS
- Others - unlisted R
- HEL variants, SuSE, Solaris, AIX, etc.

Managing Linux with Ansible is done over a Secure Shell connection using Python. When Ansible is executed, it connects to the managed node using SSH with a password or a key-pair, copies the required Python scripts for executing the tasks and runs them idempotently. Python 2.7+ or Python 3 needs to be installed on the target machine and a user with password-less sudo permissions needs to be configured to allow Ansible to execute administrative tasks without a password prompt.

Windows can also be managed by Ansible using Windows Remote Management (WinRM). It requires at least PowerShell 3.0 since Ansible uses PowerShell to execute tasks on Windows targets. The Windows target needs to be configured for PowerShell Remoting and Basic, NTLM or Kerberos authentication can be used. In a testing environment the following script can be used to configure the Windows target:

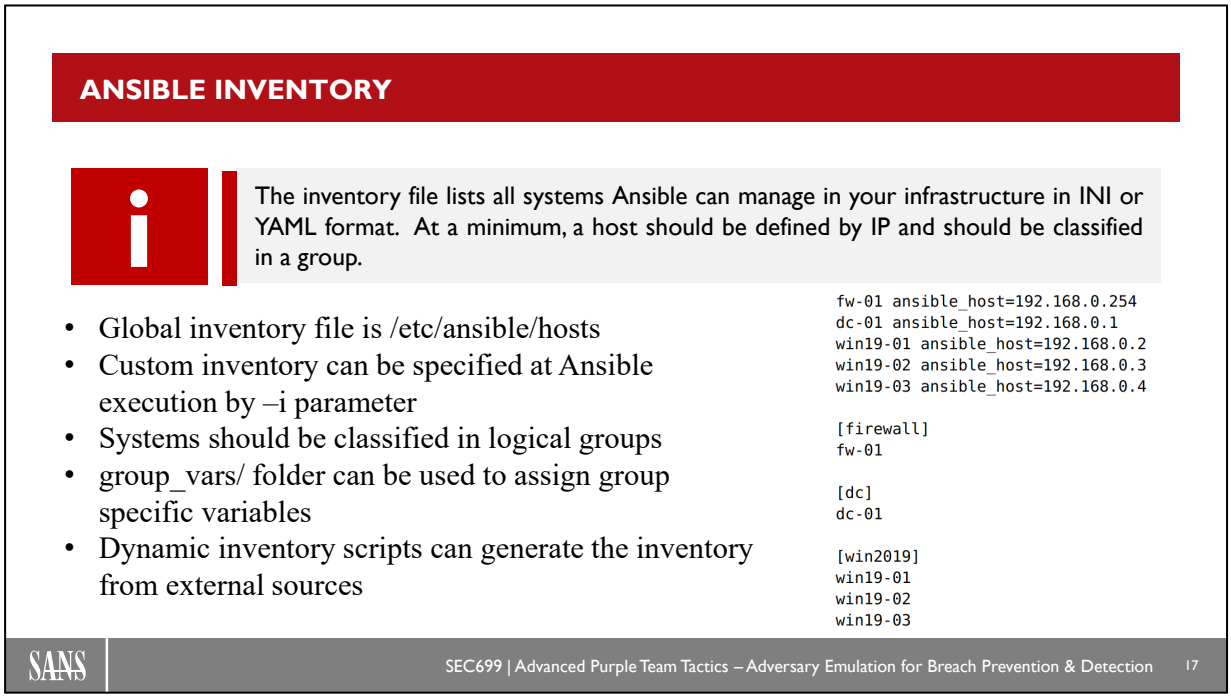
https://github.com/ansible/ansible/blob/devel/examples/scripts/ConfigureRemotingForAnsible.ps1

This script should not be used in a production environment since it configures the host for Basic authentication which sends credentials unencrypted.

Ansible can also be used for automating network administration. Vendors such as Cisco and Juniper have created Ansible modules for configuring a wide range of their devices. It uses SSH to connect but does not execute Python code on the targets such as Linux. When configuring network devices, the Ansible Playbooks are run from a control node and the configuration is pushed to the device.

Some good references include:

https://access.redhat.com/articles/3168091
https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html
https://docs.ansible.com/ansible/latest/user_guide/windows.html
https://docs.ansible.com/ansible/latest/network/getting_started/index.html

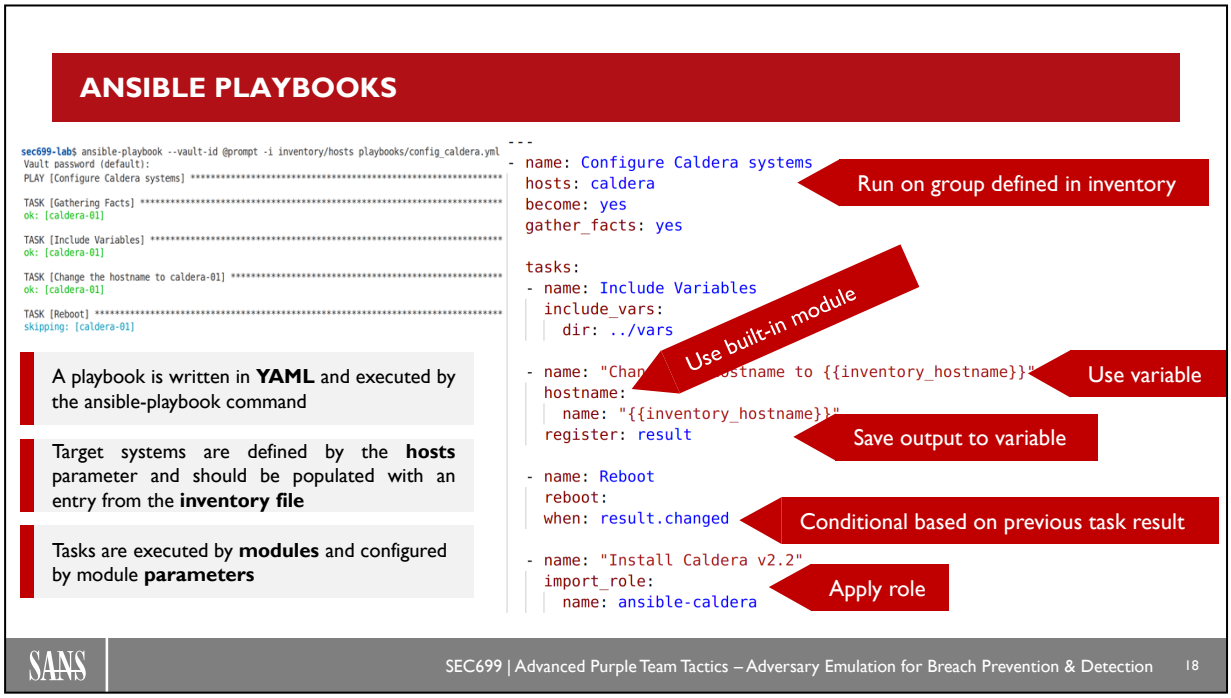


Ansible Inventory
The Ansible inventory file lists all nodes that can be managed by Ansible in INI or YAML format. At a minimum, it should contain the IP address Ansible will use to connect to the node, but I can also list hostnames and connection information such as usernames, password, and connection types. The nodes should be classified in logical groups, which can be used to target the execution of tasks or playbook.

The global Ansible inventory file is located at /etc/ansible/hosts, but a custom inventory file can be specified when executing Ansible by using the –i parameter. It is best practice to not use the global inventory file but to create a custom inventory file for each repository of Ansible playbooks as this will also be checked in into source control and it allows you to be more granular on a per playbook basis.

The settings for connecting to nodes can be specified per node in the inventory file, but for more flexibility, a groups_vars folder can be created in the folder with the inventory file. Here, connection settings can be set based on the group defined in the inventory. You can create a groups_vars/windows folder containing the connection details for the Windows inventory group.

Inventory files can be dynamically generated using dynamic inventory scripts. With such a script, you can query AWS, Azure, VMware, Red Hat Satellite, … to list all nodes on their infrastructure, and dynamically generate an Ansible inventory to target your Ansible configuration tasks.



Ansible Playbooks
Ansible playbooks are a collection of Ansible tasks written in YAML that can be targeted at a node or group defined in the Ansible inventory. These tasks are run consecutively, and a task is run concurrently on each targeted host. Playbooks are run by the ansible-playbook command:

ansible-playbook playbook.yml

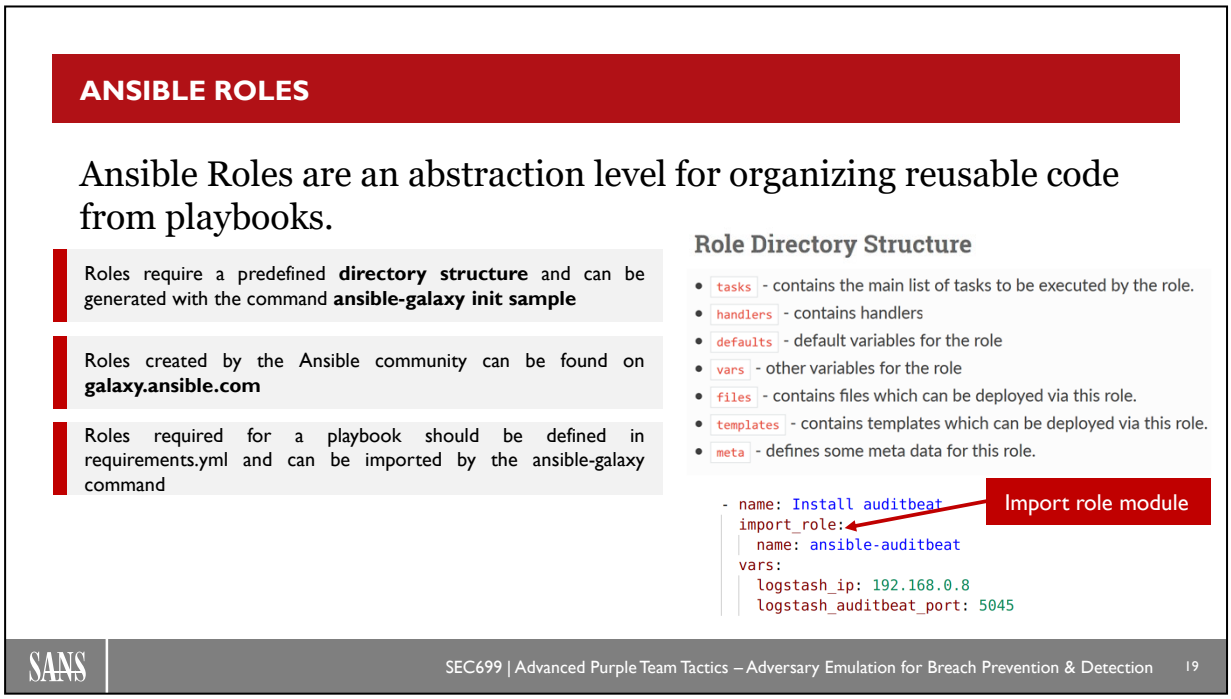
In a playbook, tasks call Ansible modules, and these are configured with the module parameters. Ansible has a huge number of built-in modules allowing you to perform a wide range of tasks. The modules are documented in the module index in the Ansible documentation:

https://docs.ansible.com/ansible/latest/collections/index_module.html

The Ansible syntax allows for the use of variables, looping, conditionals in tasks.

Please find additional details on playbooks and modules here:

https://docs.ansible.com/ansible/latest/user_guide/playbooks.html
https://docs.ansible.com/ansible/latest/user_guide/modules.html



Ansible Roles
Ansible roles are an abstraction level for organizing reusable code from playbooks. A role contains a well-defined reusable set of tasks for configuring systems. This role can be added to a playbook by the import_role module and configured by overloading variables defined within the role.

The Ansible community has created a large number of roles that can be found on galaxy.ansible.com. The roles here can be directly used in your playbooks or can serve as an inspiration for your own playbooks or roles.

Ansible roles required a specific directory structure. The following folders must be present:

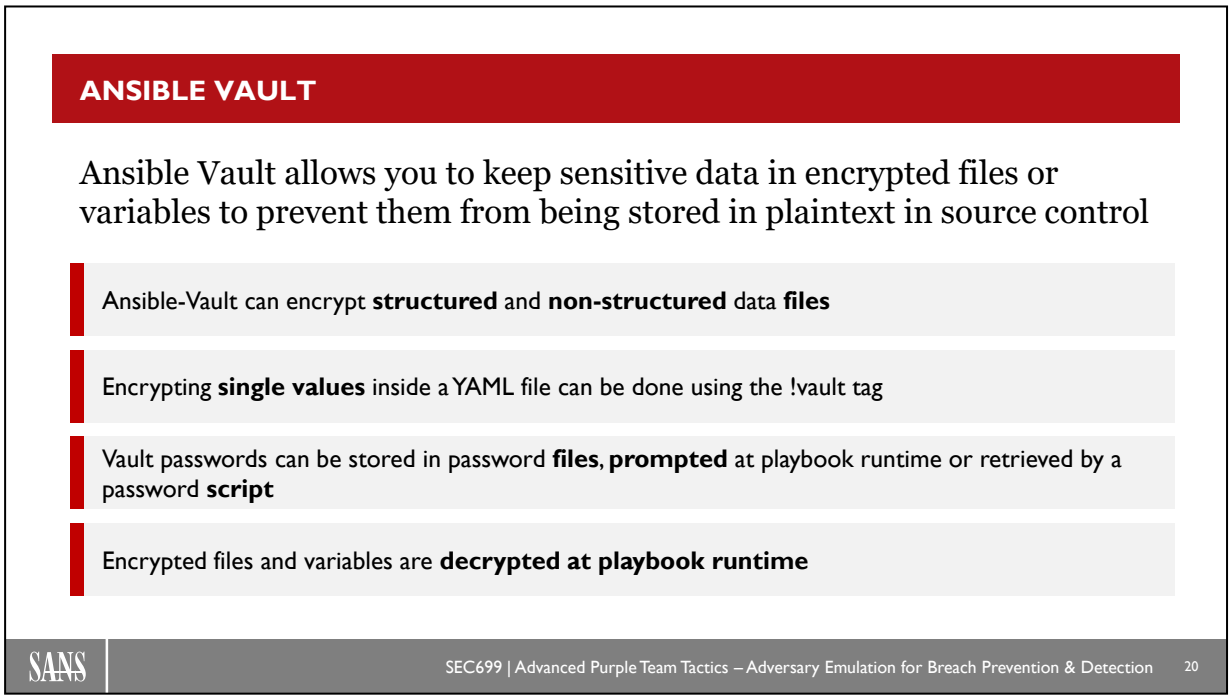
- tasks: Contains the main list of tasks to be executed by the role
- handlers: Contains handlers
- defaults: Default variables for the role
- vars: Other variables for the role
- files: Contains files which can be deployed via this role
- templates: Contains templates which can be deployed via this role
- meta: Defines some metadata for this role

This structure can be generated by the ansible-galaxy command:

ansible-galaxy init sample

The ansible-galaxy command can also be used to download Ansible roles from GitHub or galaxy.ansible.com. The required roles should be defined in a requirements.yml.

References:
https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html
https://docs.ansible.com/ansible/latest/galaxy/user_guide.html



Ansible Vault
Ansible vault is used to encrypt sensitive structured or non-structured data such as passwords, encryption keys or databases used in your Ansible playbooks. This prevents them from unauthorized access and from being stored in plaintext in source control.

Ansible vault can encrypt entire files with a password, or it can encrypt single values inside a YAML file using the !vault tag.

Encrypted files and variables are decrypted at playbook runtime by passing the --vault-id parameter to the ansible-playbook command. The decryption key can be stored in password files, prompted at playbook runtime or retrieved by a password script for use with an external key vault.

Please find additional details on the Ansible vault on the following URL:

https://docs.ansible.com/ansible/latest/user_guide/vault.html

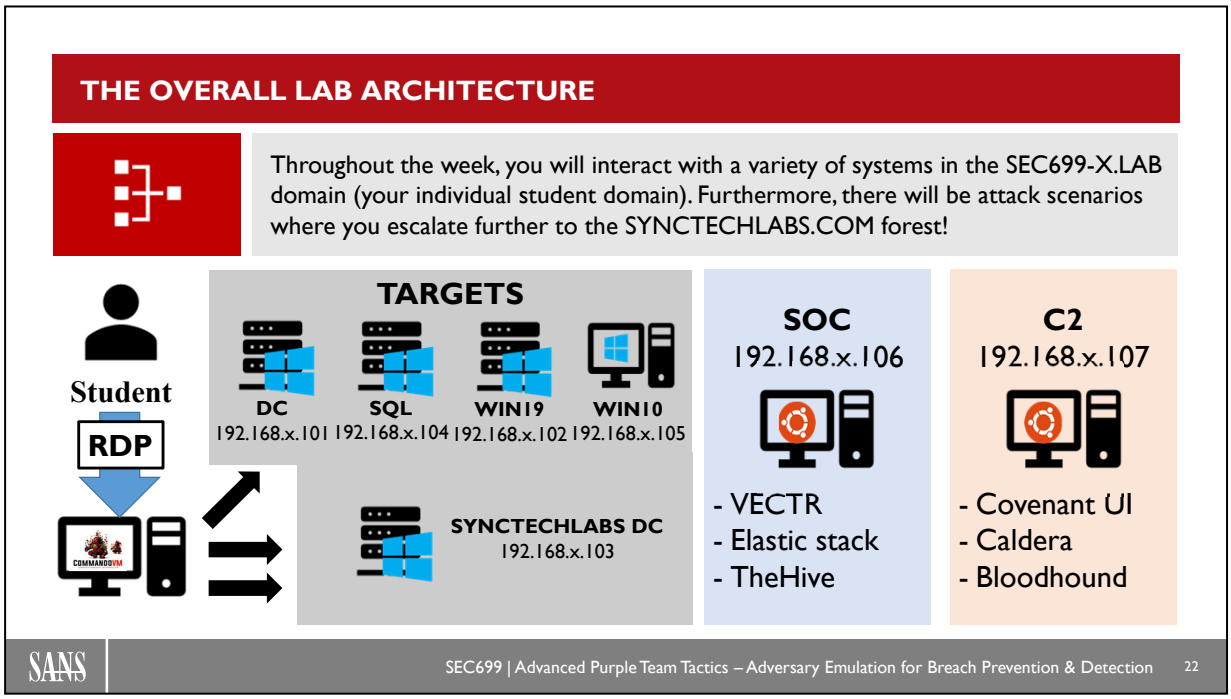
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.



The Overall Lab Architecture
Throughout the week, you will interact with a variety of systems in the SEC699-X.LAB domain (your individual student domain). Furthermore, there will be attack scenarios where you (attempt to) escalate further to the SYNCTECHLABS.COM forest. As a starting point, you will be running a CommandoVM virtual machine from where you will start the labs.

You will have the following systems at your disposal:

- A domain controller for SEC699-XX.LAB (Windows Server 2019)
- A SQL server in SEC699-XX.LAB (Windows Server 2019)
- A Windows 2019 server in SEC699-XX.LAB (Windows Server 2019)
- A Windows 10 workstation in SEC699-XX.LAB (Windows 10)
- A SOC machine that is receiving logs from all domain-joined systems (Ubuntu)
- A C2 machine that is hosting several attacker tools (Ubuntu)
- A domain controller for SYNCTECHLABS.COM (Windows Server 2019)

COMMANDOVM AS THE MAIN LAB MACHINE

For most labs, you'll work with **CommandoVM**, which you will Remote Desktop (RDP) into from the Course VM. The course VM is only used to spin up the lab environment. All functional lab activities start from the CommandoVM.
The required credentials are username "**student**" and password "**student**".

FireEye aims to provide a penetration testing / adversary emulation framework that can rival Kali Linux. Among many other tools, CommandoVM has a built-in Kali, as it leverages the Windows Subsystem for Linux (WSL).

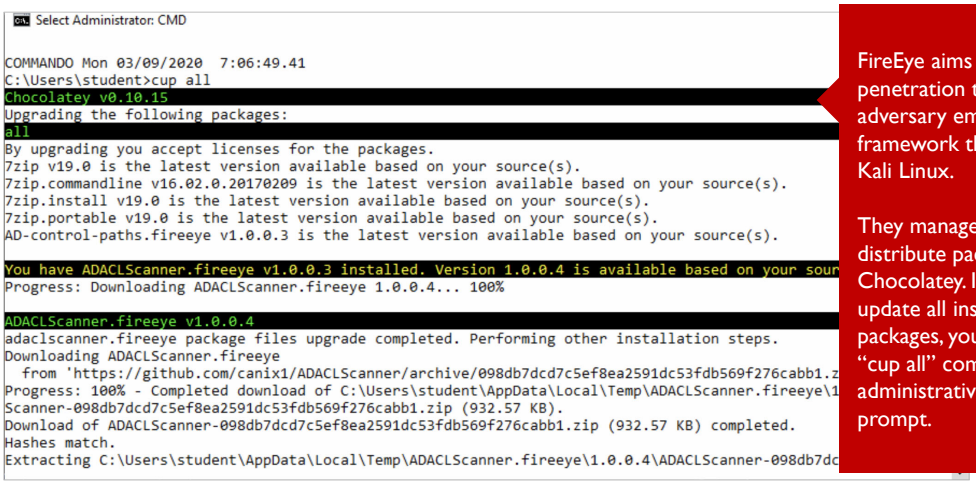
They manage and distribute packages using Chocolatey. In order to update all installed packages, you can use the "cup all" command in an administrative command prompt.

CommandoVM as the Main Lab Machine
For most labs, you'll work with CommandoVM, which you will Remote Desktop (RDP) into from the Course VM. The course VM is only used to spin up the lab environment. All functional lab activities start from the CommandoVM. The required credentials are username "student" and password "student".

So what is CommandoVM?

FireEye aims to provide a penetration testing / adversary emulation framework that can rival Kali Linux. Among many other tools, CommandoVM has a built-in Kali, as it leverages the Windows Subsystem for Linux (WSL). They manage and distribute packages using Chocolatey. In order to update all installed packages, you can use the "cup all" command in an administrative command prompt.

COMMANDOVM: UPDATING PACKAGES



FireEye aims to provide a penetration testing / adversary emulation framework that can rival Kali Linux.

They manage and distribute packages using Chocolatey. In order to update all installed packages, you can use the "cup all" command in an administrative command prompt.

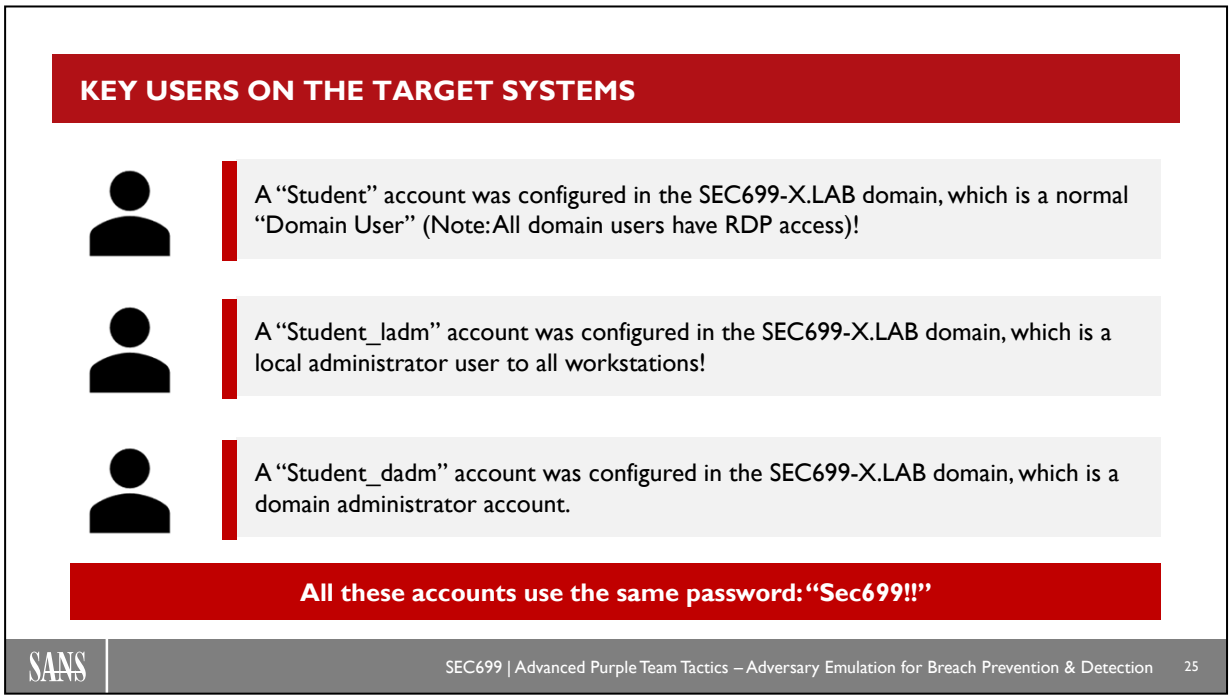
CommandoVM: Updating Packages
With CommandoVM, FireEye aims to provide a penetration testing / adversary emulation framework that can rival Kali Linux. The essence is, of course, in ease of package management and installation. In order to facilitate this on a Windows machine, CommandoVM manages and distributes packages using Chocolatey.

More information on Chocolatey:

Chocolatey is a software management solution unlike anything else you've ever experienced on Windows. Chocolatey brings the concepts of true package management to allow you to version things, manage dependencies and installation order, better inventory management, and other features.

All details can be found at https://chocolatey.org/

In order to update all installed packages, you can use the "cup all" command in an administrative command prompt.



Key Users on the Target Systems
Throughout most of the labs, you will use the following user accounts:

- Student: A "Student" account was configured in the SEC699-X.LAB domain, which is a normal "Domain User" (Note: All domain users have RDP access)!
- Student_ladm: A "Student_ladm" account was configured in the SEC699-X.LAB domain, which is a local administrator user to all workstations!
- Student_dadm: A "Student_dadm" account was configured in the SEC699-X.LAB domain, which is a domain administrator account.

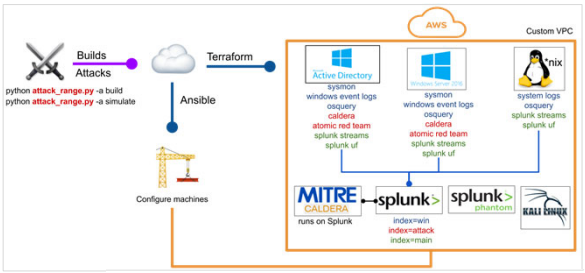
Note that all these accounts use the same password: "Sec699!!".

This combination of users with different privileges will allow us to emulate different parts of an emulation plan / kill chain (e.g., a standard business user without local admin privileges, a local administrator user,…).

SOME OTHER INITIATIVES…

DETECTIONLAB

SOURCE: https://github.com/clong/DetectionLab



SOURCE: https://github.com/splunk/attack_range

Detection Lab (by Chris Long) and Splunk Attack Range (by Jose Hernandez and Patrick Bareiss at Splunk) are two other highly interesting initiatives that help defenders easily spin up lab environments to test attack techniques and develop detection rules!

Some Other Initiatives…
While building this class, we did a lot of efforts to create a lab environment that can be used outside of the classroom and that can be further extended for custom features, scenarios or tools.

Should you be looking for an easy lab environment to perform detection engineering, there are several other ongoing community initiatives which could be worth considering:

Detection Lab was built by Chris Long and provides a base Windows domain environment that includes a Windows 2016 domain controller, a Windows 2016 server (for WEF), a Windows 10 workstation, and an Ubuntu 16.04 server that runs Splunk. Furthermore, it includes several tools for increased detection such as Microsoft ATA, Sysmon, and OSQuery. It can be easily deployed on workstations' virtualization systems (e.g., VMware of VirtualBox) or hypervisors (ESXi, Azure, AWS, HyperV,…). Please refer to https://github.com/clong/DetectionLab for full details!

The Splunk Attack Range was built by Jose Hernandez and Patrick Bareiss. Its lab environment is described above, but also includes a typical Windows domain environment and attack / defense tools (Kali Linux, Splunk, Sysmon, MITRE Caldera, Atomic Red Team…). It can be deployed locally using Vagrant and VirtualBox or in the cloud using AWS / Terraform.

References:
https://github.com/clong/DetectionLab
https://github.com/splunk/attack_range

Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

EXERCISE: GETTING TO KNOW THE LAB ENVIRONMENT

Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

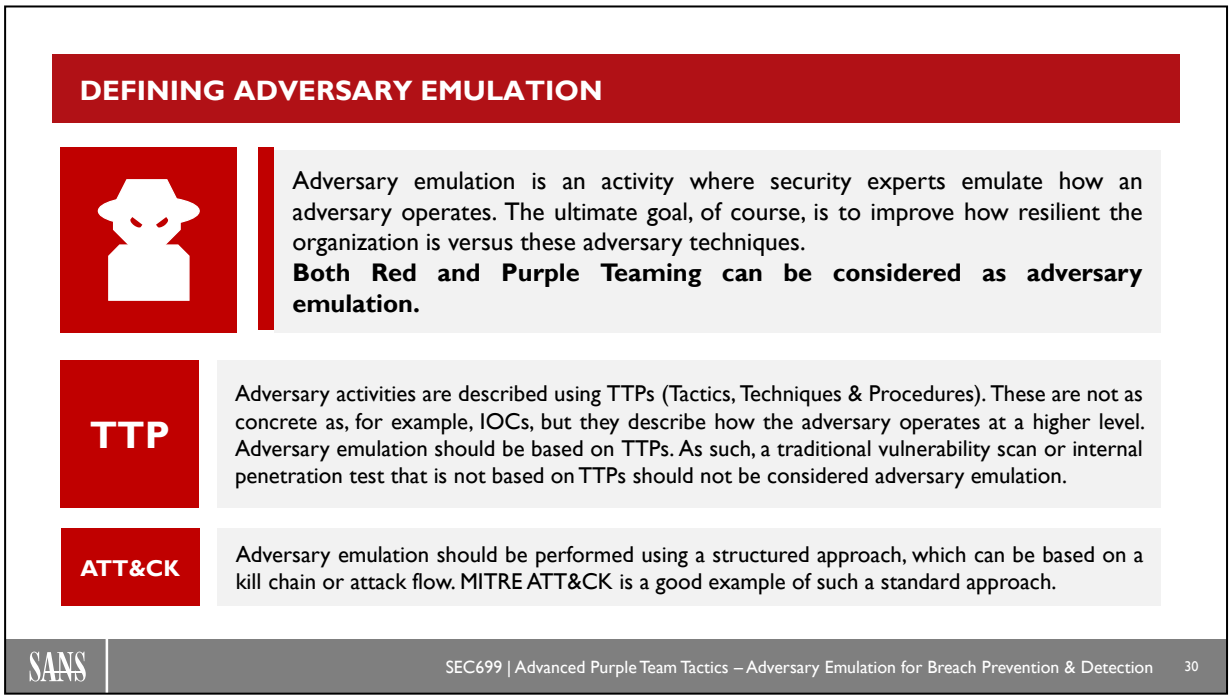
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

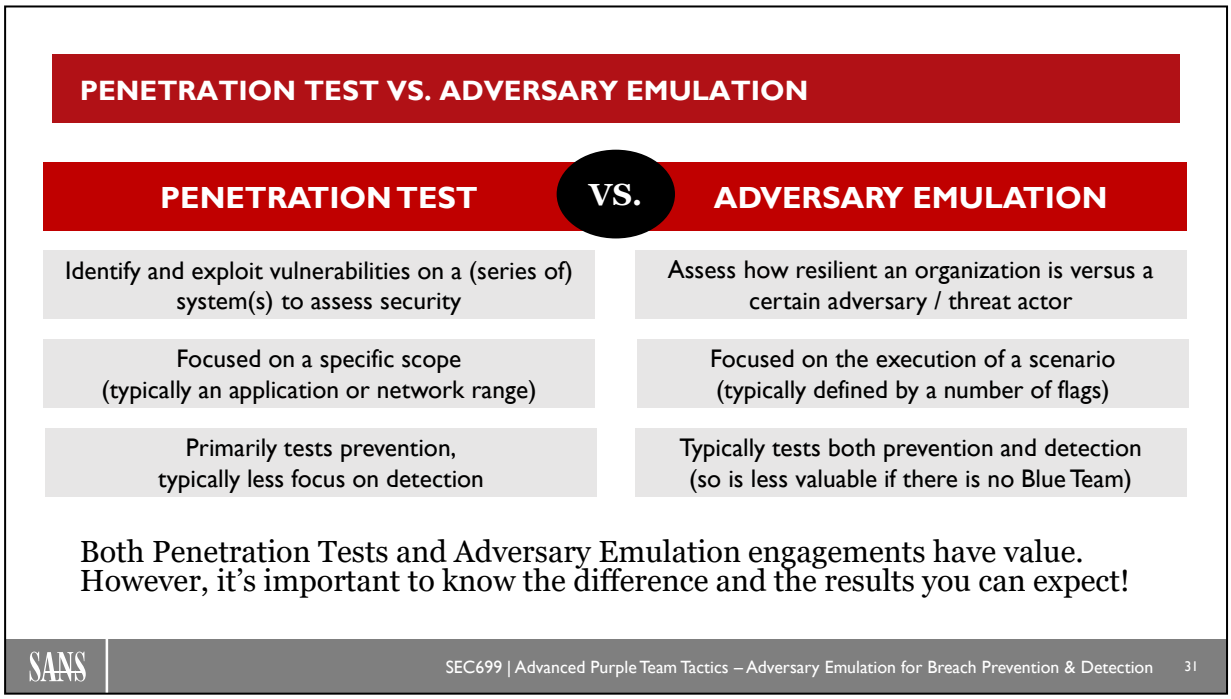


Defining Adversary Emulation

Adversary emulation is an activity where security experts emulate how an adversary operates. The ultimate goal, of course, is to improve how resilient the organization is versus these adversary techniques. Both red and Purple Teaming can be considered as adversary emulation.

One of the primary properties of adversary emulation is the use of TTPs. Adversary activities are typically described using TTPs (Tactics, Techniques & Procedures). TTPs are used by both Red / Purple Teams (when emulating attacks) and by Blue Teams (when analyzing actual attacks that are taking place). These are not as concrete as, for example, IOCs, but they describe how the adversary operates at a higher level. Adversary emulation should be based on TTPs. As such, a traditional vulnerability scan or internal penetration test that is not based on TTPs should not be considered adversary emulation.

Adversary emulation should be performed using a structured approach, which can be based on a kill chain or attack flow. MITRE ATT&CK is a good example of such a standard approach.



Penetration Test vs. Adversary Emulation
We often hear different terms used interchangeably in cybersecurity. You have probably heard of (some of) the following terms:

- Penetration test
- Adversary emulation
- Red Team

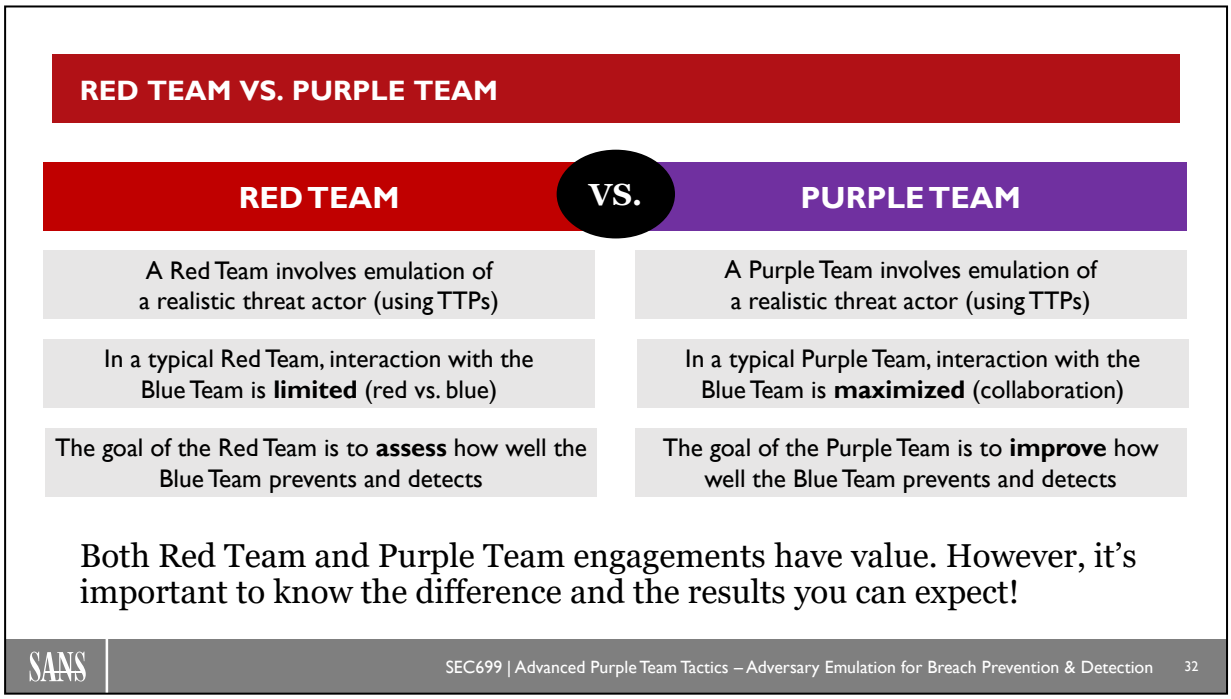
Although people can have a different understandings of different terms, it's important to have some consistency. We will define a penetration test with the following characteristics:

- The focus is to identify and exploit vulnerabilities on a (series of) system(s) to assess security
- Focused on a specific scope (typically an application or network range)
- Primarily tests prevention, typically less focus on detection

We will define adversary emulation with the following characteristics:

- Assess how resilient an organization is versus a certain adversary / threat actor
- Focused on the execution of a scenario (typically defined by a number of flags)
- Typically tests both prevention and detection (so is less valuable if there is no Blue Team)

Both Penetration Tests and Adversary Emulation engagements have value. However, it's important to know the difference and the results you can expect!

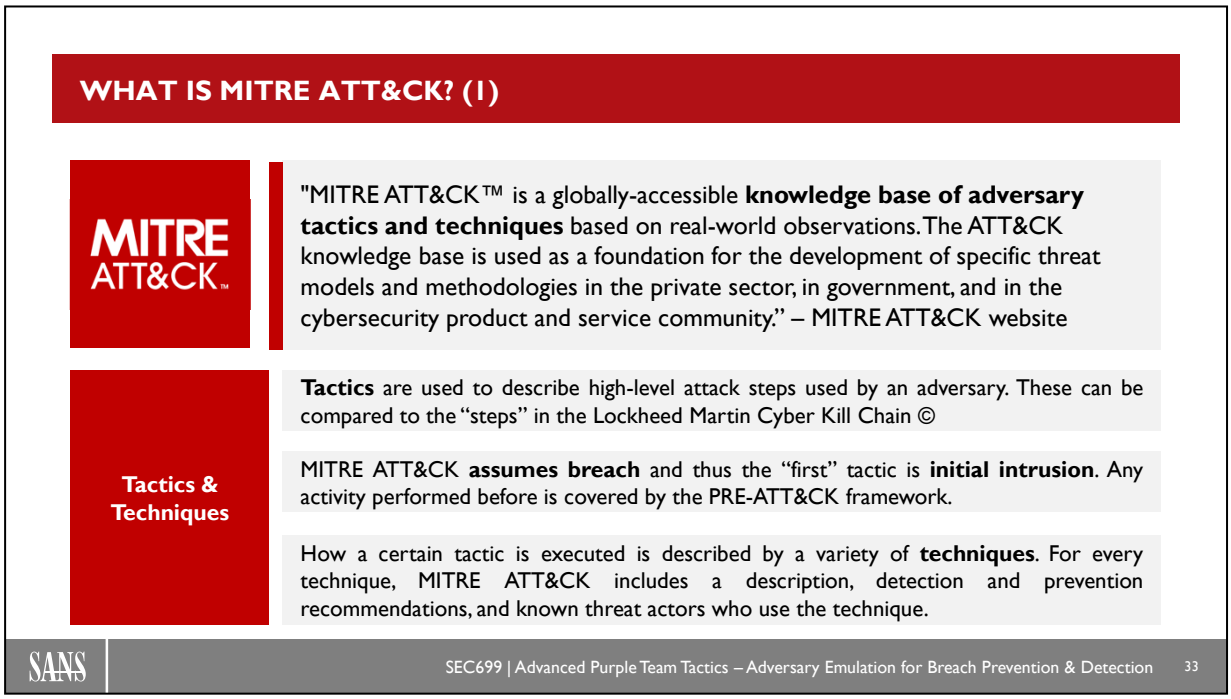


Red Team vs. Purple Team
Now that we have defined adversary emulation, let's make a distinction between two types of performing adversary emulation: Red Team engagements and Purple Team engagements. Red Team engagements have become rather well-known, and many organizations organize Red Team engagements periodically. Purple Team engagements are a bit more recent, and they aren't that well-known yet.

So, what are they all about? Both Red and Purple Team engagements involve emulation of a realistic threat actor, using known Tactics, Techniques & Procedures (TTPs). There are, however, a few distinct differences:

- In a typical Red Team, interaction with the Blue Team is limited (Red vs. Blue). In a typical Purple Team, interaction with the Blue Team is maximized, as Red and Blue collaborate.
- The goal of the Red Team is to assess how well the Blue Team prevents and detects. In a typical Purple Team, the goal of the engagement is to immediately improve how well the Blue Team prevents and detects to the Red Team efforts.

It's important to note that there is no "winner" here: Both Red Team and Purple Team engagements have value. However, it's important to know the difference and the results you can expect.



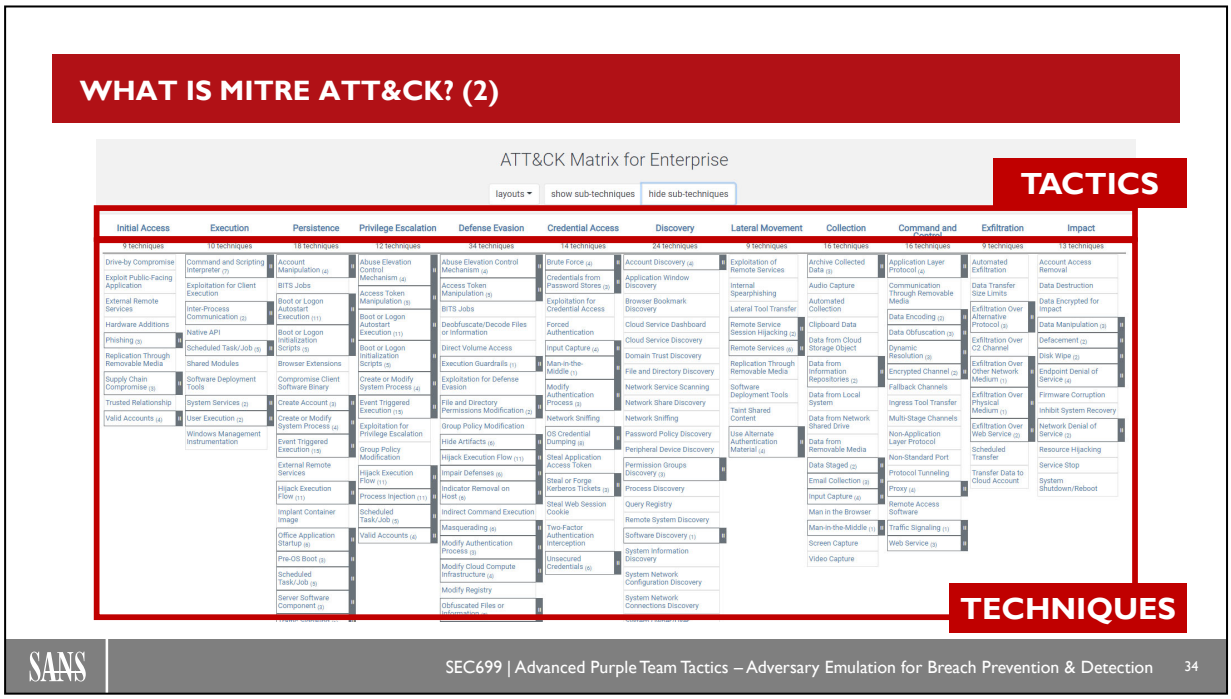
What Is MITRE ATT&CK? (1)
MITRE ATT&CK is rapidly becoming / has rapidly become a standard in the cybersecurity industry. So, what is it all about?

"MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community." – MITRE ATT&CK website

Let's add some more structure to the description:

- Tactics are used to describe high-level attack steps used by an adversary. These can be compared to the "steps" in the Lockheed Martin Cyber Kill Chain©. MITRE ATT&CK assumes breach and thus the "first" tactic is initial intrusion. Any activity performed before is covered by the PRE-ATT&CK framework.
- How a certain tactic is executed is described by a variety of techniques. For every technique, MITRE ATT&CK includes a description, detection and prevention recommendations, and known threat actors who use the technique.

Please refer to https://attack.mitre.org/ for additional details.

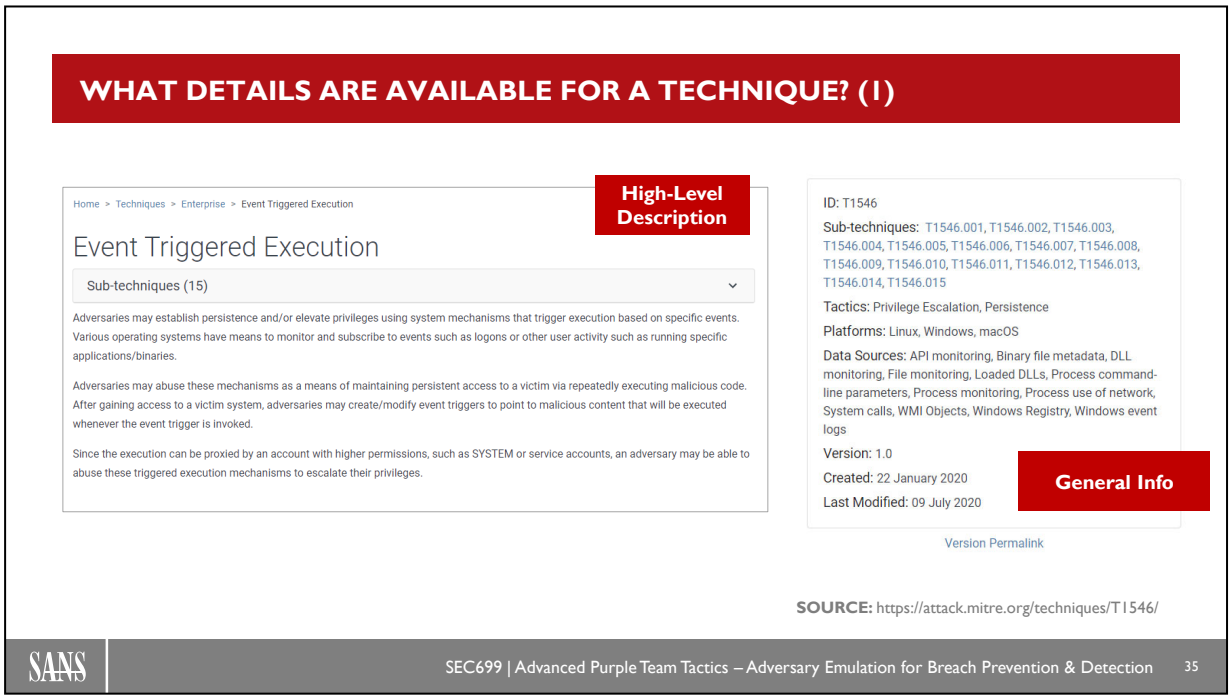


What Is MITRE ATT&CK? (2)
The above slide provides a screenshot of the current version of the MITRE ATT&CK for enterprise matrix at time of writing.

The first row includes the tactics, while the other rows include techniques that can be used to accomplish those tactics.

A relatively recent addition to MITRE ATT&CK are sub-techniques, which are concrete examples of how techniques are implemented.

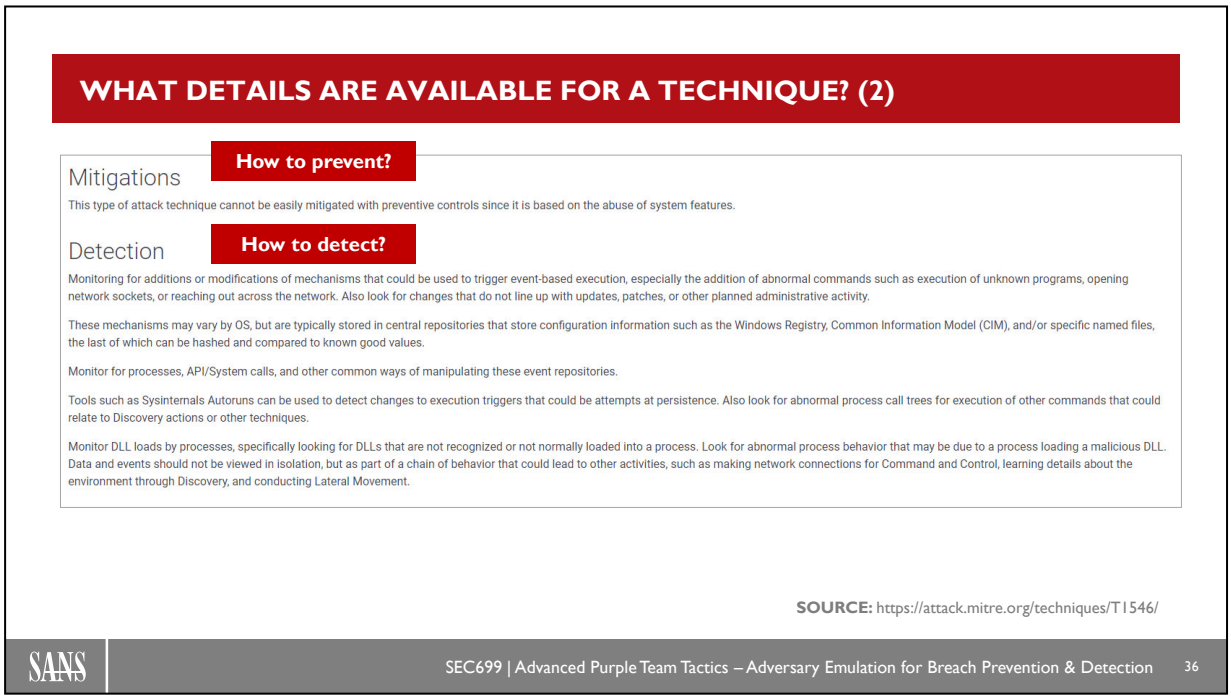
On the image in the slide, they are illustrated by the number (the number indicates how many sub-techniques a certain technique has).



