Threat Systems Management Office (TSMO)
Technical Training Team
Threat Hunting

Lawrence Comer
Cyber Risk Analyst
Rodney Visser
Chief Engineer

POC: Sara Estill, sara.n.estill.civ@army.mil

Agenda

Agenda

- Introduction to Course
- Cybersecurity Overview
- Threat Hunting Introduction
- Network Security Monitoring
- Lab 1: Virtual Machine Access
- Command Line Basics
- Situational Awareness
- Getting Help
- Input and Output
- Making Changes
- Lab 2: Wireshark Analysis

Introduction to Course

Cybersecurity Overview

Cyber Overview

Cybersecurity can be defined as the art of protecting networks, devices, and data from unauthorized access.

- Confidentiality
- Integrity
- Availability

Cyber as a domain is widespread and can encompass multiple domains:

- Security Operations
- Security Architecture/Engineering
- Risk Assessment
- Governance
- Threat Intelligence
- Framework & Standards

Cyber Overview

Cyber Threat Actors:

- Advanced Persistent Threats (Nation-State Actors)
- Cyber Criminals
- Hacktivists
- Insiders
- Script Kiddies

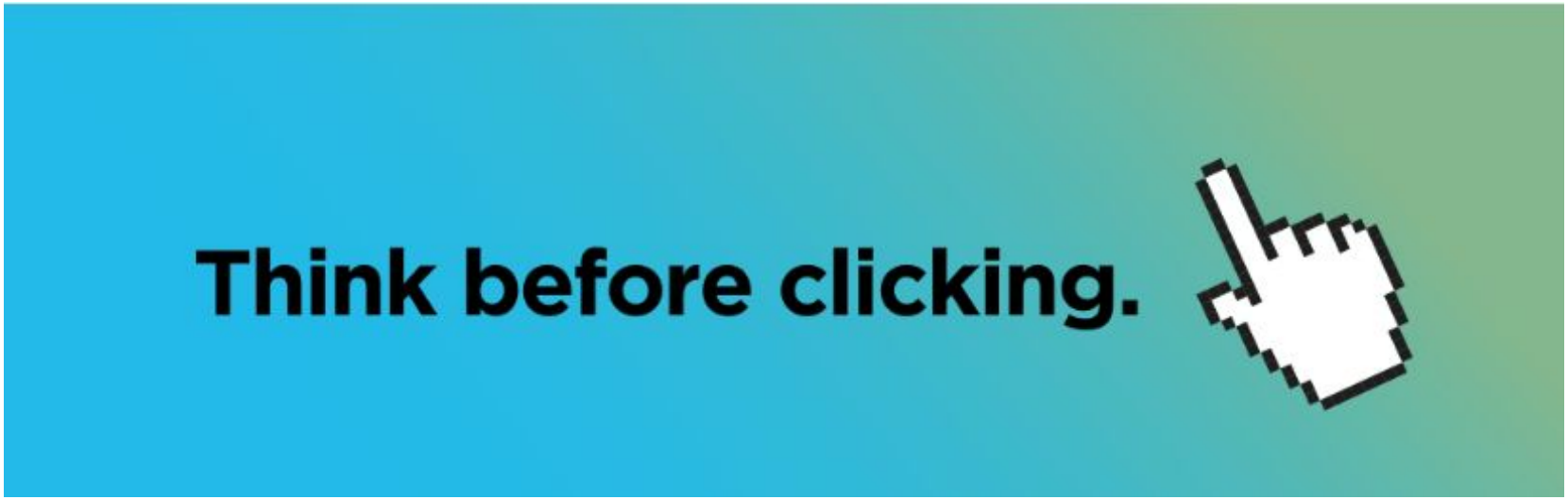
As technology advances, cyber threats and exploitation efforts advance:

- Ransomware
- Supply Chain Attacks
- Internet of Things (IoT) Devices
- Cloud Attacks
- Phishing Attacks
- Cryptocurrency and Blockchain Attacks

Cyber Overview

End users remain organizations' biggest security risk.

- Negligence
- Breaking security policy
- Phishing emails
- Downloading unknown malicious content
- External Devices



Introduction to Threat Hunting

What is Threat Hunting?

The proactive process of investigating networks to locate, close with, and eliminate advanced threats during or after a network intrusion.

Commonly, threat hunting is necessary when advanced threats have proven to evade more common security solutions for detection, prevention, quarantining, and alerting (IDS/IPS, Antivirus, EDR suites, Firewalls, and Logging).

Threat hunting typically involves a robust suite of sensors, analysis utilities, log aggregation and SIEMs, and Artificial Intelligence/Machine Learning (AI/ML).

Occasionally, attribution activities are performed during/after threat hunting and incident response efforts.

Threat Intelligence Sources

- Cyber Threat intelligence (CTI)
- Open-source intelligence (OSINT)
- Technical intelligence
- Human intelligence (HUMINT)
- Vendor-provided intelligence
- Government-provided intelligence

Proactive vs. Reactive

Proactive

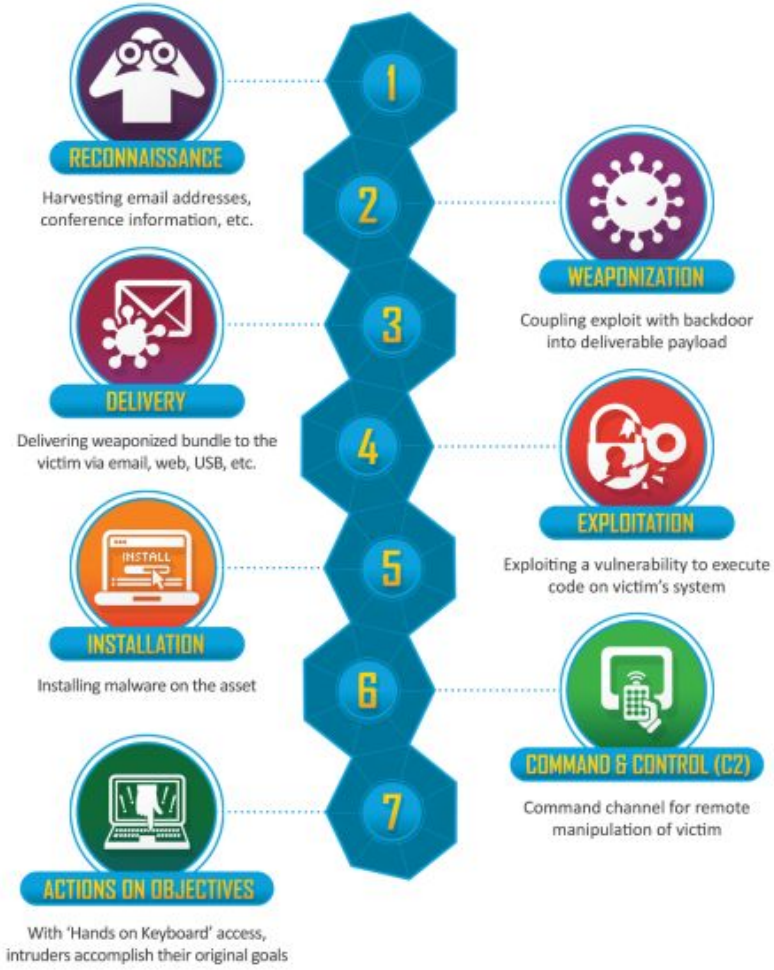
- Searching for potential threats
- Continuous monitoring
- Communicating with stakeholders
- Implementing solutions before they are needed
- Crucial for effective threat hunting

curred

n

Attacker's Mindset

The Cyber Kill Chain illustrates the mindset and process used by attackers attempting to compromise secure systems.