What Details are Available for a Technique? (1)
So, what type of information is available for the different techniques in MITRE ATT&CK? MITRE has spent a lot of time and effort to make the matrix actionable!

Thus, they provide highly useful information such as:

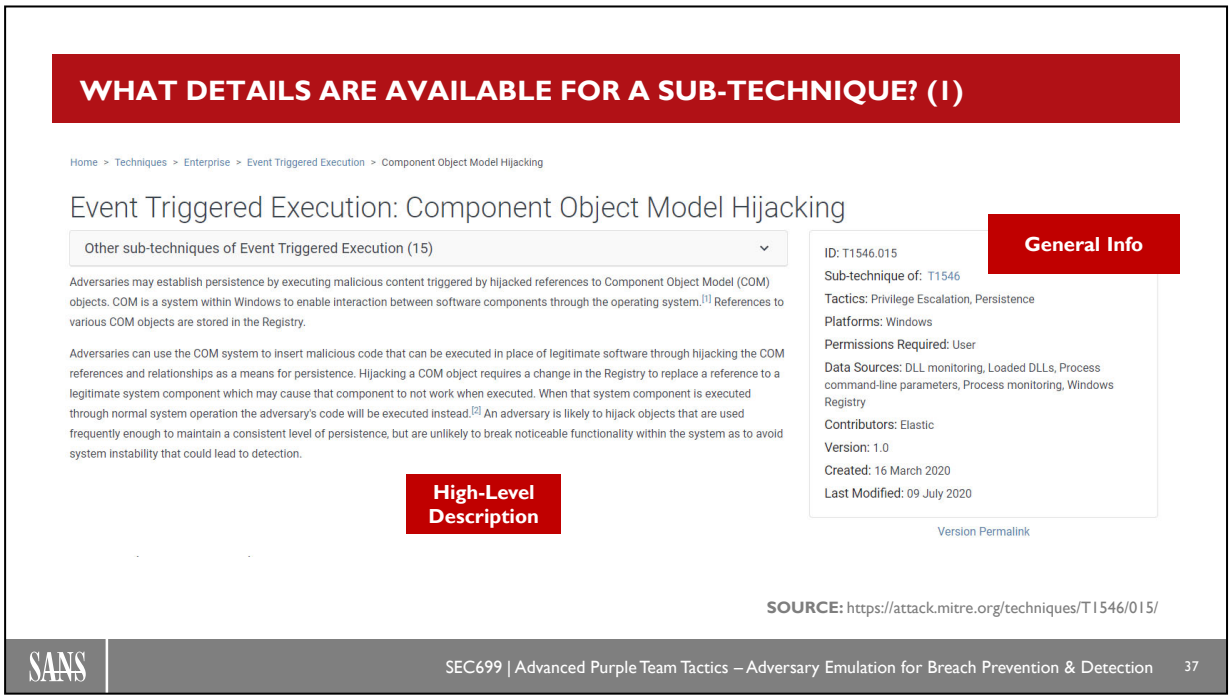
- A high-level description of the technique, explaining how it works and why an adversary would use it
- Some general information about the technique:
  - The tactics it can be found under
  - The sub-techniques that are available
  - The platform(s) it is relevant for (in this case, for example, we can see that the technique covers Linux, Windows, and MacOS)
  - Data sources for detection (we will discuss this further later in class)
  - Latest update information



What Details are Available for a Technique? (2)
Finally, MITRE ATT&CK also provides information on how these techniques can be prevented (under "mitigation") or detected (under "detection"). This can provide valuable insights for IT administrators, to better understand defensive strategies that can be used to increase the security posture of their organizations.

MITRE ATT&CK will not go to the depth of suggesting actual "command lines" to fix vulnerabilities or "detection rules" to implement in your security monitoring efforts. They will, however, provide a high-level description of the defense approach to be used.

Furthermore, note that in our example, this information is relatively high-level, as this technique entails 15 more specific different sub-techniques (as can be seen on the previous slide).

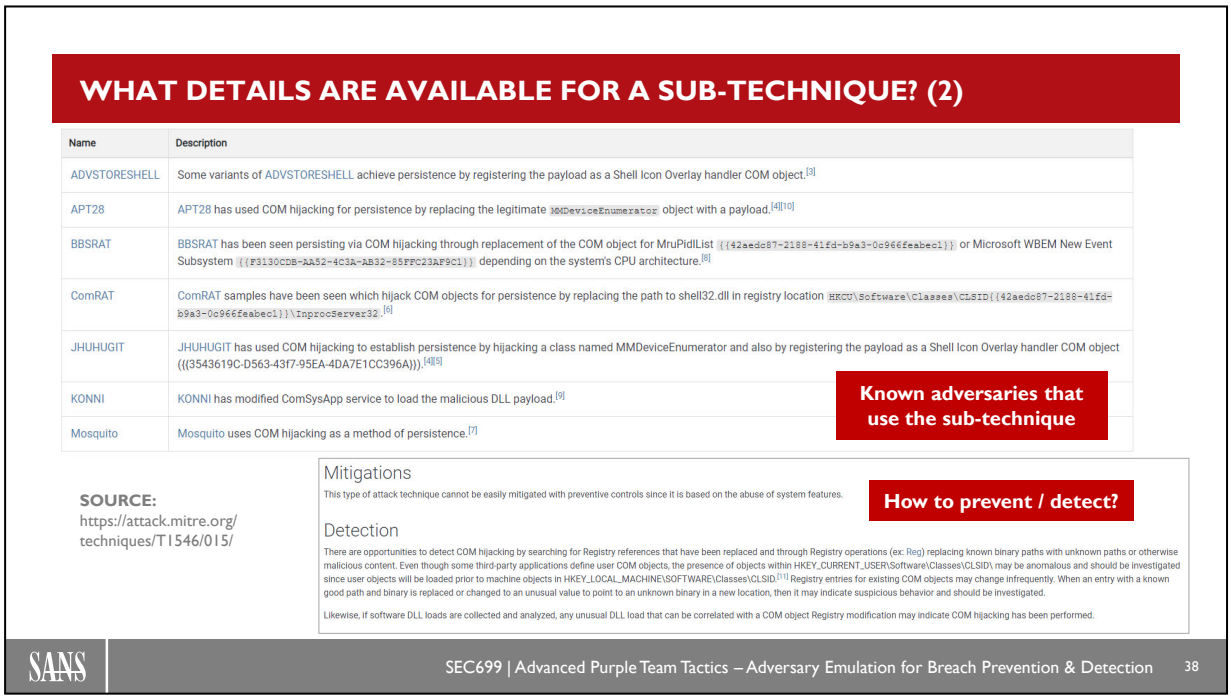


What Details are Available for a Sub-technique? (1)
Let's take a next step and go one abstraction level deeper… We will now have a look at one of the sub-techniques available in MITRE ATT&CK.

COM (Component Object Model) Hijacking is a fan-favorite technique that is often used both by red teamers and real adversaries. It's a highly interesting stealth persistence mechanism that we will further zoom in on during the remainder of the course.

As with the more generic technique, we can identify some basic information about this sub-technique:

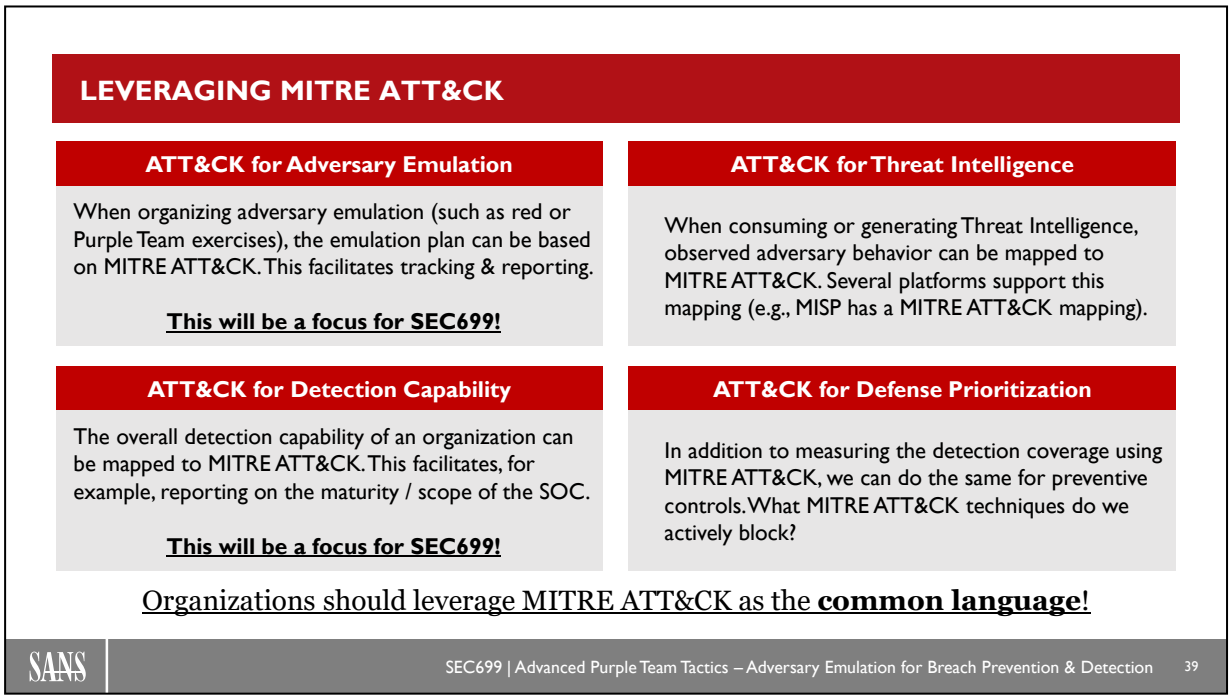
- A high-level description of the technique, explaining how it works and why an adversary would use it
- What adversaries are known to abuse the technique (threat intelligence) – see next slide
- Some general information about the technique:
  - The identifier
  - The tactics it can be found under
  - The platform it is relevant for (in this case, for example, we can see that COM hijacking is only relevant for Windows)
  - The permissions that are required (in this case, we only need normal user privileges)
  - Data sources for detection (we will discuss this further later in class)
- Contributor and version information



What Details are Available for a Sub-technique? (2)
As previously described, the sub-technique page includes additional details on when the sub-technique was used in real-life and by what adversaries.

This is hugely interesting information that can help us build realistic attack scenarios, as it helps us understand whether or not a certain adversary (that targets us) uses the technique.

The sub-technique, of course, also includes information on how to prevent or detect the sub-technique being used in your IT environment.



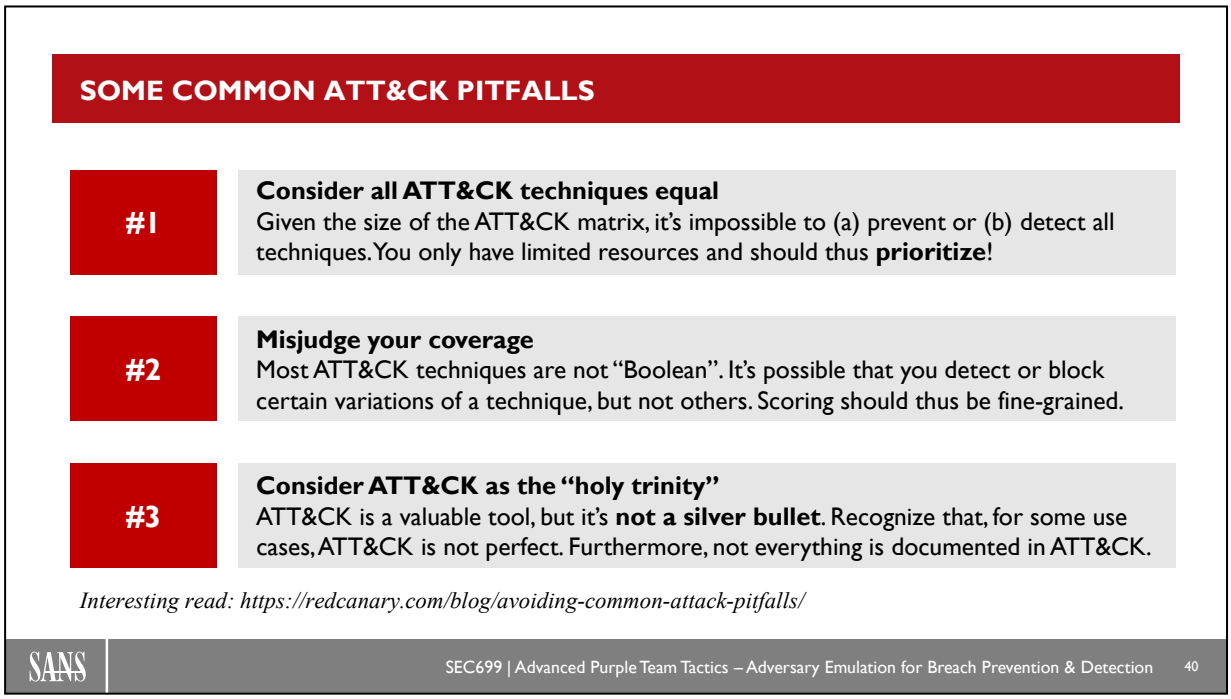
Leveraging MITRE ATT&CK
I like to introduce MITRE ATT&CK as the "common language" organizations should speak. The idea behind this statement is that all security functions (security monitoring, security assessments, threat intelligence,…) should all use MITRE ATT&CK as a standard. How can this be achieved? Consider some of the following use cases:

ATT&CK for Adversary Emulation
When organizing adversary emulation (such as red or Purple Team exercises), the emulation plan can be based on MITRE ATT&CK. This facilitates tracking and reporting. This will be a key focus for SEC699.

ATT&CK for Detection Capability
The overall detection capability of an organization can be mapped to MITRE ATT&CK. This facilitates, for example, reporting on the maturity / scope of the SOC.

ATT&CK for Threat Intelligence
When consuming or generating Threat Intelligence, observed adversary behavior can be mapped to MITRE ATT&CK. Several platforms support this mapping (e.g., MISP has a MITRE ATT&CK mapping).

ATT&CK for Defense Prioritization
In addition to measuring the detection coverage using MITRE ATT&CK, we can do the same for preventive controls. What MITRE ATT&CK techniques do we actively block?



Some Common ATT&CK Pitfalls
Although MITRE ATT&CK can be hugely beneficial to the organization, it should not be considered as a silver bullet that will solve all issues.

It can even have a negative impact on your organization if not implemented correctly. Here's a few pitfalls to take into account:

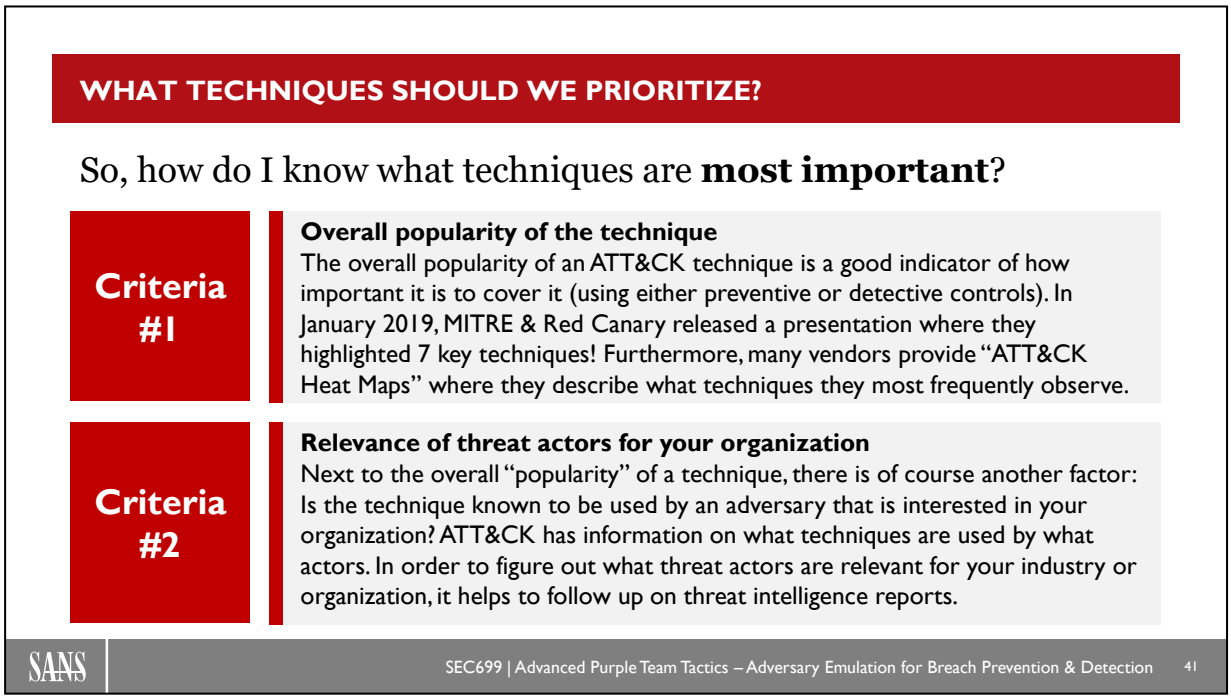
1. Consider all ATT&CK techniques equal
   Given the size of the ATT&CK matrix, it's impossible to (a) prevent or (b) detect all techniques. You only have limited resources and should thus prioritize!

2. Misjudge your coverage
   Most ATT&CK techniques are not "Boolean". It's possible that you detect or block certain variations of a technique, but not others. Scoring should thus be fine-grained.

3. Consider ATT&CK as the "holy trinity"
   ATT&CK is a valuable tool, but it's not a silver bullet. Recognize that, for some use cases, ATT&CK is not perfect. Furthermore, not everything is documented in ATT&CK.

An interesting read is the following blog post published by Red Canary on common pitfalls:

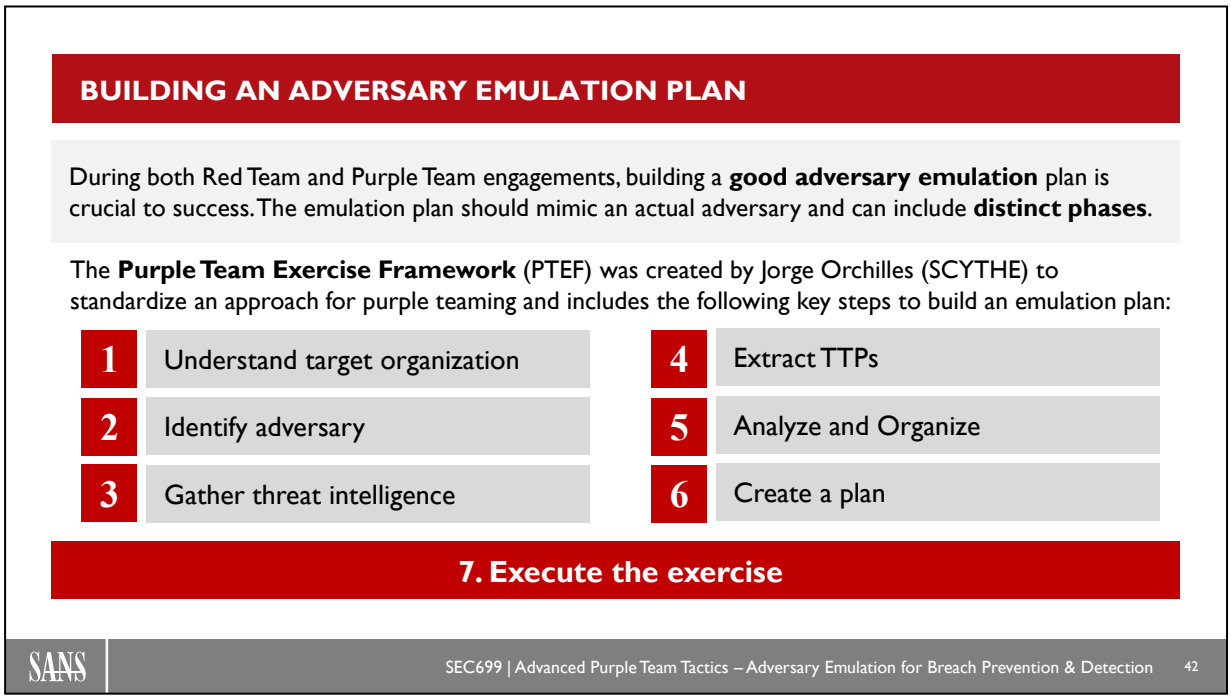
https://redcanary.com/blog/avoiding-common-attack-pitfalls/



What Techniques Should We Prioritize?
As we discussed previously, we cannot consider all techniques equal, and we need to prioritize. After all, all organizations struggle with continuous resource limitations and constraints and thus need to "pick their battles." How do they know what techniques are most important? There's two easy criteria to use:

1. Overall popularity of the technique. The overall popularity of an ATT&CK technique is a good indicator of how important it is to cover it (using either preventive or detective controls). In January 2019, MITRE & Red Canary released a presentation where they highlighted 7 key techniques (see: https://www.readkong.com/page/att-ck-your-cti-with-lessons-learned-from-four-years-in-the-1422466)! Furthermore, many vendors provide "ATT&CK Heat Maps" where they describe what techniques they most frequently observe.

2. Relevance of threat actors for your organization. Next to the overall "popularity" of a technique, there is of course another factor: Is the technique known to be used by an adversary that is interested in your organization? ATT&CK has information on what techniques are used by what actors. In order to figure out what threat actors are relevant for your industry or organization, it helps to follow up on threat intelligence reports.



Building an Adversary Emulation Plan
During both Red Team and Purple Team engagements, building a good adversary emulation plan is crucial to success. The emulation plan should mimic an actual adversary and can include distinct phases.

An interesting initiative is the Purple Team Exercise Framework (PTEF) that was created by Jorge Orchilles (SCYTHE). The goal is to standardize an approach for purple teaming. It includes the following key steps to build an emulation plan:

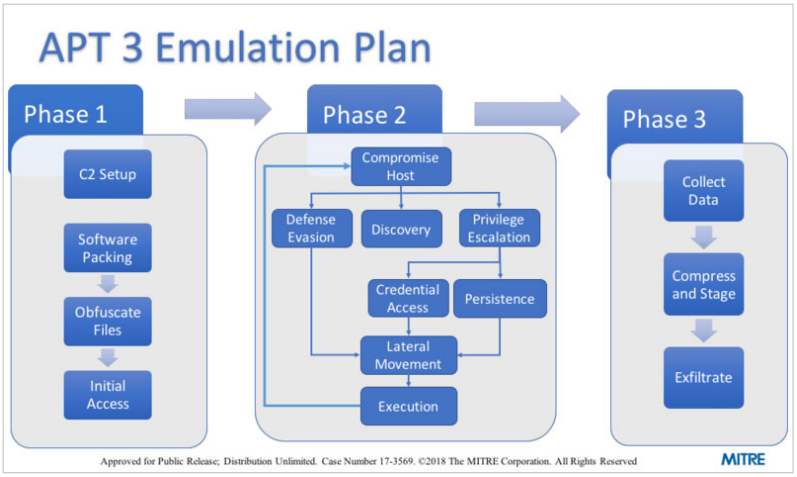
1. Understand the target organization: What is their business? What kind of data do they handle? What are their crown jewels?...
2. Identify typical adversaries that would target this organization
3. Gather threat intelligence on said adversaries
4. From the obtained threat intelligence, extract Tactics, Techniques & Procedures (TTPs)
5. Analyze and organize all obtained data that can be used
6. Create a proper emulation plan

As a final step, you of course execute the exercise!

You can get a free copy of the purple team exercise framework here:

https://www.scythe.io/ptef

ADVERSARY EMULATION – PLANS



To demonstrate the potential of ATT&CK, MITRE developed an emulation plan for APT3 (it's mainly used as a showcase for MITRE ATT&CK).

It's a great example of how the ATT&CK framework can be leveraged to develop a concrete action plan to emulate a specific adversary.

SOURCE: https://attack.mitre.org/resources/adversary-emulation-plans/

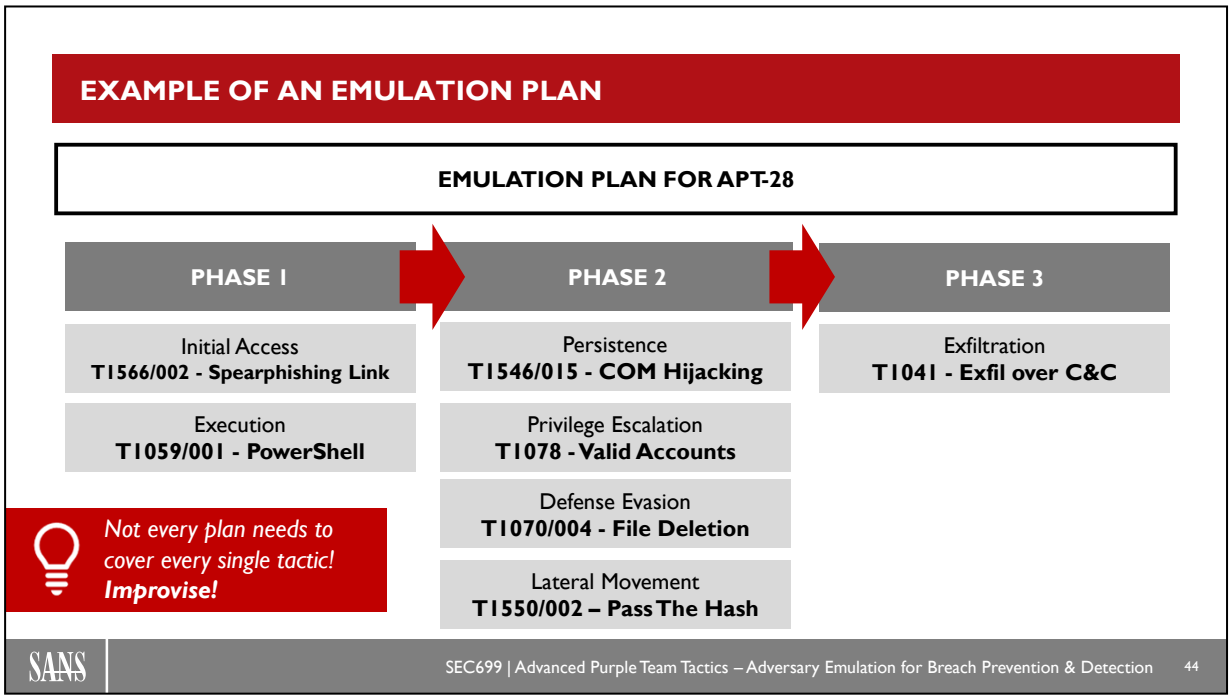
Adversary Emulation – Plans

Let's look at the ATT&CK use cases that are mostly relevant for SEC699. The first one is the definition of adversary emulation plans. Whenever you're embarking on an adversary emulation engagement (either a red or Purple Team), it makes sense to build the plan using MITRE ATT&CK. You could, for example, define three activity phases:

- Phase 1: Obtain initial access
- Phase 2: Perform lateral movement to obtain access to crown jewels / flags defined
- Phase 3: Obtain and exfiltrate data

Each of these phases will leverage different ATT&CK techniques and tactics.

To demonstrate the potential of ATT&CK, MITRE developed a full emulation plan for APT3 (it's mainly used as a showcase for MITRE ATT&CK). It's a great example of how the ATT&CK framework can be leveraged to develop a concrete action plan to emulate a specific adversary.



Example of an Emulation Plan
Let's create an example emulation plan for APT-28, in the phases we described previously.

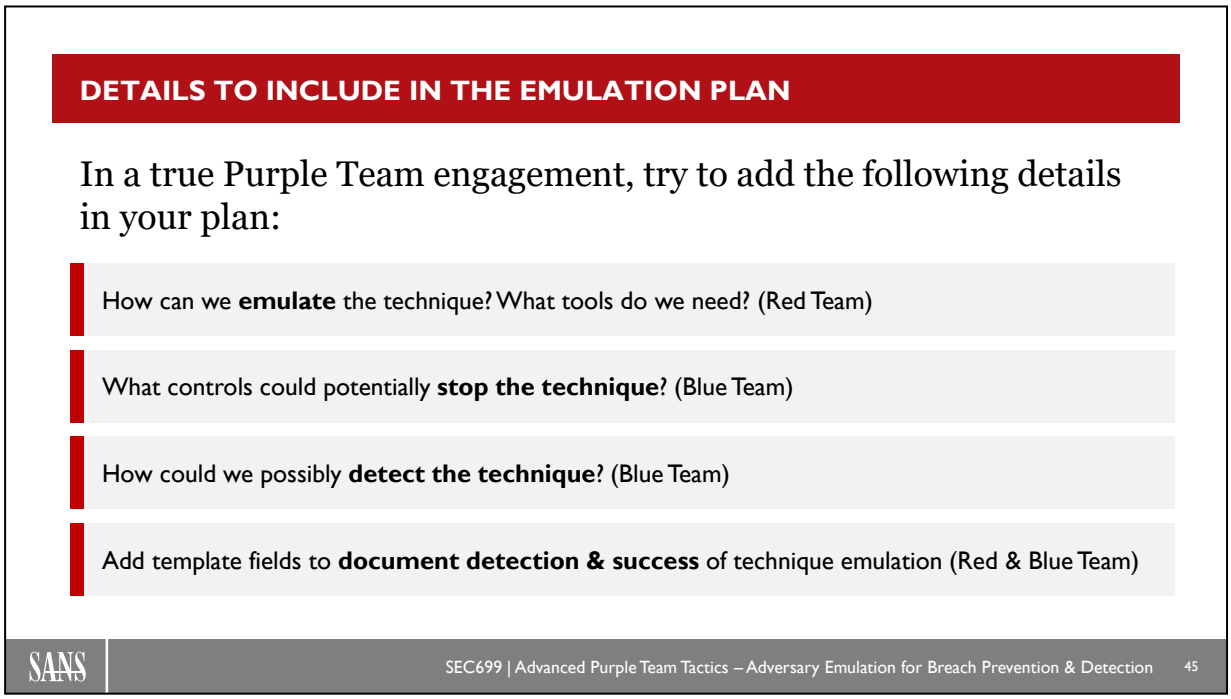
In order to obtain initial access in phase 1, we will use T1566/002 (Spearphishing link) and T1059/001 (PowerShell).

Once we have our initial execution, we will continue by using the following tactic and techniques:

- Persistence: T1546/015 – COM Hijacking
- Privilege Escalation: T1078 – Valid Accounts
- Defense Evasion: T1070/004 – File Deletion
- Lateral Movement: T1550/002 – Pass The Hash

Finally, we will exfiltrate compromised data using T1041 (Exfil over C&C).

Note that we are not using all of the available tactics or techniques. As previously discussed, we prioritized based upon what techniques are mostly relevant for APT-28 and our own organization.

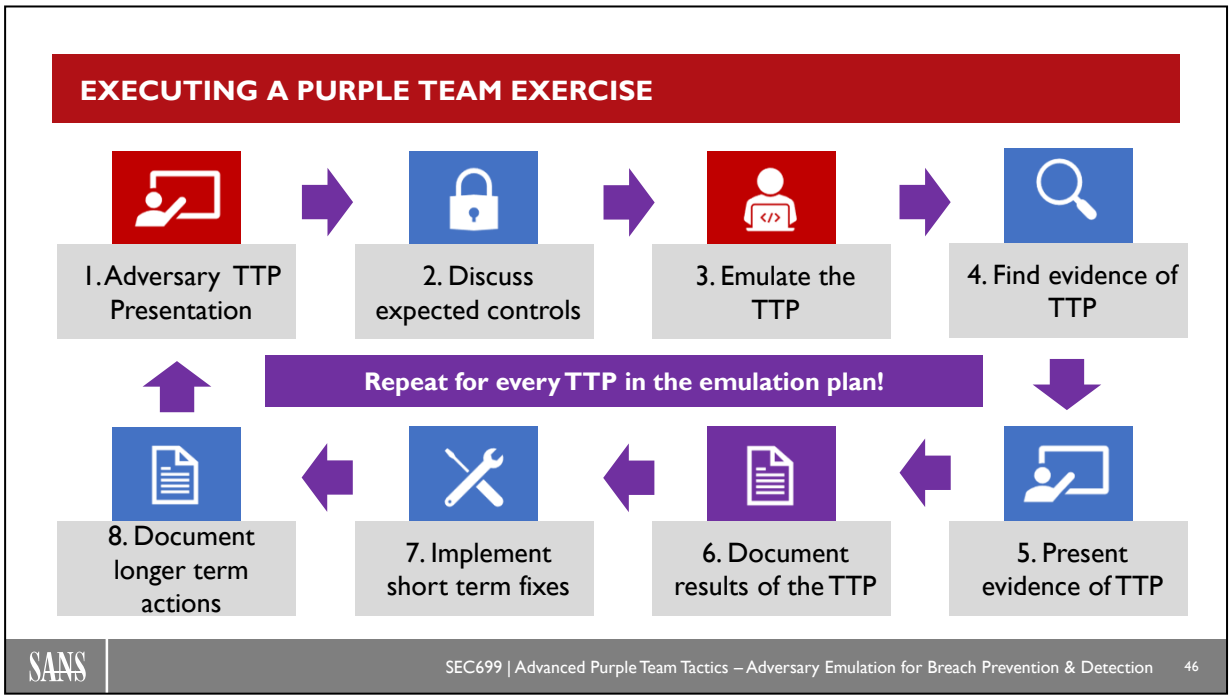


Details to Include in the Emulation Plan
As we are building an emulation plan for Purple Teaming, we need to cover both red and Blue Team information.

Here are some ideas to cover (this is not necessarily an exhaustive list):

- How can we emulate the technique? What tools do we need? (Red Team)
- What controls could potentially stop the technique? (Blue Team)
- How could we possibly detect the technique? (Blue Team)
- Add template fields to document detection & success of technique emulation (Red & Blue Team)

The more information we can add here, the more value the Purple Team could possibly have!



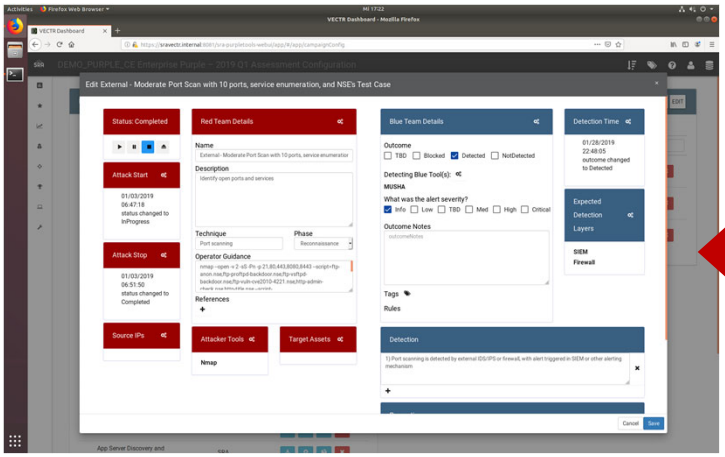
Executing a Purple Team Exercise
So how do we properly execute a Purple Team exercise? The previously referenced Purple Team Exercise Framework provides an interesting process.

First of all, we get red and blue together. Once they are ready, we run through the following steps:

1. Present the adversary and the specific TTP that will be emulated (this is typically done by the red team)
2. Discuss the expected controls that are in place to prevent success execution or detect the TTP (this is typically done by the blue team)
3. Emulate the TTP (this is typically done by the red team)
4. Find evidence of (successful) execution of the TTP (this is typically done by the blue team)
5. Present evidence of (successful) execution of the TTP (this is typically done by the blue team)
6. Document results of the TTP (joint effort between blue and red team)
7. Implement short-term fixes that are identified. This could, for example, be a change to an existing detection rule / use case (this is typically done by the blue team)
8. Finally, we document any longer term actions. This could, for example, be a change to configuration (hardening) or a new log source to configure / on-board in the SIEM (this is typically done by the blue team)

Once a TTP is fully covered throughout all of the above steps, we move on to the next TTP!

INTRODUCING VECTR™: PURPLE TEAM FOLLOW-UP (1)



SECURITYRISKADVISORS/ VECTR

VECTR is a tool developed by Security Risk Advisors that facilitates **tracking** of your red and Blue Team testing activities to measure detection and prevention capabilities across different attack scenarios.

VECTR allows blue and Red Teams to **track both progress and vectors** of performed attacks, effectively reaching the Purple Team's goal through **intel sharing**.

Introducing VECTR™: Purple Team Follow-Up (1)
VECTR is a tool developed by Security Risk Advisors that facilitates tracking of your red and Blue Team testing activities to measure detection and prevention capabilities across different attack scenarios. VECTR allows blue and Red Teams to track both progress and vectors of performed attacks, effectively reaching the Purple Team's goal through intel sharing.

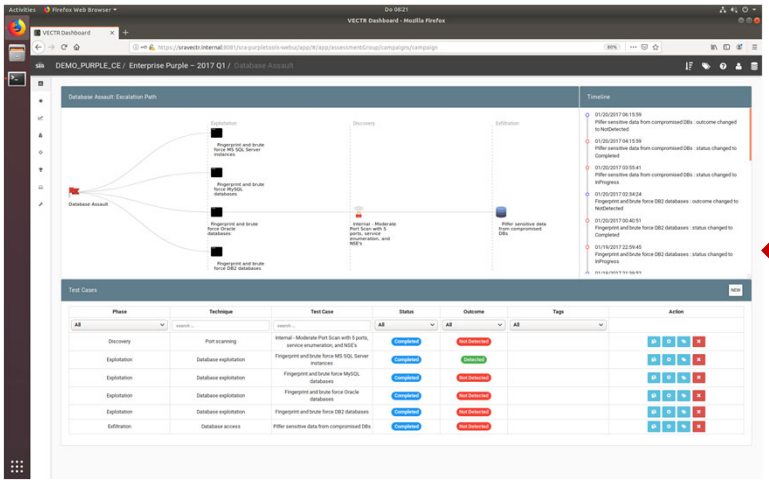
For acceptable use, Security Risk Advisors provides the following information:

"Red and Blue Teams are welcome to use the VECTR™ application for all educational, non-commercial purposes to track performance and develop detection capabilities. This community product may not be re-sold. All published or publicized work product must be attributed to VECTR™ by Security Risk Advisors."

VECTR can be found at https://vectr.io/.

The next few slides do not include extensive notes or comments, but serve as a quick overview of useful VECTR features.

INTRODUCING VECTR™: PURPLE TEAM FOLLOW-UP (2)



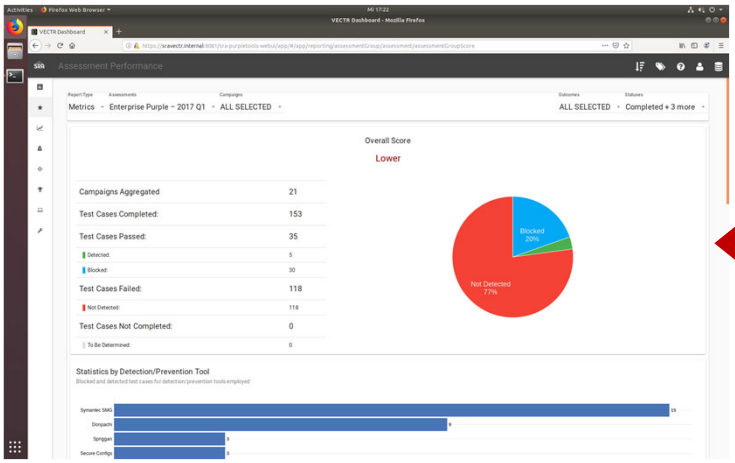
SECURITYRISKADVISORS/
VECTR

Each performed attack is carefully tracked to provide **detailed escalation paths**, **timelines,** and **test cases** in a nice and easy web interface.

The **status** and **outcome** of the different test cases is also described.

Introducing VECTR™: Purple Team Follow-Up (2)
In the above screenshot, we see a general overview of a Purple Team engagement in VECTR. Each performed attack is carefully tracked to provide detailed escalation paths, timelines, and test cases in a nice and easy web interface. The status and outcome of the different test cases is also described.

INTRODUCING VECTR™: PURPLE TEAM FOLLOW-UP (3)



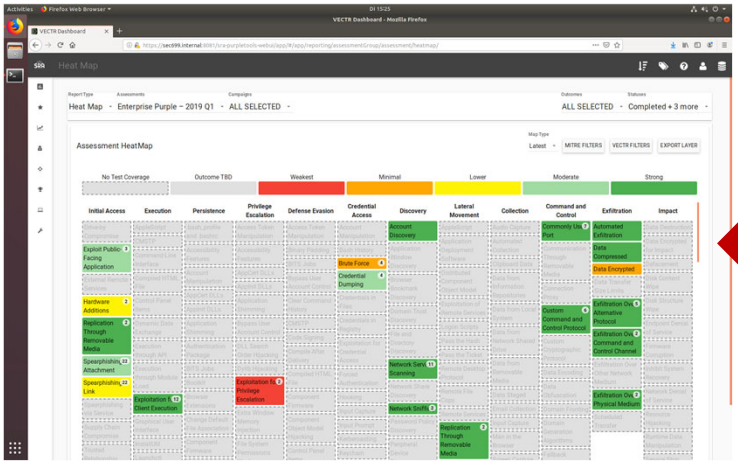
SECURITYRISKADVISORS/
VECTR

Automatically generated **reports** are available for single or multiple campaigns providing both **numerical and graphical statistics**.

This allows for overall Purple Team tracking and progress follow-up (are we actually improving?).

Introducing VECTR™: Purple Team Follow-Up (3)
VECTR has a built-in reporting engine that can be used to report on Purple Team progress. Automatically generated reports are available for single or multiple campaigns providing both numerical and graphical statistics. This allows teams to respond to management questions regarding the “added value” of Purple Team engagements and demonstrating measurable improvements.

INTRODUCING VECTR™: PURPLE TEAM FOLLOW-UP (4)

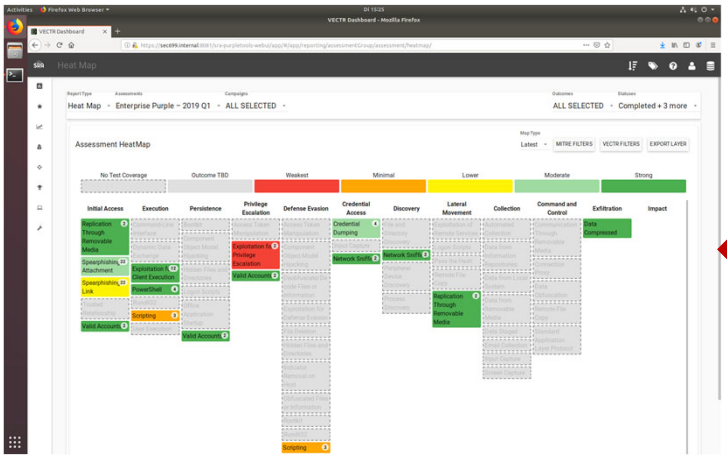


SECURITYRISKADVISORS/
VECTR

VECTR has built-in **MITRE ATT&CK** support. In the screenshot to the left, we can see that reports can be used to generate a MITRE ATT&CK heatmap to highlight strengths and weaknesses.

Introducing VECTR™: Purple Team Follow-Up (4)
VECTR has built-in MITRE ATT&CK support. In the screenshot to the left, we can see that reports can be used to generate a MITRE ATT&CK heatmap to further highlight strengths and weaknesses.

INTRODUCING VECTR™: PURPLE TEAM FOLLOW-UP (5)

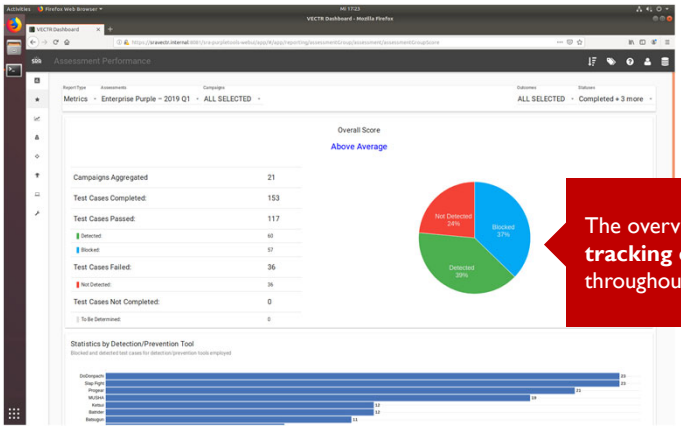


SECURITYRISKADVISORS/
VECTR

VECTR can **cross-reference** the generated heatmap with known actors such as APT28 to further highlight possible risks or areas of focus.

Introducing VECTR™: Purple Team Follow-Up (5)
The VECTR-generated heatmap can be further adapted to elect / focus on a series of controls. In the example on the slideshow, the generated heatmap is focused on techniques used by a known actor such as APT-28. This will allow further prioritization of interesting areas / techniques of focus.

INTRODUCING VECTR™: PURPLE TEAM FOLLOW-UP (6)

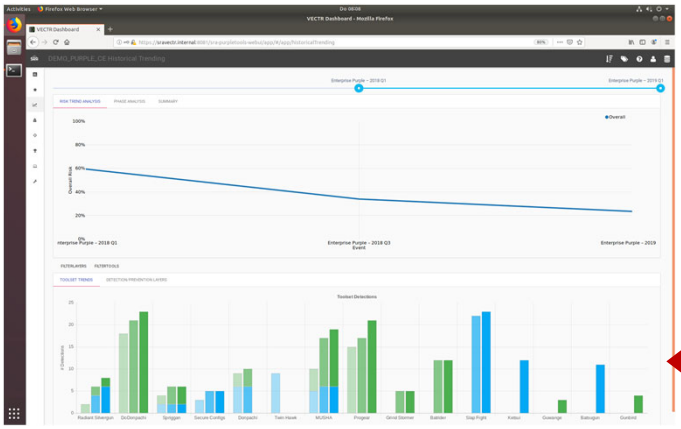


SECURITYRISKADVISORS/
VECTR

The overview provided by the reports eases the **tracking of the test cases' evolution** throughout assessments and campaigns.

Introducing VECTR™: Purple Team Follow-Up (6)
The overview provided by the reports eases the tracking of the test cases' evolution throughout assessments and campaigns.

INTRODUCING VECTR™: PURPLE TEAM FOLLOW-UP (7)



SECURITYRISKADVISORS/
VECTR

Historical data provides a **risk and tool-based overview of an enterprise's evolution**.

Introducing VECTR™: Purple Team Follow-Up (7)
Historical data provides a risk and tool-based overview of an enterprise's evolution.

Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

EXERCISE: INTRODUCTION TO VECTR™

Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

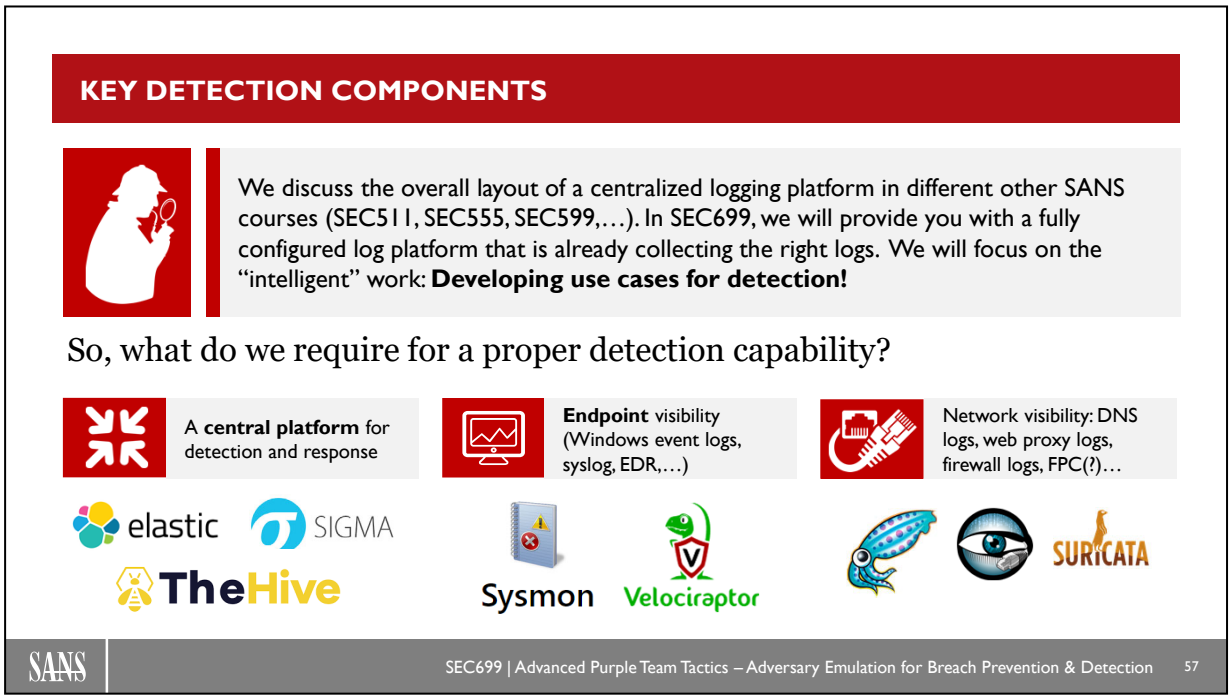
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.



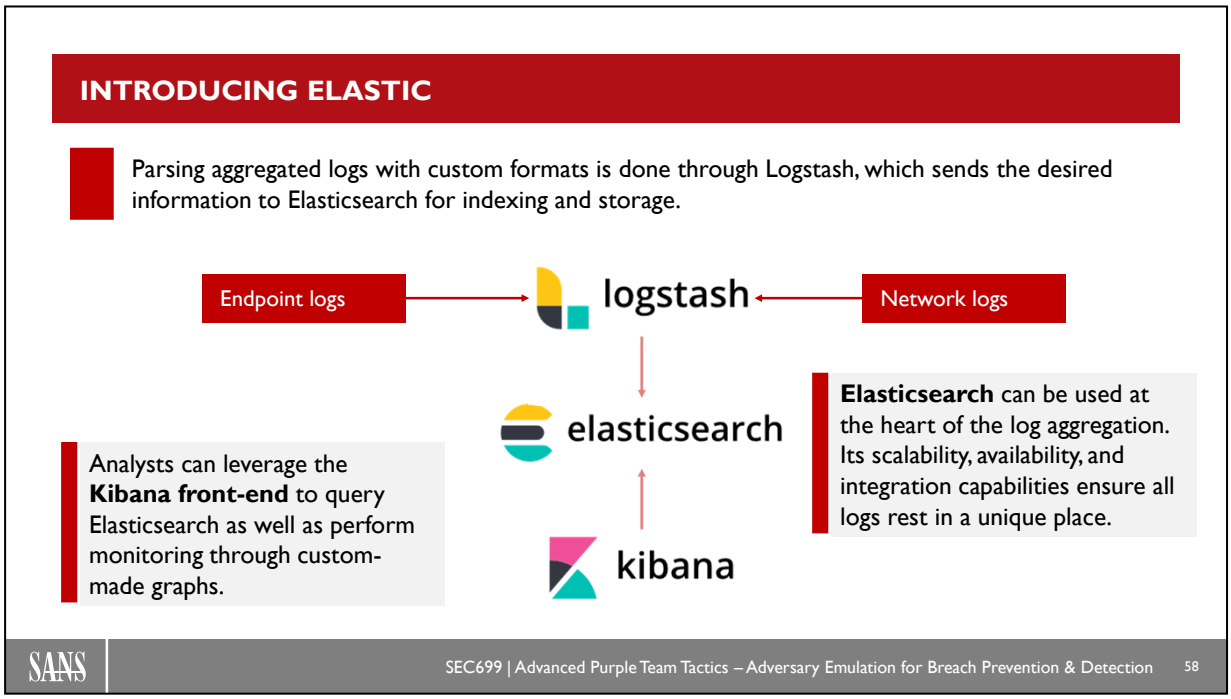
Key Detection Components

We discuss the overall layout of a centralized logging platform in different other SANS courses (SEC511, SEC555, SEC599,…). In SEC699, we will provide you with a fully configured log platform that is already collecting the right logs. We will focus on the "intelligent" work: Developing use cases for detection!

First of all, what components do we require to successfully detect adversaries in our environment:

- A central platform for detection and response. We will use the Elastic stack as a basis, with SIGMA as a use case language and TheHive as a SOAR platform.
- Endpoint visibility (Windows event logs, syslog, EDR,…). We will leverage Windows event logs and Sysmon. Furthermore, we will deploy Velociraptor for in-depth querying and response.
- Network visibility: DNS logs, web proxy logs, firewall logs, FPC (Full Packet Capture),…

Let's further look at these components!



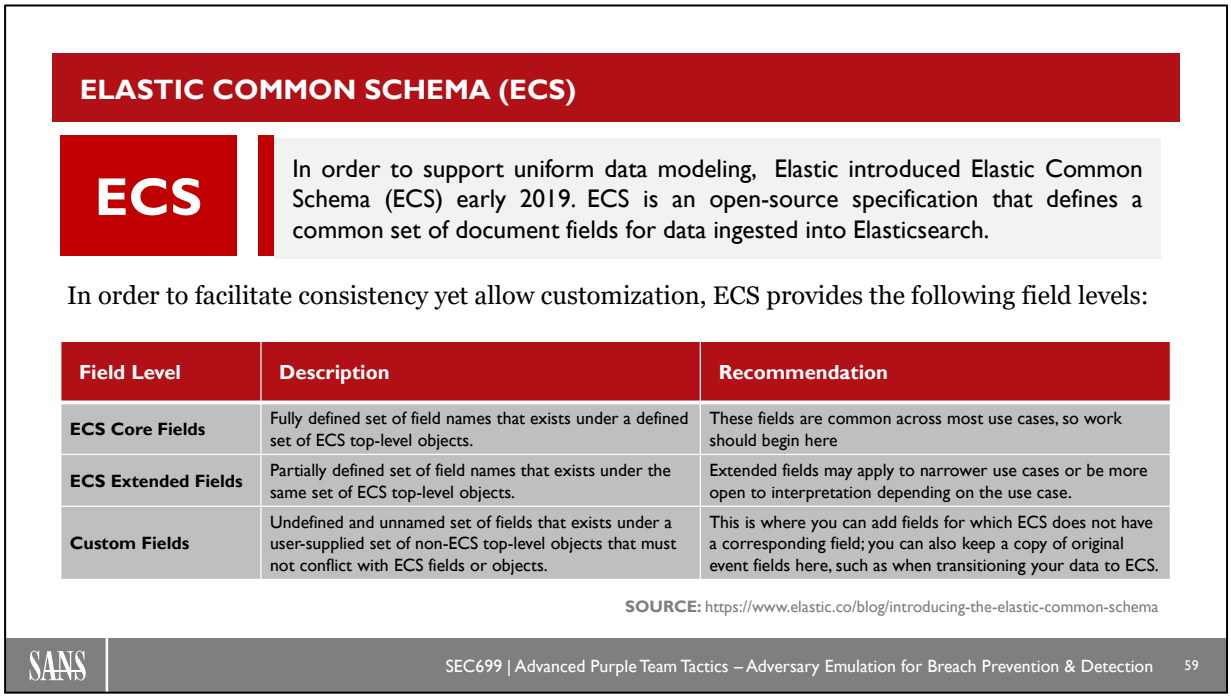
Introducing Elastic

At the center of our detection stack is Elastic. The Elastic stack (formerly "ELK") consists of three components working together, namely Elasticsearch, Logstash, and Kibana.

Elasticsearch is the big data solution and is used to store, index, and query the large volumes of data. Its functionality is similar to Splunk. However, some of the underlying technologies used are different. Elasticsearch makes use of Apache Lucene for information retrieval, originally completely written in Java, but meanwhile ported to C++ and Python, among others.

Logstash is used for parsing logs submitted to the stack and stores the results in Elasticsearch. Logstash uses Grok to transform text patterns into a meaningful structure. Grok is perfect for syslog logs, Apache, and other web server logs, mysql logs, and in general, any log format that is written for humans and not computer consumption.

Kibana takes care of the graphical component of the stack and visualizes data that it queries from Elasticsearch. Kibana can be used to implement custom dashboards, which heavily relies on JSON. Kibana has all the classics such as histograms, line graphs, and pie charts. It's also able to create geo maps, time series, and analyze relationships or anomalies using machine learning.



Elastic Common Schema (ECS)

The explosive rise of Elastic as a tool for log centralization and analysis led to a wide variety of different projects leveraging the stack. In order to increase transferability and collaboration, however, some uniformity is needed. Splunk implemented this years ago as the Splunk Common Information Model (CIM).

Elastic introduced Elastic Common Schema (ECS) early 2019. ECS is an open-source specification that defines a common set of document fields for data ingested into Elasticsearch. From the Elastic website:

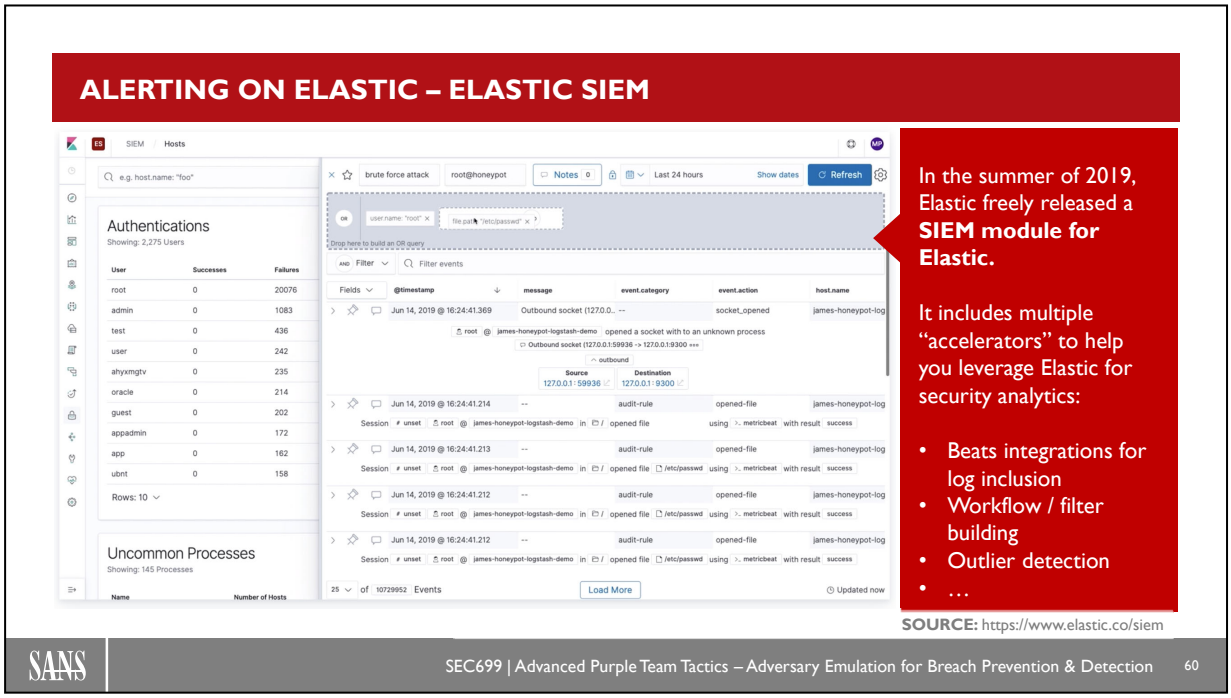
"ECS is an open-source specification that defines a common set of document fields for data ingested into Elasticsearch. ECS is designed to support uniform data modeling, enabling you to centrally analyze data from diverse sources with both interactive and automated techniques."

In order to facilitate consistency yet allow customization, ECS provides three different field levels:

- **ECS Core Fields:** Fully defined set of field names that exists under a defined set of ECS top-level objects. These fields are common across most use cases, so work should begin here.
- **ECS Extended Fields:** Partially defined set of field names that exists under the same set of ECS top-level objects. Extended fields may apply to narrower use cases or be more open to interpretation depending on the use case.
- **Custom Fields**: Undefined and unnamed set of fields that exists under a user-supplied set of non-ECS top-level objects that must not conflict with ECS fields or objects. This is where you can add fields for which ECS does not have a corresponding field; you can also keep a copy of original event fields here, such as when transitioning your data to ECS.

Reference:

https://www.elastic.co/guide/en/ecs/current/index.html



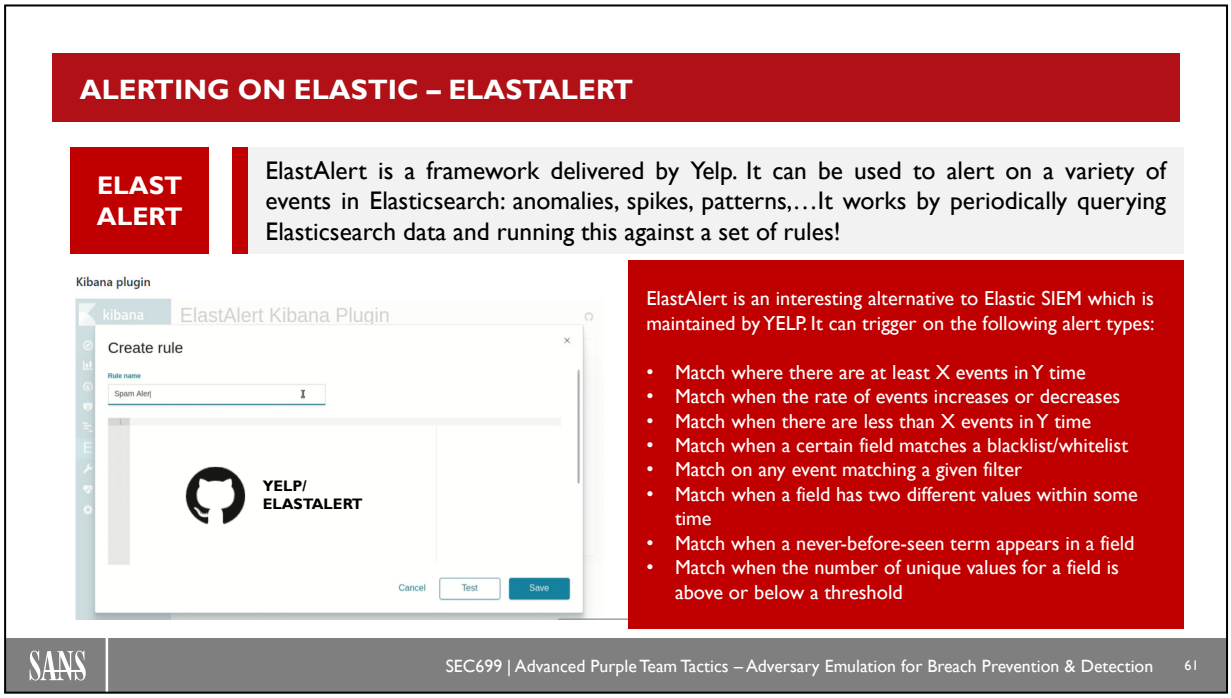
Alerting on Elastic – Elastic SIEM

On top of the standard Elastic components (Elasticsearch, Logstash, and Kibana), Elastic is building additional components for various use cases. For security, one interesting addition was the SIEM module, which was released in the summer of 2019. It includes multiple "accelerators" to help you leverage Elastic for security analytics:

- Beats integrations for log inclusion (network and host data integrations)
- Workflow / filter building
- Outlier detection
- Integration with ticketing and SOAR (Security Orchestration, Automation, and Response) platforms

Elastic SIEM is made freely available, but it's not part of the open-source Elastic license.

For additional information, please refer to https://www.elastic.co/products/siem.



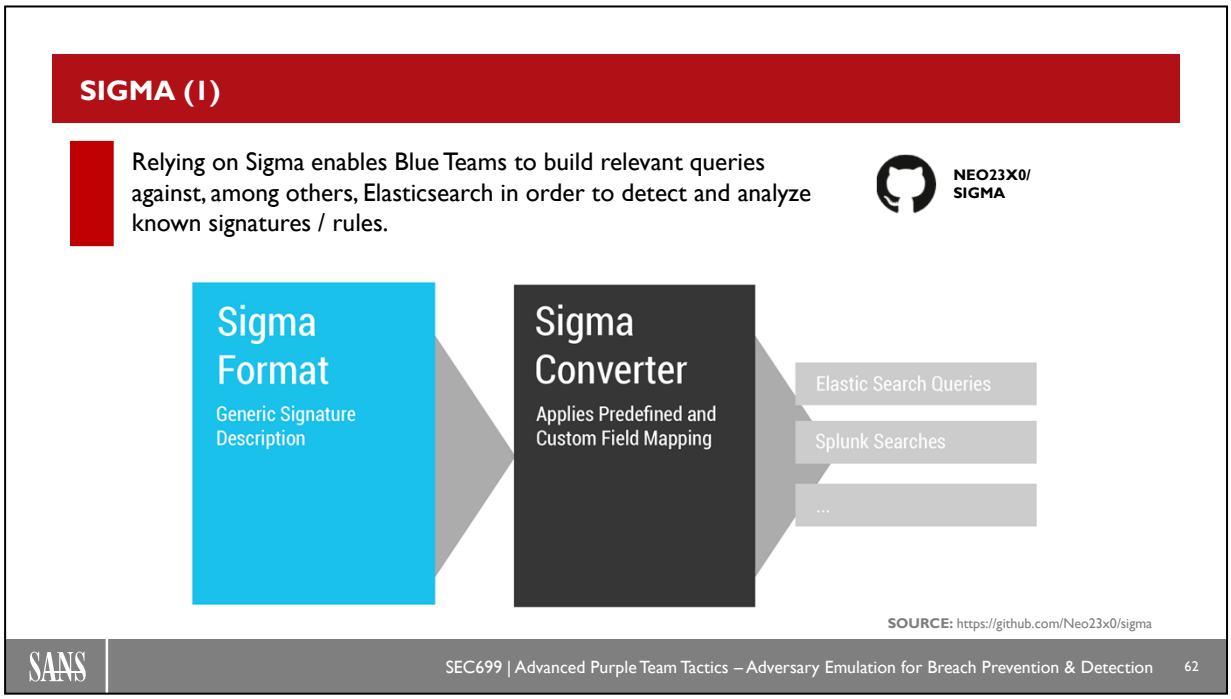
Alerting on Elastic – ElastAlert

ElastAlert is a framework delivered by Yelp. It can be used to alert on a variety of events in Elasticsearch: anomalies, spikes, patterns,…It works by periodically querying Elasticsearch data and running this against a set of rules! Whenever an alert is triggered, there are possibilities to create a variety of alerts. On the GitHub page, the following alert types are listed:

- Match where there are at least X events in Y time (frequency type)
- Match when the rate of events increases or decreases (spike type)
- Match when there are less than X events in Y time (flatline type)
- Match when a certain field matches a blacklist/whitelist (blacklist and whitelist type)
- Match on any event matching a given filter (any type)
- Match when a field has two different values within some time (change type)
- Match when a never-before-seen term appears in a field (new_term type)
- Match when the number of unique values for a field is above or below a threshold (cardinality type)

The SIGMA tool (sigmac) has built-in support to convert SIGMA rules to ElastAlert rules! Furthermore, ElastAlert has an output format for TheHive (so we can feed alerts immediately in TheHive for handling and further follow-up)! The latest documentation and version of ElastAlert can be found at https://github.com/Yelp/elastalert.

Cyber3rWard0g (Roberto Rodriguez) wrote a blog post on overall Elastalert integration at https://posts.specterops.io/what-the-helk-sigma-integration-via-elastalert-6edf1715b02?gi=fa9cd0b8ce86.



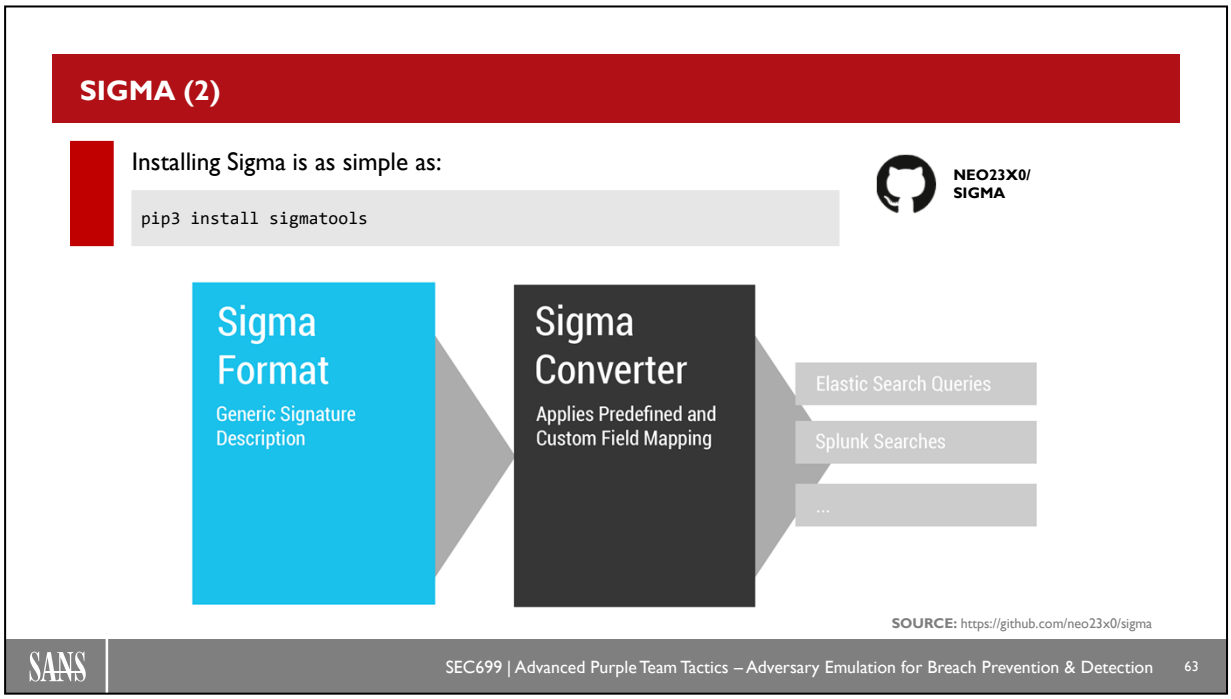
SIGMA (1)

Sigma is a project by Florian Roth that tries to provide a generic, vendor-neutral, rule format that can be used to describe suspicious or malicious behavior. Most SIGMA rules are also mapped to MITRE's ATT&CK framework. As part of the project, several "converters" have been written that allow you to convert the SIGMA rules to certain technologies. Supported technologies include (but are not limited to):

- Splunk
- Elastic
- ElastAlert
- Windows Defender ATP
- ArcSight
- Qradar
- RSA NetWitness
- …

From Florian's GitHub page (https://github.com/Neo23x0/sigma):

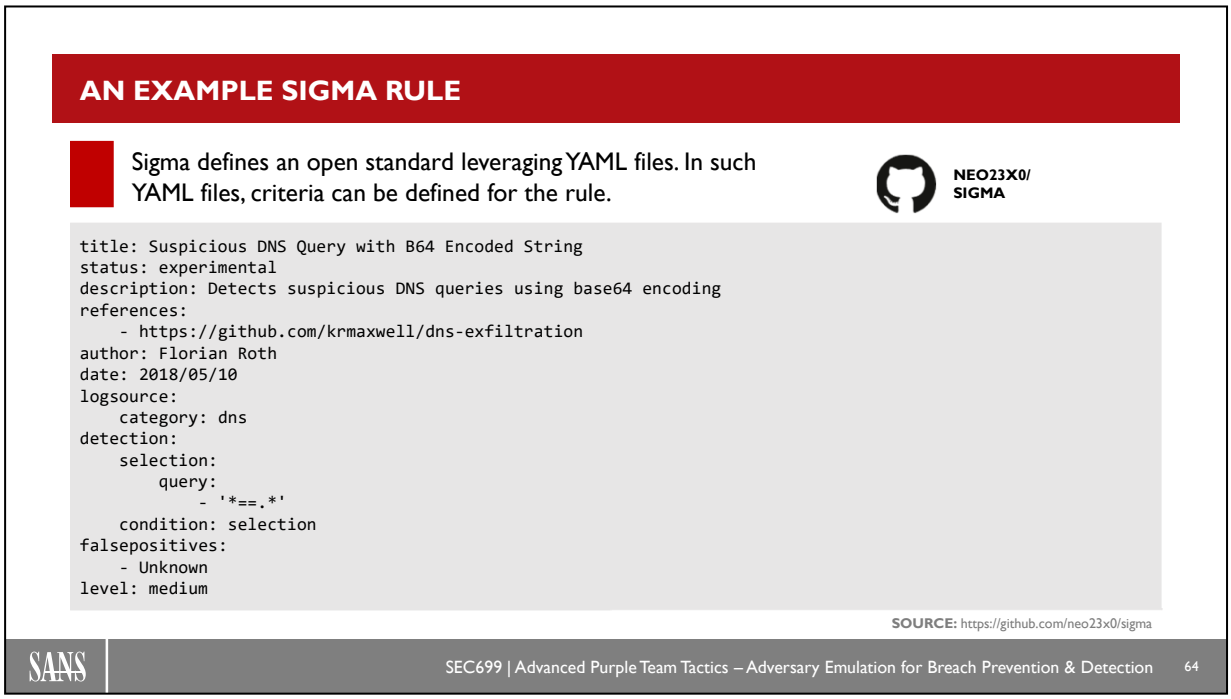
"Sigma is a generic and open signature format that allows you to describe relevant log events in a straight forward manner. The rule format is meant to be flexible, easy to write and applicable to any type of log file. The main purpose of the project is to provide a structured form in which researchers or analysts can describe their once developed detection methods and make them shareable with others."



SIGMA (2)

Installation of SIGMA is very straightforward; it can just be deployed using pip:

pip3 install sigmatools

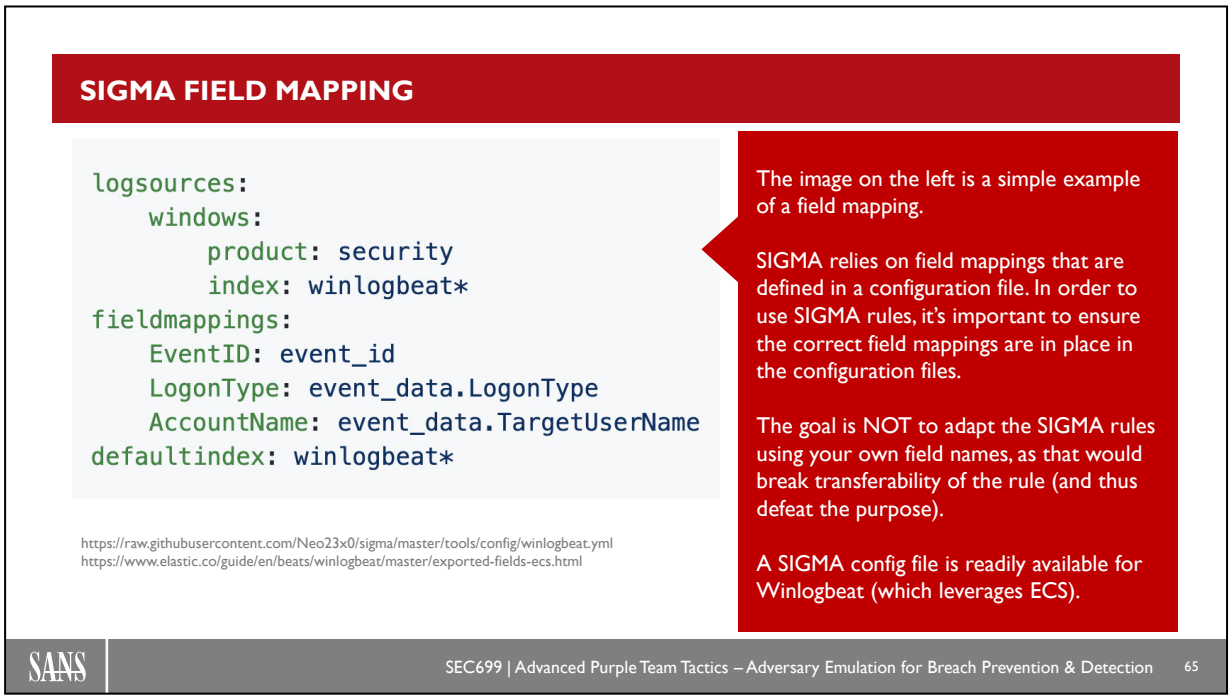


An Example SIGMA Rule

Sigma defines an open standard leveraging YAML files. In such YAML files, criteria can be defined for the rule.

From the example rule on the slide, we can deduce the following information:

- The rule title is “Suspicious DNS Query with B64 Encoded String”
- The status of the rule is “experimental”, so it’s most likely still under development
- The rule description is “Detects suspicious DNS queries using base64 encoding”
- There is a reference to a knowledge article, in this case a GitHub repository about DNS exfiltration (https://github.com/krmaxwell/dns-exfiltration)
- The author of the rule is Florian Roth
- The source logs required for successful detection are DNS logs
- The rule will fire on a DNS query that ends with “==“ (sign of Base64 padding)
- There’s no known false positives
- The confidence level of the rule is medium



SIGMA Field Mapping

The image on the slide is a simple example of a field mapping. SIGMA rules can use a wide variety of different field names (only limited by the imagination of the rule author). In order to make sure rules can be easily shared / transported though, we use generic field names and convert them to local implementations using a field mapping configuration.

In the screenshot on the slide, we can see an example mapping for Windows security logs ingested by Winlogbeat:

- EventID (in SIGMA) is mapped to event_id (in the actual Elastic cluster log)
- LogonType (in SIGMA) is mapped to event_data.LogonType (in the actual Elastic cluster log)
- AccountName (in SIGMA) is mapped to event_data.TargetUserName (in the actual Elastic cluster log)

In order to use SIGMA rules, it's important to ensure the correct field mappings are in place in the configuration files. The goal is NOT to adapt the SIGMA rules using your own field names, as that would break transferability of the rule (and thus defeat the purpose). A SIGMA config file is readily available for Winlogbeat (which leverages ECS).

For more information, please refer to:

https://raw.githubusercontent.com/Neo23x0/sigma/master/tools/config/winlogbeat.yml

https://www.elastic.co/guide/en/beats/winlogbeat/master/exported-fields-ecs.html

CONVERTING SIGMA RULES

We can convert the YAML Sigma rules to queries for a set of known SIEM systems. In the example below, we convert it to a Kibana search filter:

NEO23X0/ SIGMA

```
$ sigmac --target kibana --config config.yml rule.yml
[
  {
    "_id": "Suspicious-DNS-Query-with-B64-Encoded-String",
    "_type": "search",
    "_source": {
      "title": "Sigma: Suspicious DNS Query with B64 Encoded String",
      "description": "Detects suspicious DNS queries using base64 encoding",
      "hits": 0,
      "columns": [],
      "sort": [
        "@timestamp",
        "desc"
      ],
      "version": 1,
      "kibanaSavedObjectMeta": {
        "searchSourceJSON": "{\"index\": \"logstash-*\", \"filter\": [], \"highlight\": {\"pre_tags\": [\"@kibana-highlighted-field@\"],
\"post_tags\": [\"@/kibana-highlighted-field@\"], \"fields\": {\"*\": {}}, \"require_field_match\": false, \"fragment_size\": 2147483647},
\"query\": {\"query_string\": {\"query\": \"query.keyword:(*\\\\\\=\\\\\\=.*)\", \"analyze_wildcard\": true}}}"
      }
    }
  }
]
```

SOURCE: https://github.com/neo23x0/sigma

Converting SIGMA Rules

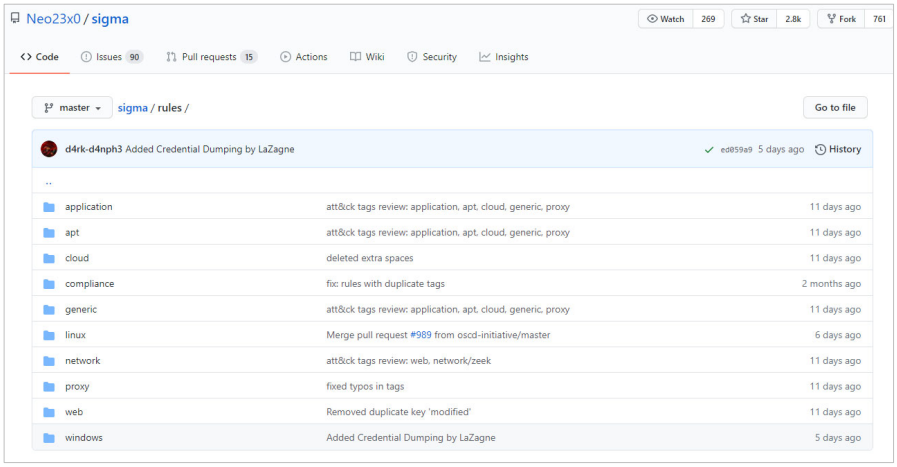
As a next step, we need to convert the YAML format to actionable searches / queries / filters / … we can use in one of the technology stacks supported by SIGMA. The above shows our translated SIGMA rule as a Kibana search filter (which is a JSON format).

We can now just copy-paste the search in a Kibana stack and assess our results! Note that this is a Kibana filter; if we are using an Elastic stack, we also have other options to implement this SIGMA rule:

- Using ElastAlert to generate automated alerts that can be fed into ticketing / communication platforms such as Teams, Slack,…
- Using raw Elastic queries

As always, it depends on your exact use case and what you are hoping to achieve.

SOURCES FOR SIGMA RULES – FLORIAN ROTH'S REPOSITORY



SOURCE: https://github.com/neo23x0/sigma/tree/master/rules

Florian Roth hosts a free and open-source SIGMA rule repository on GitHub.

The rules are split in different categories (as can be seen on the screenshot on the slide).

The repository is a moving target, as rules are periodically added, updated, merged, or even removed!

Sources for SIGMA Rules – Florian Roth's Repository

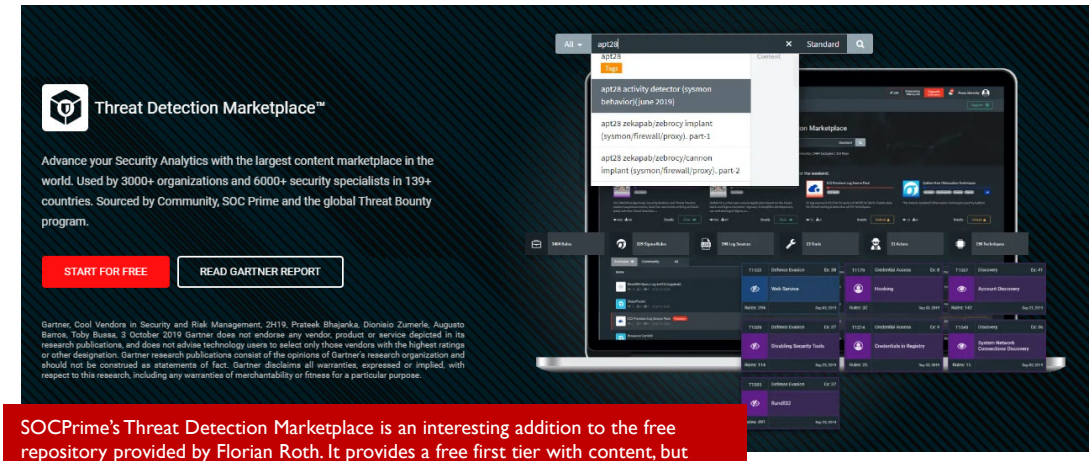
Florian Roth hosts a free and open-source SIGMA rule repository on GitHub. The rules are split into different categories (as can be seen in the screenshot on the slide).

All SIGMA rules can be easily pulled from here and can afterwards be converted to the SIEM stack of your choice.

The repository is a moving target, as rules are periodically added, updated, merged, or even removed! When this repository is used as a basis for production rulesets, it's recommended to have a forked version which is under your own control.

The repository can be found at https://github.com/neo23x0/sigma/tree/master/rules.

SOURCES FOR SIGMA RULES – SOCPRIME TDM



SOCPrime's Threat Detection Marketplace is an interesting addition to the free repository provided by Florian Roth. It provides a free first tier with content, but also provides, at a fee, premium content and features.

SOURCE: https://my.socprime.com/tdm/

Sources for SIGMA Rules – SOCPrime TDM

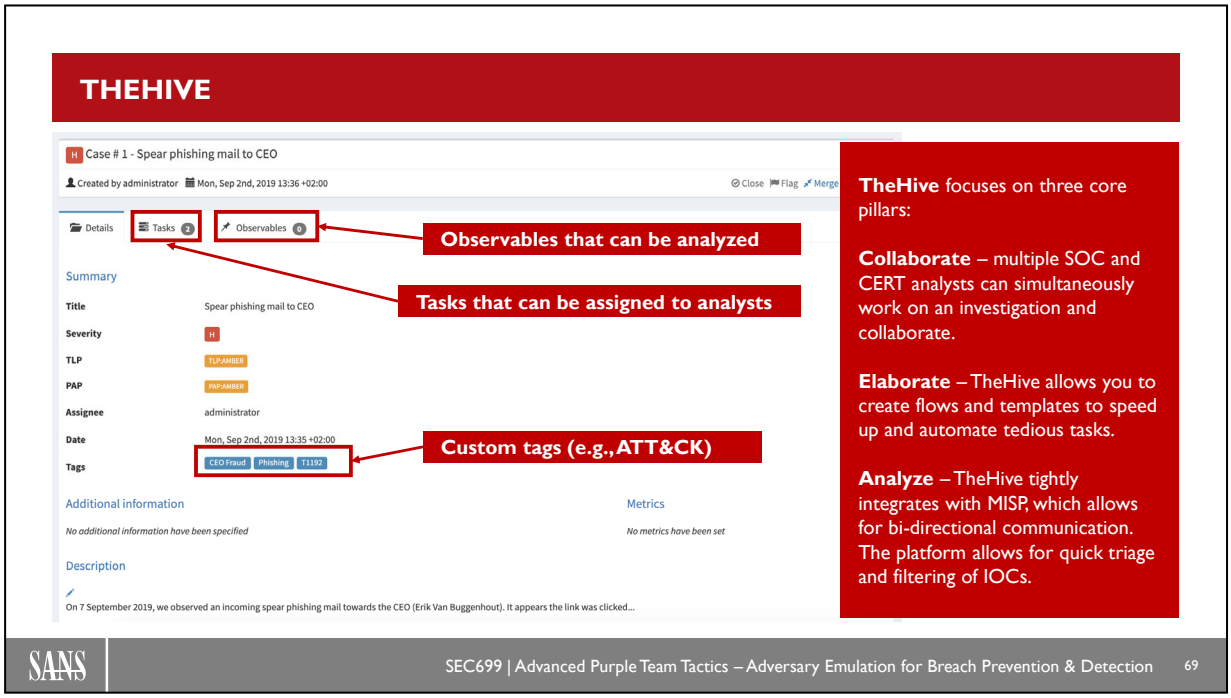
SOCPrime is a vendor that aims to help organizations leverage security tooling / products they already have in place. From their official website:

"Improve what you have, not Replace. SOC Prime helps to centrally source and support content to maximize the value of existing security investments. We have established and continue evolving the first in the world platform agnostic Threat Detection Marketplace. As of September 2019 Threat Detection Marketplace connects 6000+ users, 3000+ organizations from 139 countries with 83 Threat Bounty members and security researchers. Platform contains SOC ready dashboards, rule packages, Machine Learning recipes for the Elastic stack and Sigma rules updated daily and streamed via API. This accounts for over tens of thousands of content items mapped directly to MITRE ATT&CK methodology providing the largest in the world content repository, updated continuously."

The Threat Detection Marketplace is an interesting addition to the free repository provided by Florian Roth. It provides a free first tier with content, but also provides, at a fee, premium content and features. Some of the features they provide include:

- Sigma rules with ATT&CK tags
- Kibana dashboard configs
- Machine Learning Recipes
- Alerts for X-Pack Watchers
- Logstash configuration files
- SaaS and IaaS API integration

More information can be found at https://my.socprime.com/tdm/.



TheHive

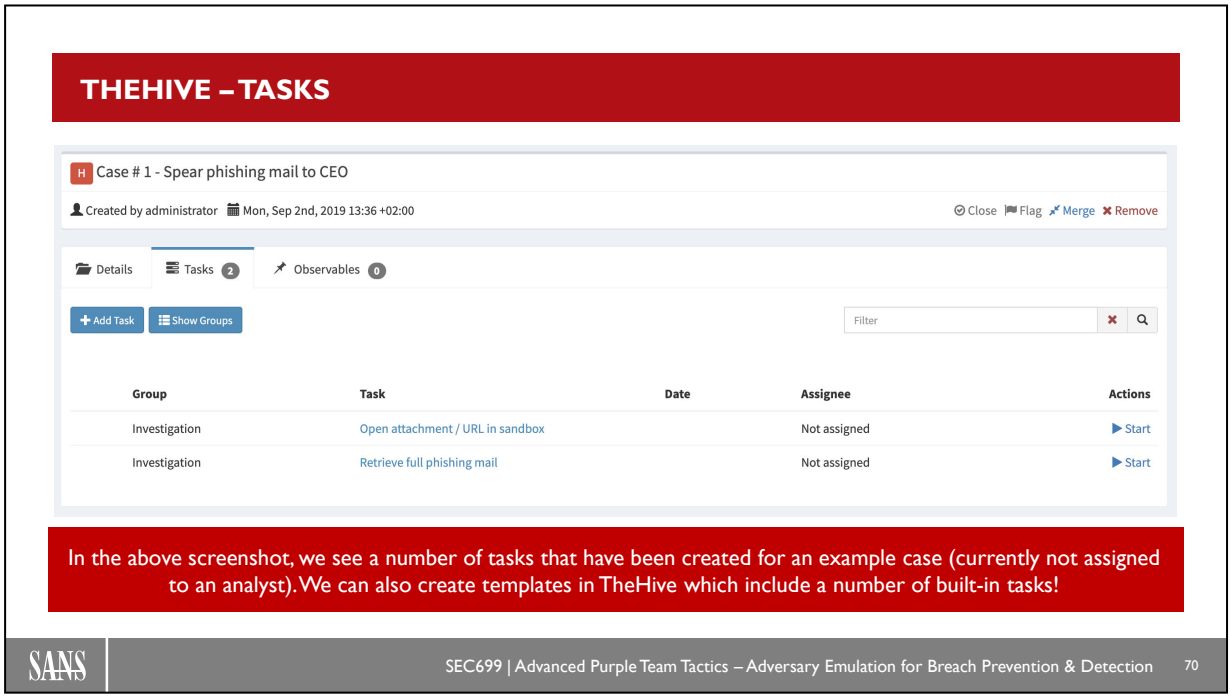
TheHive (by CERT-BDF) is an open-source incident response framework that focuses on three core pillars:

- Collaborate – Multiple SOC and CERT analysts can simultaneously work on an investigation and collaborate through the platform.
- Elaborate – TheHive allows you to create flows and templates to speed up and automate tedious tasks.
- Analyze – TheHive tightly integrates with MISP, which allows for bi-directional communication. The platform allows for quick triage and filtering of IOCs.

An important part of TheHive to highlight is the “Cortex” plugin.

“Cortex tries to solve a common problem frequently encountered by SOCs, CSIRTs and security researchers in the course of threat intelligence, digital forensics and incident response: how to analyze observables they have collected, at scale, by querying a single tool instead of several?”

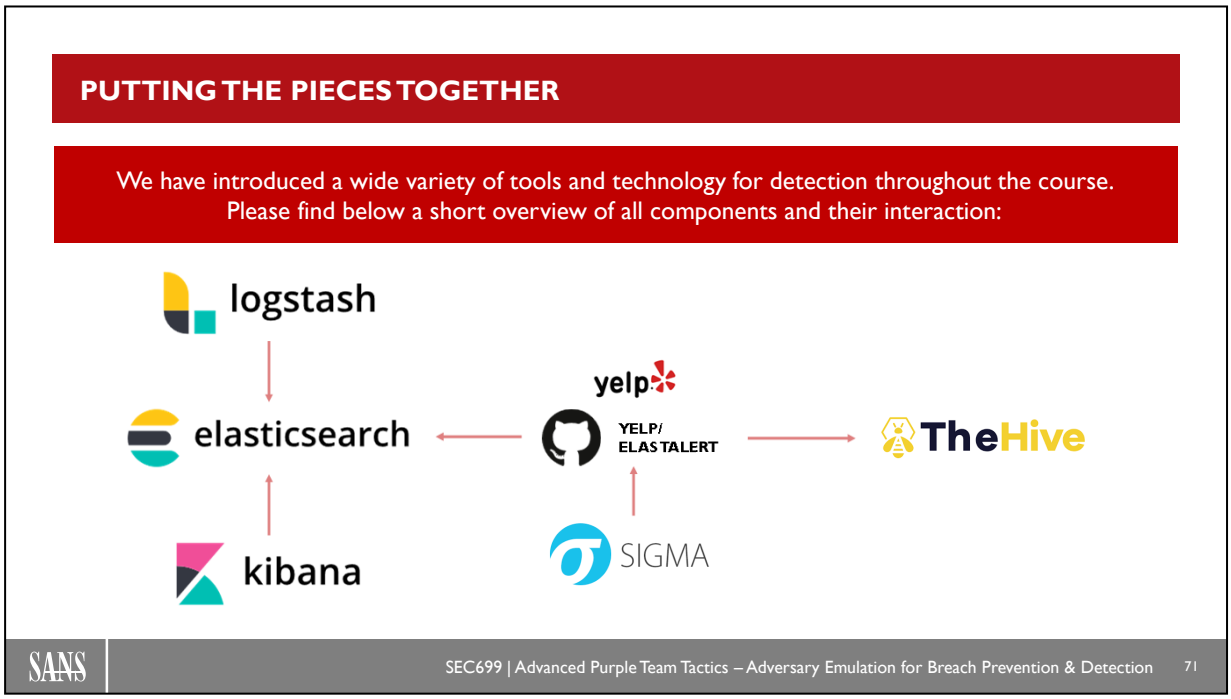
TheHive also has further integrations, for example with Cuckoo Sandbox. Additional information can be found at https://github.com/TheHive-Project.



TheHive – Tasks

In the screenshot on the slide, we see a number of tasks that have been created for an example case (currently not assigned to an analyst). We can also create templates in TheHive which include a number of built-in tasks! A number of automated templates have already been created by the community for Cortex. You can find them here:

https://github.com/TheHive-Project/Cortex-Analyzers/tree/master/thehive-templates



Putting the Pieces Together

We have introduced a wide variety of tools and technology for detection throughout the course.

Please find below a short overview of all components and their interaction:

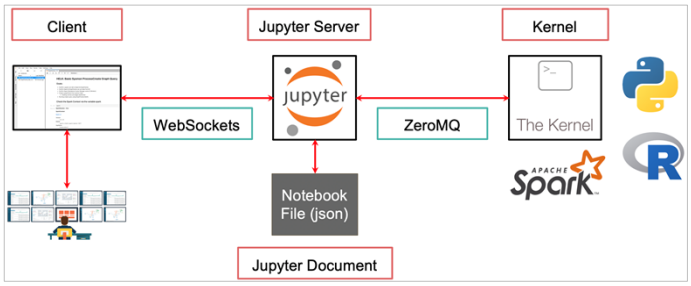
- Logs are ingested by Logstash or directly by Elasticsearch (using event forwarding, beats,…)
- Elasticsearch indexes the data and serves as the central repository
- Kibana can be used by analysts to query and visualize data in Elasticsearch
- SIGMA rules for detection are implemented by analysts and converted into ElastAlert rules
- ElastAlert queries Elasticsearch and runs its rules
- Whenever ElastAlert identifies rule hits, it will generate alerts for further follow-up in TheHive

In such a scenario, the security analysts would mainly interact with:

- TheHive for alert handling and follow-up
- SIGMA for rule addition and customization
- Kibana for further deep-dives and analysis of logs

JUPYTER NOTEBOOKS

A Jupyter Notebook is a document accessible through the web. It allows users to start "sessions" which allow input (e.g., programming code) and output (output of the code that was inputted) to be saved. Furthermore, we can add notes that document how the code works and what additional analysis needs to be done manually on the output.



SOURCE: https://threathunterplaybook.com/tutorials/jupyter/introduction.html

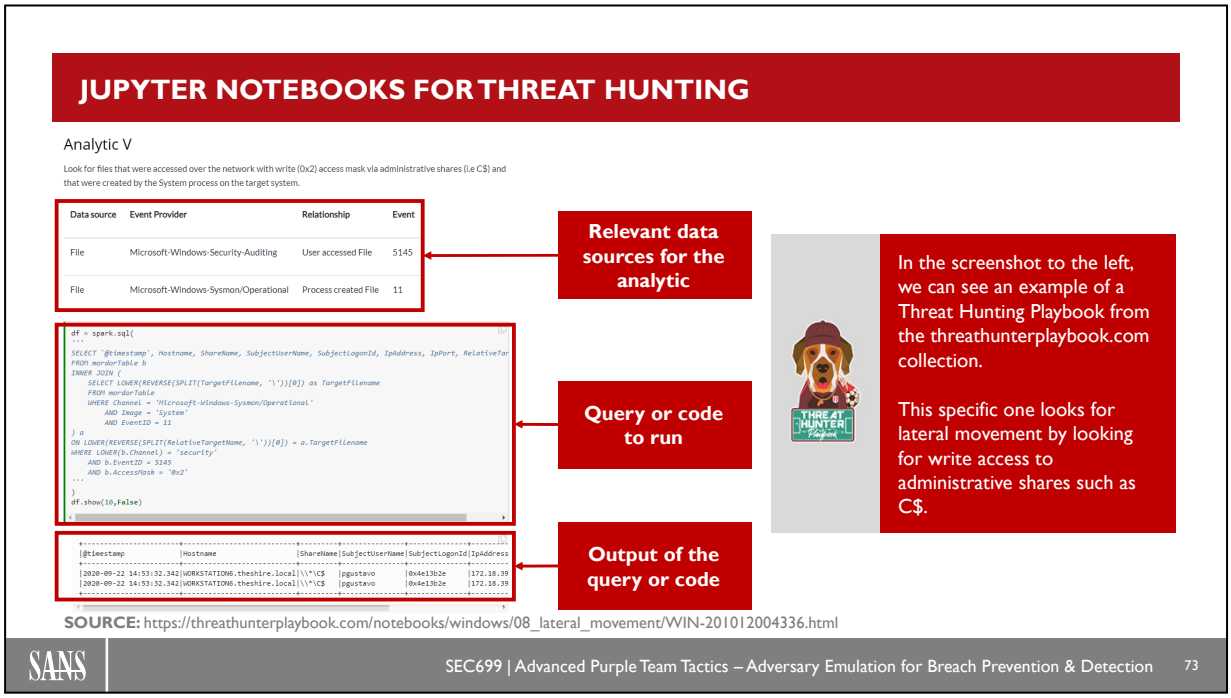
Jupyter Notebooks

A Jupyter Notebook is a document accessible through the web. It allows users to start "sessions" which allow input (e.g., programming code) and output (output of the code that was inputted) to be saved. Furthermore, we can add notes that document how the code works what additional analysis needs to be done manually on the output.