Threat Hunting Tools & Techniques

Types of Threat Hunting Tools

- Network traffic analysis
- Endpoint analysis tools
- Malware analysis tools
- Threat intelligence platforms
- Security Incident and Event Management (SIEM) systems
- Endpoint Detection and Response (EDR)
- Data visualization and correlation tools
- Scripting and automation tools
- AI/ML-based tools

Threat Hunting Techniques

- Network traffic baselining
- Network traffic analysis
- Log detection and analysis
- EDR
- Behavioral analysis
- Signature-based detection
- Anomaly detection
- Indicator of Compromise (IOC) hunting
- Attack surface reduction
- Continuous monitoring
- Automation and orchestration
- AI/ML

Incident Response

Establishing an incident response plan allows for quick and effective mitigation to any threats that are discovered during the hunting process.

This includes identifying the scope of the incident, containing the threat to prevent further damage, and taking steps to eliminate the threat.

Effective incident response also allows you to gather critical information about the attack (such as the methods and tools used) and can enable you to better protect yourself from attacks in the future.

Types of Organizations at Risk

- Financial industry
  - Fraudulent wire transfers
  - Regulatory fines as penalty for inadequate security
- Healthcare
  - Compromise of confidential patient information
  - Disruption of services and equipment
  - Loss of data pertaining to medical research
- Military & Government
  - Loss of classified information
  - Economic impact
  - Potential for loss of life
- Defensive Industrial Base (DIB)
  - Compromise and/or theft of sensitive technologies
  - Potential risk to national security
  - Financial losses

Threat Hunting Methodology

Documentation

- Clear, organized record-keeping
  - Methods used during attack
  - How the attack was detected
  - How the attack was stopped
- Makes it easier to communicate and study the exact nature of the threat
- Allows for more efficient response in the future if a similar attack occurs

Communication

- Keep stakeholders apprised of any incidents, findings, and impacts
  - Reduces number of surprises
  - Enables you to set expectations
  - Facilitates collaboration
  - Keeps analysts accountable
  - Helps the owner feel invested in the security process

- Note: Advanced threat actors often monitor communication. To avoid giving the attacker additional information, have a secure channel available that can be used to help prevent eavesdropping.

Patience & Persistence

- Conduct numerous investigations, even if it looks like there is no conclusion.
- Take time to thoroughly analyze and document all findings.
- Do not rush to conclusions or prematurely close an investigation.
- Patience and persistence help you improve your threat hunting skills.

Adaptability

Threat hunters need to be able to rapidly adapt to new technologies, attack methods, and emerging threats.

This includes:

- Staying up-to-date with new tools, techniques, and tactics used by attackers.
- Being able to adjust hunting strategies and tactics as necessary.
- Being able to quickly respond to new threats and incidents.
- Being able to pivot to new areas of focus as needed.

Collaboration

Threat hunters need to work with other members of the organization (including security analysts, network administrators, and others) in order to effectively respond to cyber threats.

By merging information and resources, everyone has a better chance of understanding the threat that you face.

Collaboration methods include:

- Sharing information with other teams and organizations, whether directly or via online platforms.
- Coordinating investigations and response tactics with other teams.
- Having regular meetings to discuss progress and share information.
- Using shared tools (such as SIEM systems) for data collection and analysis.
- Asking others to lend their expertise to a problem you may not be

Critical Thinking

Critical Thinking: the ability to analyze and evaluate information and situations in a logical and systematic manner in order to make sound decisions and judgments.

Critical thinking for Threat Hunting:

- Necessary for evaluating and analyzing large amounts of data, identifying patterns and anomalies, and making informed decisions about how to protect yourself from threats.
- Allows you to discover solutions that may not be obvious.
- Enables you to weigh the potential risks and benefits of various courses of action to best protect your organization.

Staying Current

Why is it necessary? Because change is imminent and inescapable.

Competitors, hackers, and Advanced Persistent Threats (APTs) will all embrace new technologies as they search for ways to compromise systems.

Staying current by studying the latest technologies and security techniques is the best way to protect against new threats.

Testing & Validation

Just because something worked at first does not mean that it can adequately protect you now.

Ask yourself: “Is this working correctly?”

Through testing & validation, you can:

- Identify security concerns
- Determine whether or not systems are giving you reliable information
- Improve existing systems
- Identify any gaps or weaknesses with your current security practices
- Collect evidence that will be necessary for proving compliance

Threat Hunting

Types of Threat Hunting

Proactive

- Identifying threats before they can cause damage
- Continuous monitoring, testing, and log review to find indicators of compromise before the situation escalates

Reactive

- Responding to incidents that have already occurred
- Identifying attack source, containing damage, and mitigating future threats

Opportunistic

- Taking advantage of unexpected opportunities to respond to threats
- Monitoring for anomalies and environment changes to detect potential threats before they can cause damage

Key Components of Successful Threat Hunting

- Understand the organization's assets, networks, and systems
- Have a thorough understanding of the organization's threat landscape (includes known threats, threats actors, and attack vectors)
- Utilize threat intelligence to direct your threat hunting
- Use a variety of techniques and tools to monitor for suspicious activity
- Formulate incident response plans for when threats are detected
- Continuously work to improve your organization's security
- Collaborate with other teams to ensure everyone is kept informed of the latest policies and threats
- Educate employees so that they are aware of the impact their actions have on system security

Documentation and Communication

- Keep detailed logs of all threat hunting activities and results
- Regularly share findings with stakeholders and other teams
- Create and maintain a centralized repository for threat intelligence and hunting results
- Communicate with other threat hunting teams to discuss best practices and other findings
- Document all processes to ensure consistency during future efforts
- Provide training and education to stakeholders and team members about current best practices and technologies
- Regularly review and update policies and processes to stay current with emerging threats

Threat Hunting Grounds

The Hunting Grounds

So, what do we know about our network?