Something more on Jupyter:

"Project Jupyter is a non-profit, open-source project, born out of the IPython Project in 2014 as it evolved to support interactive data science and scientific computing across all programming languages. Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license. Jupyter is developed in the open on GitHub, through the consensus of the Jupyter community. For more information on our governance approach, please see our Governance Document. All online and in-person interactions and communications directly related to the project are covered by the Jupyter Code of Conduct. This Code of Conduct sets expectations to enable a diverse community of users and contributors to participate in the project with respect and safety (https://jupyter.org/)."

An excellent introduction to the Jupyter Threat Hunting notebooks with a threat hunting use case (https://threathunterplaybook.com/tutorials/jupyter/introduction.html). Furthermore, formal documentation on Jupyter, as a whole, can be found at https://jupyter.org/.



Jupyter Notebooks for Threat Hunting

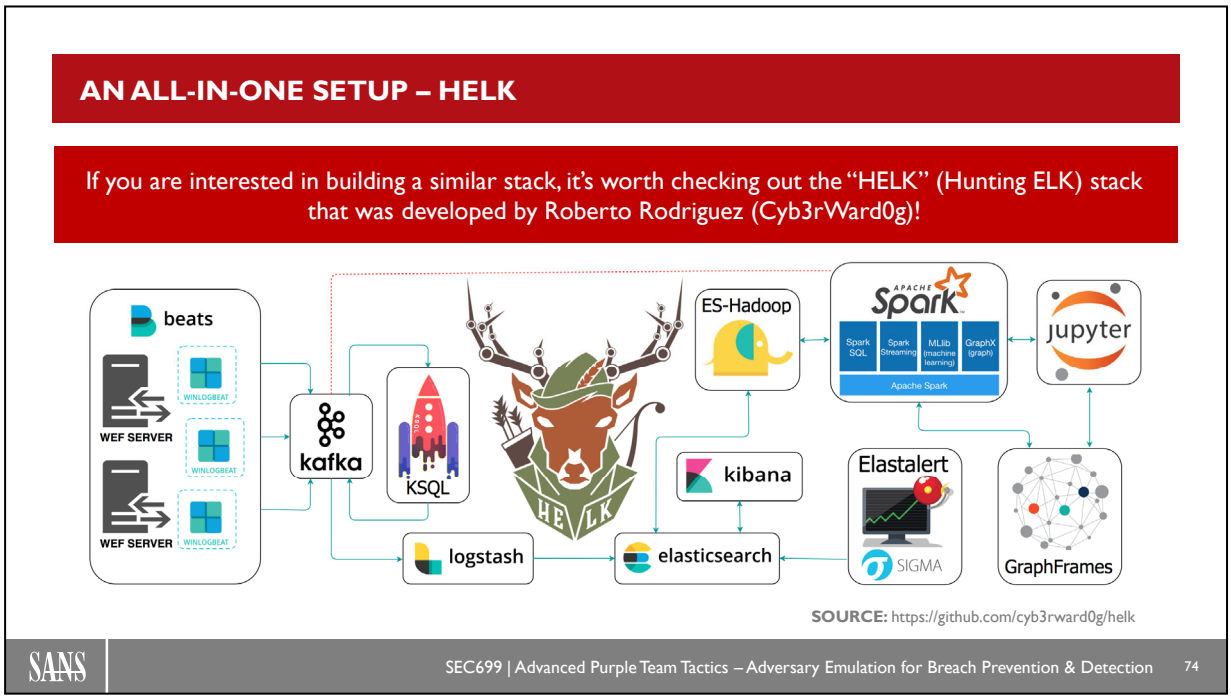
In the screenshot on the slide, we can see an example of a Threat Hunting Playbook from the threathunterplaybook.com collection. What is the threat hunting playbook?

"The Threat Hunter Playbook is a community-based open source project developed to share threat hunting concepts and aid the development of techniques and hypothesis for hunting campaigns by leveraging security event logs from diverse operating systems. This project provides not only information about detections, but also other very important activities when developing analytics such as data documentation, data modelling and even data quality assessments." (from https://threathunterplaybook.com/introduction.html)

The example looks for lateral movement by looking for write access to administrative shares such as C$. We have clearly marked the different components in the screenshot:

- Relevant data sources for this analytic (in this case Windows Security event ID 5145 and Sysmon event ID 11)
- The query to run on the back-end
- The output of the query or code

Additional information can be found on https://threathunterplaybook.com/.



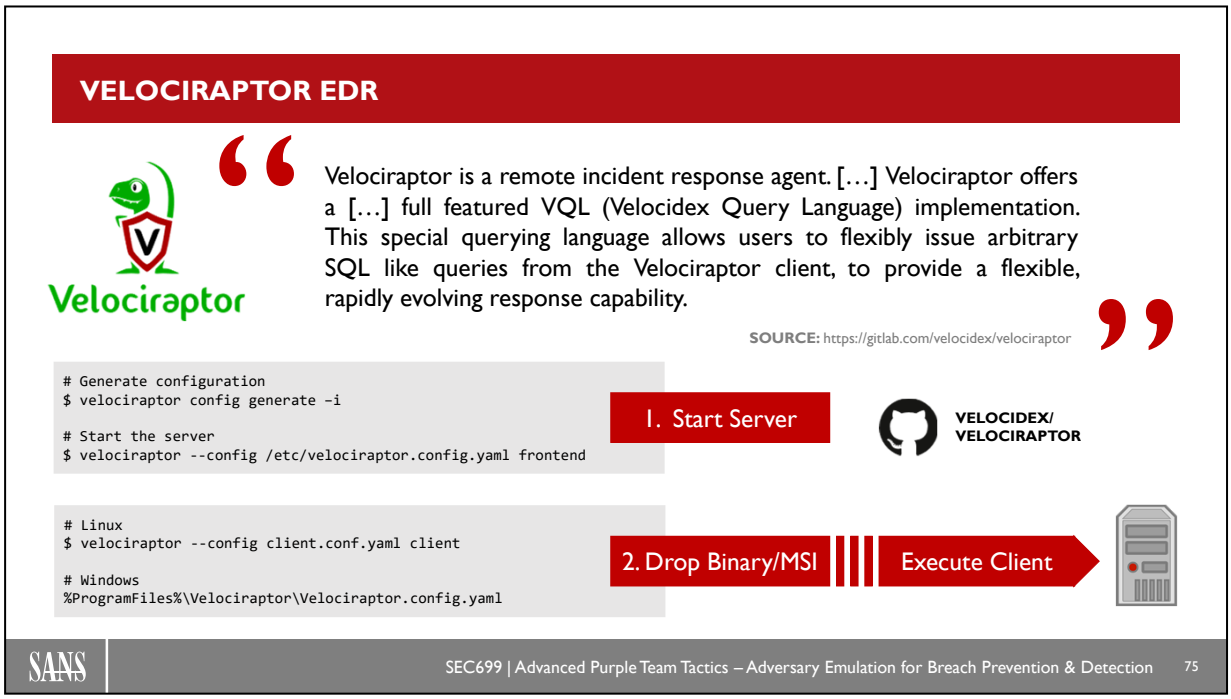
An All-in-One Setup – HELK

If you are interested in building a similar stack, it's worth checking out the "HELK" (Hunting ELK) stack that was developed by Roberto Rodriguez (Cyb3rWard0g)!

It leverages several components we also use (e.g., logstash, ElasticSearch, kibana, ElastAlert and SIGMA), but adds other components such as:

- Kafka: A distributed publish-subscribe messaging system that is designed to be fast, scalable, fault-tolerant, and durable.
- KSQL: Confluent KSQL is the open-source, streaming SQL engine that enables real-time data processing against Apache Kafka®. It provides an easy-to-use, yet powerful interactive SQL interface for stream processing on Kafka, without the need to write code in a programming language such as Java or Python.
- ES-Hadoop: An open-source, stand-alone, self-contained, small library that allows Hadoop jobs (whether using Map/Reduce or libraries built upon it such as Hive, Pig or Cascading or new upcoming libraries like Apache Spark) to interact with Elasticsearch.
- Spark: A fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python, and R, and an optimized engine that supports general execution graphs.
- GraphFrames: A package for Apache Spark which provides DataFrame-based Graphs.
- Jupyter Notebook: An open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.

You can find it over at https://github.com/cyb3rward0g/helk.



Velociraptor EDR

Velociraptor is developed by Michael Cohen (GRR developer) and Nick Klein (SANS Forensics Instructor). It is a remote incident response agent, which provides its own query language called VQL (Velocidex Query Language). This special querying language allows users to flexibly issue arbitrary SQL-like queries from the Velociraptor client, to provide a flexible, rapidly evolving response capability. This includes both acquisition of artifacts, but also execution of responsive actions (by running shell commands).

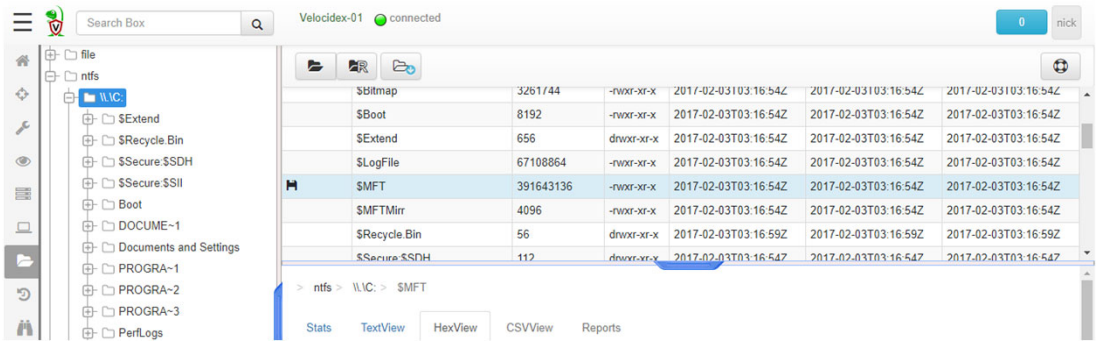
Velociraptor is managed by a central stack where a web server is running. Clients can subsequently be deployed on target machines. Velociraptor currently provides support for Windows, MacOS, and Linux platforms. As the tool was designed with security in mind, all communications between the client and the server are encrypted.

Additional information is available at https://www.velocidex.com.

VELOCIRAPTOR EDR – FILESYSTEM ACCESS

Velociraptor provides full remote file analysis capabilities allowing analysts to access all properties related to endpoint files.

VELOCIDEX/
VELOCIRAPTOR

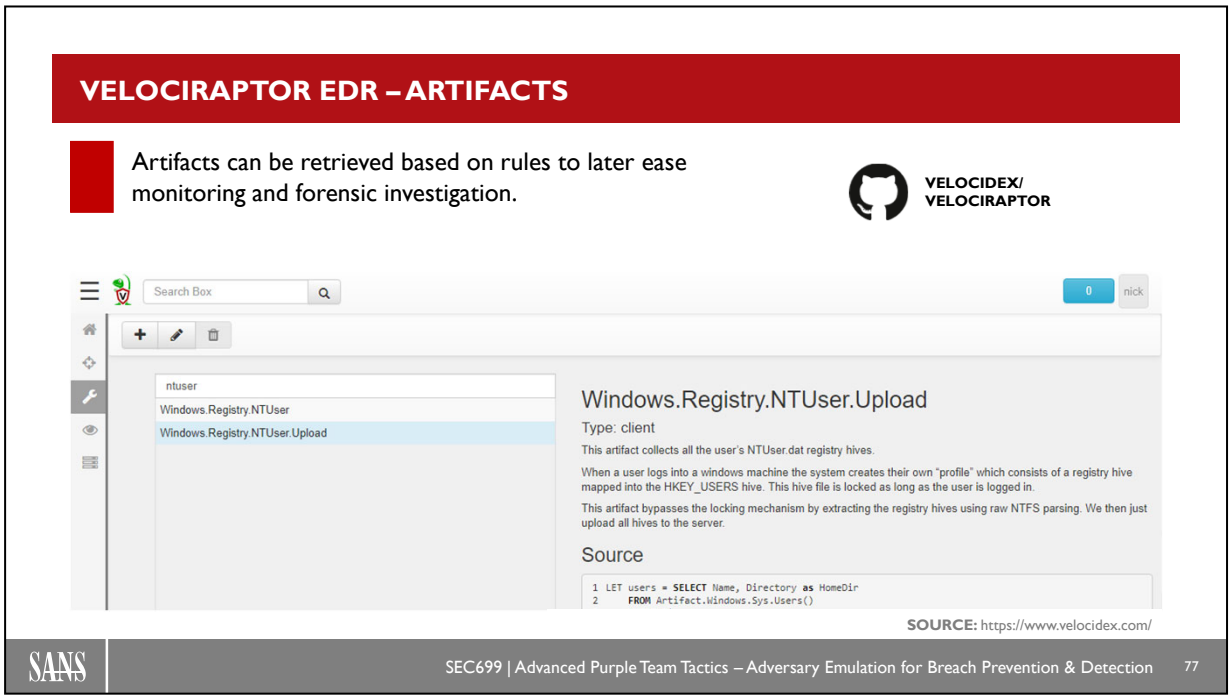


SOURCE: https://www.velocidex.com/

Our Central Stack: Velociraptor EDR – Filesystem Access

One interesting feature available in Velociraptor is full access to the remote filesystem. On this slide, we can see a remote "C:\" filesystem that is being browsed through the Velociraptor main web interface. One of the greatest strengths of this function is the ability to immediately download files from the remote system. This is highly useful for example to download suspect files that can subsequently be further investigated by analysts.

The careful observer / experienced security professional might recognize an interface that is similar to GRR (Google GRR). This is because Michael Cohen (one of the main Velociraptor developers) was one of the main developers of GRR as well.



Velociraptor EDR – Artifacts
The Velociraptor ecosystem runs on artifacts, which are methods of collection. Artifacts can be of different types and targets depending on our intended use case.

An artifact can either target a client (endpoint) or the server itself. Its type is either classic, bound to a specific moment in time, or event-based (only run it when a certain condition occurs).

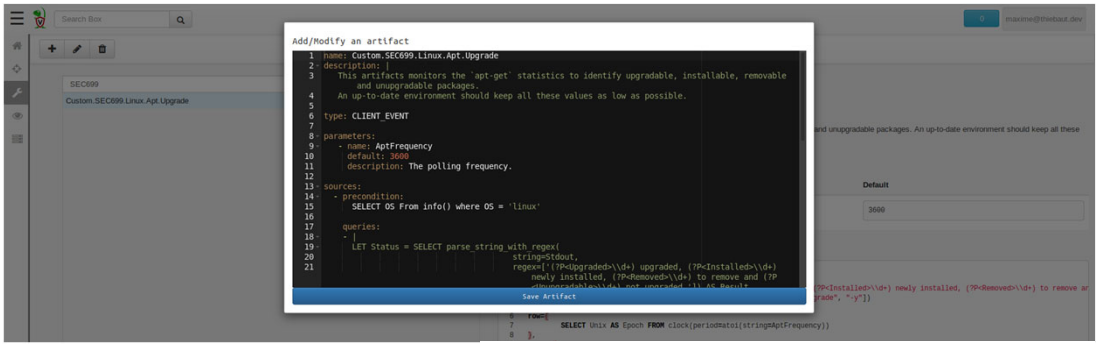
Artifacts can furthermore provide conditional execution when prerequisites have to be met. The obtained results can then be formatted to meet the analyst's desire either through classic tables and line-based charts or in a hunting format which allows the user to specify time-frames to analyze. Artifacts are written in Velociraptor's query language VQL (Velocidex Query Language). VQL is a simple SQL-like query language, except that instead of querying from tables, Velocifilter allows plugins to be defined as data sources.

It has a bit of a learning curve, but once you understand the query language, it's a highly powerful tool!

VELOCIRAPTOR EDR – CUSTOM ARTIFACTS (1)

Analysts can fully customize the artifacts and create their own client, server or event-based monitoring capabilities.

VELOCIDEX/
VELOCIRAPTOR



SOURCE: https://www.velocidex.com/

Velociraptor EDR – Custom Artifacts (1)

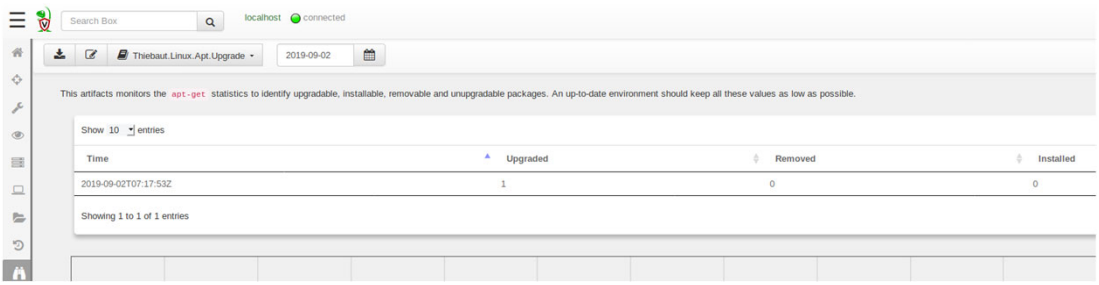
Another useful feature, especially when endpoints are geographically out of the analyst's reach, is Velociraptor's chaining capabilities. Many situations encountered through monitoring requires the retrieval of specific files and often in larger amounts than what would be done manually. Although Velociraptor enables analysts to retrieve specific files through the filesystem access, custom artifacts can be chained to build up the entire response based solely on detection and as such automatically download relevant evidence related to the chained artifacts.

Velociraptor Artifacts are written in YAML (YAML Ain't Markup Language) making it as easy to write as it is to visualize.

VELOCIRAPTOR EDR – CUSTOM ARTIFACTS (2)

Dashboards can be composed from the collected artifacts to ease their analysis.

VELOCIDEX/
VELOCIRAPTOR



SOURCE: https://www.velocidex.com/

Velociraptor EDR – Custom Artifacts (2)

The Velociraptor artifact files furthermore provide adequate reporting templates. Based on Markdown, these templates allow query results to be displayed through typical formats such as tables and line-charts. Artifact results can be filtered as shown in the above view. In the above example, where the artifact monitors the apt-get statistics to identify upgradable, installable, removable, and unupgradable packages, tables provide a useful view for time-specific events such as upgraded, removed, and installed packages whereas a line chart has more appropriate constant numbers such as packages that cannot be upgraded.

One yet to be improved feature of Velociraptor is the time-frame selection of the reports. Currently limited to 24h, this complicates the monitoring of events occurring less than once a day.

VELOCIRAPTOR EDR – EXECUTING COMMANDS

The VQL offers a wide range of usable plugins ranging from detection up until command execution through the `execve` plugin.

VELOCIDEX/ VELOCIRAPTOR

This example waits for the detection of `psexec` usages and automatically kills the processes associated to it.

```
SELECT * FROM foreach(
          row={ SELECT * FROM Artifact.Windows.Detection.PsexecService() },
          query={
             SELECT ServiceName, PathName, Modified, FileSize, Timestamp,
                    ServiceType, ChildProcess, Stdout, Stderr FROM execve(
               argv=["taskkill", "/PID", PID, "/T", "/F"])
       })
```

SOURCE: https://www.velocidex.com/

Velociraptor EDR – Executing Commands
To provide its users with full customization capabilities, Velociraptor provides a range of client-side and server-side plugins. Each plugin, specialized for a specific use-case, provides results for a given set of arguments as shown with “execve” in the above example.

The example shown on the slide will use the Artifact.Windows.Detection.PsexecService() function to continuously check for processes which have been started via the PSExec service. In case a process is detected, it will kill this process.

In addition to plugins, Velociraptor exposes functions such as the showcased “foreach” or documented “clock”, “atoi”, “base64decode” and others.

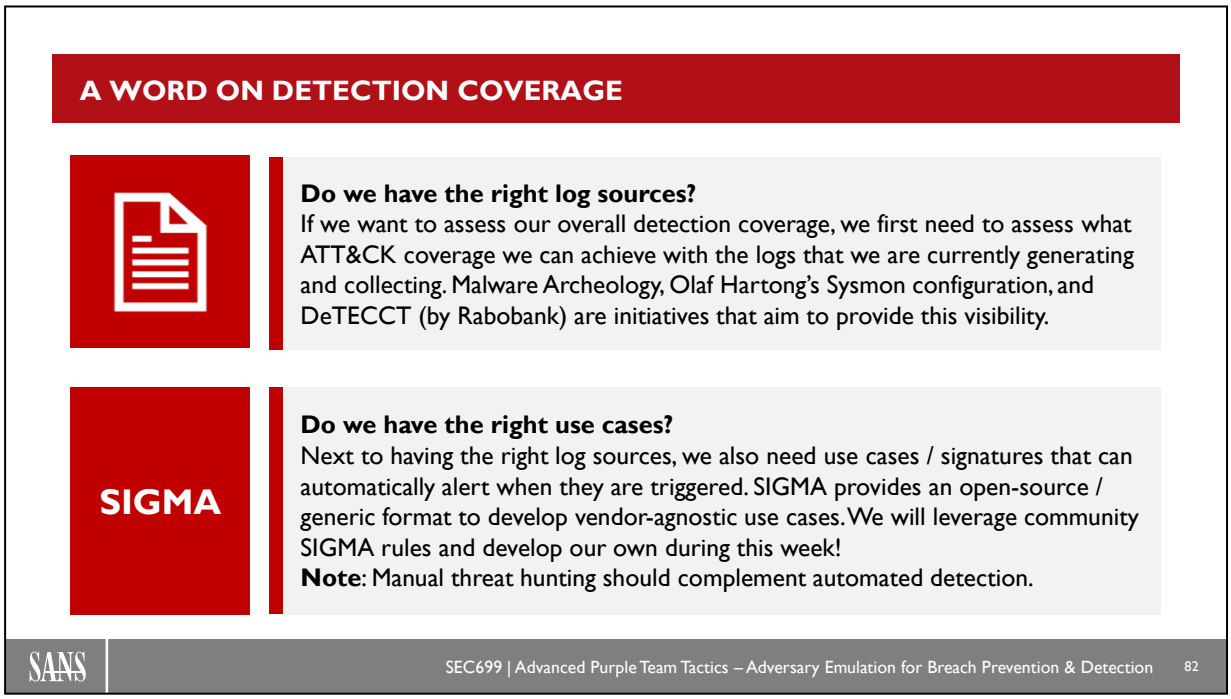
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.



A Word on Detection Coverage
Next to using MITRE ATT&CK for the definition of an emulation plan, we can also use it to track and report on detection coverage.

In order to successfully detect adversary steps in your environment, there's two main questions to ask:

Do we have the right log sources?

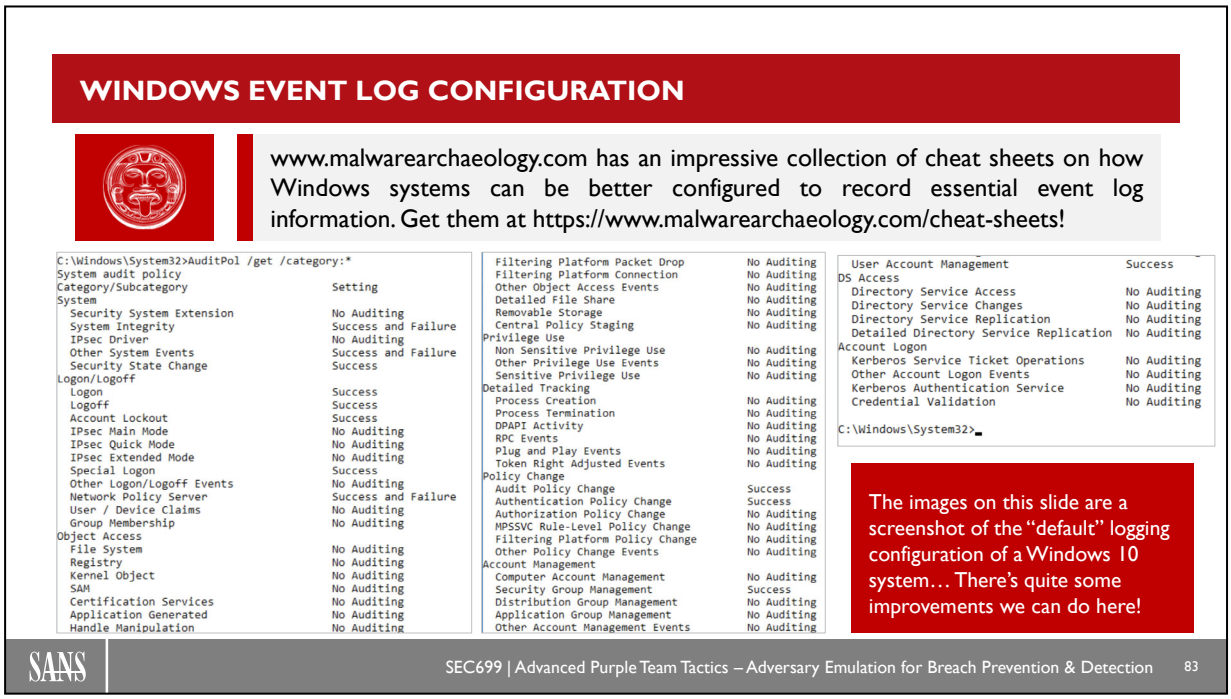
If we want to assess our overall detection coverage, we first need to assess what ATT&CK coverage we can achieve with the logs that we are currently generating and collecting. Malware Archeology, Olaf Hartong's Sysmon configuration, and DeTACCT (by Rabobank) are initiatives that aim to provide this visibility. You can find these projects here:

- https://www.malwarearchaeology.com/cheat-sheets
- https://github.com/olafhartong/sysmon-modular
- https://github.com/rabobank-cdc/DeTTECT

Do we have the right use cases?

Next to having the right log sources, we also need use cases / signatures that can automatically alert when they are triggered. SIGMA provides an open-source / generic format to develop vendor-agnostic use cases; many (if not all), SIGMA rules are mapped to MITRE ATT&CK techniques. We will leverage community SIGMA rules and develop our own during this week!

Note: Manual threat hunting should complement automated detection.



Windows Event Log Configuration

Windows event logs are an absolute prerequisite for proper detection! The slide above provides a full insight of the local audit policy for a standard Windows 10 system. For some people, there's bound to be some surprises:

- Failed logons (for example due to a bad password) are not logged
- There is no object access logging configured whatsoever
- The use of "sensitive privileges" is not logged at all
- …

So, how do we improve this and what logs are most valuable for us? The people over at www.malwarearchaeology.com have an impressive collection of cheat sheets on how Windows systems can be better configured to record essential event log information. Get them at https://www.malwarearchaeology.com/cheat-sheets!

INTRODUCING SYSMON

Sysmon is short for System Monitoring

Sysmon installs a Windows service and a device driver

These components monitor activity on a system:

- Creation and termination of processes
- Loading of executable images
- Network connection establishing
- …

Sysmon provides **unrivaled visibility** on Windows endpoints!

Introducing Sysmon

Sysmon is a system monitoring tool that is part of the Sysinternals Suite.

It is a tool that was originally developed for Microsoft. It is deployed on many of their servers and workstations to monitor system activity. In case of incidents, Sysmon provides a valuable log of system activities that can help forensic investigators to reconstruct an incident. Sysinternal tools are stand-alone tools that don't come with an installer (like setup.exe or install.msi). For Sysmon, there is Sysmon.exe and Sysmon64.exe.

Sysmon.exe is a 32-bit version that embeds the 64-bit version, too. Sysmon64.exe is a 64-bit version only; it is provided for Windows systems that only support 64-bit executables, and not 32-bit (Windows Servers without 32-bit subsystem).

If the 32-bit version is executed on a 64-bit OS, it will extract the 64-bit version and run that instead. When Sysmon is installed on a Windows machine, it installs a Windows service and a device driver. These components are necessary to detect and record system activities like the creation and termination of processes, loading of executable images, creation of network connections, loading of drivers, …

All this activity is logged in a dedicated Windows event log.

SYSMON EVENT TYPES

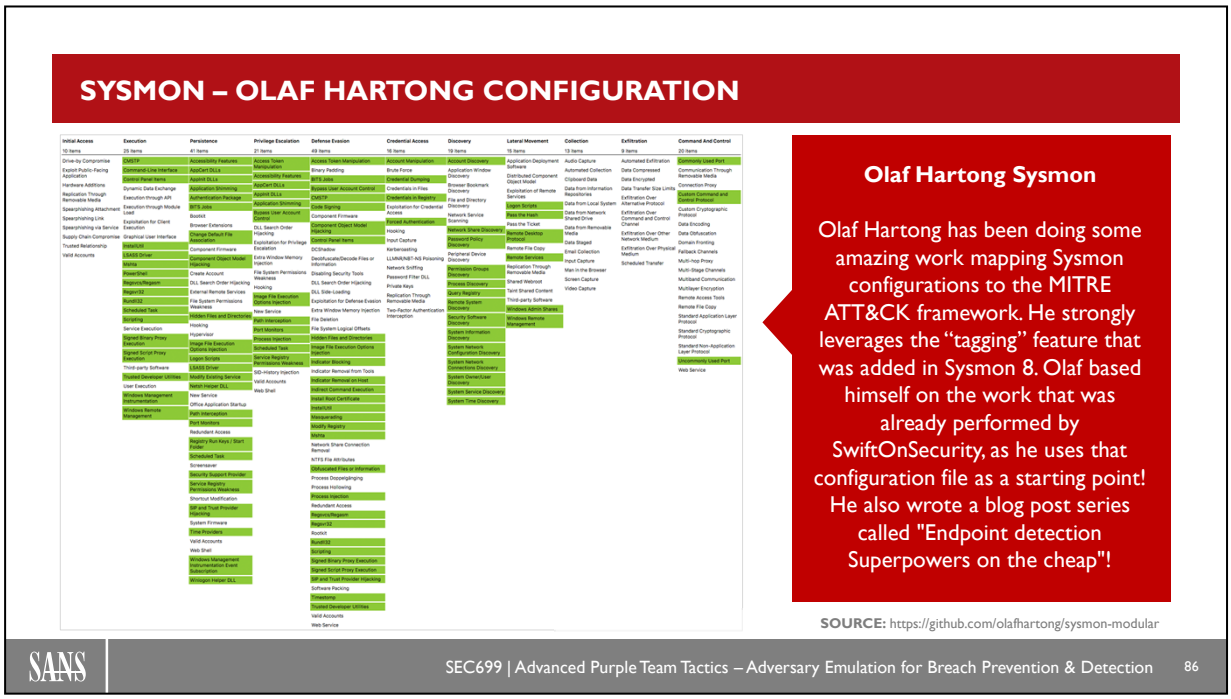
Event ID	Description	Event ID	Description
1	Process creation	14	RegistryEvent (Key and Value Rename)
2	A process changed a file creation time	15	FileCreateStreamHash
3	Network connection	16	Sysmon Configuration Changed
4	Sysmon service state changed	17	Pipe Created
5	Process terminated	18	Pipe Connected
6	Driver loaded	19	WmiEventFilter activity detected
7	Image loaded	20	WmiEventConsumer activity detected
8	CreateRemoteThread	21	WmiEventConsumerToFilter activity detected
9	RawAccessRead	22	DNS event (DNS query)
10	ProcessAccess	23	FileDelete (A file delete was detected)
11	FileCreate	24	ClipboardChange (new clipboard content)
12	RegistryEvent (Object create and delete)	25	ProcessTampering (Process image change)
13	RegistryEvent (Value Set)	255	Error

Sysmon Event Types

Since Sysmon version 13, it records 25 different types of events. These events monitor objects like processes, files, registry objects, … But, also, events to monitor changes to Sysmon itself (IDs 4 and 16) can be logged. This can indicate tampering attempts.

Other event IDs that can be indicative of tampering by malicious actors are changes in file creation times (ID 2), creation of remote threads (ID 8), often used for code injection, opening of processes for process tampering (ID 10),… Recent additions to Sysmon have included:

- DNS query logging (event ID 22)
- File deletion activity (event ID 23), which could be useful for ransomware scenarios
- Clipboard activity (event ID 24)
- Process tampering (event ID 25), which could be useful to detect advanced attacks such as process hollowing



Sysmon – Olaf Hartong Configuration

Olaf Hartong has been doing some amazing work mapping Sysmon configurations to the MITRE ATT&CK framework. He strongly leverages the "tagging" feature that was added in Sysmon 8. Olaf based himself on the work that was already performed by SwiftOnSecurity, as he uses that configuration file as a starting point! He also wrote a blog post series called "Endpoint detection Superpowers on the cheap", which is definitely worth reading:

- Endpoint detection Superpowers on the cheap - part 1 - MITRE ATT&CK, Sysmon and my modular configuration (https://medium.com/@olafhartong/endpoint-detection-superpowers-on-the-cheap-part-1-e9c28201ac47)
- Endpoint detection Superpowers on the cheap - part 2 - Deploy and Maintain (https://medium.com/@olafhartong/endpoint-detection-superpowers-on-the-cheap-part-2-deploy-and-maintain-d06580329fe8)
- Endpoint detection Superpowers on the cheap - part 3 - Sysmon Tampering (https://medium.com/@olafhartong/endpoint-detection-superpowers-on-the-cheap-part-3-sysmon-tampering-49c2dc9bf6d9)

Olaf's GitHub repository can be found here: https://github.com/olafhartong/sysmon-modular

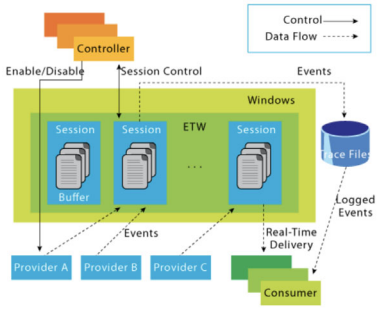
For "quick and dirty" implementation, Olaf's consolidated configuration file can be found here: https://github.com/olafhartong/sysmon-modular/blob/master/sysmonconfig.xml

DIGGING A BIT DEEPER: INTRODUCING ETW

ETW

"Event Tracing for Windows (ETW) is an efficient kernel-level tracing facility that lets you log kernel or application-defined events to a log file. You can consume the events in real time or from a log file and use them to debug an application or to determine where performance issues are occurring in the application."

SOURCE: HTTPS://https://docs.microsoft.com/EN-US/WINDOWS/WIN32/ETW/ABOUT-EVENT-TRACING



Controllers

Controllers allow us to start / stop tracing sessions
Example: Logman.exe

Providers

Providers provide the events that are being traced
Example: Microsoft-Windows-Security-Auditing

Consumers

Consumers consume the events that are being traced
Example: Event Viewer

SOURCE: https://docs.microsoft.com/en-us/archive/blogs/ntdebugging/part-1-etw-introduction-and-overview

Digging a Bit Deeper: Introducing ETW

In order to fully understand how Sysmon works, we need to understand a bit more about its internals. For several event types, Sysmon relies on Event Tracing for Windows (EWT):

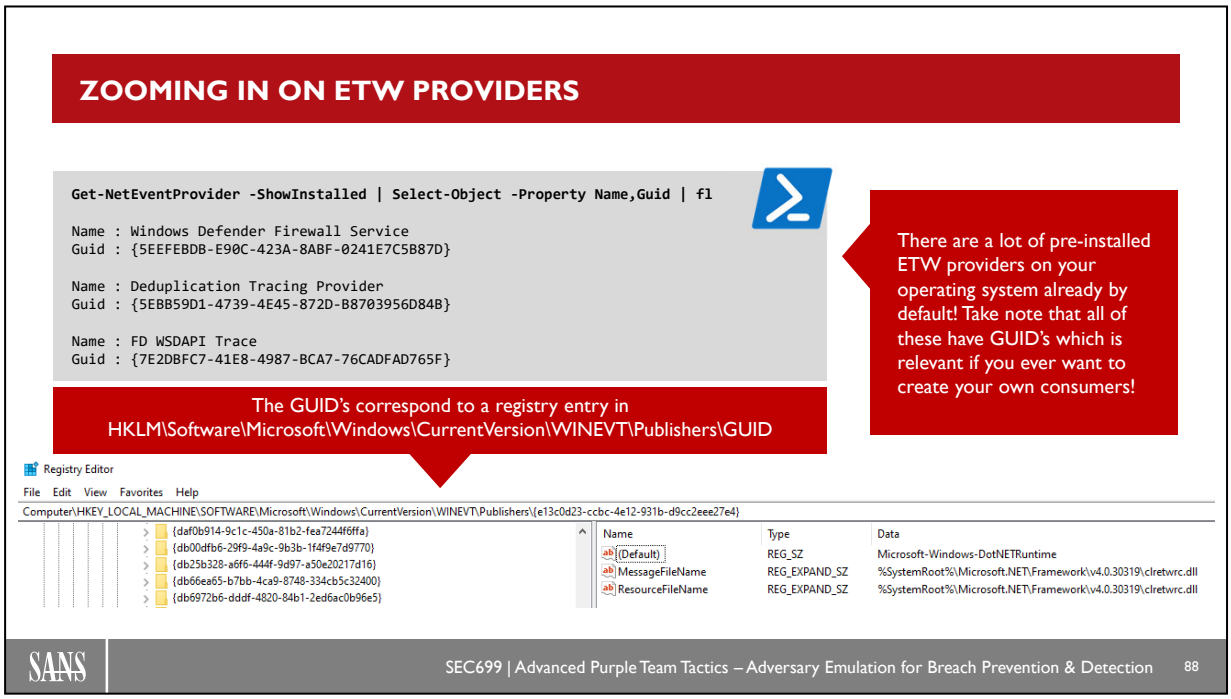
"Event Tracing for Windows (ETW) is an efficient kernel-level tracing facility that lets you log kernel or application-defined events to a log file. You can consume the events in real time or from a log file and use them to debug an application or to determine where performance issues are occurring in the application." (Source: https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing).

The overall ETW architecture is divided in three main components:

- Controllers allows us to start or stop tracing sessions. They also enable providers. An example of a controller is logman.exe (built-in Windows).
- Providers provide the events that are being traced. An example of a prover is Microsoft-Windows-Security-Auditing.
- Consumers consume the events that are being traced. An example of a consumer is the Event Viewer.

Some good reads on ETW can be found at:

- https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing
- https://blogs.msdn.microsoft.com/ntdebugging/2009/08/27/part-1-etw-introduction-and-overview/
- https://medium.com/threat-hunters-forge/threat-hunting-with-etw-events-and-helk-part-1-installing-silketw-6eb74815e4a0



Zooming in on ETW Providers

Let's investigate ETW providers a little bit more. Microsoft Windows systems come packed with a large number of providers out of the box. We can easily enumerate them using either PowerShell or traditional Windows command-line syntax. For Powershell, we could use the below syntax:

Get-NetEventProvider -ShowInstalled | Select-Object -Property Name,Guid | Sort-Object Name

As demonstrated on the slide, this command will return the names of the different providers, along with their GUID (unique identifier). In order to achieve the same using the command line, we can use the "logman" utility:

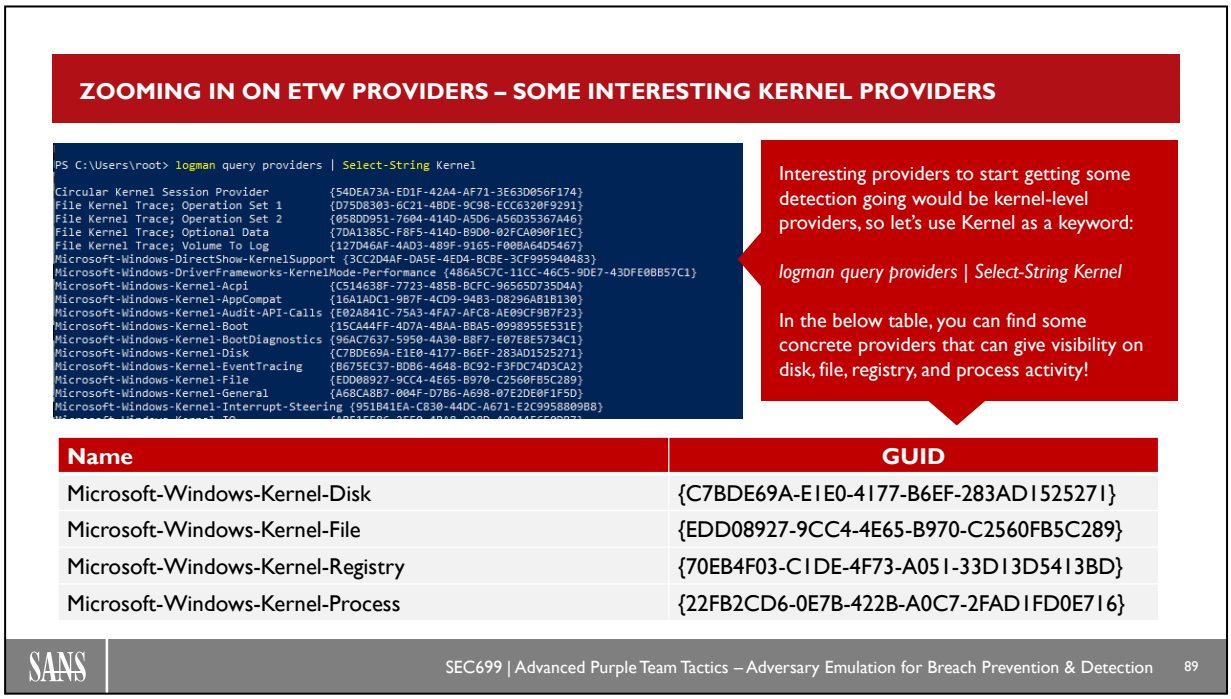
logman query providers

Providers can also be identified by analyzing the registry of a Windows system, as all providers are listed in the following registry location:

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{GUID}

For additional insights on how ETW can be leveraged, please refer to the following documentation:

- https://docs.microsoft.com/en-us/powershell/module/eventtracingmanagement/get-etwtraceprovider?view=win10-ps
- https://medium.com/palantir/tampering-with-windows-event-tracing-background-offense-and-defense-4be7ac62ac63



Zooming in on ETW Providers – Some Interesting Kernel Providers

The list of ETW providers can look a bit daunting and there's so much information to look into…

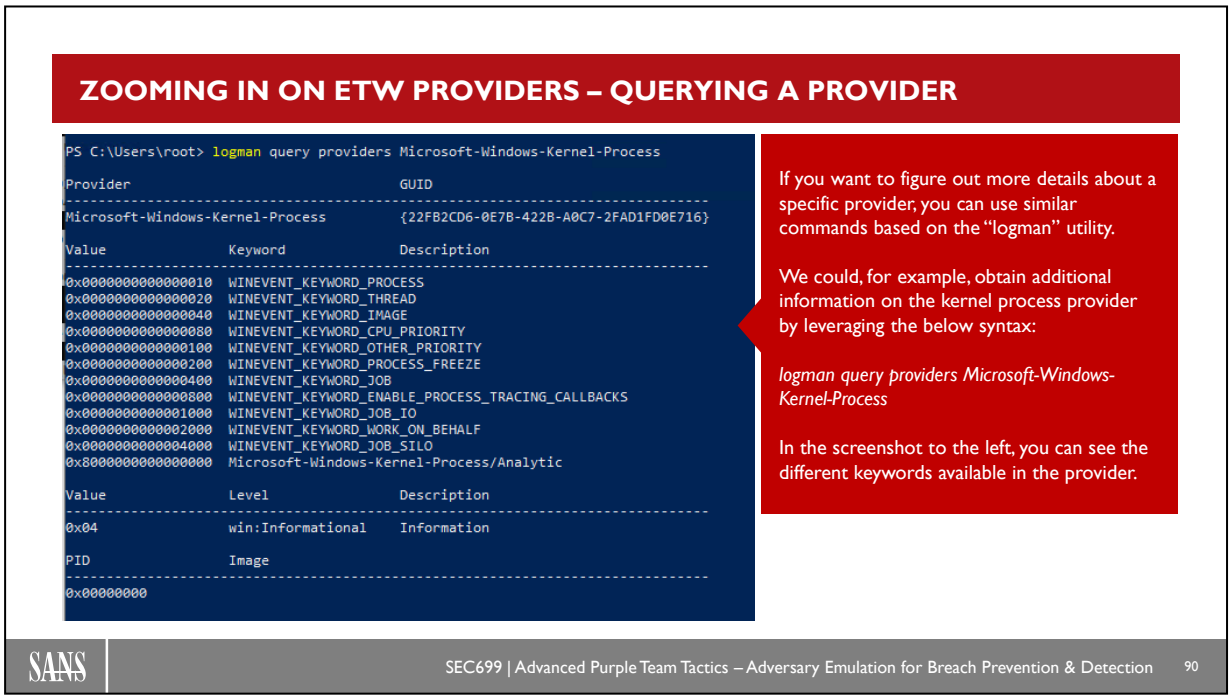
Let's start with the obvious stuff first!

Interesting providers to start getting some detection going would be kernel-level providers, so let's use Kernel as a keyword:

logman query providers | Select-String Kernel

Below, you can find some concrete providers that can give visibility on disk, file, registry, and process activity:

- Microsoft-Windows-Kernel-Disk (GUID C7BDE69A-E1E0-4177-B6EF-283AD1525271)
- Microsoft-Windows-Kernel-File (GUID EDD08927-9CC4-4E65-B970-C2560FB5C289)
- Microsoft-Windows-Kernel-Registry (GUID 70EB4F03-C1DE-4F73-A051-33D13D5413BD)
- Microsoft-Windows-Kernel-Process (GUID 22FB2CD6-0E7B-422B-A0C7-2FAD1FD0E716)



Zooming in on ETW Providers – Querying a Provider

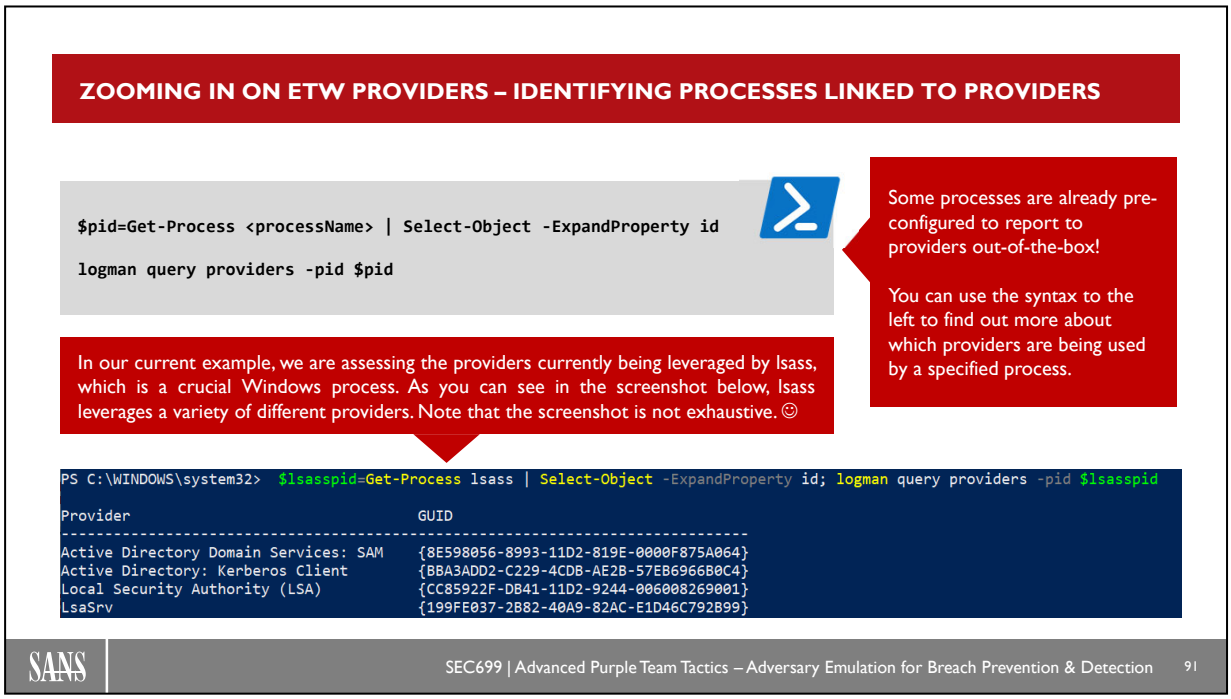
If you want to figure out more details about a specific provider, you can use similar commands based on the "logman" utility.

We could, for example, obtain additional information on the kernel process provider by leveraging the below syntax:

logman query providers Microsoft-Windows-Kernel-Process

In the screenshot on the slide, you can see the different keywords available in the provider (and their associated values).

These values can afterwards be used in tracing sessions, which we'll discuss in further slides.



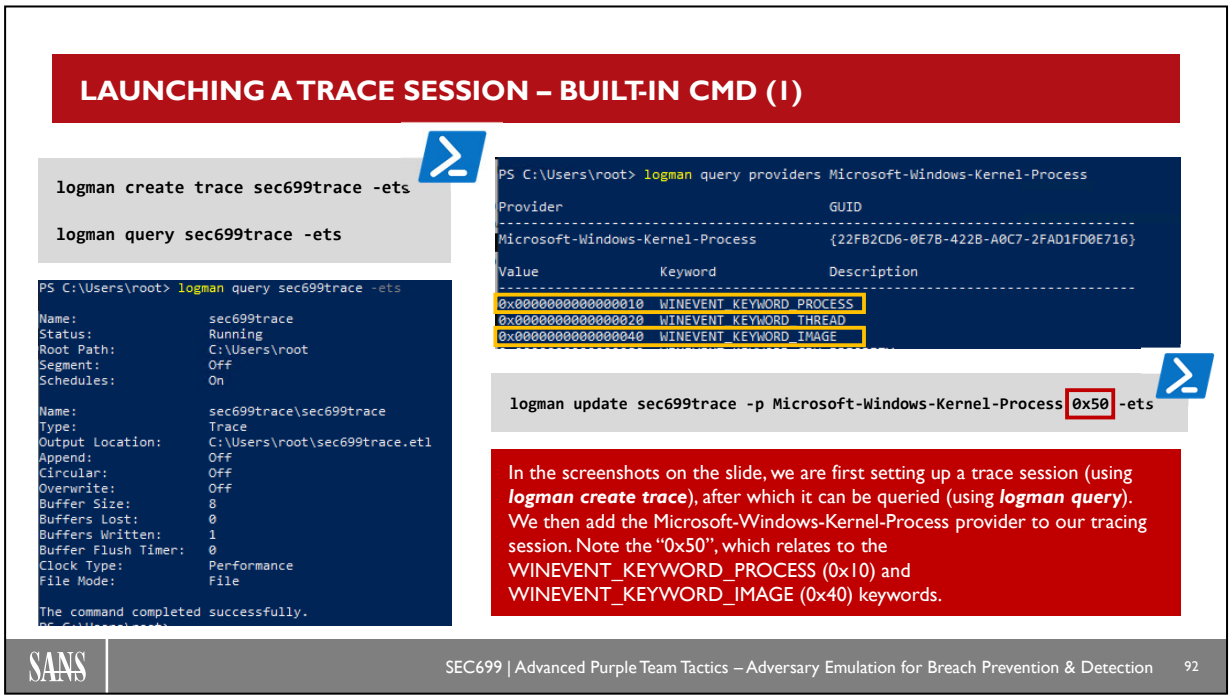
Zooming in on ETW Providers – Identifying Processes Linked to Providers

Some processes are already preconfigured to report to providers out-of-the-box! Let's investigate this a little further…

On the slide, you can see that we are leveraging the following syntax:

$pid=Get-Process <processName> | Select-Object -ExpandProperty id
logman query providers -pid $pid

In our current example, we are assessing the providers currently being leveraged by lsass, which is a crucial Windows process. This syntax will first fetch the process ID for a process named "lsass", after which it will use this variable to fetch related ETW providers. The screenshot highlights several providers lsass is reporting to. Given the importance of lsaass, it should not come as a surprise that the screenshot above is not exhaustive. You might recognize some terminology (SAM, Kerberos,…), which we will discuss in a lot more detail throughout the course!



Launching a Trace Session – Built-in CMD (1)

Now that we've looked at the different providers and their keywords, let's start actually using it!

As a first step, let's start a logman trace session called "sec699trace" using the "logman create trace" syntax:

logman create trace sec699trace -ets

The result of this command will be rather short, as it should just tell you the command was executed successfully. Once the trace session is started, we can now query to understand its properties. We can do this by using the "logman query" syntax:

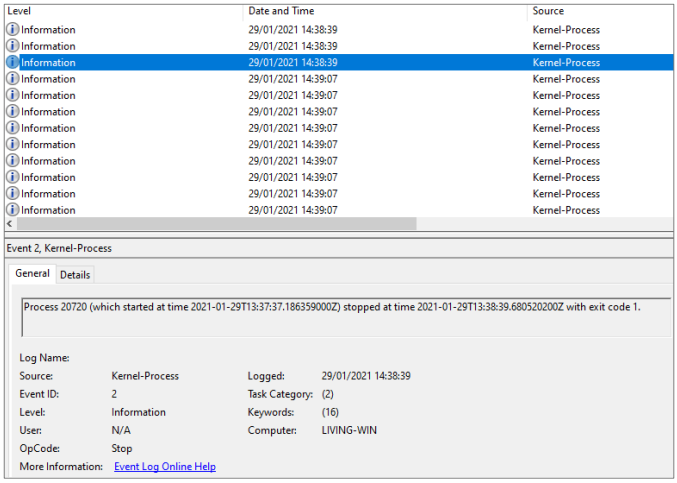
logman query sec699trace -ets

You will notice that the trace session is being written to a file on disk (in this case C:\Users\root\sec699trace.etl). Furthermore, you'll notice that there's no mention of any providers just yet. Let's add a provider to the trace session! For our example, we'll add the Microsoft-Windows-Kernel-Process provider to our trace session. We can do this by using the "logman update" syntax:

logman update sec699trace -p Microsoft-Windows-Kernel-Process 0x50 –ets

What does the 0x50 stand for? If you recall the details we enumerated previously, you might remember that there are certain keywords listed under the Microsoft-Windows-Kernel-Process provider. In our example, we are interested in the WINEVENT_KEYWORD_PROCESS (0x10) and WINEVENT_KEYWORD_IMAGE (0x40) keywords. When adding these up, you end up with 0x50.

LAUNCHING A TRACE SESSION – BUILT-IN CMD (2)



Now that our trace session is running, let's try to have a look at some of the events.

As seen on the previous slide, the trace session is writing to the C:\Users\root\sec699trace.etl file.

This .etl file can be opened in the Windows Event Viewer, where we can see the events as illustrated to the left.

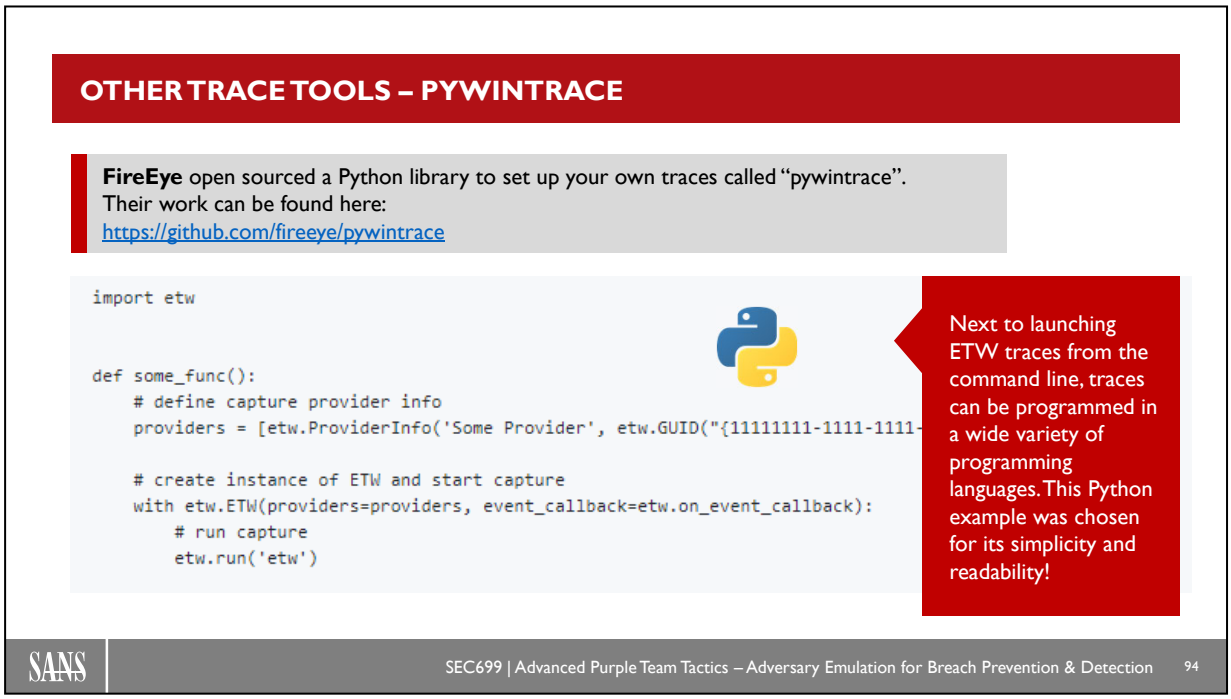
Launching a Trace Session – Built-in CMD (2)

Now that our trace session is running, let's try to have a look at some of the events. As seen on the previous slide, the trace session is writing to the C:\Users\root\sec699trace.etl file. This .etl file can be opened in the Windows Event Viewer, where we can see the events as illustrated above.

The event opened on the slide is a process exit event. With event ID 2, you can see we can deduce the following types of information:

- Process ID
- Start time of the process
- Stop time of the process
- Exit code of the process (in this case 1)

When compared to Sysmon, this is of course much more of a "raw" log, but it could be used to do some deeper analysis / correlation.

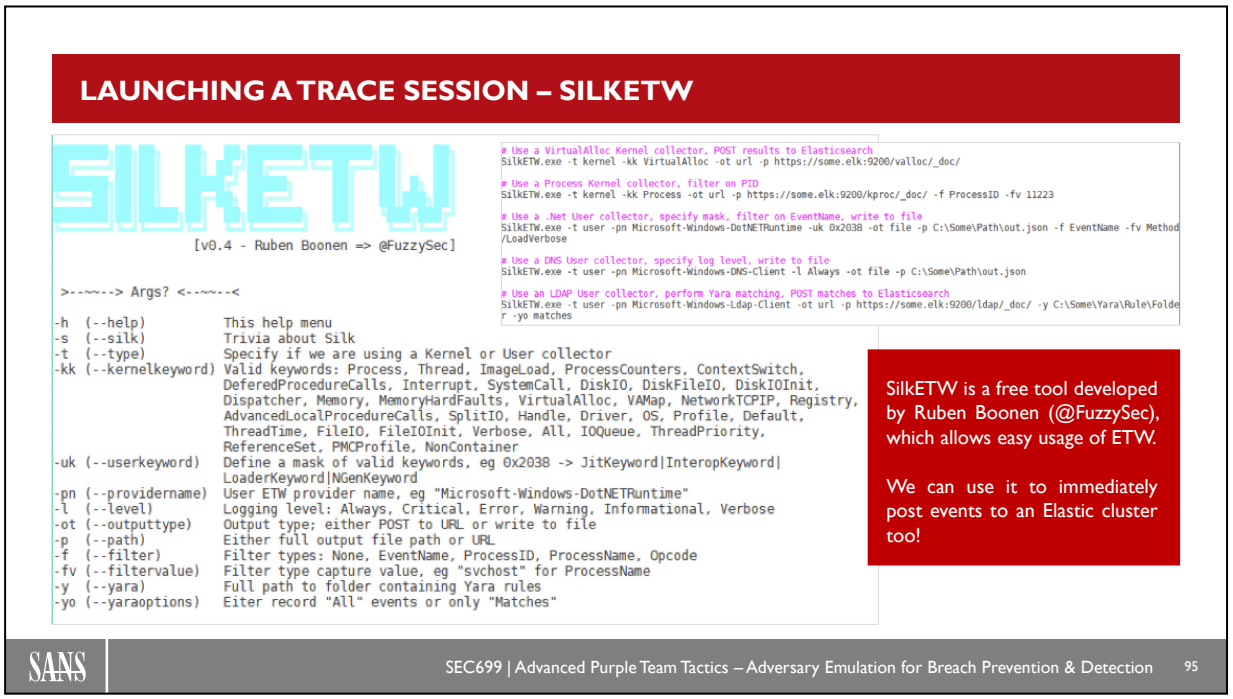


Other Trace Tools – Pywintrace

Next to launching ETW traces from the command line, traces can be programmed and leveraged in a wide variety of programming languages. This way, the detailed events generated by ETW can be used as part of programming logic. Quite often, EDR tools leverage ETW internally to obtain deep visibility on what is happening on the OS.

FireEye open sourced quite an interesting Python library to leverage ETW called "pywintrace".

The library and its supporting documentation can be found at https://github.com/fireeye/pywintrace.



Launching a Trace Session – SilkETW
Using ETW, we have virtually unlimited visibility into Windows kernel-level activity. We could thus see the tricks previously mentioned as they appear (e.g., a call to "CreateProcess" with an explicitly set parent process). SilkETW is a free tool by FireEye that facilitates use of ETW!

From their GitHub page:

"SilkETW & SilkService are flexible C# wrappers for ETW, they are meant to abstract away the complexities of ETW and give people a simple interface to perform research and introspection. While both projects have obvious defensive (and offensive) applications they should primarily be considered as research tools.

For easy consumption, output data is serialized to JSON. The JSON data can either be written to file and analyzed locally using PowerShell, stored in the Windows eventlog or shipped off to 3rd party infrastructure such as Elasticsearch."

Full information can be found on https://github.com/fireeye/SilkETW.

INTRODUCING DETTECT

DeTT&CT

> DeTT&CT aims to assist Blue Teams using ATT&CK to score and compare data log source quality, visibility coverage, detection coverage and threat actor behaviors. All of which can help, in different ways, to get more resilient against attacks targeting your organization.

SOURCE: https://github.com/rabobank-cdc/DeTTECT

Data sources for endpoints-example

Data source name	Date registered	Date connected	Products	Comment	Available for data analytics	DQ: device completeness	DQ: data field completeness	DQ: timeliness	DQ: consistency	DQ: retention	DQ: score
API monitoring					False	0	0	0	0	0	0
Access tokens					False	0	0	0	0	0	0
Anti-virus	2019-01-10	2000-01-01	AV Product		True	4	2	3	5	5	3.8
Application logs					False	0	0	0	0	0	0
Asset management					False	0	0	0	0	0	0
Authentication logs					False	0	0	0	0	0	0
BIOS					False	0	0	0	0	0	0
Binary file metadata					False	0	0	0	0	0	0
Browser extensions					False	0	0	0	0	0	0
Component firmware					False	0	0	0	0	0	0
DLL monitoring					False	0	0	0	0	0	0
DNS records	2019-03-01	2017-04-01	Windows DNS server		True	5	5	5	5	5	5
Data loss prevention					False	0	0	0	0	0	0
Detonation chamber					False	0	0	0	0	0	0
Digital certificate logs					False	0	0	0	0	0	0
Disk forensics	2019-01-10	2019-01-01	Manual, Commercial tool		True	5	5	1	5	0	4
EFI					False	0	0	0	0	0	0
Email gateway	2019-01-10	2000-01-01	Email Gateway Product		True	4	3	3	5	3	3.6
Environment variable					False	0	0	0	0	0	0
File monitoring					False	0	0	0	0	0	0
Host network interface					False	0	0	0	0	0	0
Kernel drivers					False	0	0	0	0	0	0
Loaded DLLs					False	0	0	0	0	0	0
MBR					False	0	0	0	0	0	0
Mail server					False	0	0	0	0	0	0
Malware reverse engineering					False	0	0	0	0	0	0
Named Pipes					False	0	0	0	0	0	0
Netflow/Enclave netflow					False	0	0	0	0	0	0
Network device logs					False	0	0	0	0	0	0
Network intrusion detection system	2019-01-10	2016-01-01	NIDS		True	4	3	3	4	4	3.6
Network protocol analysis					False	0	0	0	0	0	0
Packet capture					False	0	0	0	0	0	0

SANS SEC699 | Advanced Purple Team Tactics – Adversary Emulation for Breach Prevention & Detection 96

Introducing DeTTECT

DeTTECT (Detect Tactics, Techniques & Combat Threats) was introduced by the Dutch bank Rabobank. From its official documentation, we can deduce the following definition:

"DeTT&CT aims to assist Blue Teams using ATT&CK to score and compare data log source quality, visibility coverage, detection coverage and threat actor behaviors. All of which can help, in different ways, to get more resilient against attacks targeting your organization."

DeTTECT enables Blue Teams to track the quality of their data sources using YAML source files. YAML syntax and structure is available on the Git wiki.

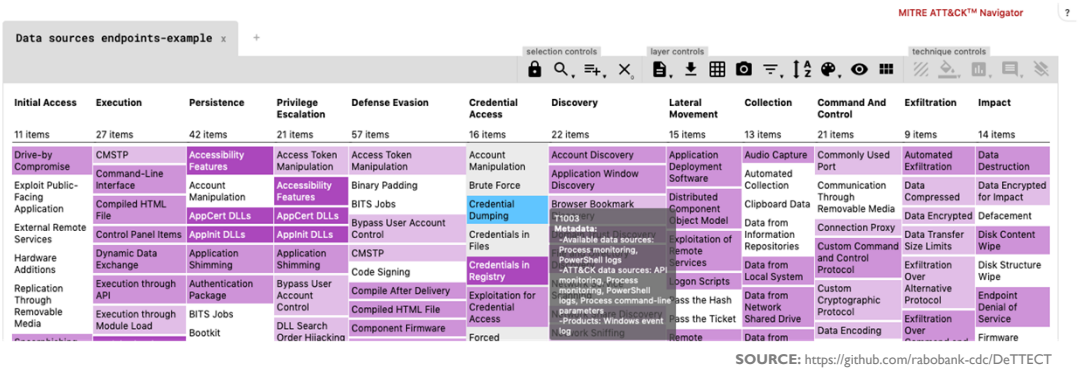
You can find the official repository at https://github.com/rabobank-cdc/dettect.

DeTTECT works by describing your capabilities at multiple levels in order to obtain an overview, as shown above, of your current status and gaps to improve. The different scored regions start from low-level details such as the available "data-sources" and work up the chain through the "visibility" of interpretation up until "detection" capabilities and "threat actor" resiliency.

ASSESSING DATA SOURCE & VISIBILITY COVERAGE USING DETTECT

The source scoring can be used to provide an overview of your source coverage using MITRE's ATT&CK navigator.

RABOBANK-CDC/ DETTECT



SOURCE: https://github.com/rabobank-cdc/DeTTECT

Assessing Data Source & Visibility Coverage Using DeTTECT

The source of relationships between ATT&CK entities (Group, Techniques, Tactics, …) lays in the quality of a Blue Team's data sources. DeTTECT's first stage of usage requires Blue Teams to list and evaluate their data source of which 50 different types are preincluded in the framework. Each data source is scored according to a predefined list of criteria evaluated on a scale of 0 to 5:

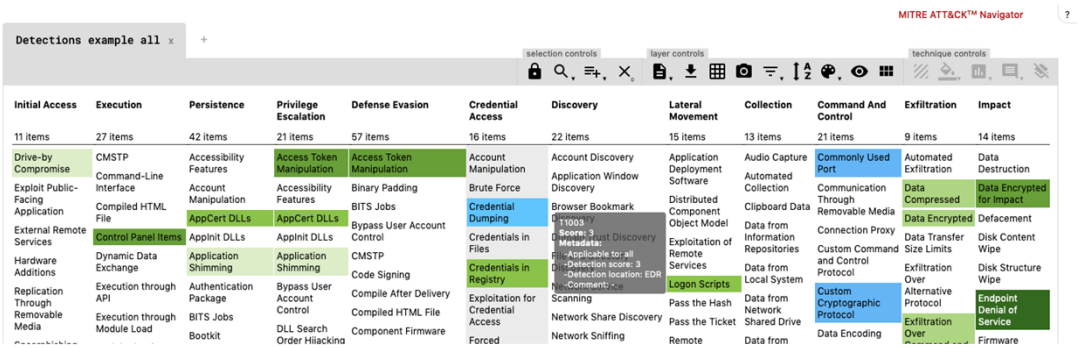
- Data Completeness: Whether the data is available for all devices/users.
- Data Field Completeness: Whether the data is populated with the appropriate fields of information; i.e., in the case of Procmon, do we have the PIDs?
- Timeliness: Whether data is available right away; i.e., in the case of Windows Event Logs, are they constantly aggregated, or do we have a delay of multiple hours?
- Consistency: Whether data is standardized or not; i.e., can we cross-reference fields with other data sources?
- Retention: Whether data retention is done as desired; i.e., are logs retained for the desired period?

The next step critical to making improvements is to identify visibility gaps. Using our previously scored data sources and cross-referencing them with the MITRE ATT&CK data-source suggestions enable us to score our visibility on a scale of 0 (None – No visibility at all) up to 4 (Excellent – All data sources and quality required to see all the aspects of the technique are available).

ASSESSING DETECTION COVERAGE USING DETTECT (1)

The same can be done with your detection coverage if you score each technique's visibility.

RABOBANK-CDC/ DETTECT



SOURCE: https://github.com/rabobank-cdc/DeTTECT

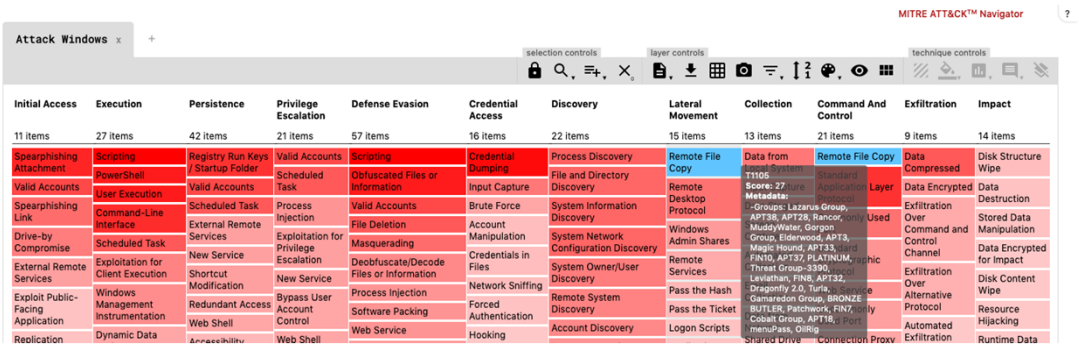
Assessing Detection Coverage Using DeTTECT (1)

Once we have scored our visibility, we can proceed to evaluate our detection capabilities. Scoring detection capabilities is done on a scale from -1 (None) to 0 (Forensics – Events are logged) and up to 5 (Excellent – All known aspects of the technique can be detected in real-time). As for all the previous and future steps, scoring and evaluating is done through YAML files for both readability and usability.

ASSESSING DETECTION COVERAGE USING DETTECT (2)

Given threat actor data, DeTTECT can provide a heat map highlighting the most used techniques.

RABOBANK-CDC/ DETTECT



SOURCE: https://github.com/rabobank-cdc/DeTTECT

Assessing Detection Coverage Using DeTTECT (2)

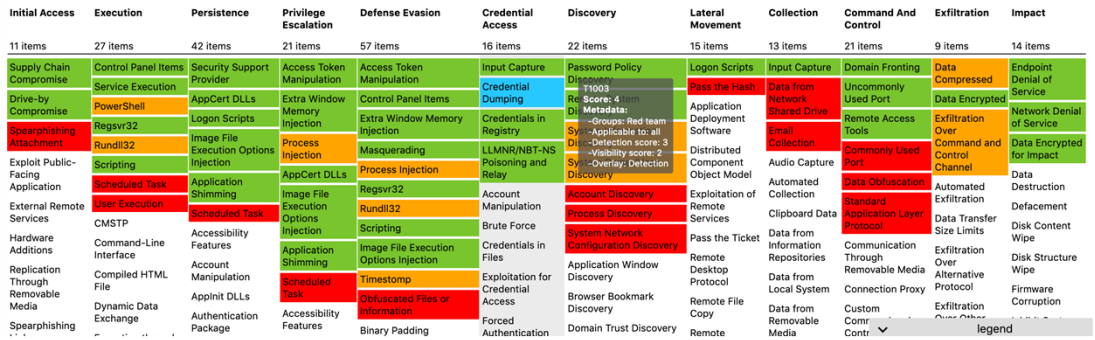
With our evaluated detection capabilities serving as one input to the final DeTTECT objective, a second needed input is the identification of relevant threat actor groups. By identifying relevant threat groups, DeTTECT empowers Blue Teams to identify the most commonly used attacks and desired points of focus.

It should, however, be noted that this kind of information is heavily subject to bias as outlined by MITRE: https://medium.com/mitre-attack/building-an-attack-sightings-ecosystem-b43d52cac151

IDENTIFYING GAPS AND PRIORITIZING THROUGH DETTECT

DeTTECT additionally allows the cross-comparison of threat actor data (MITRE/Red Teams) with the Blue Team's detection capabilities.

RABOBANK-CDC/ DETTECT



SOURCE: https://github.com/rabobank-cdc/DeTTECT

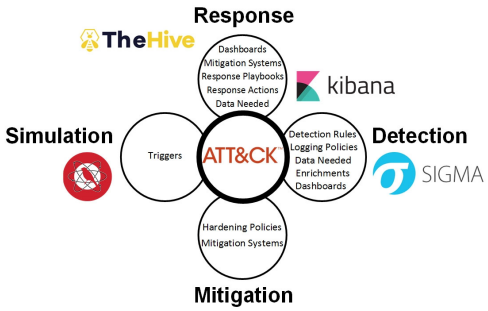
Identifying Gaps and Prioritizing Through DeTTECT

With both our capabilities and threat actors identified, DeTTECT's final feature empowers us to cross-reference both inputs in order to identify gaps in our data sources, visibility or detection capabilities. These gaps can be prioritized based on the likelihood of the equivalent techniques being used against our organization. When fully used, DeTTECT empowers Blue Teams with the needed overview to build roadmaps aimed at orienting their defense efforts.

Using DeTTECT can also be a data-driven approach to provide management with the needed overview and justification for choices made toward further desired or needed improvement as well as a way to report and track Blue Team's evolution.

PUTTING IT ALL TOGETHER: ATOMIC THREAT COVERAGE

"Atomic Threat Coverage is a tool which allows you to automatically generate actionable analytics, designed to combat threats (based on the MITRE ATT&CK adversary model) from Detection, Response, Mitigation and Simulation perspectives."



Atomic Threat Coverage simplifies the representation of SIGMA rules by turning them into human-readable wiki-style pages and other analytics in order to "evangelize" the MITRE ATT&CK framework.

Furthermore, the goal of ATC is to integrate different ATT&CK-mapped initiatives such as SIGMA, TheHive, Atomic Red Team,…

SOURCE: https://github.com/atc-project/atomic-threat-coverage

Putting it All Together: Atomic Threat Coverage
Atomic Threat Coverage is a tool which allows you to automatically generate actionable analytics, designed to combat threats (based on the MITRE ATT&CK adversary model) from Detection, Response, Mitigation and Simulation perspectives.

Atomic Threat Coverage simplifies the representation of sigma rules by turning them into human-readable wiki-style pages and other analytics in order to "evangelize" the MITRE ATT&CK framework. Furthermore, the goal of ATC is to integrate different ATT&CK-mapped initiatives such as SIGMA, TheHive, Atomic Red Team,…

On the project GitHub page, the following use cases are listed:

- Detection Rules based on Sigma — Generic Signature Format for SIEM Systems
- Data Needed to be collected to produce detection of specific Threat
- Logging Policies need to be configured on data source to be able to collect Data Needed
- Enrichments for specific Data Needed which are required for some Detection Rules
- Triggers based on Atomic Red Team — detection tests based on MITRE's ATT&CK
- Response Actions which executed during Incident Response
- Response Playbooks for reacting on specific threat, constructed from atomic Response Actions
- Visualizations for creating Threat Hunting / Triage Dashboards
- Hardening Policies need to be implemented to mitigate specific Threats
- Mitigation Systems need to be deployed and configured to mitigate specific Threats
- Customers of the analytics — could be internal or external. This entity needed for implementation tracking

Full documentation can be found at github.com/atc-project/atomic-threat-coverage.

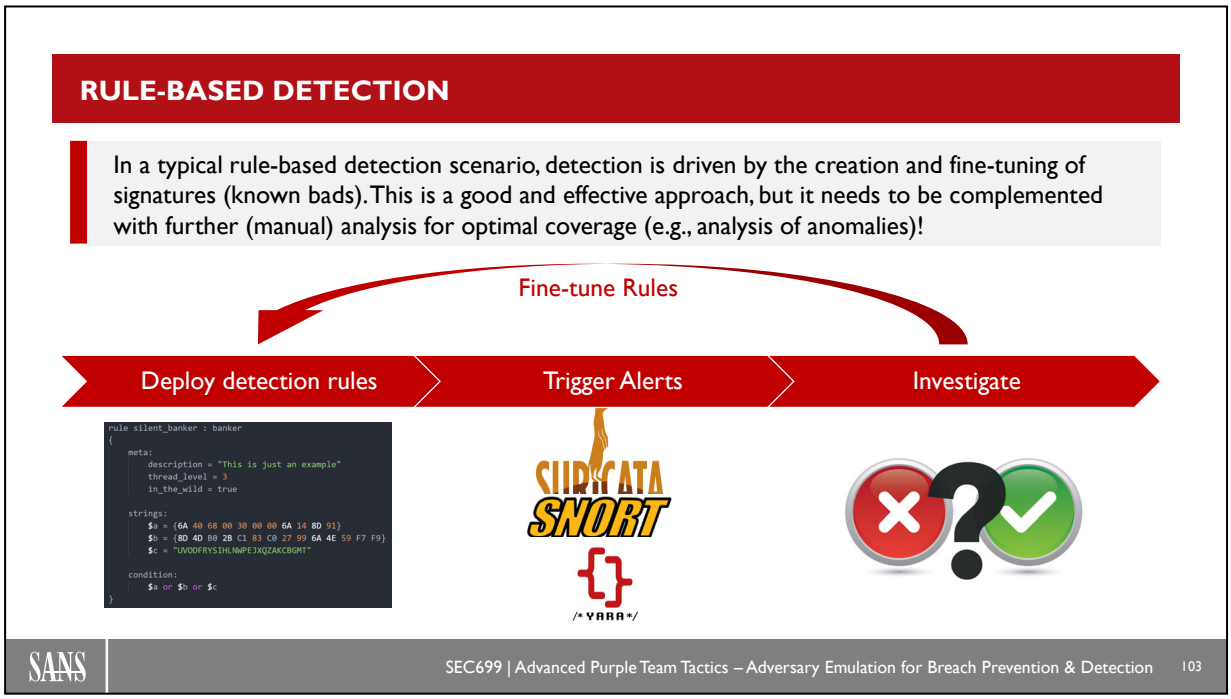
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

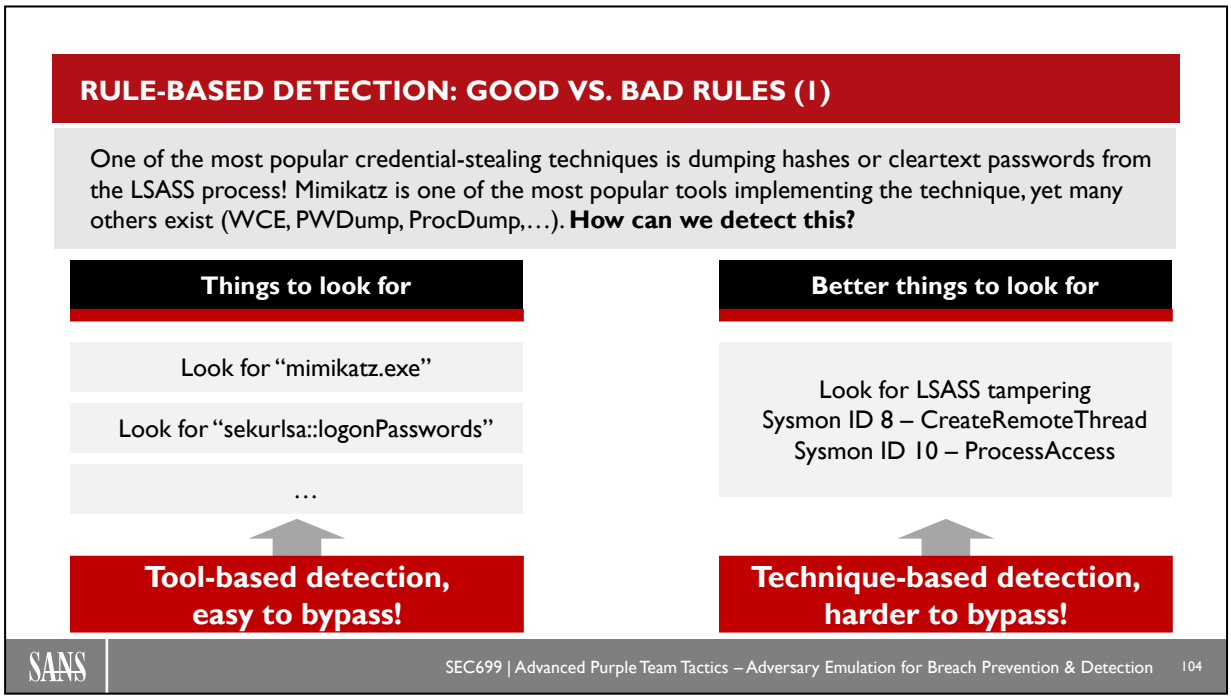
This page intentionally left blank.



Rule-Based Detection
In a typical rule-based detection scenario, detection is driven by the creation and fine-tuning of signatures (known bads). This is a good and effective approach, but it needs to be complemented with further (manual) analysis for optimal coverage (e.g., analysis of anomalies)!

In the slide, we see a typical rule-based detection workflow:

- Detection rules are deployed (YARA rules, IDS signatures, SIEM use cases,…)
- Alerts are triggered by a variety of tools
- The alerts are investigated by analysts, after which the rules are further fine-tuned (if required)



Rule-Based Detection: Good vs. Bad Rules (1)
One of the most popular credential-stealing techniques is dumping hashes or cleartext passwords from the LSASS process! Mimikatz is one of the most popular tools implementing the technique, yet many others exist (WCE, PWDump, ProcDump,…).

So, how could we possibly detect this behavior?

First, let's have a look at a few basic examples of tool-based detection:

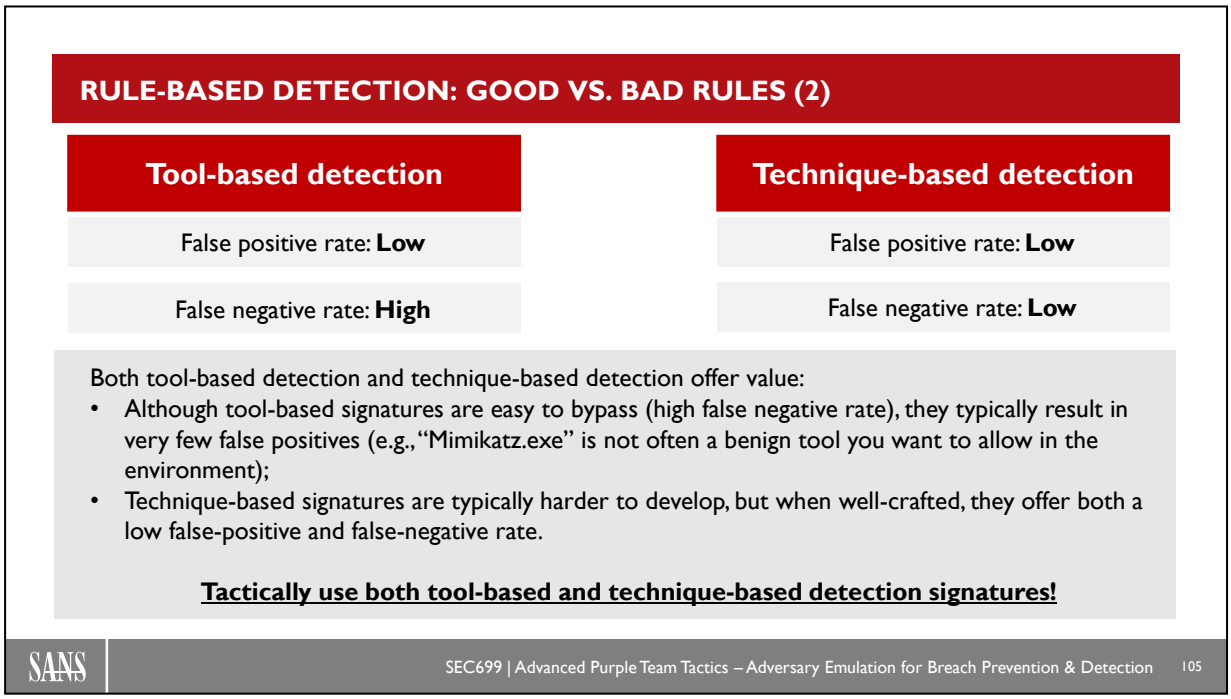
- A very simple, yet naïve, method could be to look for the tool names in process creation logs (event ID 1)
- Additionally, in the same type of logs, we could look for typical command-line arguments

Although there will be few false positives with such rules, there will be plenty of false negatives, as this is trivial to bypass! Microsoft Defender at one point detected the following command-line "notepad.exe privilege::debug sekurlsa::logonPasswords" as malicious, purely based on the command-line arguments!

So, what's a better approach? We could try to attempt detection of the technique as opposed to detecting the tool:

- Sysmon event ID 8 (CreateRemoteThread) – Look for target "lsass.exe" and assess who is attempting to interact with lsass.exe
- Sysmon event ID 10 (ProcessAccess) – Look for target "lsass.exe" and assess who is attempting to interact with lsass.exe

Throughout the week, we will very much focus on technique-based detection versus tool-based detection!



Rule-Based Detection: Good vs. Bad Rules (2)
Does this mean we should only deploy technique-based signatures?

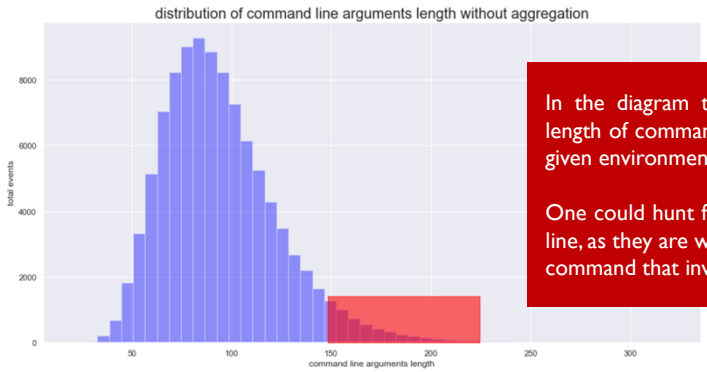
It's not that simple:

- Although tool-based signatures are easy to bypass (high false negative rate), they typically result in very few false positives (e.g., "Mimikatz.exe" is not often a benign tool you want to allow in the environment);
- Technique-based signatures are typically harder to develop, but when well-crafted, they offer both a low false-positive and false-negative rate.

A typical scenario could be that you initially develop a tool-based signature, which you try to evolve over time to include technique-based detections! All in all, a tool-based detection signature is better than no signature at all. We should thus use them both!

ANOMALY-BASED DETECTION

At the other side of the detection spectrum is detection based on anomalies…
Let's have a look at how this could typically work



In the diagram to the left, we have mapped the overall length of command-line arguments across all processes in a given environment.

One could hunt for processes that have a longer command line, as they are worth investigating (e.g., a PowerShell "IEX" command that invokes a long Base64-encoded payload)…

Anomaly-Based Detection
At the other side of the detection spectrum is detection based on anomalies… Let's have a look at how this could typically work. The length of a command line (including full arguments) upon execution of a process can be indicative of malicious activity.

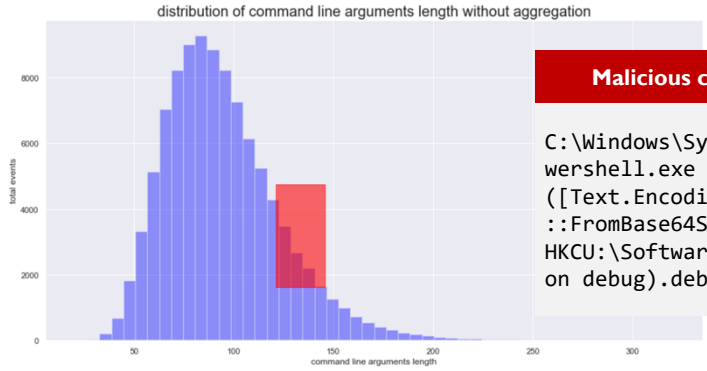
A PowerShell command with a long, encoded payload (this is what PowerShell Empire used to do), is thus something we would like to detect!

In the diagram on the slide, we have mapped the overall length of command-line arguments across all processes in a given environment.

One could hunt for processes that have a longer command line, as they are worth investigating (e.g., a PowerShell "IEX" command that invokes a long Base64-encoded payload)…

ANOMALY-BASED DETECTION: MISSING MALICIOUS ACTIVITY

We might, however, miss out on malicious activity…



Malicious command hidden around the peak:

```
C:\Windows\System32\WindowsPowerShell\v1.0\po
wershell.exe -NonI -W hidden -c "IEX
([Text.Encoding]::UNICODE.GetString([Convert]
::FromBase64String((gp
HKCU:\Software\Microsoft\Windows\CurrentVersi
on debug).debug)))"
```


Anomaly-Based Detection: Missing Mal ious Activity

There's a few risks with such an approach, however, as we might miss malicious activity that doesn't use a statistically longer command-line entry.

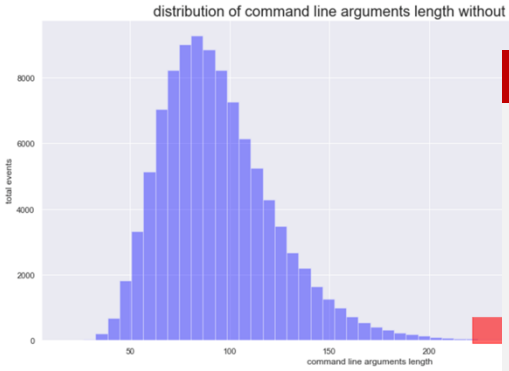
Consider the following example:

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NonI -W hidden -c "IEX ([Text.Encoding]::UNICODE.GetString([Convert]::FromBase64String((gp HKCU:\Software\Microsoft\Windows\CurrentVersion debug).debug)))"

In this example, the adversary has placed a malicious payload in the registry. As the full payload does not have to be referenced on the command line, this would not be considered a statistical outlier.

ANOMALY-BASED DETECTION: FALSE POSITIVES

Or create false positives…



Legitimate command hidden in long tail (Slack.exe):

```
"C:\Users\nviso\AppData\Local\slack\app-3.3.7\slack.exe" --type=renderer --autoplay-policy=no-user-gesture-required --force-color-profile=srgb --no-sandbox --enable-features=FixAltGraph --service-pipe-token=DF5A6C5D08D740B851EF712110E697D1 --lang=en-US --standard-schemes=slack-resources,slack-sounds,slack-webapp-dev --secure-schemes=slack-resources,slack-sounds,slack-webapp-dev --app-user-model-id=com.squirrel.slack.slack --app-path="C:\Users\draman\AppData\Local\slack\app-3.3.7\resources\app.asar" --node-integration=true --webview-tag=false --no-sandbox --preload="C:\Users\draman\AppData\Local\slack\app-3.3.7\resources\app.asar\src\static\index.js" --background-color=#FFFFFF --device-scale-factor=1.75 --num-raster-threads=2 --enable-main-frame-before-activation --enable-compositor-image-animations --service-request-channel-token=DF5A6C5D08D740B851EF712110E697D1 --renderer-client-id=6 --mojo-platform-channel-handle=2208 /prefetch:\\resources\\app.asar\\src\\static\\index.js\"},\"icon\":{},\"title\":\"Slack\",\"autoHideMenuBar\":false,\"show\":false,\"windowType\":\"main\",\"bootstrapScript\":\"C:\\Users\\gherinckx\\AppData\\Local\\slack\\app-3.3.7\\resources\\app.asar\\src\\renderer\\main.tsx\",\"acceptFirstMouse\":true,\"backgroundColor\":\"#FFFFFF\",\"frame\":false,\"minHeight\":300,\"minWidth\":300},\"perfTimer\":{\"mainPid\":12200,\"BOOT\":[300,521113619],\"numTeamsAtLaunch\":1,\"SHELL\":[300,761690119],\"APP_READY\":[302,541762842],\"MAIN_WINDOW_CREATING\":[302,874784110]},\"identifier\":\"slack_preload_metadata_arguments\"}"
```

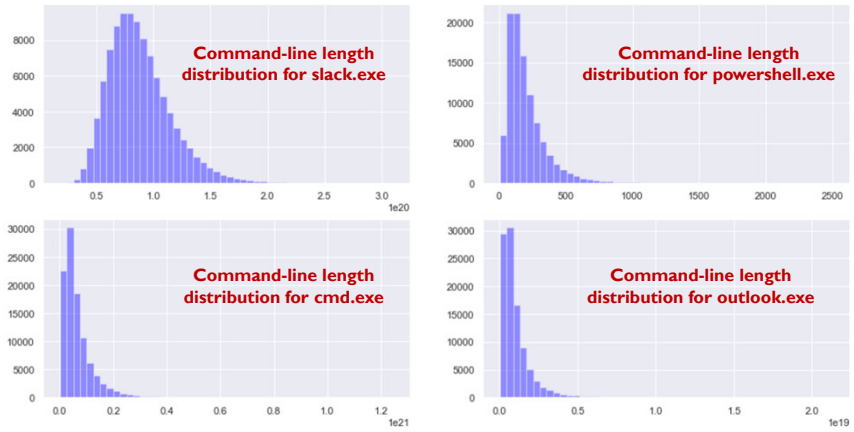

Anomaly-Based Detection: False Positives

On the other hand, there could be long command lines which are perfectly valid. In this example on the slide, we can observe invocation of the popular collaboration software Slack. The full command-line entry is displayed in the slide above.

In our analysis, this would be a statistical outlier, though, and can be found in the long tail of the diagram.

ANOMALY-BASED DETECTION: DATA AGGREGATION

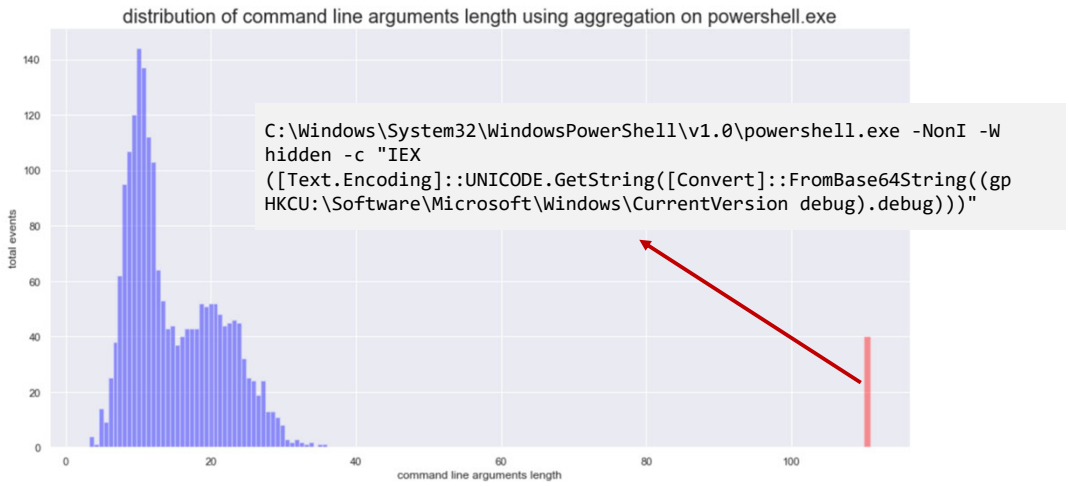
The data should be aggregated based on the process name to maximize detection results



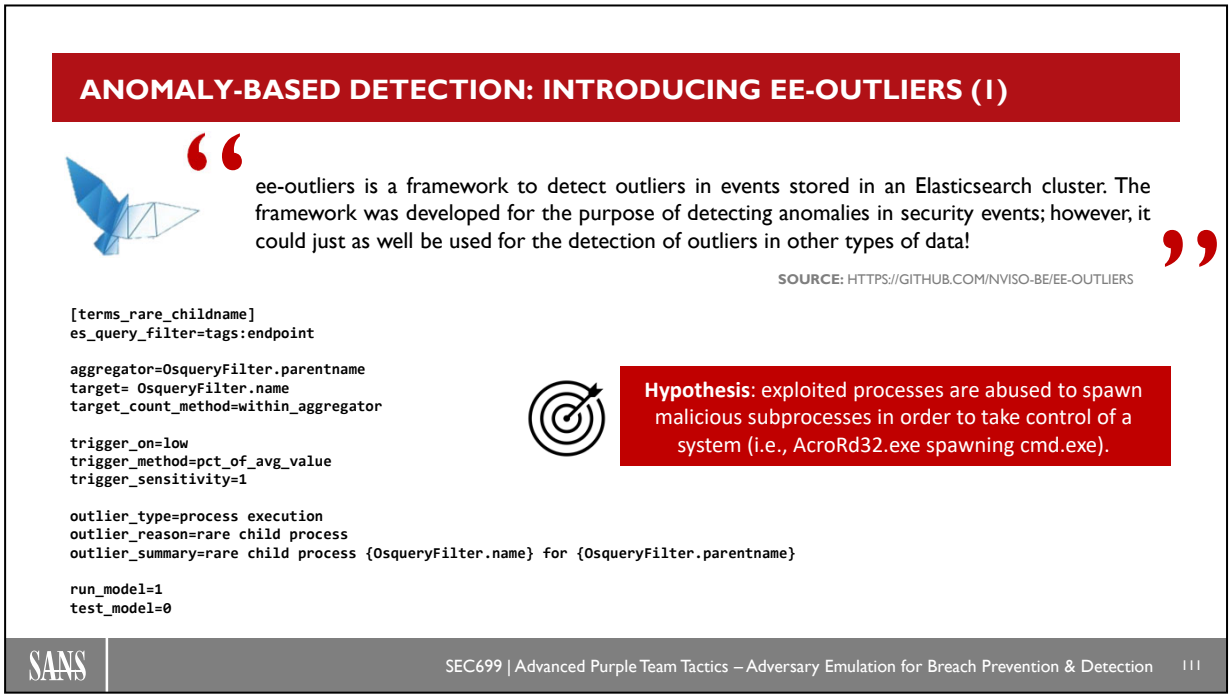
Anomaly-Based Detection: Data Aggregation

A solution to this particular problem is to further aggregate the data and start building models. In the images above, we can see the command-line length distribution aggregated based on the name of the process. Note that there is a distinct difference between the command-line lengths of slack.exe, powershell.exe, cmd.exe, and outlook.exe. This is to be expected: Outlook.exe is, for example, typically not expected to have a lot of command-line arguments. As this can be different in different organizations (e.g., different software installed or different uses of software), this model is a good way of building a diagram of expected values!