- Is it flat?
- Is it segmented?
- Do we have a DMZ (Demilitarized Zone)?
  - What is located inside our DMZ?
- Do we have asset detection?
- Do we have service up/down detection?
- Do we know what kind of network traffic will be in each network segmentation?

The Hunting Grounds

Answering these questions is fundamentally important. If we cannot answer them, it will be difficult to move forward, as we will be unable to differentiate what is normal and what is not.

- We may not be able to answer all of them!
  - We could be assessing a new network.
  - Tools could be in the middle of deployment or not deployed at all.
- It may not be your responsibility to know every piece of traffic, where it is coming from/to, and why.
- However, it is your responsibility to be aware of the roles between hosts and servers within your network.
- For instance, should a workstation in the accounting department be accessing files on an administrator file share? How could we detect this?

The Hunting Grounds

- The reality is you most likely have an established environment.
  - You may already have a baseline of what your configurations are, how your network is segmented, and what normal traffic looks like.
  - Methods to capture and forward Events and Syslogs that identify system changes.

- If not, have a conversation with your system and network administration team about incorporating methods to understand what “normal” looks like and how we are monitoring and collecting information in our environment.

- Threat Hunting is a team sport.

- We skimmed the surface of properly baselining our network, but the idea is that we need to ensure that we have a firm understanding of our network environment (“Threat Hunting Ground”). Our goal is to understand our environment so we can make it as difficult as possible for an adversary to move in, and ensure we rapidly respond if they do.

Examples of Hunting Grounds

Utility company

- Includes control systems and SCADA systems, as well as any external networks that interact with company assets.
- Any third-party vendors' systems would also be considered a part of the hunting grounds.

Automobile manufacturer

- Includes plants used to build automobiles, as well as any other vehicles or stores used to transport and sell cars once they have been built.
- Public-facing websites used to advertise or sell vehicles could also be a way for attackers to exploit the system.

Shipping company

- Includes vessels used to transport goods, as well as any communications or navigation systems used to monitor and track those goods during transit.

Hospital

- Includes hospital networks and equipment, as well as any vendors responsible for providing data management and record-keeping capabilities.

Network Security Monitoring

Network Security Monitoring

Network Security Monitoring helps to detect and respond to cyber threats by continuously monitoring network traffic and identifying abnormal or suspicious activity.

Types of Network Monitoring Tools

Network Intrusion Detection Systems (NIDS): used to monitor network traffic and detect any suspicious activity.

Network Intrusion Prevention Systems (NIPS): similar to NIDS, except these are capable of responding to and preventing these threats from continuing.

Endpoint Detection and Response (EDR): monitor endpoint devices (laptops, servers, phones, etc.) for suspicious activity and can respond to threats in real time.

Security Information and Event Management (SIEM): collect, correlate, and analyze system logs in order to identify potential threats and provide threat intelligence.

Types of Network Monitoring Tools

Network Access Control (NAC): monitor network access and ensure that only authorized devices and users have network access.

Sandboxing: simulate the execution of suspicious files in a controlled environment where their behavior can be studied.

Behavioral Analytics: use machine learning algorithms to detect anomalies in network or device usage.

Cloud Security Solutions: specifically designed to monitor and protect cloud-based infrastructure and services.

Command Line Basics

Command Line Basics

- What is Bash?
  - When we think of a “terminal” in Linux, we are typically thinking of Bash.
  - Bash is a shell; at its heart, it is a command interpreter.
  - There are alternatives, but Bash is the most common shell in production today.

```
When you type this:
$ echo 'hi'
Bash interprets it and does this:
hi
```

Command Line Basics

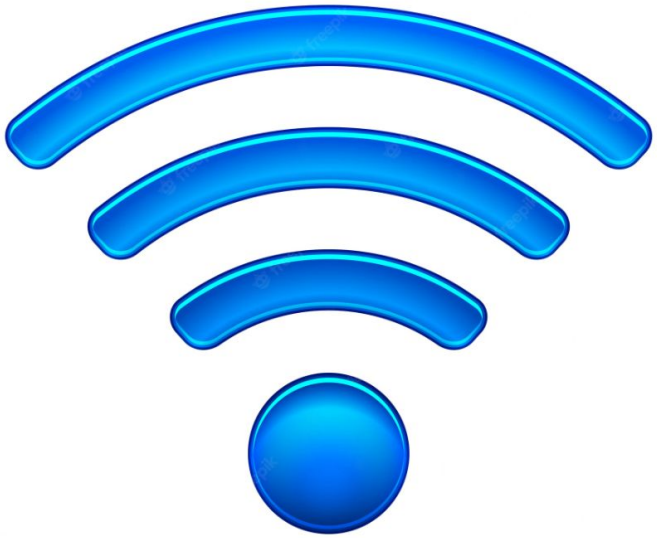
- Be aware of what shell you are running; if you enter a new system, it is possible that something other than Bash is the default.
- The **$SHELL** environment variable (discussed later) typically contains the executable of the running shell.
- If you encounter something other than **bash**, you can usually just run **bash** to start a new session

```
$ echo $SHELL
/usr/bin/zsh
$ bash
```

Situational Awareness

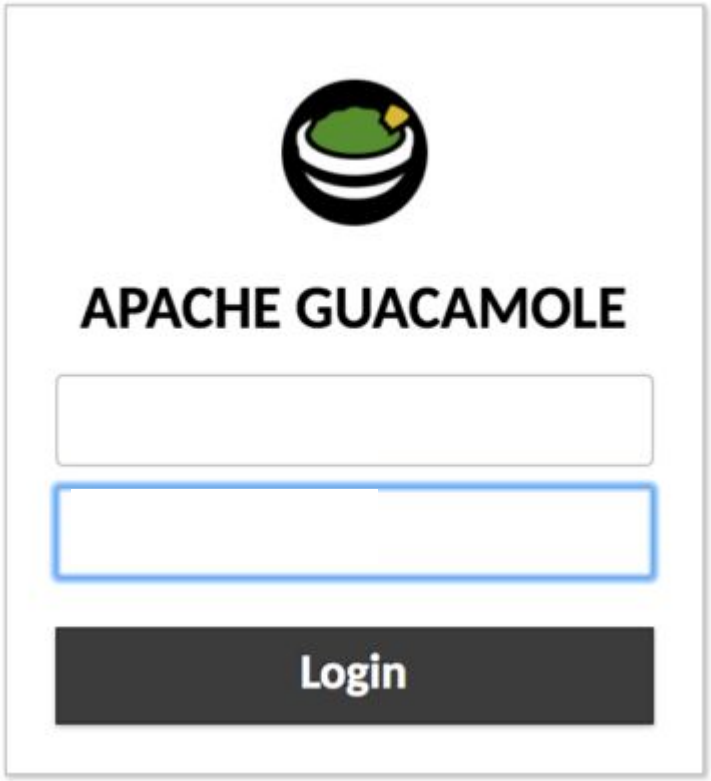
Lab – Virtual Machine Access

- Use the information provided by your instructor to access the Wi-Fi network we will use for our lab environment.
- Once connected, open your browser and go to the address that will be provided by your instructor.