ANOMALY-BASED DETECTION: POWERSHELL.EXE ANALYSIS



Anomaly-Based Detection: PowerShell.exe Analysis
Using the new diagram dedicated to Powershell.exe, our payload of +- 130 characters is clearly in the long tail and we can thus easily spot it!



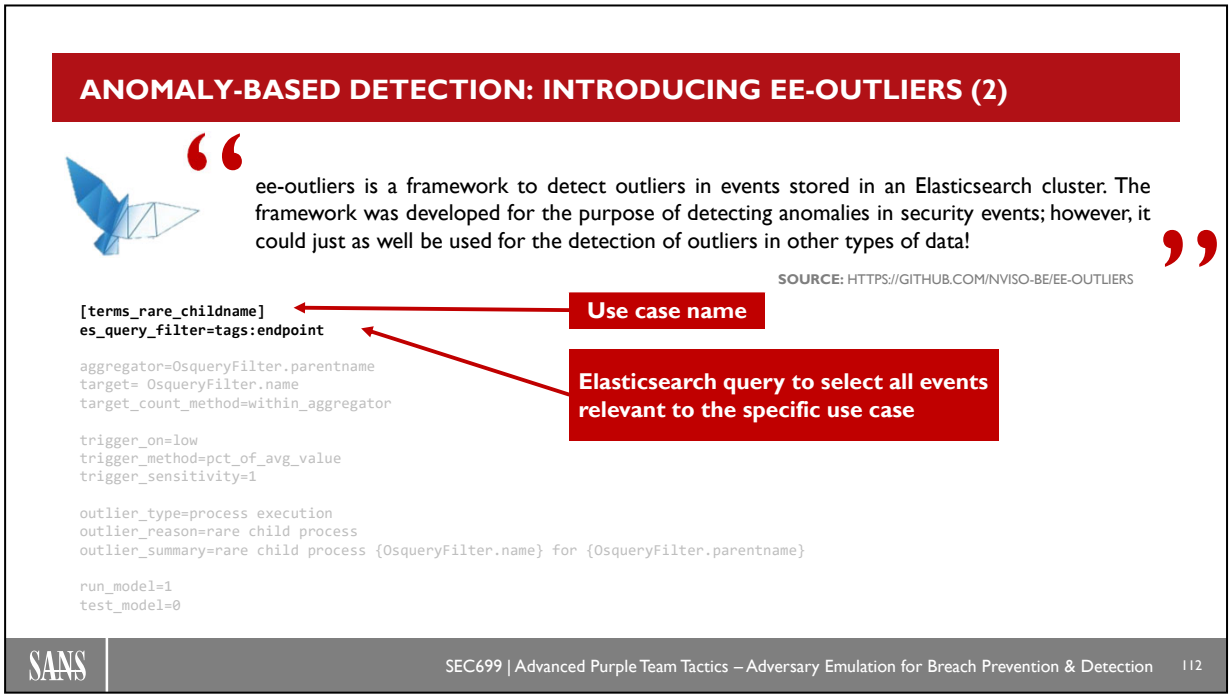
Anomaly-Based Detection: Introducing ee-outliers (1)
ee-outliers was developed by NVISO Labs and is a framework to detect outliers in events stored in an Elasticsearch cluster. The framework was developed for the purpose of detecting anomalies in security events; however, it could just as well be used for the detection of outliers in other types of data!

The framework makes use of statistical models that are easily defined by the user in a configuration file. In case the models detect an outlier, the relevant Elasticsearch events are enriched with additional outlier fields. These fields can then be dashboarded and visualized using the tools of your choice (Kibana or Grafana, for example).

The possibilities of the type of anomalies you can spot using ee-outliers is virtually limitless. A few examples of types of outliers we have detected ourselves using ee-outliers during threat hunting activities include:

- Detect beaconing (DNS, TLS, HTTP, etc.)
- Detect geographical improbable activity
- Detect obfuscated and suspicious command execution
- Detect fileless malware execution
- Detect malicious authentication events
- Detect processes with suspicious outbound connectivity
- Detect malicious persistence mechanisms (scheduled tasks, auto-runs, etc.)
- …

All information can be found on https://github.com/nviso-be/ee-outliers.

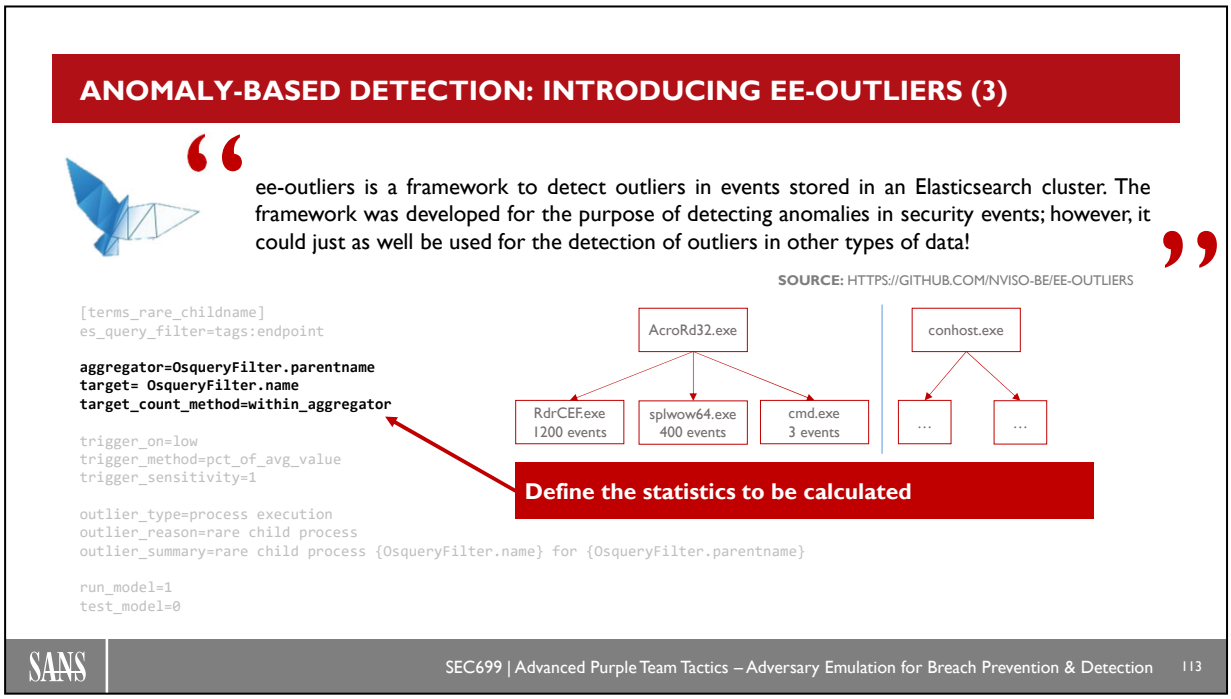


Anomaly-Based Detection: Introducing ee-outliers (2)
Let's assume the following hypothesis: Exploited processes are abused to spawn malicious subprocesses in order to take control of a system (e.g., AcroRd32.exe spawning cmd.exe). How could we detect this using ee-outliers?

We can build a simple use case for this, which we will walk through as an example in the next few slides.

We first define a name for the use case (in this example, "terms_rare_childname"). You can, of course, be as creative as you'd like with the name, but it's probably a good idea to respect a certain naming convention…

The other value is the Elasticsearch query that will be used to select the events that are to be evaluated by the use case. In this example, we have added a tag "endpoint" to all endpoint logs, which we believe to be relevant for our use case.



Anomaly-Based Detection: Introducing ee-outliers (3)

As a next step, we need to define what the "aggregator" will be, plus what field we are looking to analyze. In our current example, we are using OSQuery to periodically collect a list of running processes. We will thus use the following configuration:

- The aggregator is "OsqueryFilter.parentname". This is a field name in Elastic, used for the parent process name.
- The target is "OsqueryFilter.name". This is a field name in Elastic, used for the process name.
- The target_count_method is "within_aggregator", as we are looking for process names that are spawned by the parent process name.

ANOMALY-BASED DETECTION: INTRODUCING EE-OUTLIERS (4)

> ee-outliers is a framework to detect outliers in events stored in an Elasticsearch cluster. The framework was developed for the purpose of detecting anomalies in security events; however, it could just as well be used for the detection of outliers in other types of data!

SOURCE: HTTPS://GITHUB.COM/NVISO-BE/EE-OUTLIERS

```
[terms_rare_childname]
es_query_filter=tags:endpoint

aggregator=OsqueryFilter.parentname
target= OsqueryFilter.name
target_count_method=within_aggregator

trigger_on=low
trigger_method=pct_of_avg_value
trigger_sensitivity=1

outlier_type=process execution
outlier_reason=rare child process
outlier_summary=rare child process {OsqueryFilter.name} for {OsqueryFilter.parentname}

run_model=1
test_model=0
```

Define which data is considered an outlier

Anomaly-Based Detection: Introducing ee-outliers (4)

We also need to define when we consider a value to be an outlier. In our example, we have configured the following values:

- The "trigger_on" is set to low, as we are looking for child process names that are uncommon for this parent process name.
- The "trigger_method" is simply set to a percentage (pct_of_avg_value).
- The "trigger_sensitivity" is set to 1, indicating an alert will be raised when the child process occurs in less than 1% of the average count.

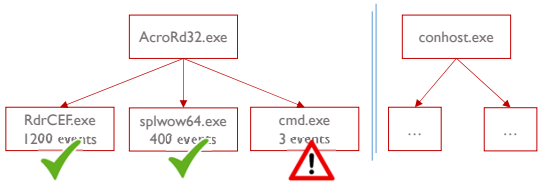
During production use, these values might, of course, need to be refined.

ANOMALY-BASED DETECTION: INTRODUCING EE-OUTLIERS (5)

> ee-outliers is a framework to detect outliers in events stored in an Elasticsearch cluster. The framework was developed for the purpose of detecting anomalies in security events; however, it could just as well be used for the detection of outliers in other types of data!

SOURCE: HTTPS://GITHUB.COM/NVISO-BE/EE-OUTLIERS

```
[terms_rare_childname]
es_query_filter=tags:endpoint

aggregator=OsqueryFilter.parentname
target= OsqueryFilter.name
target_count_method=within_aggregator

trigger_on=low
trigger_method=pct_of_avg_value
trigger_sensitivity=1

outlier_type=process execution
outlier_reason=rare child process
outlier_summary=rare child process {OsqueryFilter.name} for {OsqueryFilter.parentname}

run_model=1
test_model=0
```

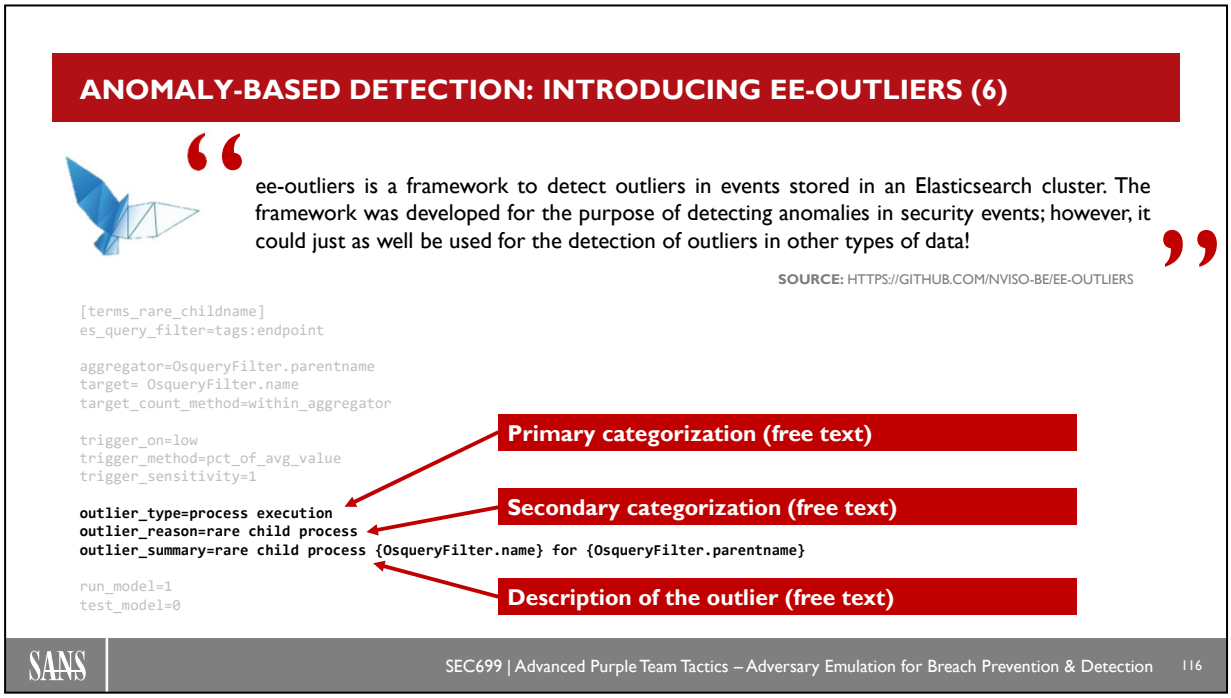


Avg(1200, 400, 3) = 534
534 x 1% = 5 (trigger threshold)
3 (cmd.exe) < 5 = OUTLIER!

Anomaly-Based Detection: Introducing ee-outliers (5)
As described on the previous slide, we can see the actual numbers here, indicating that cmd.exe will be triggered as an outlier!

The full reasoning is as follows:

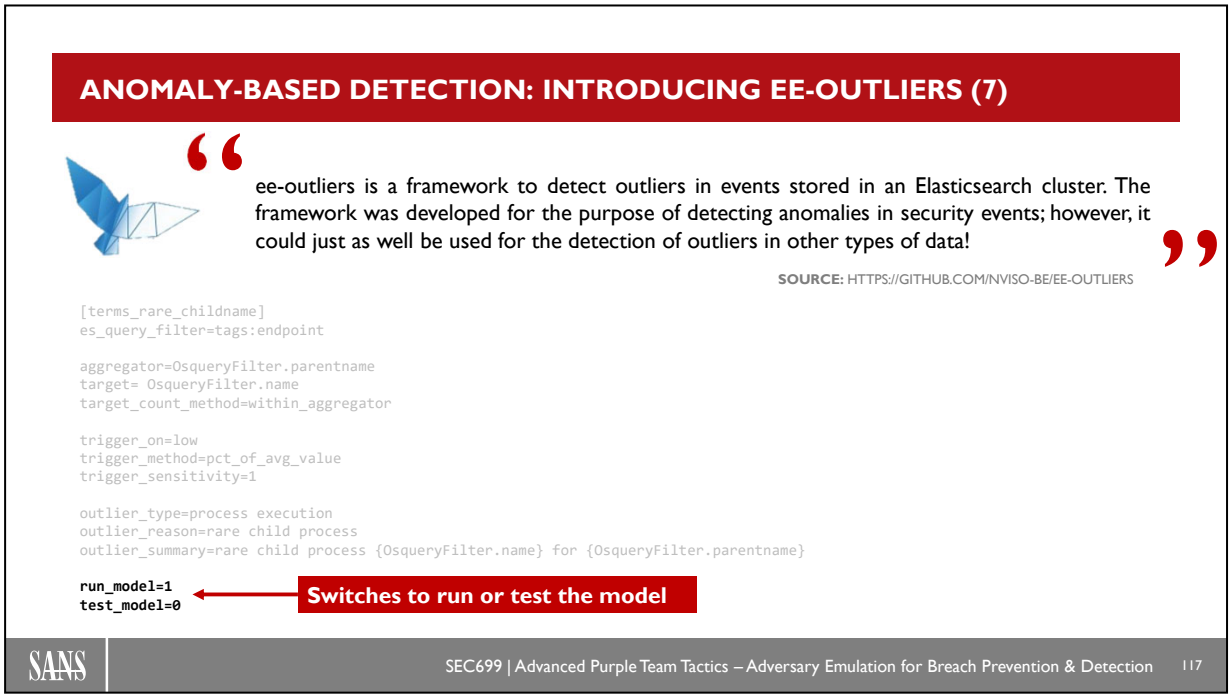
- In our example, there are a total of 1603 events (1200, 400 and 3).
- Thus, the average is 534.
- 1% of 534 is 5, which will be our trigger threshold.
- As the value for cmd.exe is 3 (below the threshold), an outlier will be detected!



Anomaly-Based Detection: Introducing ee-outliers (6)

Finally, we need to also configure the way ee-outliers will generate results. We can do this using the following fields:

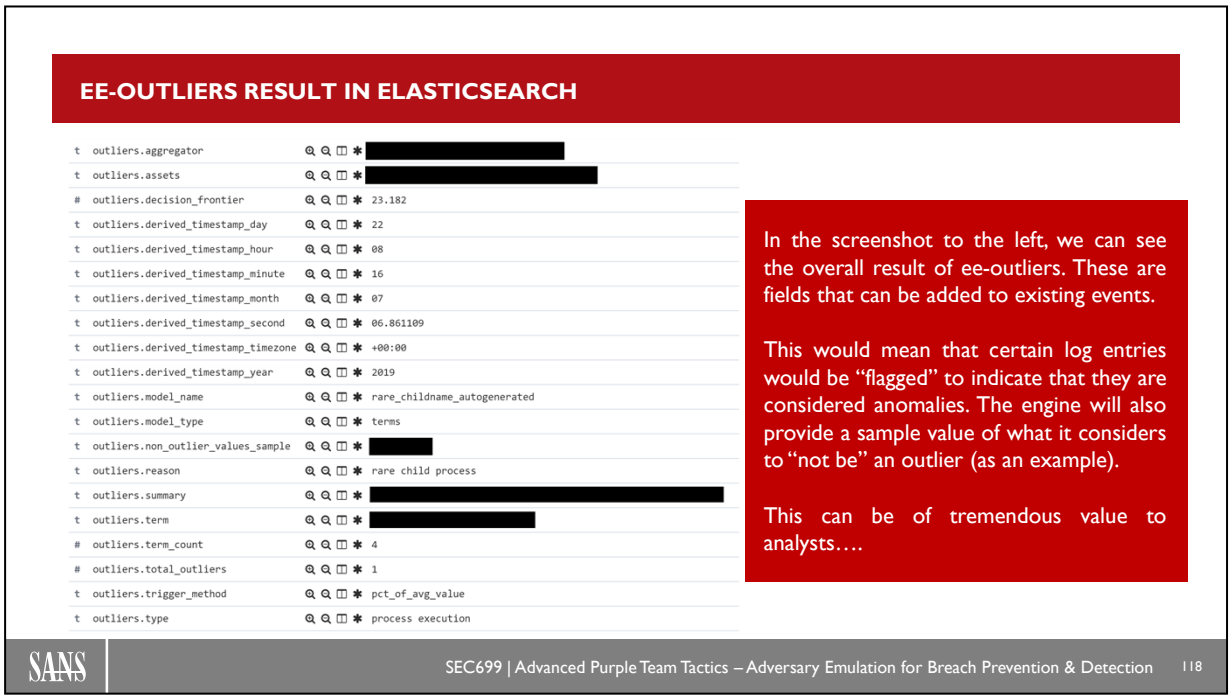
- Outlier_type can be used to configure a category of outliers (in this case, we opted for "process execution").
- Outlier_reason can be used to add a "short" reason for the outlier to be raised.
- Outlier_summary has additional details on the actual outlier (including variables).



Anomaly-Based Detection: Introducing ee-outliers (7)
At the end of the configuration are two switches that can be used to either run or test the model.

Additional example of how ee-outliers can be used can be found here:

https://blog.nviso.eu/2018/12/21/detecting-suspicious-child-processes-using-ee-outliers-and-elasticsearch/
https://blog.nviso.eu/2018/12/11/tls-beaconing-detection-using-ee-outliers-and-elasticsearch/



EE-Outliers Result in Elasticsearch

In the screenshot, we can see the overall result of ee-outliers. These are fields that can be added to existing events. This would mean that certain log entries would be "flagged" to indicate that they are considered anomalies. The engine will also provide a sample value of what it considers to "not be" an outlier (as an example). This can be of tremendous value to analysts….

Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

EXERCISE: PREPARING OUR ELASTIC AND SIGMA STACK

Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

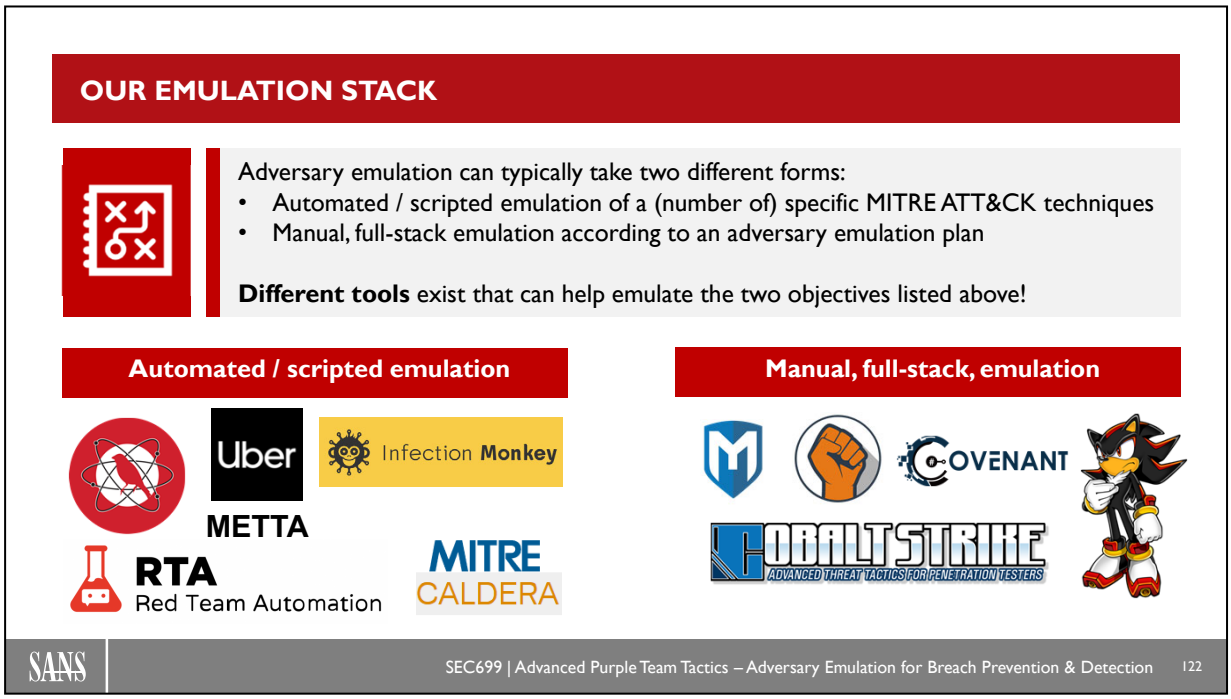
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.



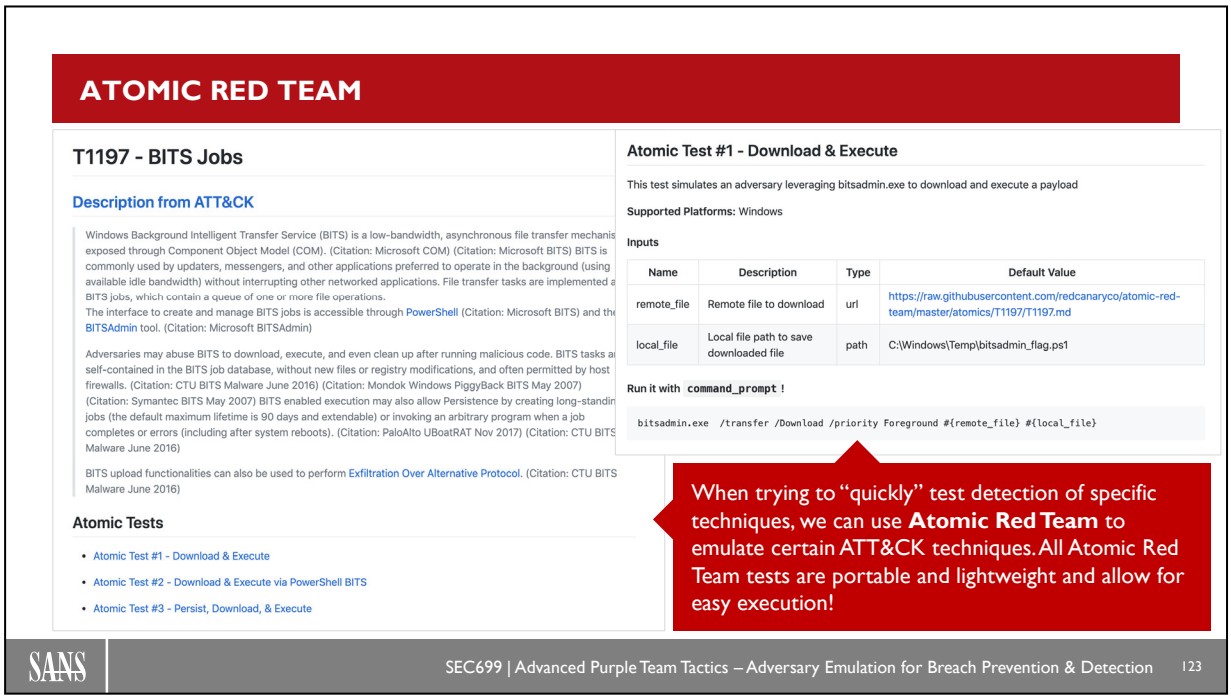
Our Emulation Stack

Now that we've discussed a number of key tools that are required for proper monitoring and detection of adversary techniques, let's zoom in on some toolkits available for emulation of said techniques.

Adversary emulation can typically take two different forms:

- Automated / scripted emulation of (a number of) specific MITRE ATT&CK techniques. Typical tools / initiatives include Atomic Red Team, Uber Metta, Infection Monkey (closed source), RTA (Red Team Automation), and MITRE Caldera.
- Manual, full-stack emulation according to an adversary emulation plan. Typical tools that fall in this category include Metasploit (although a bit more pen test-focused), Faction C2, Covenant, and Cobalt Strike (commercial).

Both of the approaches can have significate added value for organizations and we will walk through some of these tools throughout this section. We will also select a number of tools that will be used throughout the rest of the week.



Atomic Red Team
Atomic Red Team is a collection of atomic tests mapped to the MITRE ATT&CK framework. These tests are sample command lines that can be easily executed to test the success / detection of a specific ATT&CK technique.

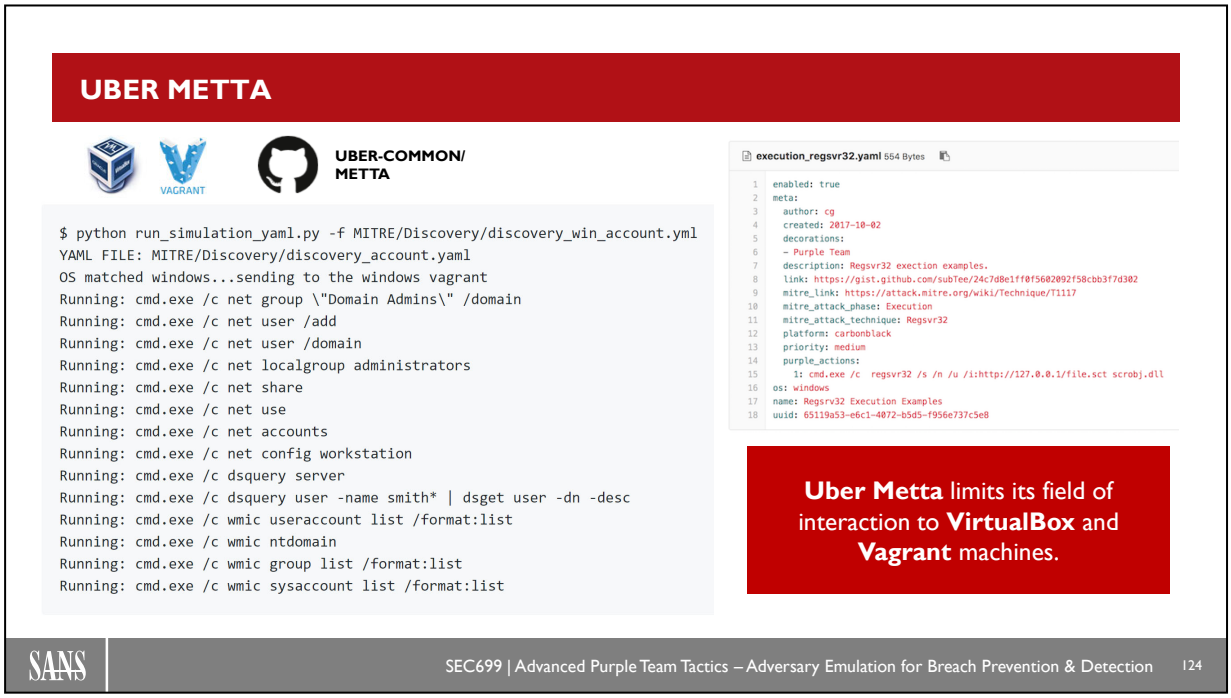
We can find three key principles in the official Atomic Red Team documentation (extracted from https://atomicredteam.io/):

1. Teams need to be able to test everything from specific technical controls to outcomes.
   Security teams do not want to operate with a "hopes and prayers" attitude toward detection. We need to know what our controls and program can detect, and what they cannot. We don't have to detect every adversary, but we do need to believe in knowing our blind spots.

2. We should be able to run a test in less than five minutes.
   Most security tests and automation tools take a tremendous amount of time to install, configure, and execute. We coined the term "atomic tests" because we felt there was a simple way to decompose tests so most could be run in a few minutes.
   The best test is the one you actually run.

3. We need to keep learning how adversaries are operating.
   Most security teams don't have the benefit of seeing a wide variety of adversary types and techniques crossing their networks every day. Even at Red Canary, we only come across a fraction of the possible techniques being used, which makes the community working together essential to making us all better.

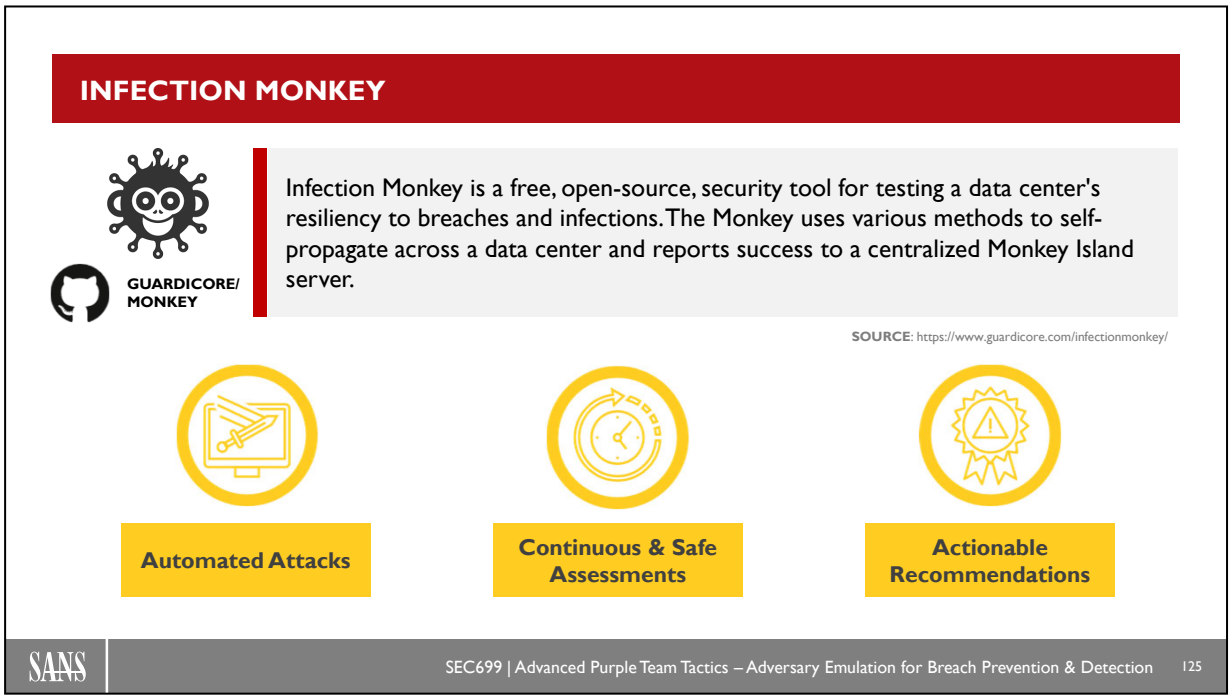
The repository is maintained at https://github.com/redcanaryco/atomic-red-team, while the project homepage is https://atomicredteam.io/.



Uber Metta
Metta was developed by UBER as an "information security preparedness tool." Metta uses Redis/Celery, Python, and vagrant with VirtualBox to do adversarial simulation. Actions are defined in YAML files.
In the screenshots above, we can see:

- The output of Metta, when it's running the "discovery_win_account.yml" file
- An example YAML file (execution_regsvr32.yaml) that is used to test an application whitelisting bypass is included as well

Based upon the above, it should be clear that Metta is highly customizable, as the YAML files are easy to adapt / fine-tune.

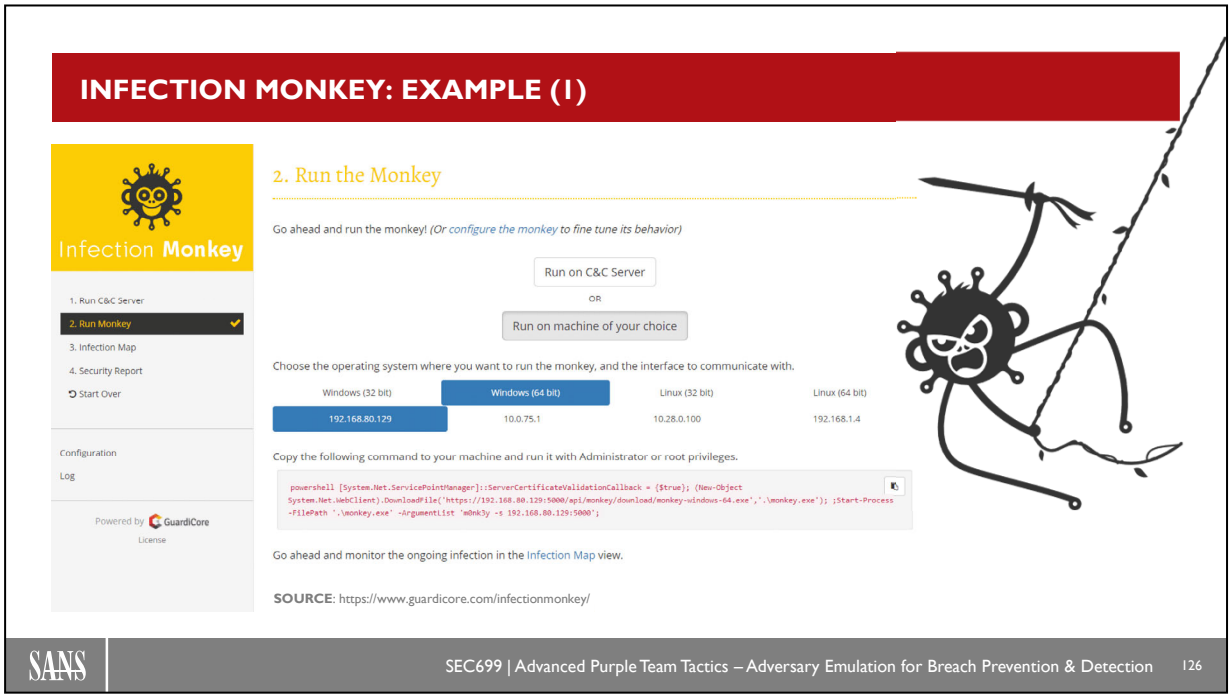


Infection Monkey
Infection Monkey is a free, open-source, security tool for testing a data center's resiliency to breaches and infections. The Monkey uses various methods to self-propagate across a data center and reports success to a centralized Monkey Island server. Infection Monkey has three focus areas:

- Automated attacks
- Continuous and safe assessments
- Actionable recommendations

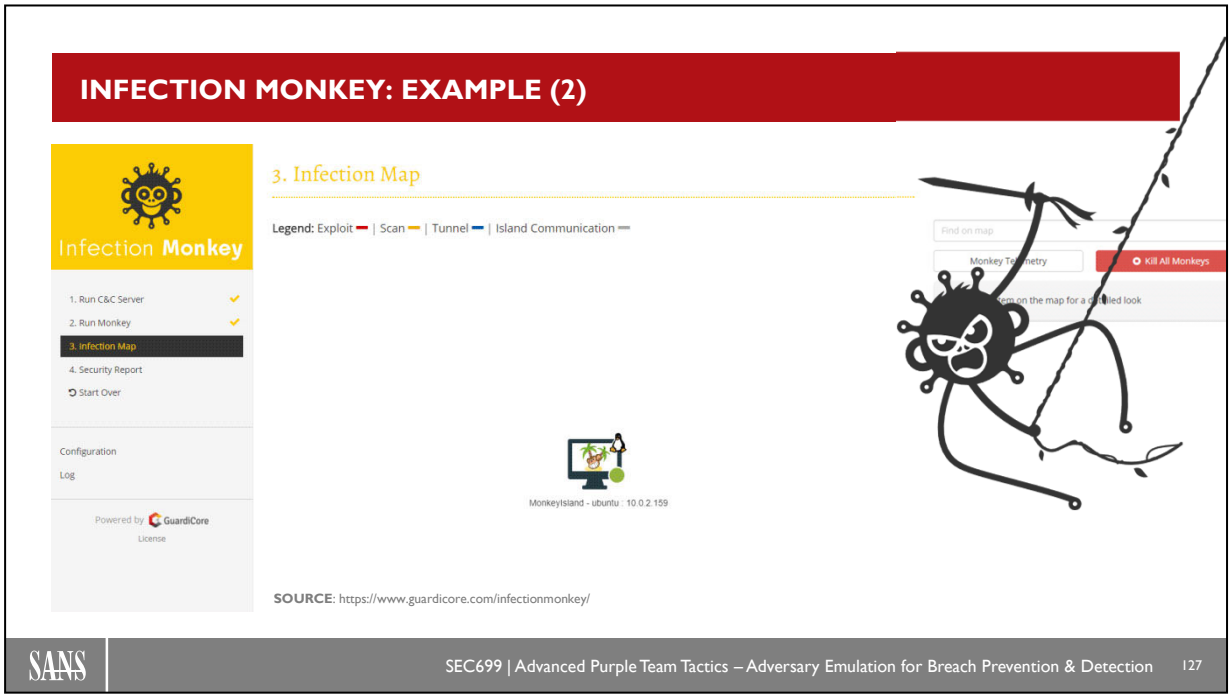
More information on Infection Monkey can be found at:

https://github.com/guardicore/monkey
https://www.guardicore.com/infectionmonkey/



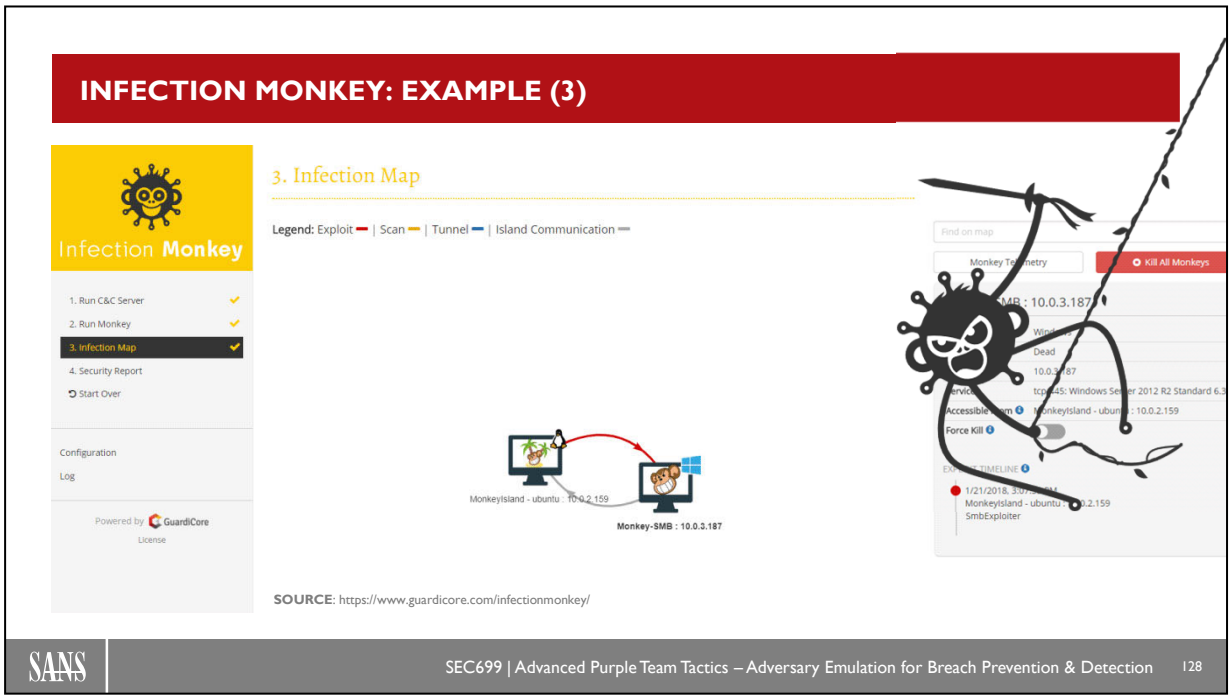
Infection Monkey: Example (1)

To give you an idea on the overall look and feel of Infection Monkey, let's have a look at an example scenario. In the above screenshot, we can observe how the Infection Monkey can be invoked on a Windows system. As can be seen on the screenshot, it has support for both 32-bit and 64-bit Windows or Linux systems. The example above uses PowerShell to download and execute the "monkey-windows-64.exe" executable. This is similar to other platforms such as Caldera. It's important to note that Infection Monkey is less customizable and will just opportunistically search for vulnerabilities it can abuse (it's thus not limited to a predefined number of ATT&CK techniques).



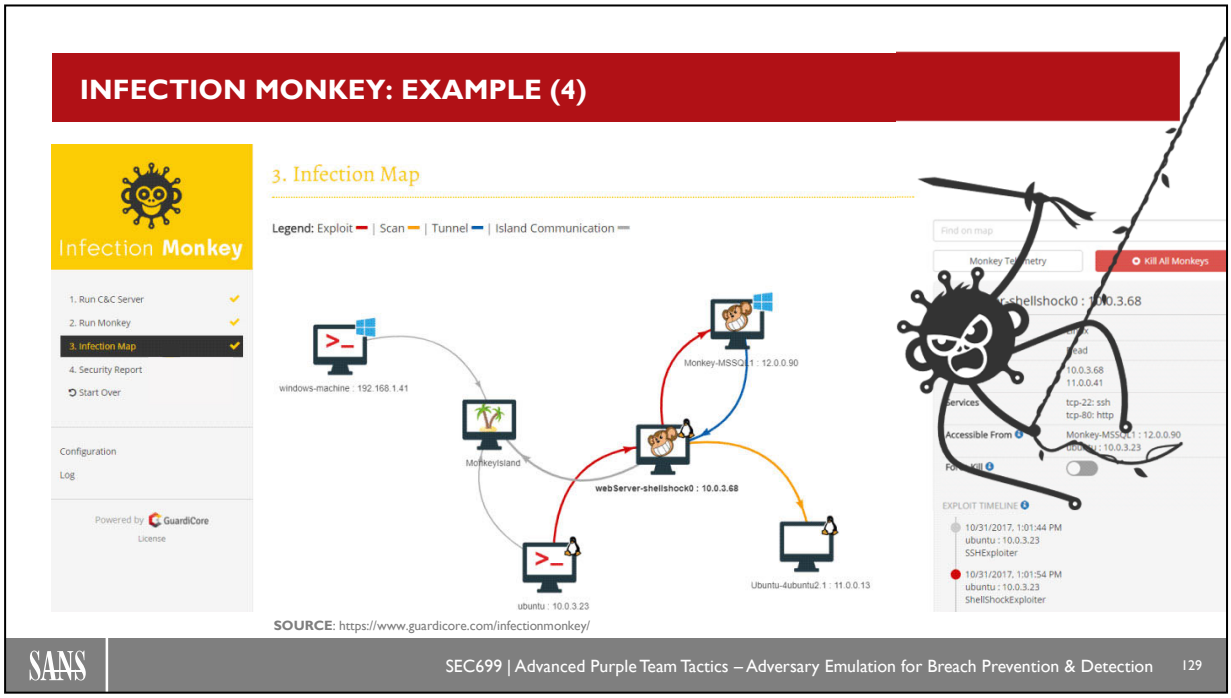
Infection Monkey: Example (2)
In the screenshot above, we can see the "infection map", which shows all currently infected / compromised systems. We only see one system currently, which is the central system configured as "MonkeyIsland"; it's hosted on an Ubuntu machine with IP address 10.0.2.159.

We will now start infecting other machines!



Infection Monkey: Example (3)

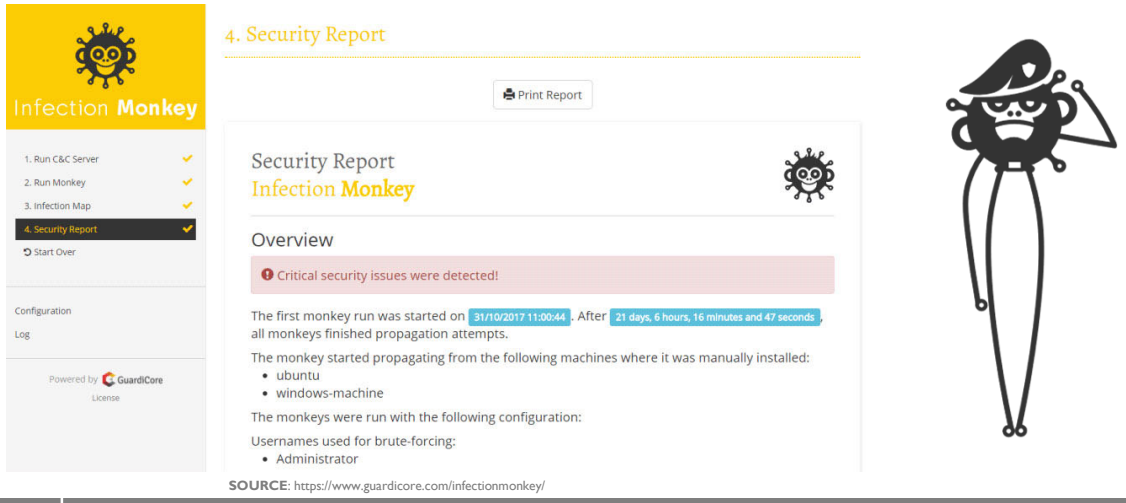
In the screenshot above, we can see that Infection Monkey has managed to compromise a second system (10.0.3.187) using an SMB Exploit. In the slide, we can see that it's running Windows Server 2012 R2. Most likely, this means a vulnerable version of SMB was running (e.g., SMBv1), which could be abused by Infection Monkey. The exploit is symbolized by the red arrow, while we also see a gray arrow that indicates a Command & Control channel toward the Monkey Island.