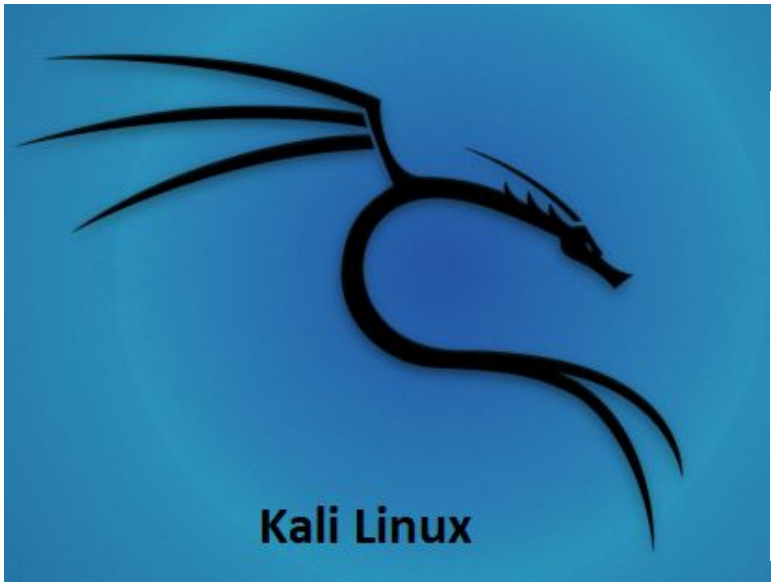
Lab – Virtual Machine Access

- Log into Apache Guacamole using the provided username and password



Lab – Virtual Machine Access

- You can now access the Kali Linux and Windows virtual machines.
- You can also access useful tools using the IP addresses provided by your instructors.



Windows®

Situational Awareness

- Launch a terminal in your VM
  - Click the blue Kali icon on the upper-left corner of the screen
  - type '**terminal**' into the search box
  - select the '**Terminal**' application to launch
- Type '**hostname**' into the terminal and press **enter** to run the command
- You should see the name of the computer you are currently using
- Type '**whoami**' into the terminal and press **enter** to run the command
- You should see your **login username** printed to the screen
  - **whoami** simply responds with the currently logged in user

```
$ hostname
computer name

$ whoami
username
```

Situational Awareness

- **pwd** (i.e., **present working directory)** returns the current path where you are operating on the file system
- Type **pwd** and press enter to see which directory you are working in
  - Note that by default when you enter a terminal/shell, you are placed in your user's **home** directory
- **ls** lists the contents of the directory you are currently in
- Type **ls** to see the contents of your home directory
  - try **ls -l** for more details

```
$ ls -l
total 4223088
-rw-r--r-- 1 student student   15235403 Jul  5 20:57  ex_1.json
-rw-r--r-- 1 student student    1511189 Jul  5 20:58  ex_2.json
...
```

Situational Awareness

- **cd** changes directories
- Change directories to Desktop with the command **cd Desktop**
- Now type **pwd** to see that the value has changed
- **..** is shorthand for 'parent directory'
- Type **cd ..** to return to the parent directory

```
$ cd Desktop
$ cd ..
```

Situational Awareness

- A note about specifying files and directories
  - When a **path** (to a file or directory) starts with '/', that is an **absolute path**
  - When a path does not start with a '/', it is **relative**
  - **Relative** paths are interpreted as relative to the **pwd**
  - **.** indicates the current directory
  - Note the two **cd** commands below

```
$ pwd
/home/student
$ cd Desktop
- IS EQUIVALENT TO -
$ cd /home/student/Desktop
```

Situational Awareness

- File and folder names in Linux are case-sensitive
  - Any objects whose names are spelled the same but capitalized differently will be treated as unique
  - "**FOLDER,**" "**Folder,**" & "**folder**" would be the names of three separate destinations in the file system
  - Note the two **cd** commands shown below

```
$ cd Desktop
- IS NOT EQUIVALENT TO -
$ cd desktop
```

Situational Awareness

- **~** is shorthand for your home directory
- You can **cd ~** to return to your home directory
- **~/** can also be used as the beginning of a path; everything after ~/ is relative to your home directory

```
$ cd Desktop/
$ pwd
/home/student/Desktop
$ cd ~
$ pwd
/home/student
$ cd ~/Desktop
$ pwd
/home/student/Desktop
```

Situational Awareness

- use **file** to get some basic information about files
- pass a **directory** and file will simple tell you that it is a directory
- pass a **filename** and **file** will attempt to give you information about the contents

```
$ file examples_folder/
examples_folder/: directory
$ file a_list.json
a_list.json: JSON data
$ file slowly.py
slowly.py: ASCII text
$ file data.bin
data.bin: data
$ file /usr/bin/bash
/usr/bin/bash: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=a0a6a0d2519910abdcb4ffb1312d9241a8e594cf, for GNU/Linux 4.4.0, stripped
```

Situational Awareness

- **cat** will print the contents of a file to the screen
  - This is analogous to **type** on windows
- cat is useful for **text files**
- Outputting **binary** files to the screen is rarely useful

```
$ cat input.json
{"a": 1, "b": 2}
{"a": 3, "b": 4}
$ cat slowly.py
import random
import sys
...
```

Commands Introduced

- **whoami**
  - Returns the currently logged in user
- **pwd**
  - Prints the "present working directory"
- **ls**
  - List directory contents
- **cd**
  - Change directory
- **cd .. , cd ~**
  - Shortcuts to cd to parent directory, home directory
- **file**
  - Utility to show basic file information
- **cat**
  - Print file contents to screen

Getting Help

Getting Help

- **--help** can typically be passed to a command to display **usage** information
- Try some of the commands we have covered with **--help** now

```
$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by
default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is
specified.
Mandatory arguments to long options are mandatory for short options
too.
-a, --all                  do not ignore entries starting with .
-A, --almost-all           do not list implied . and ..

...
```

Getting Help

- For more information, Linux provides **man** which, if requirements are installed (and they typically are), displays information about commands
- Try typing **man cat** now to see information about the **cat** command
- Note that in this view, you can scroll up and down with the **arrow keys**
- Exit by pressing '**q**'

Review 1

What can be used along with any command to provide a more detailed description and use of the command you are running?