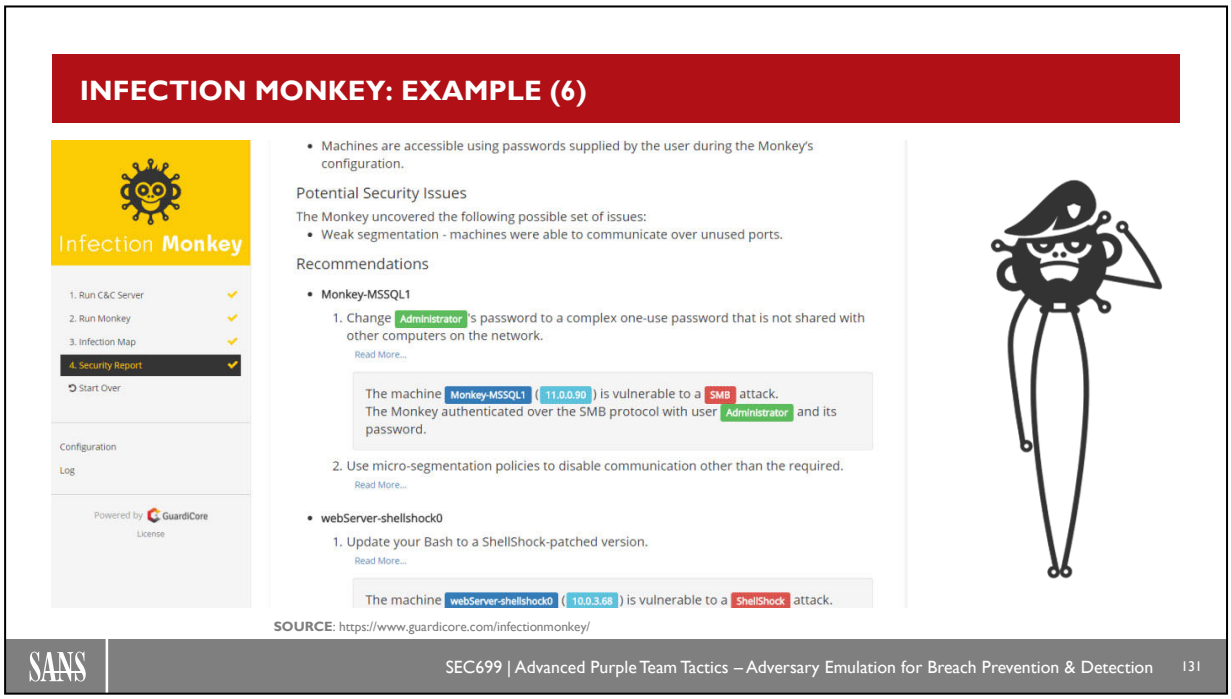
Infection Monkey: Example (4)
In the screenshot above, we see another situation:

- "Infection Monkey" was manually executed on 192.168.1.41 (windows-machine) and 10.0.3.23 (ubuntu). We can deduce this as there is only a gray arrow from the system toward Monkey Island (for C&C), there is no exploit or scan in the other direction.
- The machine with IP address 10.0.3.68 (webserver-shellshock0) was exploited by 10.0.3.23 (ubuntu). It appears the ShellShock vulnerability was used for this.
- From 10.0.3.58 (webserver-shellshock0), the following pivots took place:
  - A scan was launched against 11.0.0.13 (Ubuntu-4ubuntu2.1), without successful exploitation
  - 12.0.0.90 (Monkey-MSSQL1) was successfully exploited and is connecting back to MonkeyIsland through webserver-shellshock0 (blue arrow indicates tunnel)

INFECTION MONKEY: EXAMPLE (5)



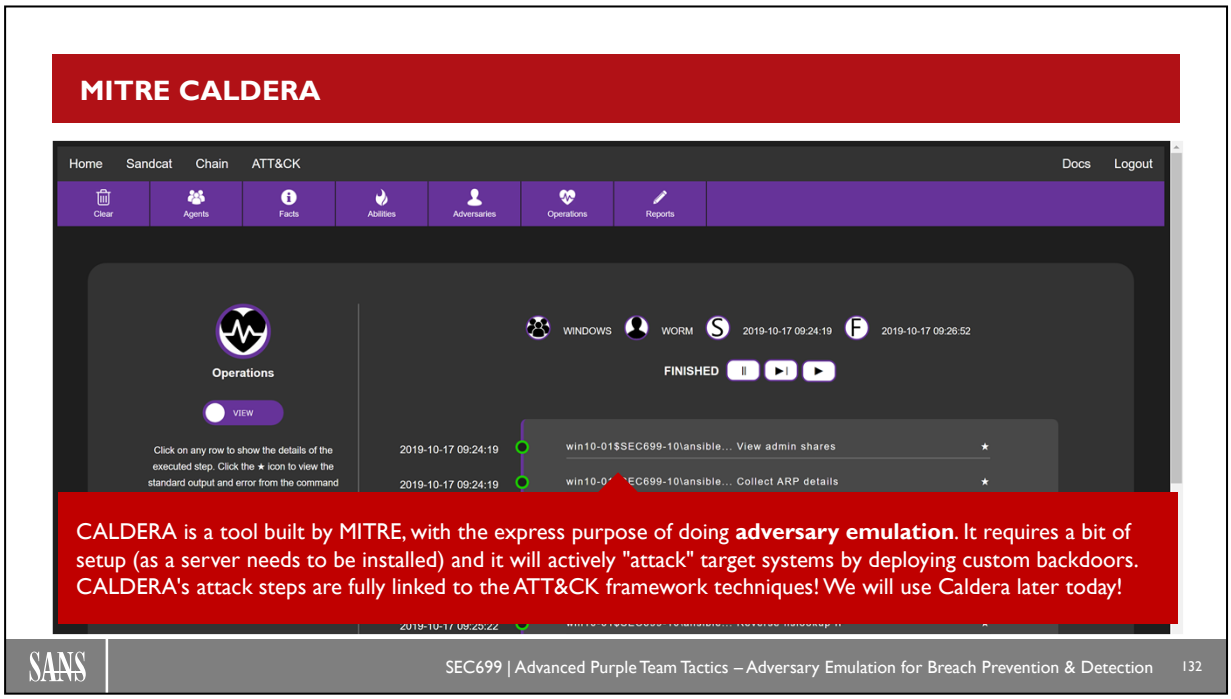
Infection Monkey: Example (5)
Once Infection Monkey has finished running through the environment, it will generate a security report. This security report includes some basic information on the work that was done (from what machines did it start, what usernames were used,…).



Infection Monkey: Example (6)
The final report also includes security recommendations that can help further improve the overall security level of the company.

In the example above, you can see that:

- The Monkey was able to obtain access to Monkey-MSSQL1 by password guessing the Administrator account over SMB. The recommendations are thus to (1) change the password and (2) implement network segmentation to only allow network protocols that should explicitly be allowed.
- The Monkey was able to obtain access to webserver-shellshock0 by abusing the ShellShock vulnerability. The recommendation is thus to update the version of Bash to a non-vulnerable one.

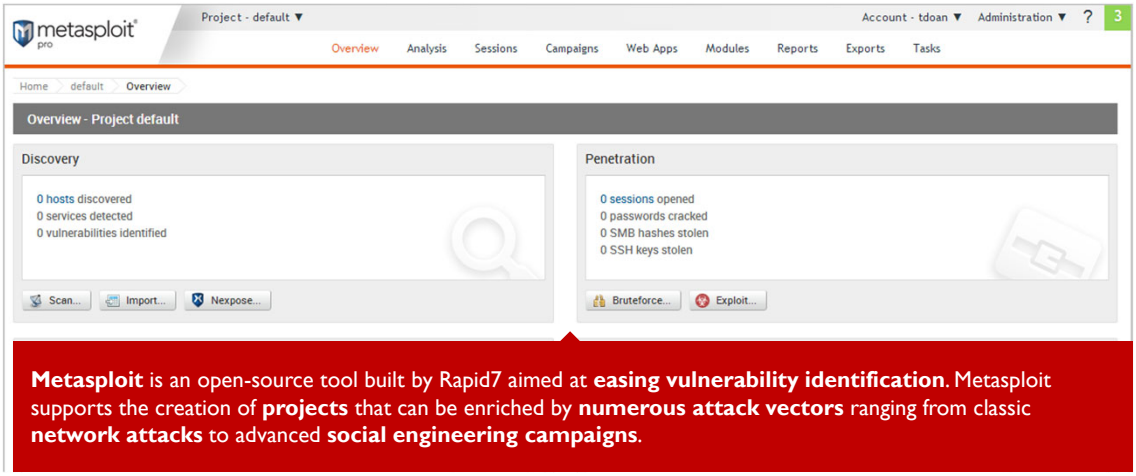


MITRE Caldera

CALDERA is a tool built by MITRE, with the express purpose of doing adversary emulation. It requires a bit of setup (as a server needs to be installed) and it will actively "attack" target systems by deploying custom backdoors. CALDERA's attack steps are fully linked to the ATT&CK framework techniques! CALDERA doesn't currently cover all ATT&CK techniques, but it's a work in progress. MITRE is continuously improving the overall platform, while there are also community efforts to develop and implement additional techniques.

CALDERA will be used as a basis for automated adversary emulation throughout SEC699 (we will write our own modules, techniques,…)! A more detailed section on CALDERA will follow today!

METASPLOIT

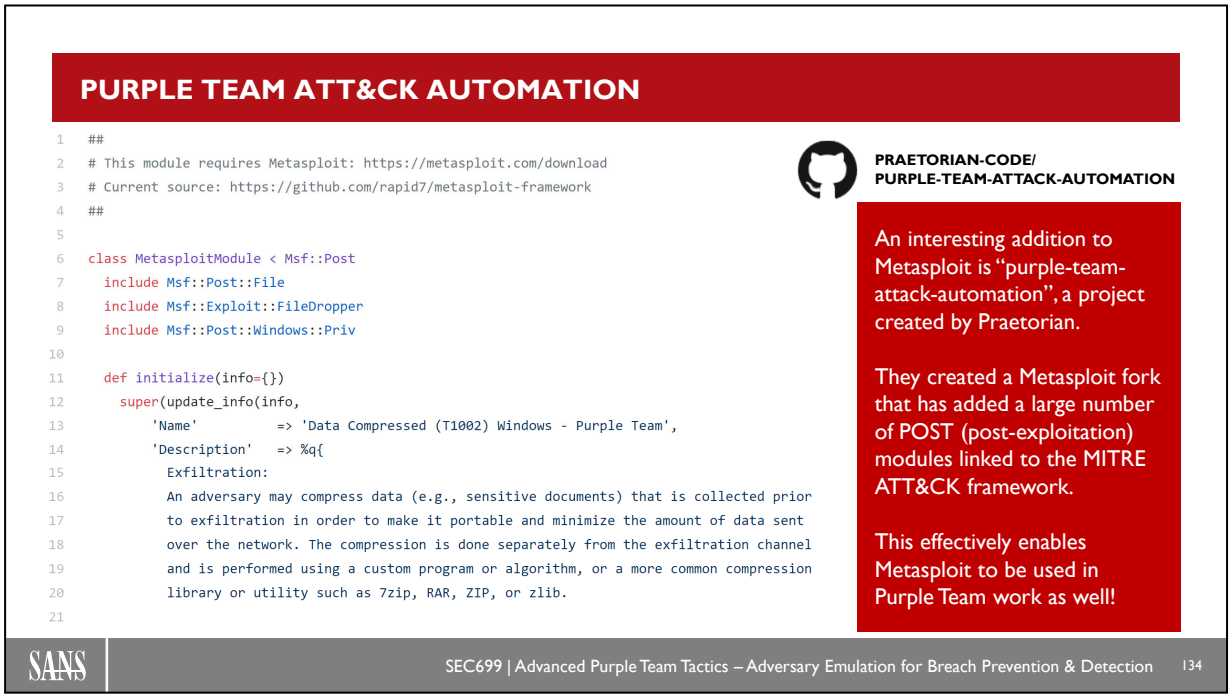


Metasploit is an open-source tool built by Rapid7 aimed at **easing vulnerability identification**. Metasploit supports the creation of **projects** that can be enriched by **numerous attack vectors** ranging from classic **network attacks** to advanced **social engineering campaigns**.

SOURCE: https://docs.rapid7.com/metasploit/

Metasploit
Metasploit has been around for a long time and has become one of the de facto penetration testing tools, used by Red Teamers and actual adversaries alike. Initially built by HD Moore, it's now being maintained as an open-source tool by Rapid7 (note that they also offer a commercial version called Metasploit Pro, which includes a fancy GUI). Metasploit is focused on "standardization" of exploitation techniques, where exploits and payloads can be easily combined during an engagement. Metasploit does not have a strong mapping to MITRE ATT&CK by default and is thus less suited for Purple Team operations. Also note that it doesn't focus on stealth operations, but is more of a penetration testing tool aimed at looking for vulnerabilities.

Additional information can be found here:
https://www.metasploit.com/
https://github.com/rapid7/metasploit-framework

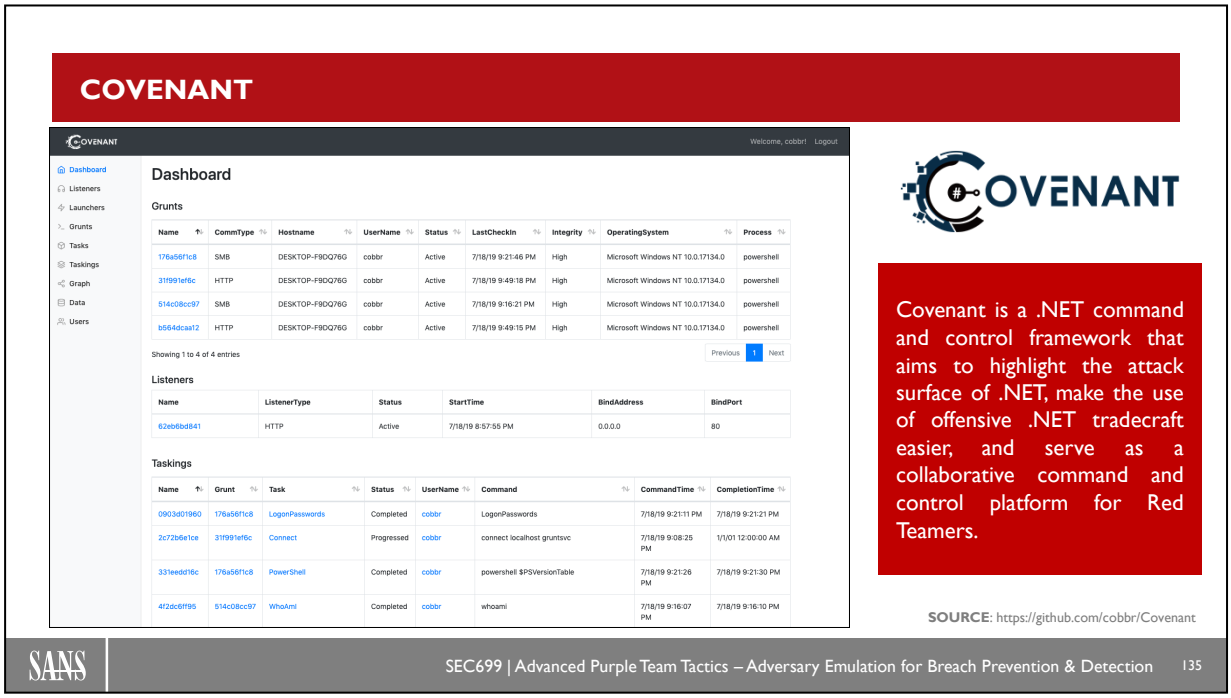


Purple Team ATT&CK Automation
An interesting addition to Metasploit is "purple-team-attack-automation", a project created by Praetorian. They created a Metasploit fork that has added a large number of POST (post-exploitation) modules linked to the MITRE ATT&CK framework. This effectively enables Metasploit to be used in Purple Team work as well!

From Praetorian's GitHub page:

"At Praetorian, we were seeking a way to automatically emulate adversary tactics in order to evaluate detection and response capabilities. Our solution implements MITRE ATT&CK™ TTPs as Metasploit Framework post modules. As of this release, we've automated a little over 100 TTPs as modules. Metasploit's advantage is its robust library, capability to interact with operating system APIs, and its flexible license. In addition, we're able to emulate the features of other tools such as in-memory .NET execution via leveraging Metasploit's execute_powershell functionality. This allows Blue Teams to ensure that their tools are alerting on the actual TTP behavior and not execution artifacts (such as encoded PowerShell)."

Additional information and documentation can be found here:
https://github.com/praetorian-inc/purple-team-attack-automation



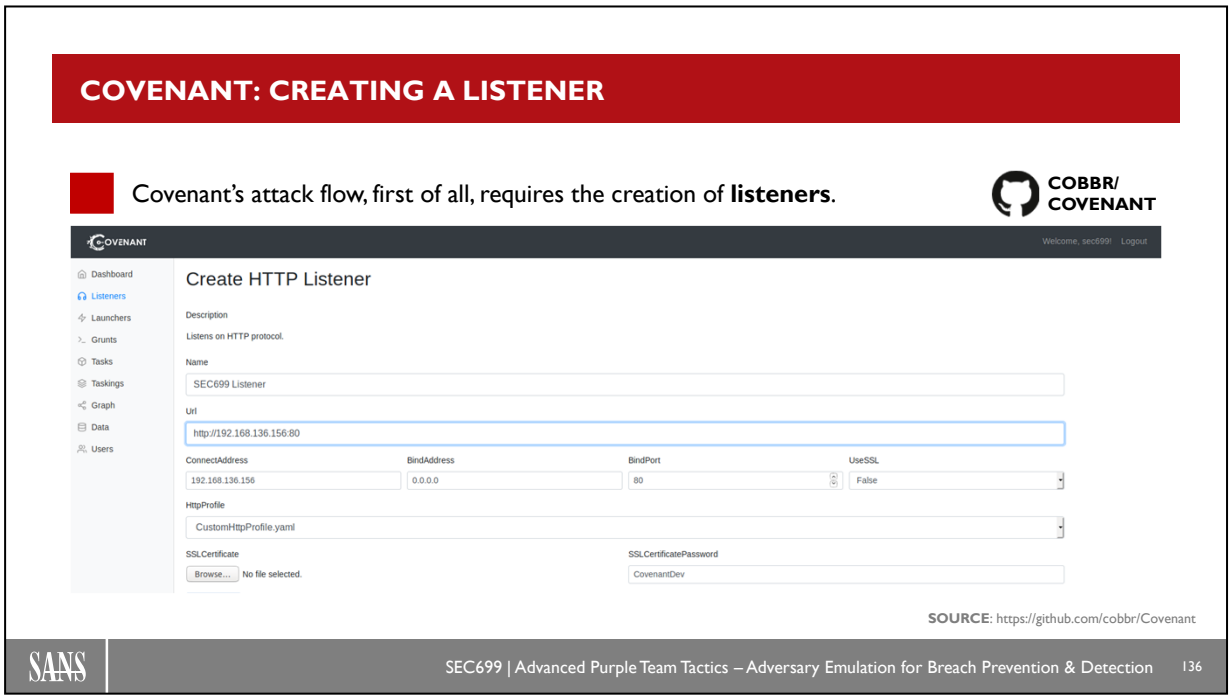
Covenant
Covenant is a .NET command and control framework that aims to highlight the attack surface of .NET, make the use of offensive .NET tradecraft easier, and serve as a collaborative command and control platform for Red Teamers. It's developed by Ryan Cobb (SpecterOps) and is a natural follow-up after PowerShell Empire was deprecated (due to an increased number of security controls in PowerShell).

In the screenshot above, we can see the central Covenant dashboard, which covers:

- Grunts: Grunts are compromised systems
- Listeners: Listeners are listeners (☺) that are handling connections coming in from the grunts
- Taskings: Taskings are tasks that are assigned to grunts

We will use Covenant during an upcoming lab! The full documentation and tool can be found at https://github.com/cobbr/Covenant.

We will quickly run through the setup of Covenant in the next few slides!

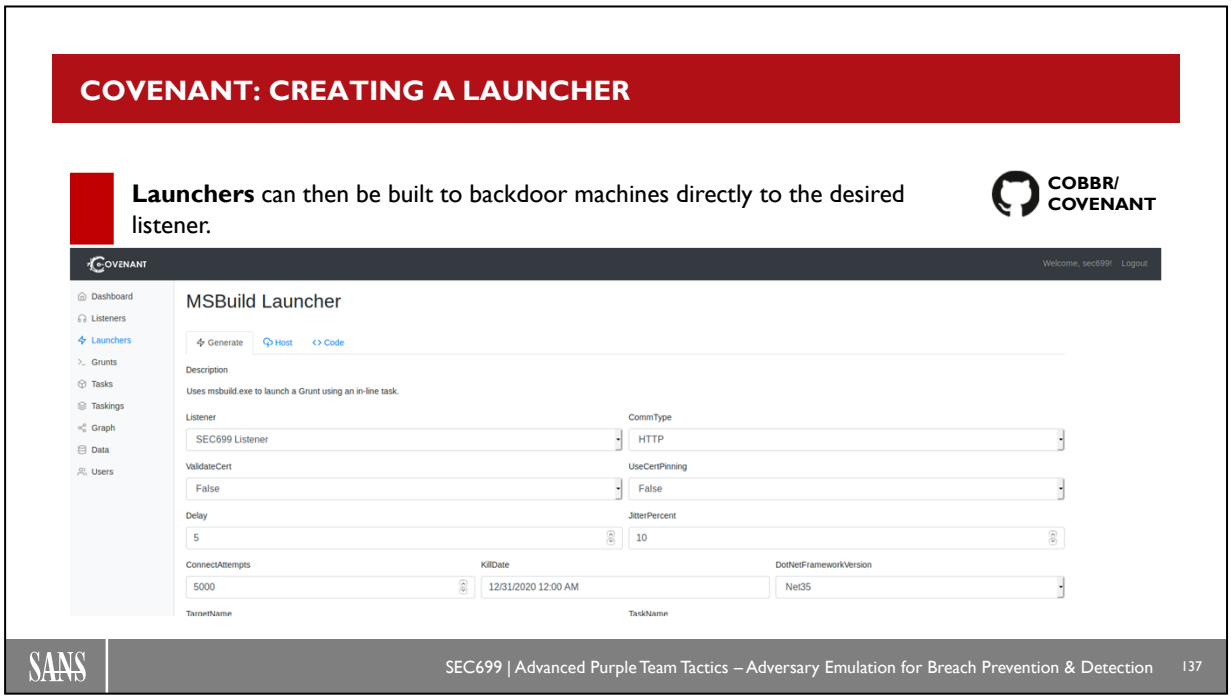


Covenant: Creating a Listener
As a first step, we need to configure Covenant to start listening for incoming connections from infected systems (systems with a grunt). We can do this by creating a listener.

Note that Covenant currently only supports HTTP listeners. Similar to Empire, there are a number of properties that can be configured for a listener:

- The IP address and port it needs to bind to
- The URL that is used for communication
- Whether or not SSL / TLS should be used
- An HTTPProfile (which can include fully customized HTTP request and response headers for stealth operations!)

Additional information on Listener configuration can be found at https://github.com/cobbr/Covenant/wiki/Listeners.



Covenant: Creating a Launcher
Launchers can be used to generate different commands, binaries, one-liners,… to infect machines in the target environment (to become grunts).

Currently, Covenant has the following launchers built-in:

- Binaries
- PowerShell
- MSBuild
- InstallUtil
- Mshta
- Regsvr32
- Wmic
- Wscript
- Cscript

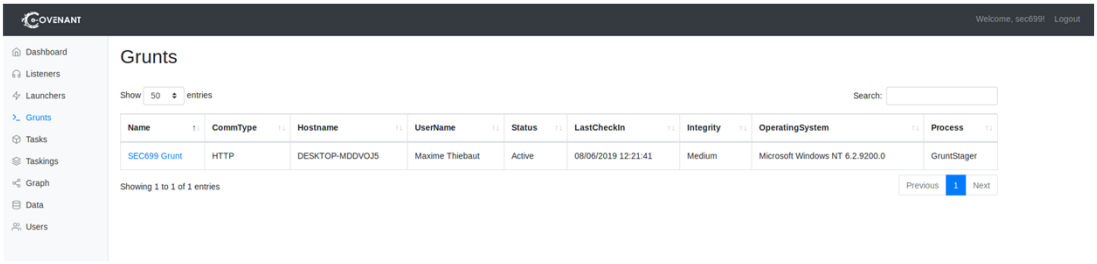
This impressive list provides many different possibilities to allow tailoring toward implemented security controls. Furthermore, these launchers can be further customized. As an example, grunts can be configured to use different communication types (e.g., a grunt can be configured to use an SMB named pipe toward another grunt, which finally connects outbound over HTTP toward the listener).

More information on Covenant launcher configuration can be found at https://github.com/cobbr/Covenant/wiki/Launchers.

COVENANT: GRUNTS

Compromised machines, a.k.a. **grunts**, become available for remote tasking.

COBBR/ COVENANT



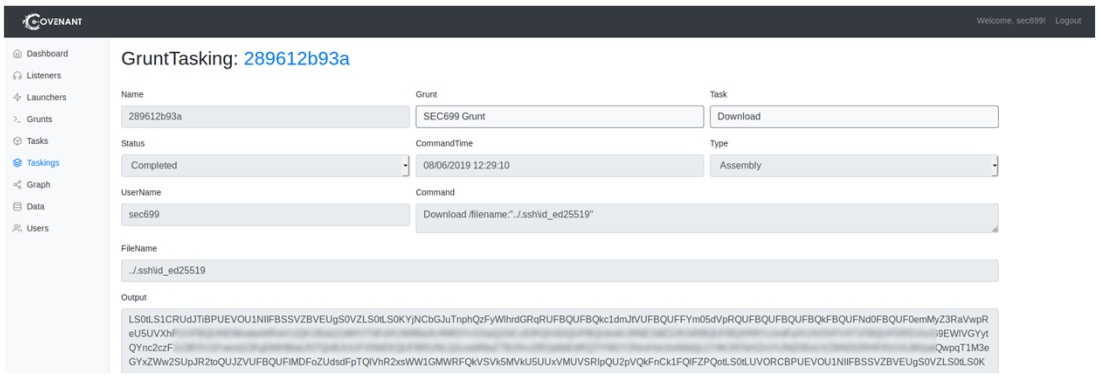
Covenant: Grunts
Once compromised (when a launcher is executed), machines become grunts that are available for tasks. In the screenshot above, we can see that one grunt is available that is using the HTTP Communication Type. Grunts can either be controlled dynamically (through an interactive session), or tasks can be configured. A detailed history of previously executed tasks is also available.

Full documentation on how grunts can be leveraged as part of our work can be found at https://github.com/cobbr/Covenant/wiki/Grunt-Interaction.

COVENANT: TASKS

Predefined, yet customizable, **tasks** can be manually scheduled for grunt execution with their output being sent back to Covenant.

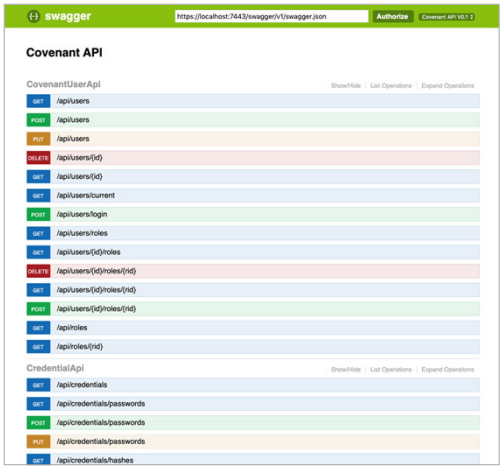
COBBR/ COVENANT



Covenant: Tasks

As previously discussed, predefined, yet customizable, tasks can be manually scheduled for grunt execution with their output being sent back to Covenant. The “tasking” view shows what tasks were previously executed and what their output was.

COVENANT: API



COBBR/
COVENANT

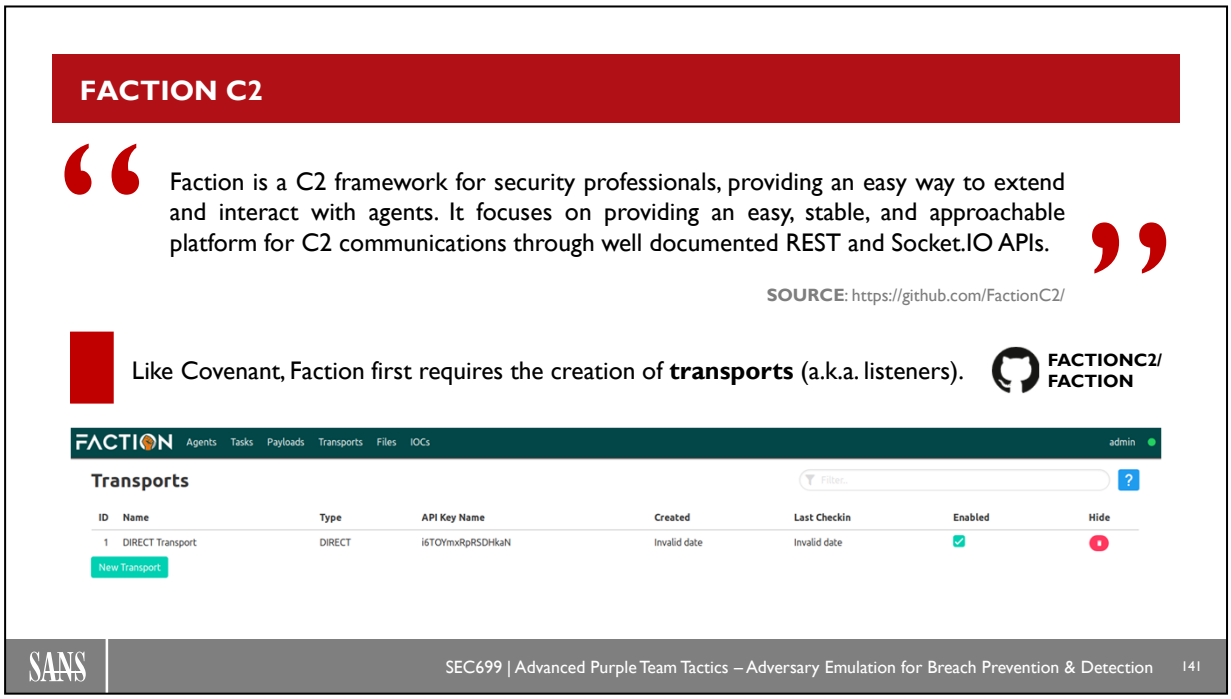
For further automation, Covenant includes a **full-blown API**. Power users can leverage this to automate the execution of adversary emulation steps.

Furthermore, it can also be used to create Covenant extensions and even gain access to valuable (debug) data that would not be visible in the web application.

SOURCE: https://github.com/cobbr/Covenant/wiki/Using-The-API

Covenant: API
For further automation, Covenant includes a full-blown API. Power users can leverage this to automate the execution of adversary emulation steps. Furthermore, it can also be used to create Covenant extensions and even gain access to valuable (debug) data that would not be visible in the web application.

The Covenant API uses Swagger UI, which also visualizes the overall API.



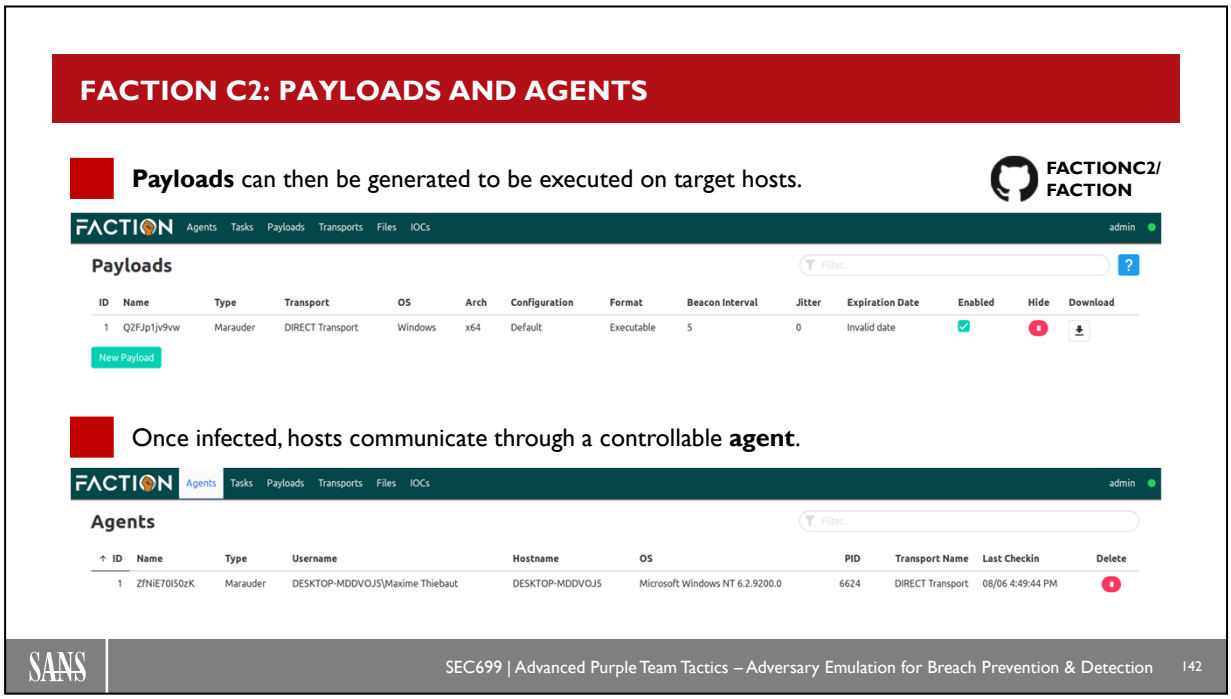
Faction C2

Faction is a C2 framework for security professionals, providing an easy way to extend and interact with agents. It focuses on providing an easy, stable, and approachable platform for C2 communications through well documented REST and Socket.IO APIs. It's currently a bit less stable than Covenant, but might be a useful addition to your toolkit. According to Faction's documentation, here's why it's "special" (extracted from https://www.factionc2.com/):

- It's flexible: Faction was designed to interact with any agent that speaks its language. This means that you can easily create your own agent for Faction either for your internal team or the world at large.
- You can create an entirely standalone agent with all its functionality baked in, but agents greatly benefit when they can load Faction modules. These modules are stand-alone libraries or code that bring new commands and features to an agent. An important aspect of Faction modules is that they are designed to be language specific, not agent specific.
- In most engagements, you're not going to have your agents connecting back directly to your C2. Faction was designed with redirects in mind in the form of Transport Servers. Transport Servers sit between Faction and your agent and handle masking your communications, and since Faction is all API based, these servers can be written in whatever language you're most comfortable in.

We will shortly walk through some Faction C2 functionality!

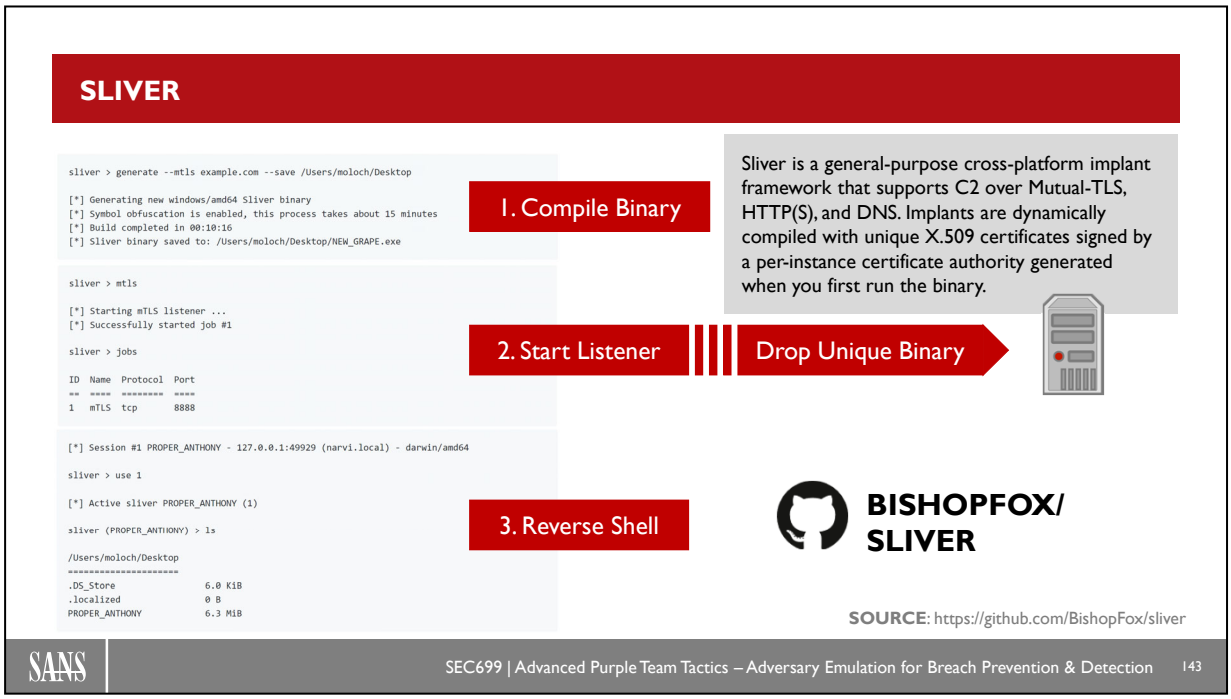
First of all, like Covenant, Faction requires the creation of a "Listener". In Faction language, however, a listener is called a "transport".



Faction C2: Payloads and Agents
In order to compromise systems, Faction C2 relies on payloads and agents.

Payloads and agents are defined as follows (from https://www.factionc2.com/docs/using):

- Payloads are run on targets to establish an Agent. They control the initial settings for an agent, such as beacon interval, jitter, transports, and expiration dates. Payloads use the same password to stage an agent; as part of the staging process, an agent gets its own password for communications.
- Once a Payload stages, it becomes an agent. Agents allow you to interact with the target system. Faction agents are extensible through modules that provide additional commands.

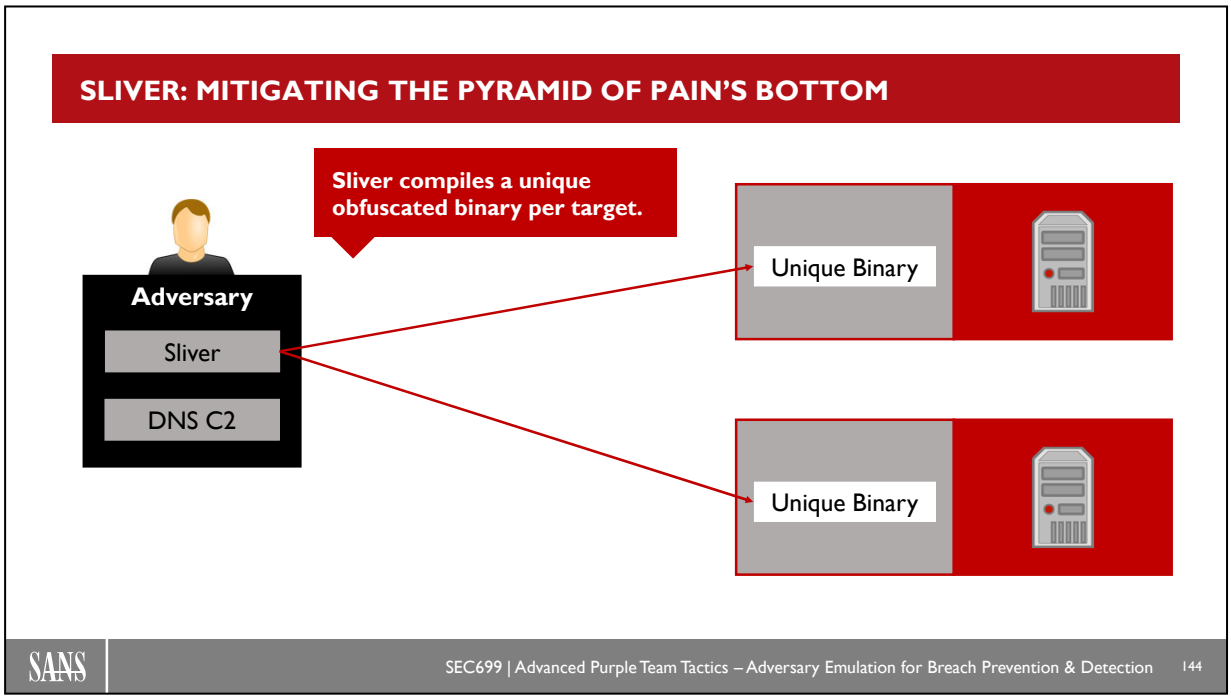


Sliver
Sliver is a project currently in "alpha" stage by Bishop Fox. It is a general-purpose cross-platform implant framework that supports C2 over Mutual-TLS, HTTP(S), and DNS. Focused on stealth operations and AV / EDR evasion, implants are dynamically compiled with unique X.509 certificates signed by a per-instance certificate authority generated when you first run the binary.

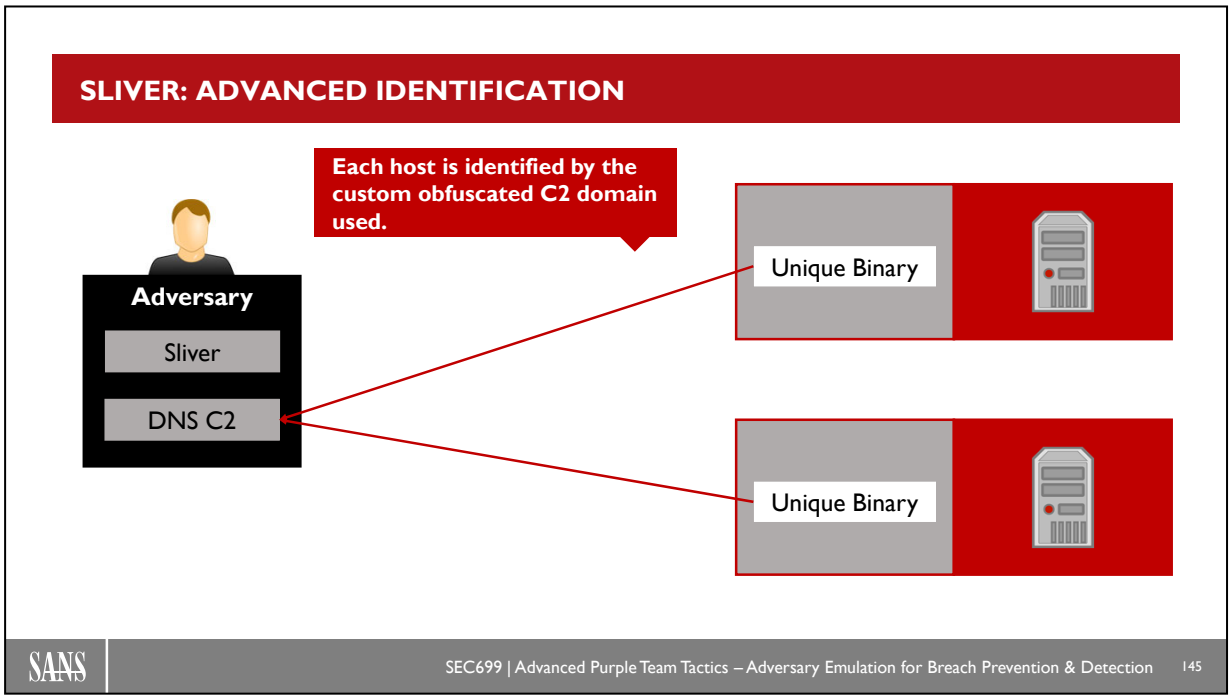
How the typical Sliver infection chain works:

- A unique binary is compiled
- A listener is started, after which the unique binary is dropped
- Upon execution of the binary, a reverse shell is obtained by the adversary

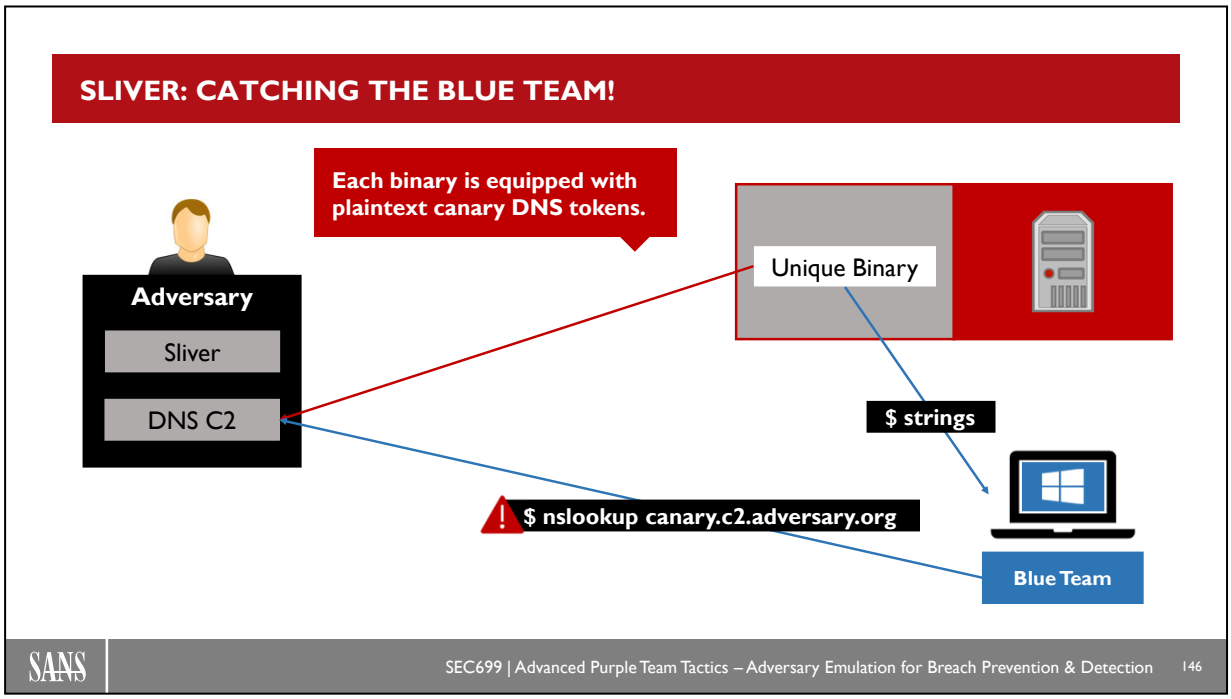
Additional information can be found at https://github.com/bishopfox/sliver.



Sliver: Mitigating the Pyramid of Pain's Bottom
Sliver relies on binaries compiled per target host. This approach has the advantage of avoiding the bottom of the Pyramid of Pain as each binary has a variable hash and supports variable C2 domain addresses and IPs. The cross-platform binaries can be compiled against Darwin, Linux, and Windows. Multiple communication protocols are furthermore supported, ranging from classic HTTP to the more complex "Mutual HTTPS".



Sliver: Advanced Identification
Rather than relying on host-specific identifiers such as a FQDN, Sliver ships each binary with a unique obfuscated subdomain. Using this approach ensures we identify not only the host but also the spread binary, allowing Red Teams to easily associate infection vectors with infected hosts.



Sliver: Catching the Blue Team!
One very interesting feature in Sliver is the addition of canaries to catch the Blue Team. All Sliver binaries are equipped with a plaintext DNS canary token. This is a hostname that's actually controlled by the adversary, but is NOT used as part of the Sliver activities. Whenever the Blue Team catches one of the Sliver binaries, they are likely to run tools such as strings against these binaries. The DNS canary would clearly jump out and *might* be visited by the Blue Team. When the Blue Team resolves the hostname, the Red Team now knows one of their implants has been detected.

Although undeniably useful, this feature requires a bit of setup as DNS delegation must be performed for the desired domain.

INTRODUCING SHAD0W

BAT3C/
SHAD0W

Shad0w is a relatively new framework (first released in 2020) that was developed by **bats3c**. It has a highly interesting focus on **stealth operations** and has implemented some very interesting features…

- Support for **multiple process injection techniques**
- Payloads frequently updated to **avoid AV detection**
- Leverages **Donut** for payload generation (including .NET assemblies, EXEs, DLLs,…)
- Relies on **dynamically resolved syscalls** to avoid userland API hooking

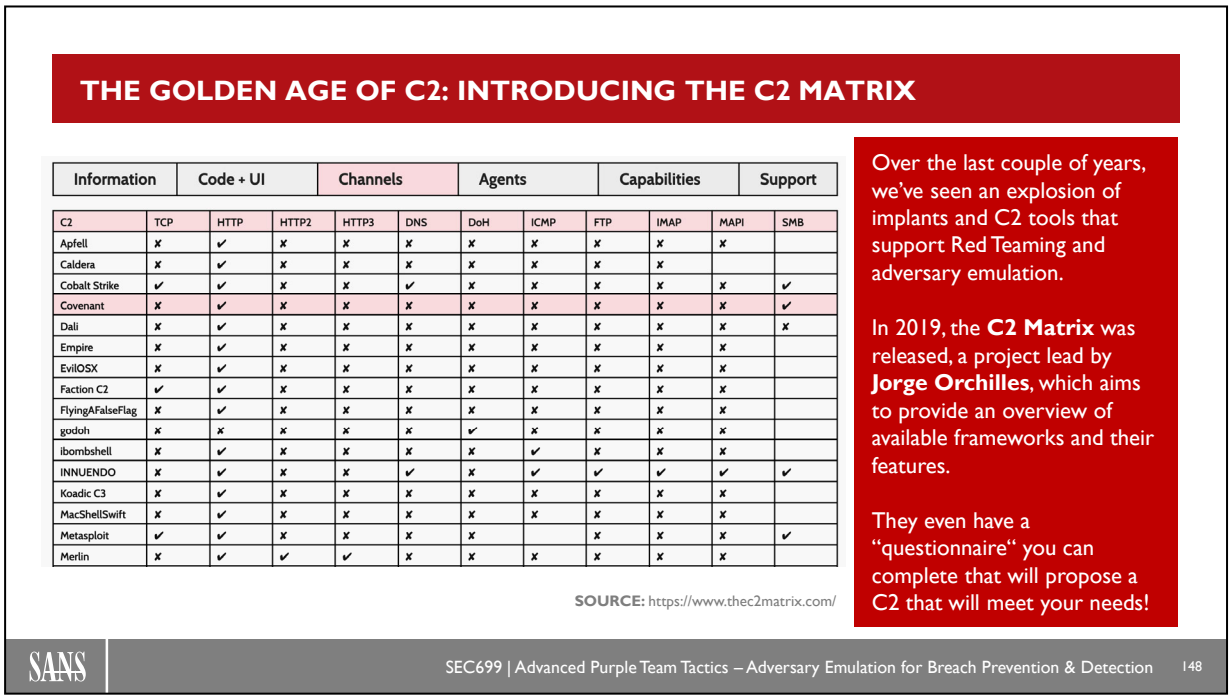
We will review several of these stealth techniques in-depth throughout the week!

SOURCE: https://github.com/bats3c/shad0w

Introducing Shad0w
Shad0w is a relatively new framework that was developed by bats3c. It was first released in 2020 and has a highly interesting focus on stealth operations and has implemented some very interesting features.
Some of the most interesting features are:

- Support for multiple process injection techniques. This includes, for example, DLL injection, which we'll touch upon during the remainder of this course.
- The payloads used are frequently changed to help avoid AV detection.
- Leverages Donut for payload generation. Payloads it can use include traditional executables (EXE) and DLLs, but also .NET assemblies.
- Shad0w relies on dynamically resolved syscalls that can help evade detection through userland API hooks.

Please feel free to have a look at https://github.com/bats3c/shad0w.



The Golden Age of C2: Introducing the C2 Matrix
Over the last couple of years, we've seen an explosion of implants and C2 tools that support Red Teaming and adversary emulation. It can be tricky to keep a good overview and understand what C2 platform is best for your needs. In 2019, the C2 Matrix was released, which aims to provide an overview of available frameworks and their features. One of the co-authors of the C2 Matrix is SANS Instructor Jorge Orchilles.