What **ls** option displays all files (including hidden ones)?

What is the name of the machine?

What command displays the current directory you are working from? What is the current directory?

What is the proper **cd** command for returning to the home directory?

Input & Output

Input & Output

- Generally speaking, programs in Linux have three input/output streams:
  - **stdin** is input
  - **stdout** is normal output
  - **stderr** is an output stream for error
- Many (or even most) command line applications on Linux machines understand how to read from **stdin** when an input is expected.
  - For example, if no filename is passed to **cat**, it will print **stdin** back to the screen.
- In Bash, we can use **pipes** to redirect **stdout** from one process to **stdin** of another.
  - Pipes are represented by a '**|**' in bash

Input & Output

cat – Displays the contents of a file.

```
[user01@localhost ~]$ cat names1.txt
barry
mike
steve
charles
kaley
marcus
lucas
ian

[user01@localhost ~]$
```

more – Allows the user to scroll up and down through the page when viewing a large file. The **'more'** command also allows a user to scroll up and down when using the **'ls'** command if the listing is too large to display with the terminal window. Using the **spacebar** will scroll down one page, while using **enter** will only scroll a single line at a time.

```
[root@localhost etc]# ls | more
abrt
adjtime
aliases
alsa
alternatives
anaconda
anacrontab
asound.conf
at.deny
audit
authselect
autofs.conf
autofs_ldap_auth.conf
auto.master
auto.master.d
auto.misc
auto.net
auto.smb
avahi
bash_completion.d
bashrc
bindresvport.blacklist
binfmt.d
bluetooth
brlapi.key
brltty
brltty.conf
chkconfig.d
chromium
```

Input & Output

‘head’ and **‘tail’** commands are useful when reading long files in which you need to know the first portion or end portion of the file (for example, log files).

Remember that the ‘**cat’** command shows the entire contents of a file (or with the use of the **‘more’** command and **spacebar** to view additional portions of the file).

Head

- `head file` (Show the first 10 lines)
- `head -15 file` (Show the first 15 lines)

Tail

- `tail file` (Show the last 10 lines)
- `tail -15 file` (Show the last 15 lines)

Input & Output

```
[root@workstation01 log]# head messages-20210927
Sep 13 13:19:16 localhost kernel: Linux version 4.18.0-305.el8.x86_64 (mockbuild@x86-vm-07.build.eng.bos.redhat.com)
(gcc version 8.4.1 20200928 (Red Hat 8.4.1-1) (GCC)) #1 SMP Thu Apr 29 08:54:30 EDT 2021
Sep 13 13:19:16 localhost kernel: Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-305.el8.x86_64 root=UUID=55825
76d-15f4-45e2-ad33-63e147354e87 ro crashkernel=auto resume=UUID=fddec587-defa-4aa7-b1b1-36bada8f890a rhgb quiet
Sep 13 13:19:16 localhost kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
Sep 13 13:19:16 localhost kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
Sep 13 13:19:16 localhost kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
Sep 13 13:19:16 localhost kernel: x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256
Sep 13 13:19:16 localhost kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'compacted'
format.
Sep 13 13:19:16 localhost kernel: BIOS-provided physical RAM map:
Sep 13 13:19:16 localhost kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000097bff] usable
Sep 13 13:19:16 localhost kernel: BIOS-e820: [mem 0x000000000097c00-0x000000000009ffff] reserved
```

```
[root@workstation01 log]# tail messages-20210927
Sep 27 09:09:41 workstation01 NetworkManager[1182]: <info>  [1632751781.4930] dhcp4 (ens160): state changed bound ->
extended, address=192.168.88.147
Sep 27 09:09:41 workstation01 systemd[1]: Starting Cleanup of Temporary Directories...
Sep 27 09:09:41 workstation01 dbus-daemon[990]: [system] Activating via systemd: service name='org.freedesktop.nm_dis
patcher' unit='dbus-org.freedesktop.nm-dispatcher.service' requested by ':1.16' (uid=0 pid=1182 comm="/usr/sbin/Netwo
rkManager --no-daemon " label="system_u:system_r:NetworkManager_t:s0")
Sep 27 09:09:41 workstation01 systemd[1]: Starting Network Manager Script Dispatcher Service...
Sep 27 09:09:41 workstation01 dbus-daemon[990]: [system] Successfully activated service 'org.freedesktop.nm_dispatche
r'
Sep 27 09:09:41 workstation01 systemd[1]: Started Network Manager Script Dispatcher Service.
Sep 27 09:09:41 workstation01 systemd[1]: systemd-tmpfiles-clean.service: Succeeded.
Sep 27 09:09:41 workstation01 systemd[1]: Started Cleanup of Temporary Directories.
Sep 27 09:09:51 workstation01 systemd[1]: NetworkManager-dispatcher.service: Succeeded.
Sep 27 09:14:13 workstation01 chronyd[1017]: Selected source 38.229.60.9
[root@workstation01 log]#
```

Input & Output

Operator	Function	Example
;	Process the command on the right after you're done processing the command on the left.	echo one ; echo two
>	Place the output of the thing on the left in the **empty** file named on the right.	ls /home/me > myfilesonce.txt ; ls /home/me > myfilesonce.txt
>>	**Append** the output of the thing on the left to the end of the **existing** file on the right.	ls /home/me > myfilestwice.txt ; ls /home/me >> myfilestwice.txt
<	Use the file on the right as the standard input of the command on the left.	cat < sourcefile > targetfile
|	Pipe the standard output of the thing on the left into the standard input of the thing on the right.	echo "test123" | mail -s "subjectline" emailaddress

Most Linux commands can use the operators listed above.

Text Editors

vi: Text editor that has two modes (insert mode and command mode)

- **Insert mode**: used to edit/create new and existing files
- **Command mode**: used to perform functions such as copying and pasting, saving documents, or exiting vi when done

Command	Purpose
vi [filename]	opens specified file using the vi editor
*i	switch to insert mode and insert text before cursor position
esc	exit insert mode (switch to command mode)
/[string]	search current file for specified string
yy	copy the current line
p	paste the copied line
:w	save changes made to the file
:q	exit vi

Text Editors

nano: Text editor with only one mode (insert mode). Command instructions are given using keyboard shortcuts rather than a dedicated command mode.

Command	Purpose
nano [filename]	opens specified file using the nano editor
ctrl+w	search
alt+a	begin selecting text
alt+6	copy selected text
ctrl+k	cut selected text
ctrl+u	paste selected text
ctrl+x	save and close current file
ctrl+o	save current file and continue editing

Review 2

Which command and option would you use to see the last 25 lines of a file?

What redirection operator is used to place an output from one command into the end of an existing file?

When using the **'more'** command, what is the difference between pressing the **enter** button and the **spacebar**?

Name the primary difference(s) between **vi** and **nano**.

A note about permanence

- As we start making changes to the file system, remember that there is no Recycle Bin here.
- If you delete critical files, you could severely impact or disable the machine on which you are working.
- Use caution when making changes.

Making Changes

Making Changes - Basic Operations

- Before we start, return to your home directory with **cd ~**
- Make new directories with **mkdir**
- By default, **mkdir** will only create a single directory; use **-p** to create all directories on a path that do not exist
- Notice the use of **ls -R** below; that tells **ls** to '**recurse**' into subdirectories

```
$ mkdir example
$ mkdir example/first/second
mkdir: cannot create directory ‘example/first/second’: No such file or directory
$ mkdir -p example/first/second
$ ls -R example
example:
first

example/first:
second

example/first/second:
```

Making Changes - Basic Operations

- **cp** copies a file
- **cp** -r copies a directory recursively
- **cp** ***from to***

```
$ echo "abc" > alpha.txt
$ cp alpha.txt example/first/second/
$ cat example/first/second/alpha.txt
abc
$ cp example/first example/copyoffirst
cp: -r not specified; omitting directory 'example/first'
$ cp -r example/first example/copyoffirst
$ ls example/copyoffirst/
second
```

Making Changes - Basic Operations

- **mv** moves files and directories
- **mv** ***from to***

```
$ mv alpha.txt moved.txt
$ cat moved.txt
abc
$ mv example movedexample
$ ls movedexample
copyoffirst  first
```

Making Changes - Basic Operations

- **rm** removes or **deletes** things
- Delete files with **rm filename**
- Delete directories by passing the **-r** (recursive) argument
- This is a good time to remember the fact that changes are **permanent**

```
$ rm moved.txt
$ rm -r movedexample/copyoffirst/
$ ls movedexample/
first
```

Tools - grep

grep – general regular expression parser

grep is a search command for Linux that is used to search for text strings and regular expressions within one or more files.

grep [options] pattern [files]

-b Display the block number at the beginning of each line.
-c Display the number of matched lines.
-h Display the matched lines, but do not display the filenames.
-i Ignore case sensitivity.
-l Display the filenames, but do not display the matched lines.
-n Display the matched lines and their line numbers.
-s Silent mode.
-v Display all lines that do **not** match.
-w Match whole word.

Tools - grep

- Use **grep** to filter lines of text to match (or not match) regular expressions
- The full scope of regular expressions is beyond the scope of this class
- We'll cover a few common uses

```
$ echo -e "cat1\nbat2\ndog3" | grep at
# Matches any line that contains 'at'
$ echo -e "cat1\nbat2\ndog3" | grep -v at
# Matches any line that does not contain 'at'
$ echo -e "cat1\nbat2\ndog3" | grep [12]
# Matches any line that contains 1 or 2
$ echo -e "cat1\nbat2\ndog3" | grep "^b"
# Matches any line that starts with 'b'
```

Tools - grep

- Try the following commands in your terminal window
- **man grep** to get more information on how grep works

```
$ echo -e "cat1\nbat2\ndog3" | grep at
# Matches any line that contains 'at'
$ echo -e "cat1\nbat2\ndog3" | grep -v at
# Matches any line that does not contain 'at'
$ echo -e "cat1\nbat2\ndog3" | grep [12]
# Matches any line that contains 1 or 2
$ echo -e "cat1\nbat2\ndog3" | grep "^b"
# Matches any line that starts with 'b'
```

Tools - grep

Search file for a user:

```
$ grep mike /etc/passwd
```

Search file ignoring word case:

```
$ grep -i “mike" /etc/passwd
```

Search all files and directories recursively under given directory:

```
$ grep -r “mike" /etc/
```

Search for a specific word in file:

```
$ grep -w “mike" /Documents/names.txt
```

Search for two different words in file:

```
$ grep -w ‘mike|steve’ /Documents/names.txt
```

Count lines that matched in file:

```
$ grep -c ‘mike’ /Documents/names.txt
```

Tools - grep

Dot (.) – matches 1 character
Asterisk (*) – matches multiple characters
Examples:

`grep b.g myfile` → finds the words “big,” “bag”
`grep b*k myfile` → finds the word “back,” “buck,” “book”

System Management

System Information

uptime – displays time; how long the system has been running; number of active users; and system load averages for the past 1, 5, and 15 minutes

```
09:10:18 up 106 days, 32 min, 2 users, load average: 0.22, 0.41, 0.32
```

free – displays the amount of free and used space in memory

```
dave@howtogeek:~$ free
              total        used        free      shared  buff/cache
available
Mem:        2038576      670716      327956       14296     1039904
  1187160
Swap:       1557568      769096      788472
```

df – displays the amount of free space in storage that the user can access

```
[hydn@alien ~]$ df
Filesystem      Size  Used Avail Use% Mounted on
dev             7.8G     0  7.8G   0% /dev
run             7.8G  1.6M  7.8G   1% /run
/dev/md0        218G   47G  161G  23% /
tmpfs           7.8G  447M  7.4G   6% /dev/shm
tmpfs           4.0M     0  4.0M   0% /sys/fs/cgroup
tmpfs           7.8G  9.4M  7.8G   1% /tmp
/dev/sda1       511M  344K  511M   1% /boot/efi
tmpfs           1.6G   76K  1.6G   1% /run/user/1000
```

Processes

ps – provides a listing of the current processes running on the machine.

```
[user01@localhost ~]$ ps
    PID TTY          TIME CMD
   3006 pts/1    00:00:00 bash
   3442 pts/1    00:00:00 ps
[user01@localhost ~]$
```

lsof (LiSt Open Files) – use to determine what files are opened by what process.