The screenshot on the slide provides an overview of the different channels supported by C2 tools listed in the C2 Matrix.

They even have a "questionnaire" you can complete that will propose a C2 that will meet your needs. Go check it out at https://www.thec2matrix.com/.

Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

EXERCISE: PREPARING ADVERSARY EMULATION STACK

Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

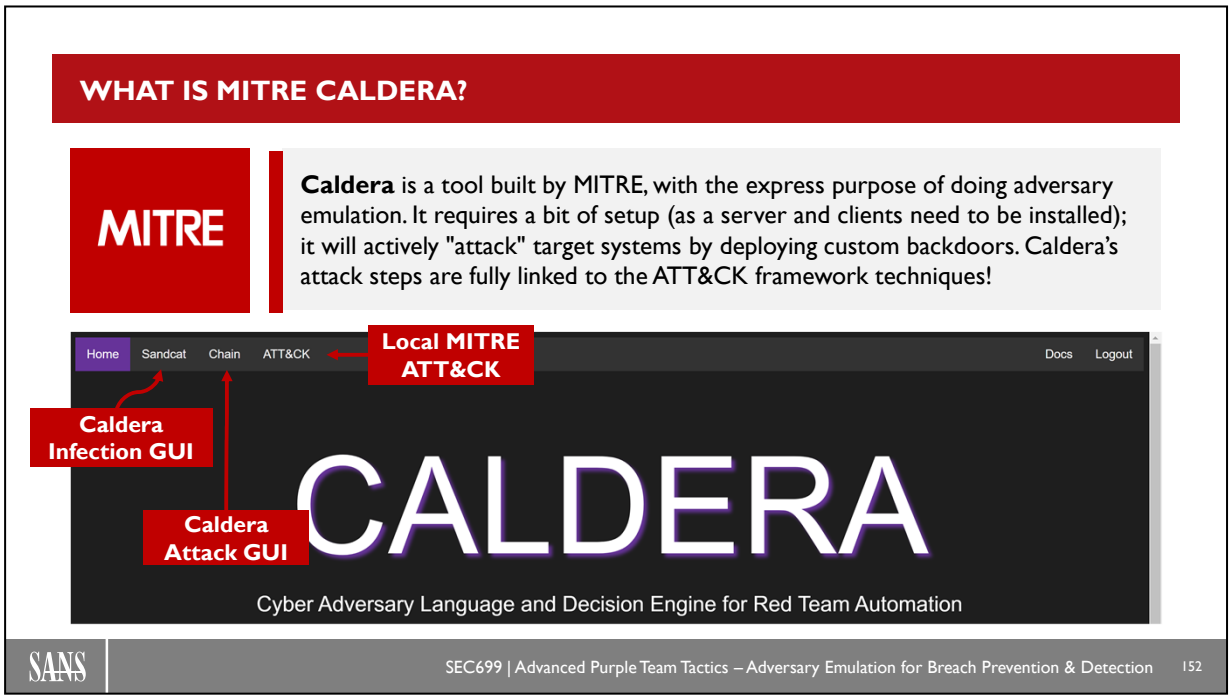
Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

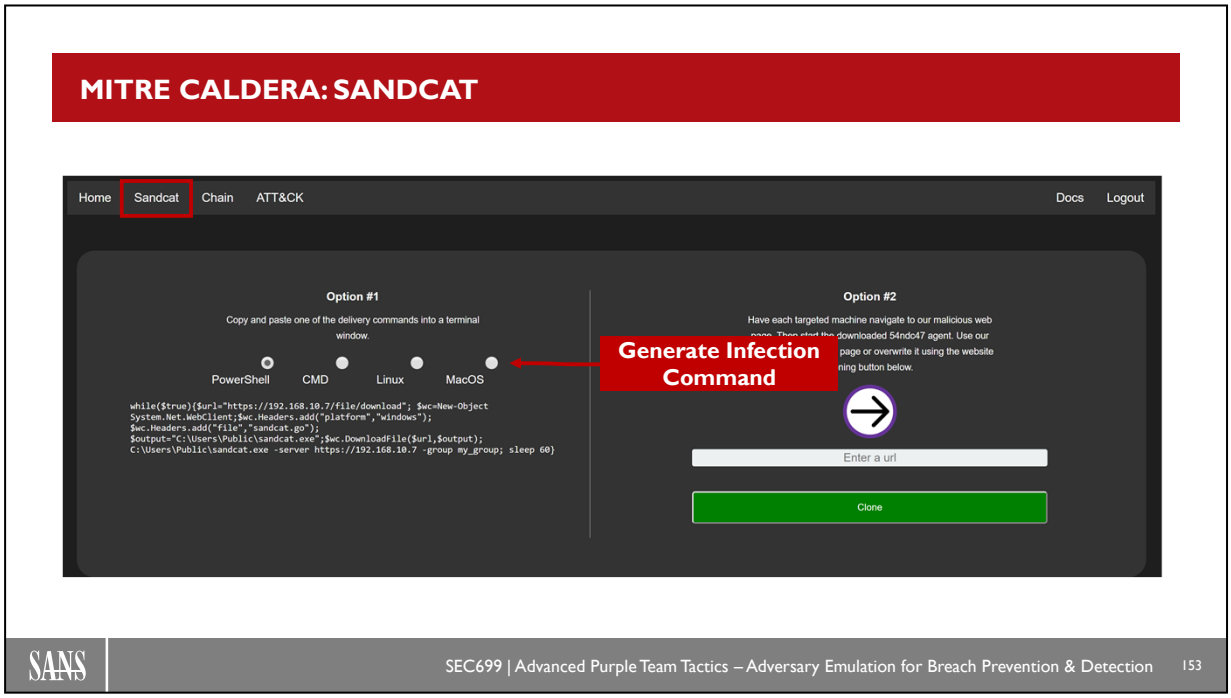


What Is MITRE Caldera?

Caldera is one of the most promising open-source adversary emulation tools built by MITRE. It requires a bit of setup (as a server and clients need to be installed); it will actively "attack" target systems by deploying custom backdoors. Caldera's attack steps are fully linked to the ATT&CK framework techniques!

It includes a full-blown local implementation of the MITRE ATT&CK framework and has a GUI that allows us to configure and run adversary emulation campaigns. A fair warning, though: Caldera is under continuous development and thus quickly changes! We will illustrate this by developing our own module during the upcoming Caldera lab! As Caldera will be the main tool we will use for automated adversary emulation through SEC699, we will spend quite some time walking through its different features in this section!

All Caldera documentation can be found at https://github.com/mitre/caldera.

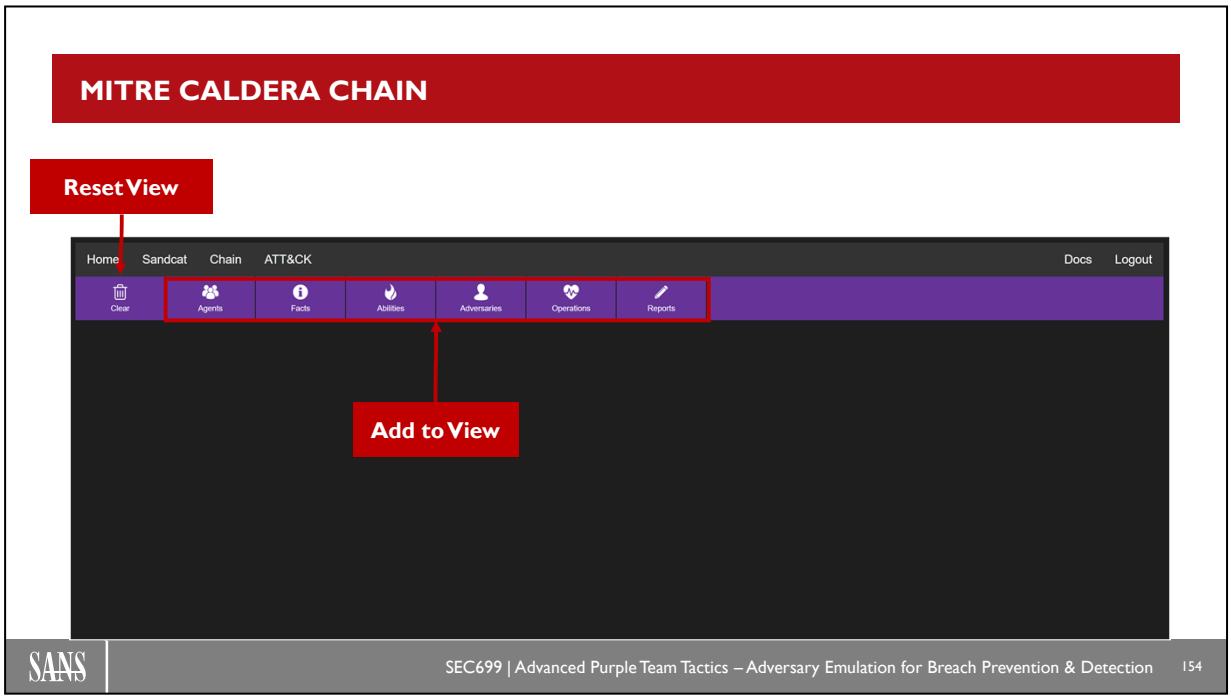


MITRE Caldera: Sandcat

In order to "enlist" a host in Caldera, we need to deploy the Caldera agent "Sandcat" on the system. Caldera provides two main methods to accomplish this:

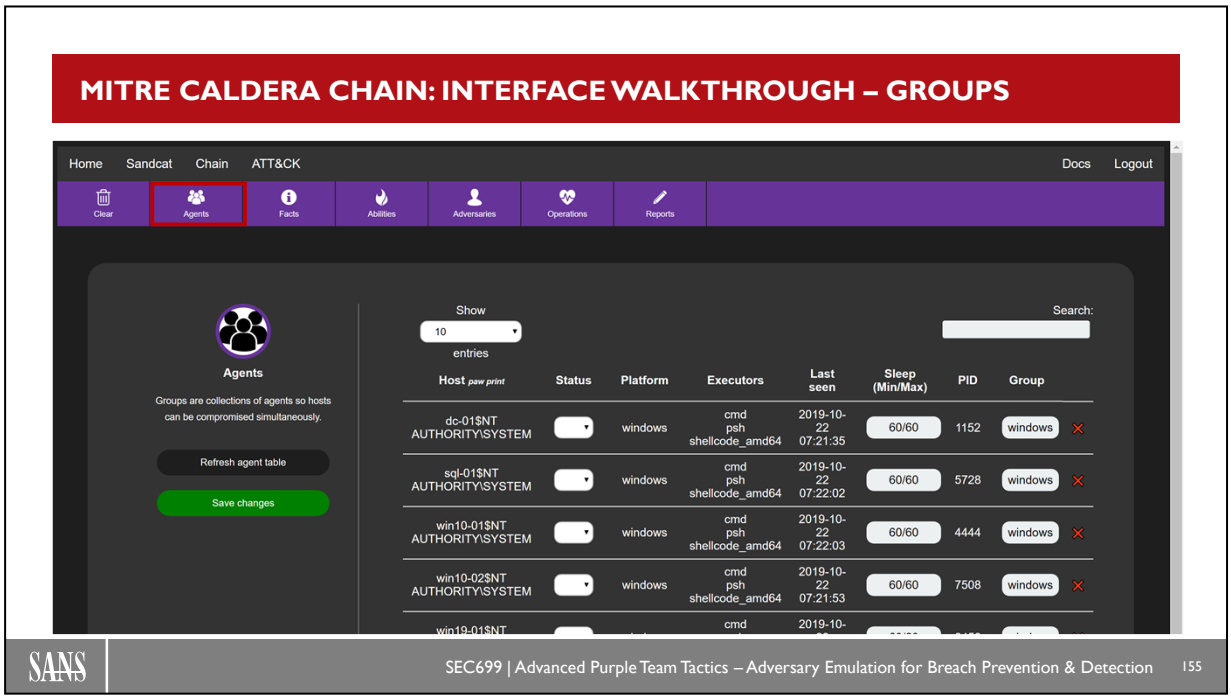
- Option 1: Generating an infection command (in PowerShell, Windows CMD, Linux or MacOS). This command-line syntax can subsequently be copy / pasted in a suitable command prompt for execution.
- Option 2: Using a URL that provides a downloadable binary for the Sandcat agent.

Depending on your requirements and exact setup, you can use any of the two options listed above. It's good to remember that Caldera is typically not used in a pure, stealth, Red Team mode, so the infection methods haven't been designed with "stealth" in mind.



MITRE Caldera Chain
MITRE Caldera is fully-customizable; all of its features are structured in plugins. A set of basic plugins are shipped by default to provide a functioning boilerplate.

The **Chain** plugin, as observed in the above image, provides us with a GUI to manage our operations. Although quite rudimental at the moment, the Chain plugin works by pinning sub-views to our main view which enables us to manage groups, facts, adversaries, and operations. Once our view is over-populated, a "clear" button provides us with the ability to reset the view.

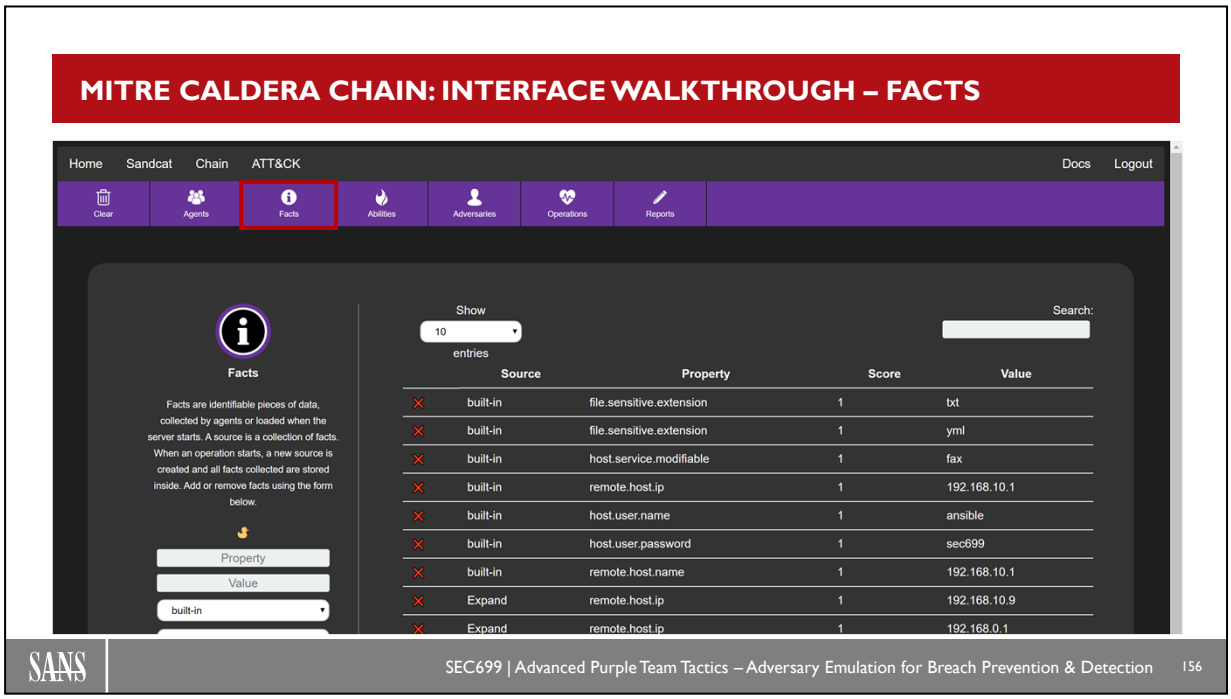


MITRE Caldera Chain: Interface Walkthrough – Groups

The "Agents" view shows all systems that are currently "connected" to Caldera (i.e., that have the Caldera client running). We can create and structure groups ourselves. Note that a system can be part of different groups at the same time! On the slide above, we have the following systems:

- dc-01 (member of the "windows" group)
- sql-01 (member of the "windows" group)
- ubuntu18-01 (member of the "linux" group)
- win10-01 (member of the "windows" group)
- win10-02 (member of the "windows" group)
- win19-01 (member of the "windows" group)
- win19-02 (member of the "windows" group)
- Workstation (member of the "windows" group)

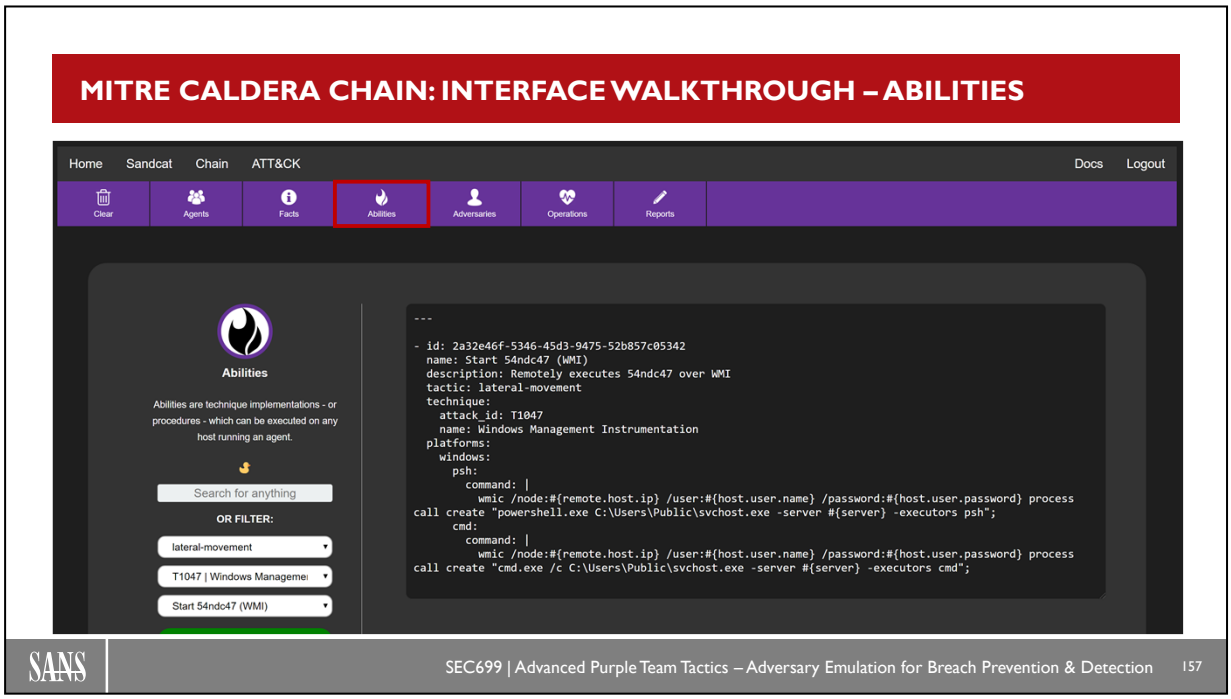
We will further discuss how these groups can be used later in this section!



MITRE Caldera Chain: Interface Walkthrough – Facts

The **Facts** enable us to rely on variables to customize our operations. These can be set statically (e.g., usernames, passwords, …) as well as dynamically retrieved from previous outputs through regex expressions and full line retrievals.

Each fact can be granted a score influencing its precedence over similar variables while individual variables can be temporarily disabled through blacklisting.



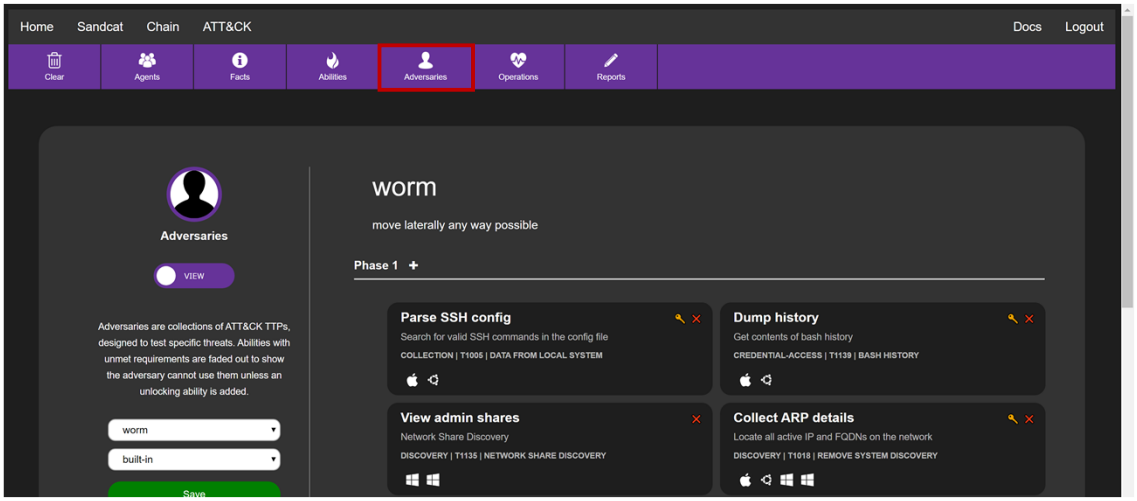
MITRE Caldera Chain: Interface Walkthrough – Abilities

The **Abilities** view is a very interesting one! This is the core of Caldera's emulation capabilities. An ability can be thought of as an implementation of a specific ATT&CK technique. As you can see in the slide, the following information is available for an ability:

- GUID identified for the ability
- A link to the ATT&CK tactic and technique
- A name and description
- The platform that it is valid for
- A command that is to be run to emulate the technique
- If applicable, clean-up activities

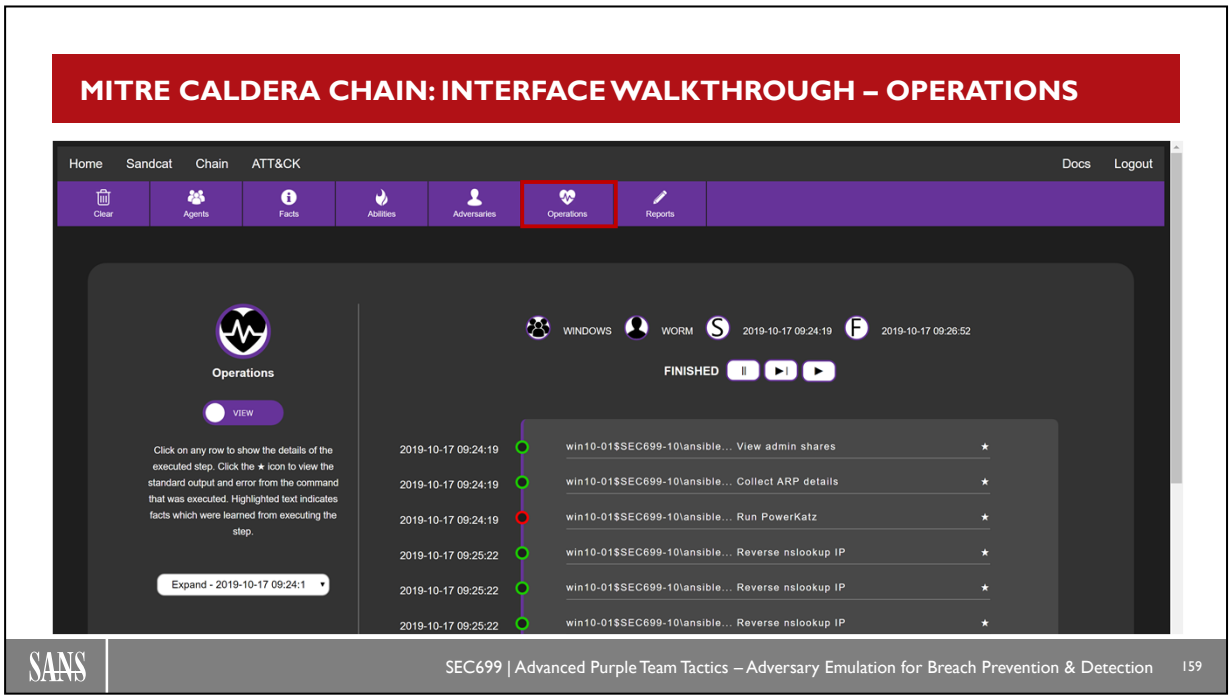
We will further discuss how these abilities can be used later in this section!

MITRE CALDERA CHAIN: INTERFACE WALKTHROUGH – ADVERSARIES



MITRE Caldera Chain: Interface Walkthrough – Adversaries

The **Adversaries** view is where we configure adversary profiles. This is where we could, for example, develop a threat actor that can afterwards be emulated. As part of the adversary profile, we create a number of phases that consist of different techniques that are to be executed!



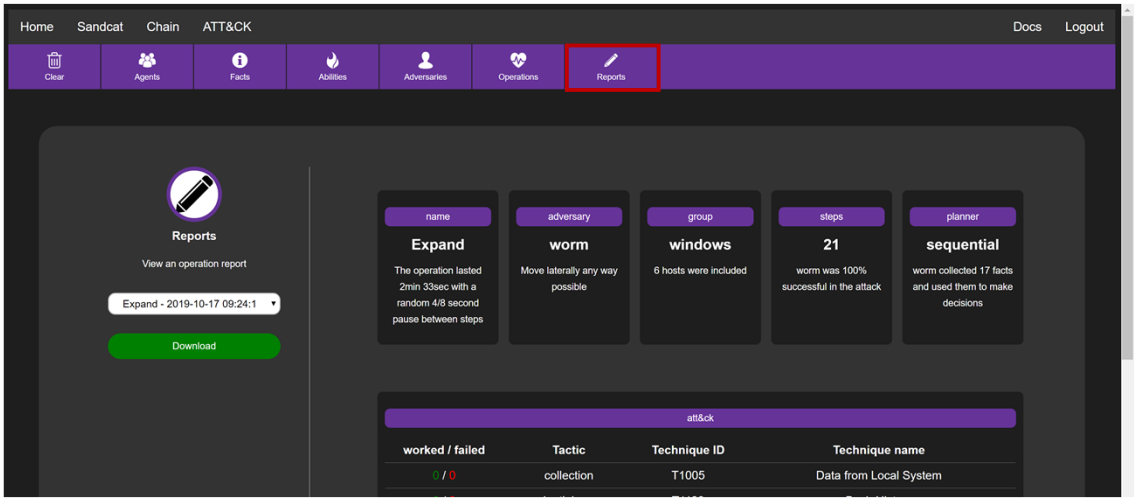
MITRE Caldera Chain: Interface Walkthrough – Operations
Operations are what we really seek in Caldera. An operation runs an adversary *(a.k.a. threat actor; chain of actions/abilities)* against a group of predefined infected hosts.

One notable feature is Caldera's ability to clean up behind itself, avoiding payloads and temporary files from staying on the targeted systems after use.

Besides the clean-up, Caldera offers stealth-related options, one of which is the ability to automatically obfuscate its operation. One more advanced option is the Jitter which gives us the ability to customize the noise made during the operation by setting both the minimal and maximal duration between two C2 polls.

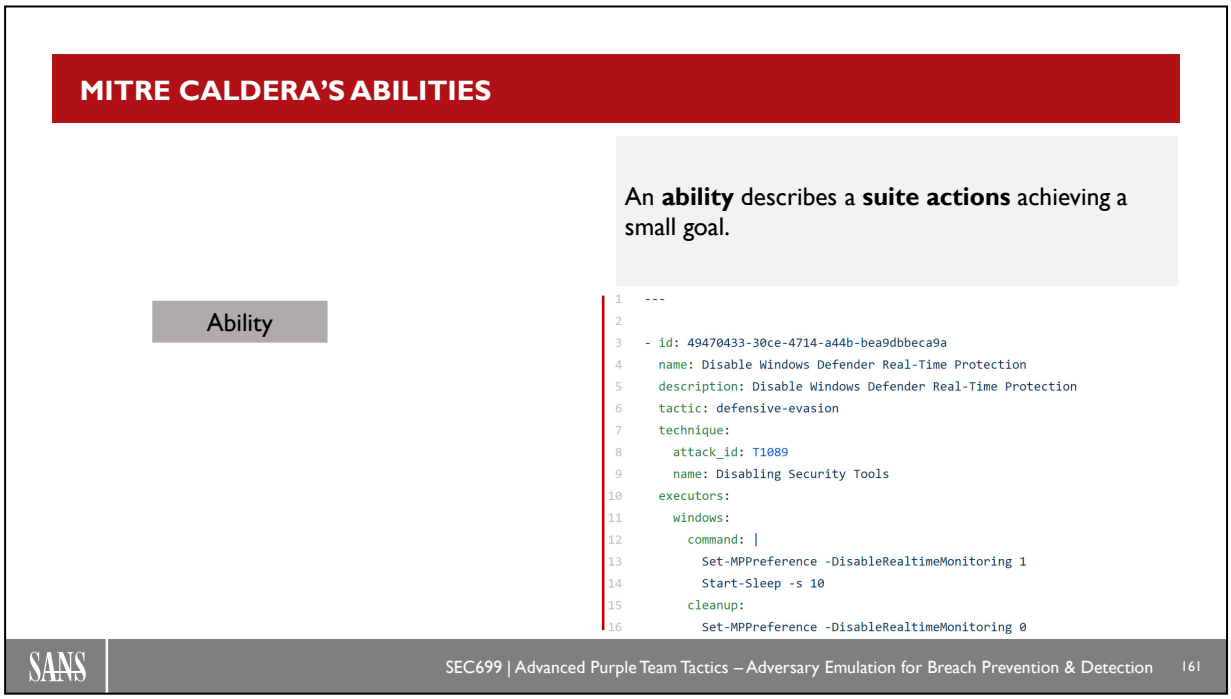
On the right-hand side, we can see the results of the operation as it runs!

MITRE CALDERA CHAIN: INTERFACE WALKTHROUGH – REPORTS



MITRE Caldera Chain: Interface Walkthrough – Reports

When we have performed some operations, we can consult its reports. This high-level overview allows us to assert the successful execution of our adversary against our target group. The usage of the **Reports** comes in handy as some operations, when combined with the dynamic facts, generates more than a couple hundred actions, rendering the Operations tab useless.

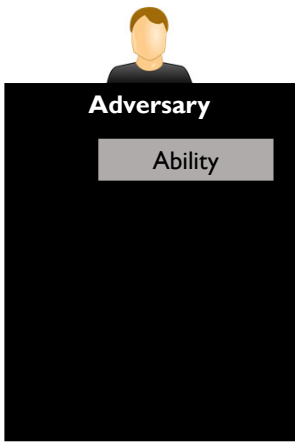


MITRE Caldera's Abilities
Abilities are implementations of MITRE ATT&CK Techniques. Written in the user-friendly YAML format, abilities as we will cover are part of the Stockpile plugin; a custom plugin could easily support another format such as JSON, for example.

A Caldera (Stockpile) ability is typically composed of its information (id, name and description), its mapping (ATT&CK tactic and technique), and its executors which implement the ability itself.

Executors are split into the three supported operating systems (Windows, Linux, and Darwin) and support custom payloads for each OS. Besides the technique implementation itself, each executor can furthermore describe the steps needed to perform a clean-up.

MITRE CALDERA'S ADVERSARIES (1)



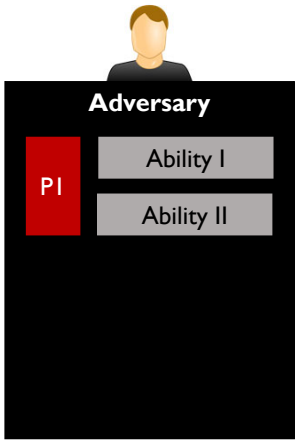
An **adversary** describes a **malicious actor** equipped with **abilities**.

```
---

- name: Windows
  description: Dump Windows Credentials
  phases:
    1:
      - 43b3754c-def4-4699-a673-1d85648fda6a # Clears out the bash history
    2:
      - 49470433-30ce-4714-a44b-bea9dbbeca9a # Disable Windows Defender Real-Time
    3:
      - b08240d0-ff35-444d-b20b-1671a7f65011 # Ensure AMSI is disabled for each n
    4:
      - baac2c6d-4652-4b7e-ab0a-f1bf246edd12 # Use powerkatz to execute mimikatz
```


MITRE Caldera's Adversaries (1)

A Caldera (Stockpile) adversary is nothing else than a chain of abilities. Each adversary is described in its YAML file by referencing its information (name and description) as well as the chained abilities' identifiers.

MITRE CALDERA'S PHASES



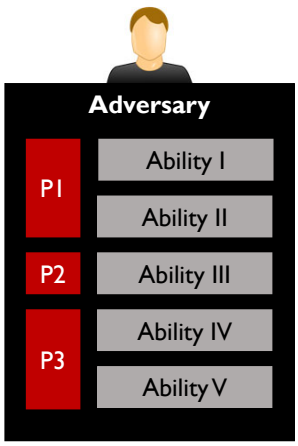
Multiple abilities can be grouped in a **phase**.

```
---

- name: Windows
  description: Dump Windows Credentials
  phases:
    1:
      - 43b3754c-def4-4699-a673-1d85648fda6a # Clears out the bash history
    2:
      - 49470433-30ce-4714-a44b-bea9dbbeca9a # Disable Windows Defender Real-Tim
    3:
      - b08240d0-ff35-444d-b20b-1671a7f65011 # Ensure AMSI is disabled for each
    4:
      - baac2c6d-4652-4b7e-ab0a-f1bf246edd12 # Use powerkatz to execute mimikatz
```


MITRE Caldera's Phases
To ensure proper order, each ability is grouped in phases. Although the GUI offers up to 10 phases, no reasonable limit has been observed.

It is, of course, possible (and recommended) to put multiple abilities in a same phase, although order is not guaranteed within a single phase.

MITRE CALDERA'S ADVERSARIES (2)



Multiple phases describe an **adversary**.

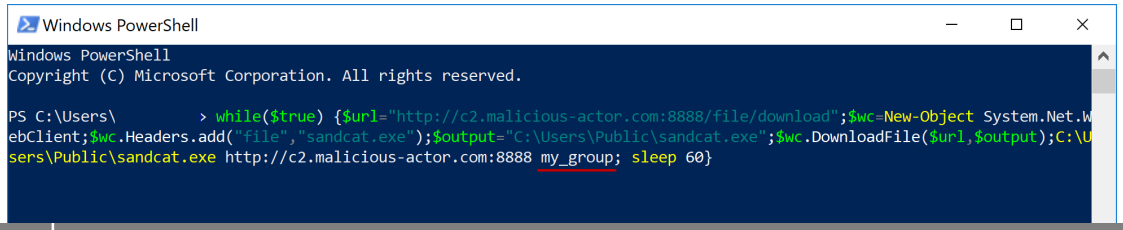
```
---

- name: Windows
  description: Dump Windows Credentials
  phases:
    1:
      - 43b3754c-def4-4699-a673-1d85648fda6a # Clears out the bash history
    2:
      - 49470433-30ce-4714-a44b-bea9dbbeca9a # Disable Windows Defender Real-Tim
    3:
      - b08240d0-ff35-444d-b20b-1671a7f65011 # Ensure AMSI is disabled for each
    4:
      - baac2c6d-4652-4b7e-ab0a-f1bf246edd12 # Use powerkatz to execute mimikatz
```


MITRE Caldera's Adversaries (2)

To sum up: A MITRE Caldera adversary describes an ordered list of phases that contains, in turn, an unordered list of abilities. The entire structure is reflected through YAML files and directory structures, which will be loaded by the Stockpile plugin to later initiate the desired operations.

MITRE CALDERA'S INFECTED HOSTS

A newly infected **host**, by the Sandcat plugin, joins a predefined **group**.

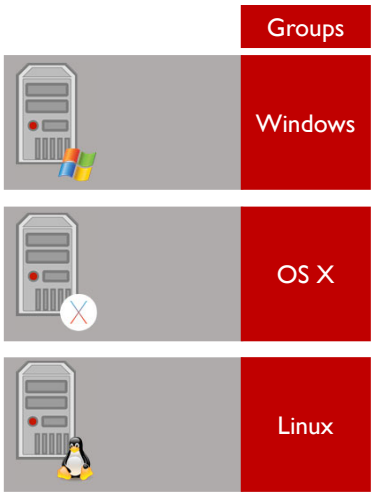


MITRE Caldera's Infected Hosts

While the Stockpile plugin handles the adversary emulation logic, the Sandcat plugin handles the communication between endpoints and the Caldera C2 server. Sandcat offers a cross-platform (Windows, Linux, and Darwin) infector, which can be run through a single command line as shown above. Although optional, it is a good practice to pass as Sandcat argument a group identifying the infected host such as "synctechlabs_windows" or, as seen above, "my_group".

MITRE CALDERA'S GROUPS (1)

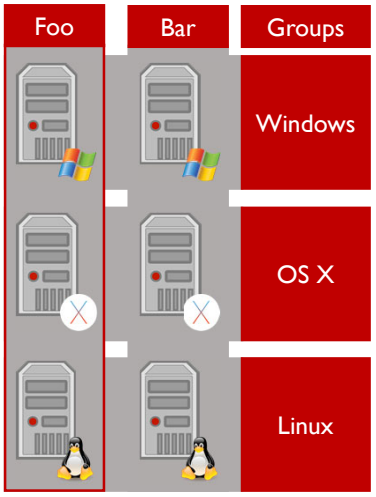
Multiple newly infected **hosts** can have **different** initial **groups**.



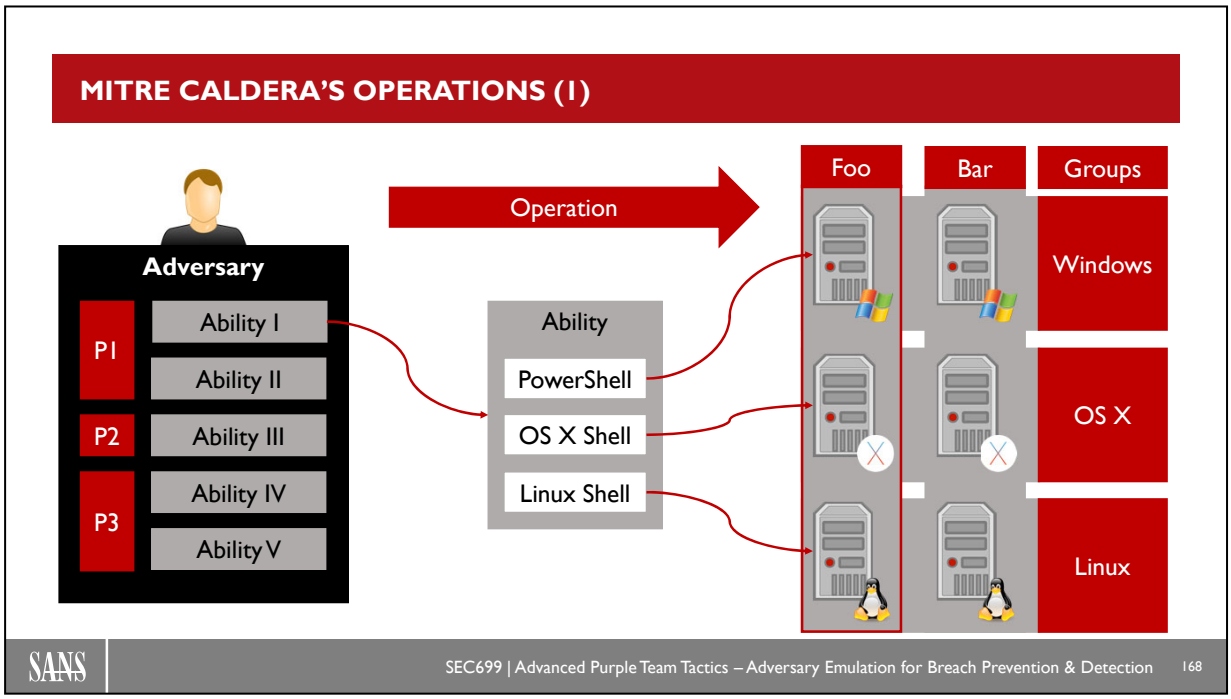
MITRE Caldera's Groups (1)
Associating different groups to infected hosts enables us to better identify our current reach and prioritize interesting infection vectors.

MITRE CALDERA'S GROUPS (2)

Different **groups** can contain **multiple hosts** with **multiple operating systems**.

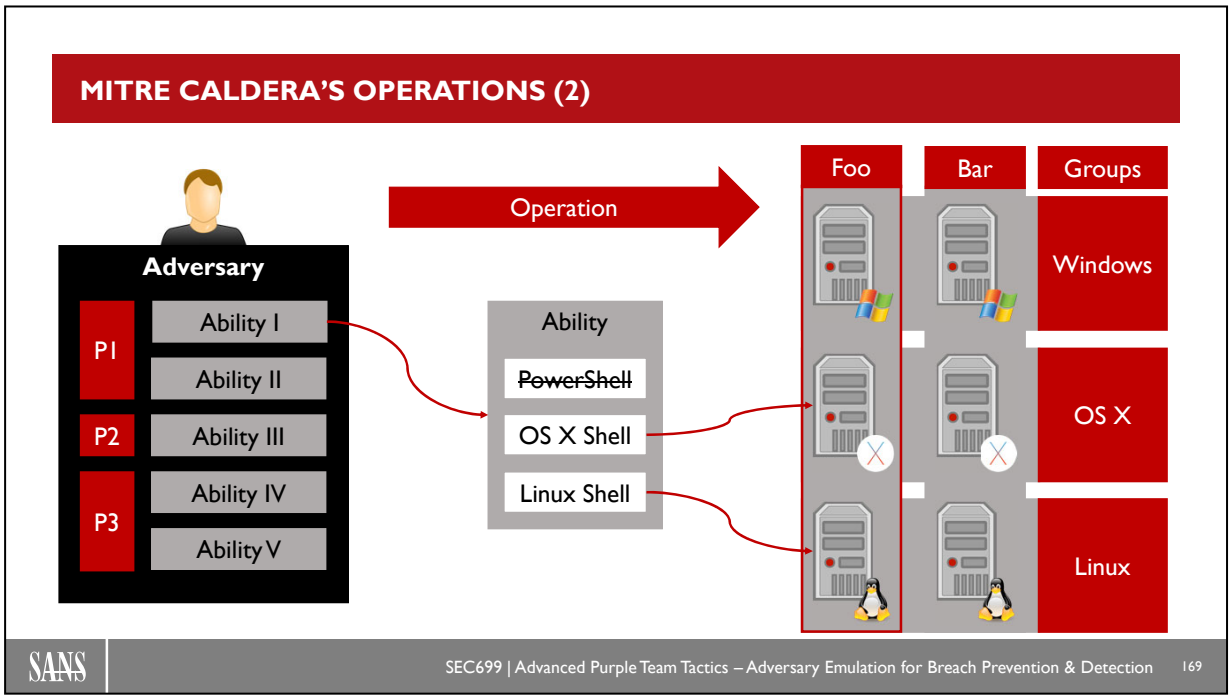


MITRE Caldera's Groups (2)
The Caldera GUI allows us to later assign multiple groups to each host. As operations must target a group, the more groups, the more targeting choices.

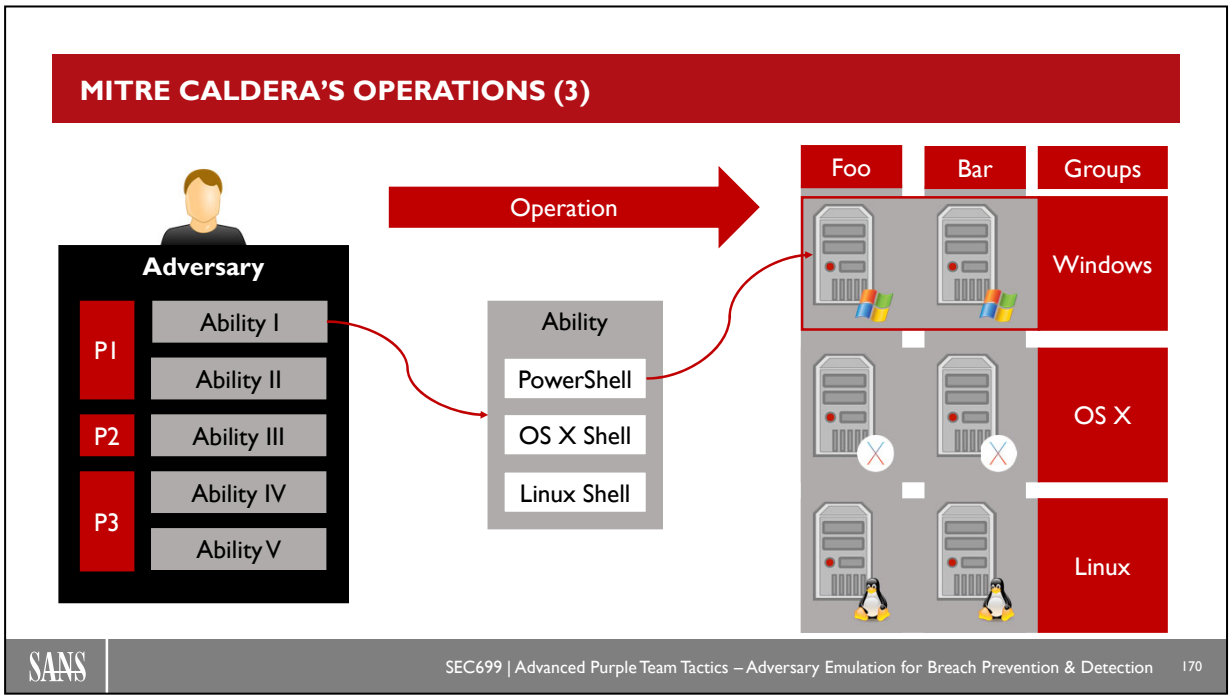


MITRE Caldera's Operations (1)

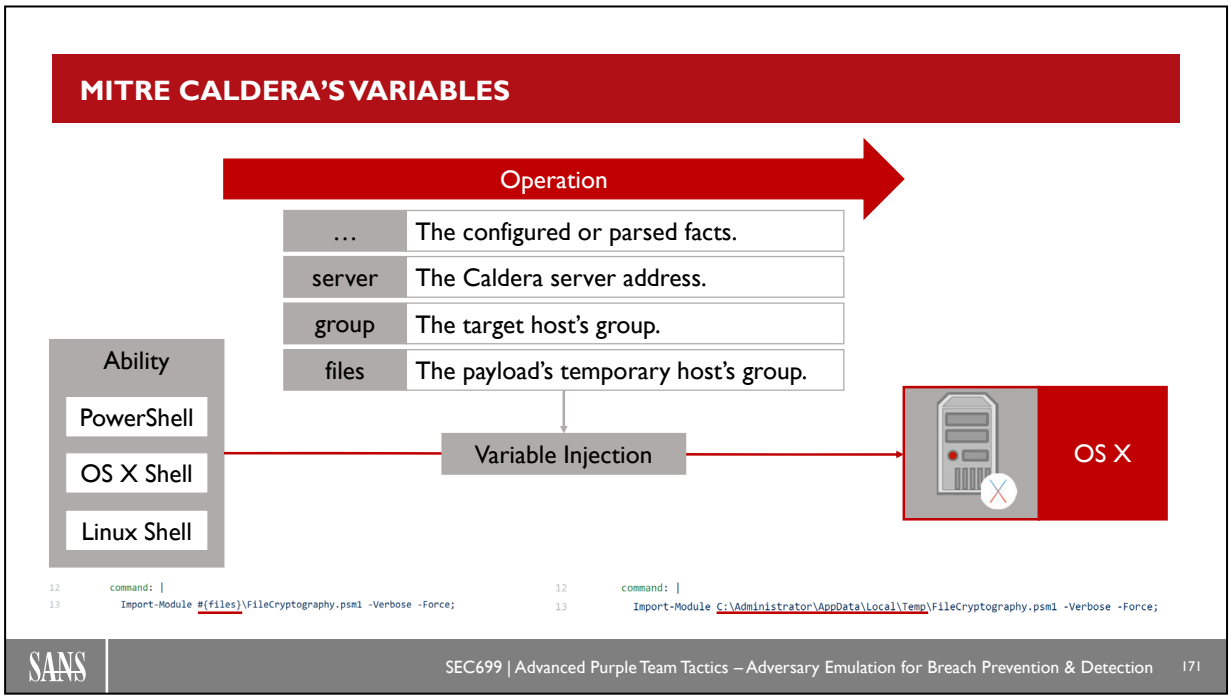
Each Caldera operation targets a specific group which can contain multiple operating systems. This property highlights the necessity to properly implement abilities across the different operating systems where relevant.



MITRE Caldera's Operations (2)
Should an ability, however, not be implemented for a specific operating system, i.e. Windows above, Caldera will skip the missing implementation and resume with the following abilities.



MITRE Caldera's Operations (3)
Using the same principles, and although it may be self-speaking, a cross-platform ability can run without problem against a single operating system.



MITRE Caldera's Variables
At each ability's execution, Caldera passes the commands through the templating engine, which injects both variables and facts.

Three variables are constantly injected and defined by Caldera itself:

- server – The address to the Caldera C2 server as can be used to access additional services provided by the server (network shares, …).
- group – The name of the targeted group the host is part of.
- files – The path to the payload folder, usually the user's temporary folder ("%TEMP%", …).

The facts injected by the templating engine are both the one's pre-selected at the operation's creation as well as any other facts parsed during the operation.

Course Roadmap

- **Introduction & Key Tools**
- Initial Access
- Lateral Movement
- Persistence
- Azure AD & Emulation Plans
- Adversary Emulation Capstone

SEC699.1

Introduction
Course objectives
Building our lab environment
Introducing the lab architecture
Exercise: Deploying the lab environment
Purple teaming organization
Exercise: Introduction to VECTR™
Key tools
Building a stack for detection
Assessing detection coverage
Rule-based versus anomaly-based detection
Exercise: Preparing our Elastic and SIGMA stack
Building a stack for adversary emulation
Exercise: Preparing adversary emulation stack
Automated emulation using MITRE Caldera
Exercise: Caldera

This page intentionally left blank.

EXERCISE: CALDERA

Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

This page intentionally left blank.