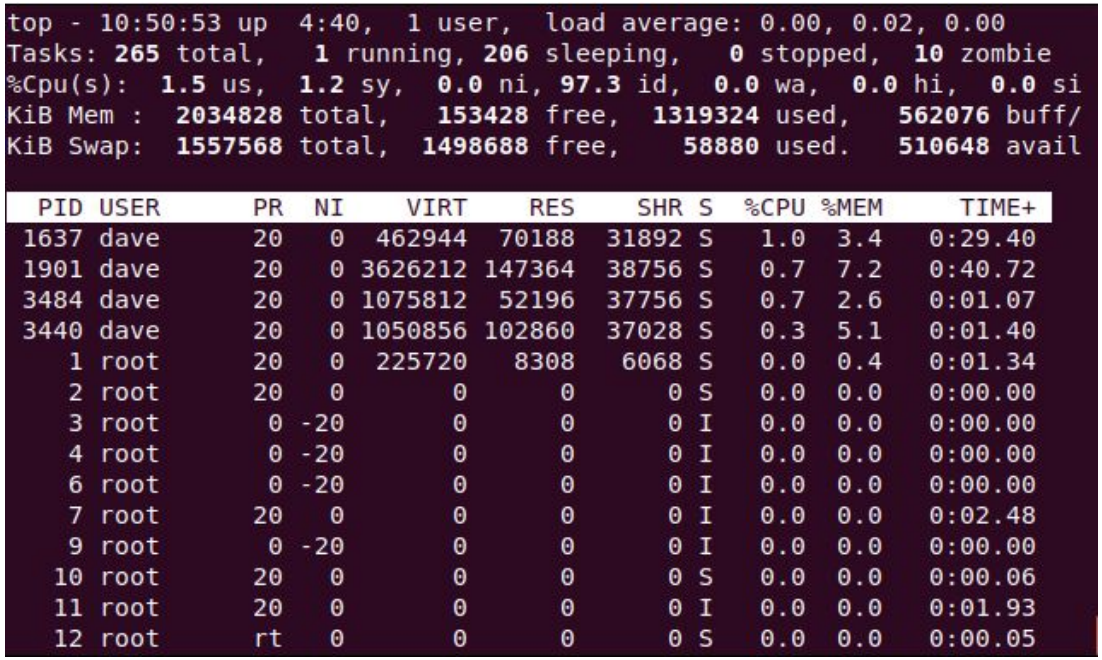
```
[root@localhost user01]# lsof
```

```
  PID  TID TASKCMD           USER   FD      TYPE             DEVICE SIZE/OFF       NODE NAME
    1                        root  cwd       DIR              259,3      224        128 /
    1                        root  rtd       DIR              259,3      224        128 /
    1                        root  txt       REG              259,3  1588952   34236096 /usr/lib/systemd/systemd
    1                        root  mem       REG              259,3   147336     267587 /usr/lib64/libnl-3.so.200.26.0
    1                        root  mem       REG              259,3   549824     267595 /usr/lib64/libnl-route-3.so.200.26.0
    1                        root  mem       REG              259,3   131056    1172329 /usr/lib64/libibverbs.so.1.11.32.0
```

Processes

kill – terminates the specified process or process group

top – displays uptime information, as well as task counts, CPU utilization, and active processes



Processes

systemctl –allows you to manage services in ways such as starting and stopping them, setting which services run automatically when the system boots, and restarting active services

```
$ sudo systemctl start application.service
```

```
$ sudo systemctl stop application.service
```

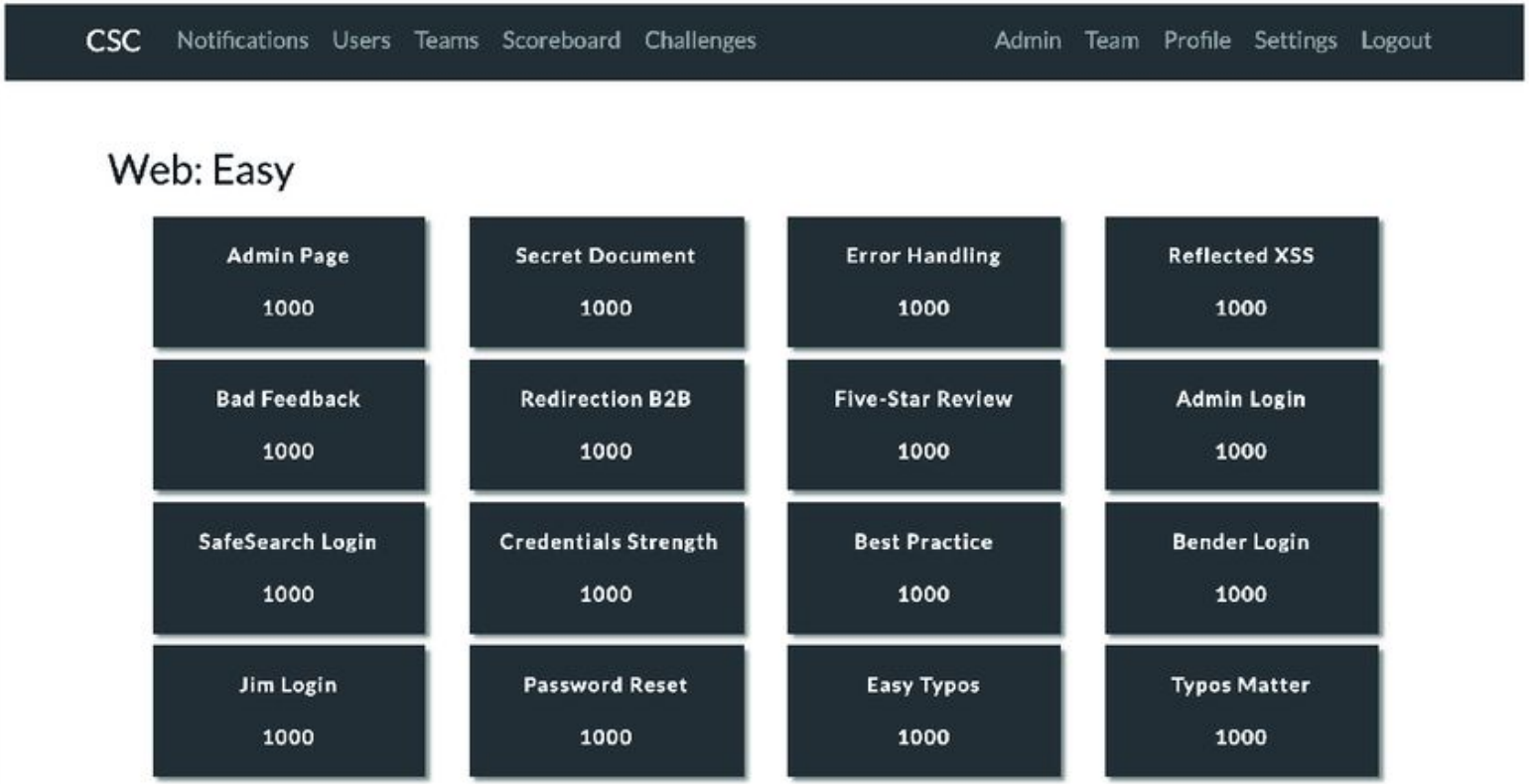
```
$ sudo systemctl enable application.service
```

```
$ sudo systemctl disable application.service
```

```
$ sudo systemctl restart application.service
```

Available Tools - CTFd

- **CTFd** is a tool used to provide a controlled environment for building and performing capture-the-flag challenges.
- Challenges can fall into a number of categories including Networking, Web-based, and Exploitation.

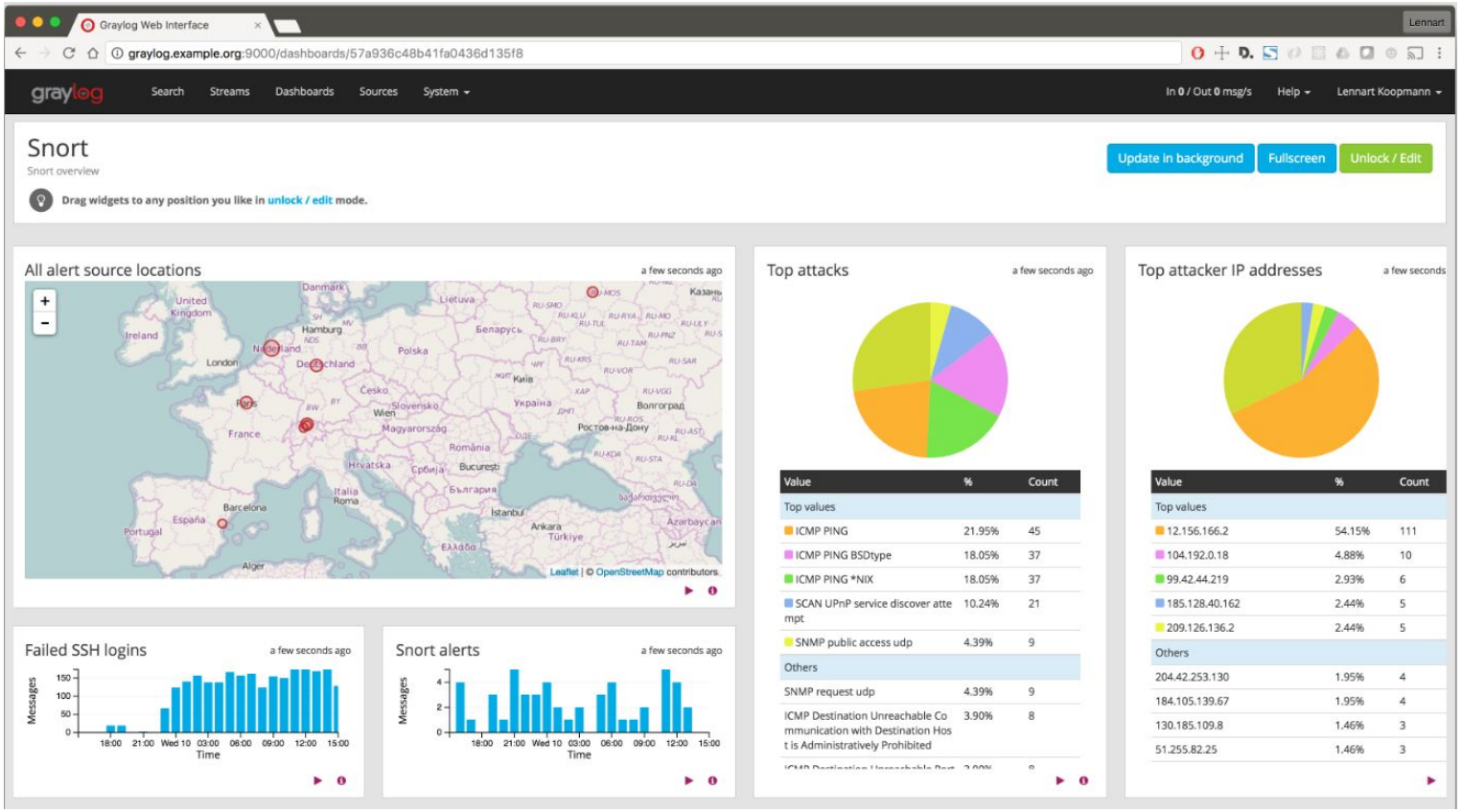


Available Tools - LibreTranslate

- **LibreTranslate** is an open-source tool that can translate text into a wide variety of different languages.
- This tool is being used to assist in translation between instructors and students throughout this course.

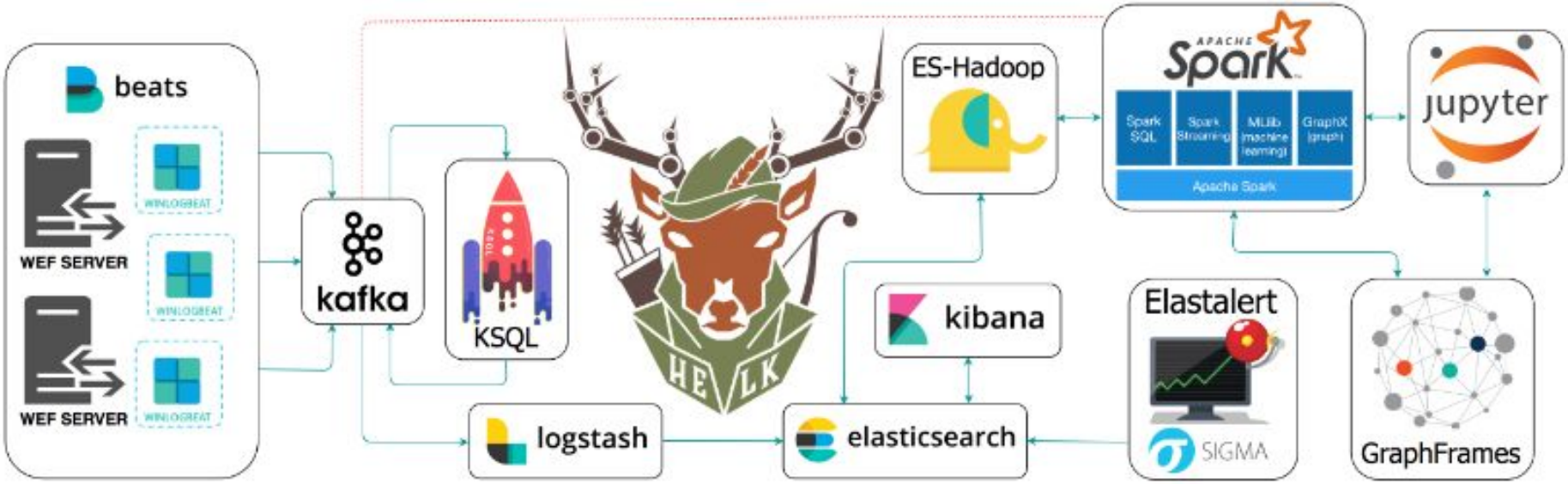
Available Tools - Graylog

- **Graylog** is a Security Information and Event Management (SIEM) tool that is used to collect and manage log information that could reveal potential threats.
- Offers an open-source version.



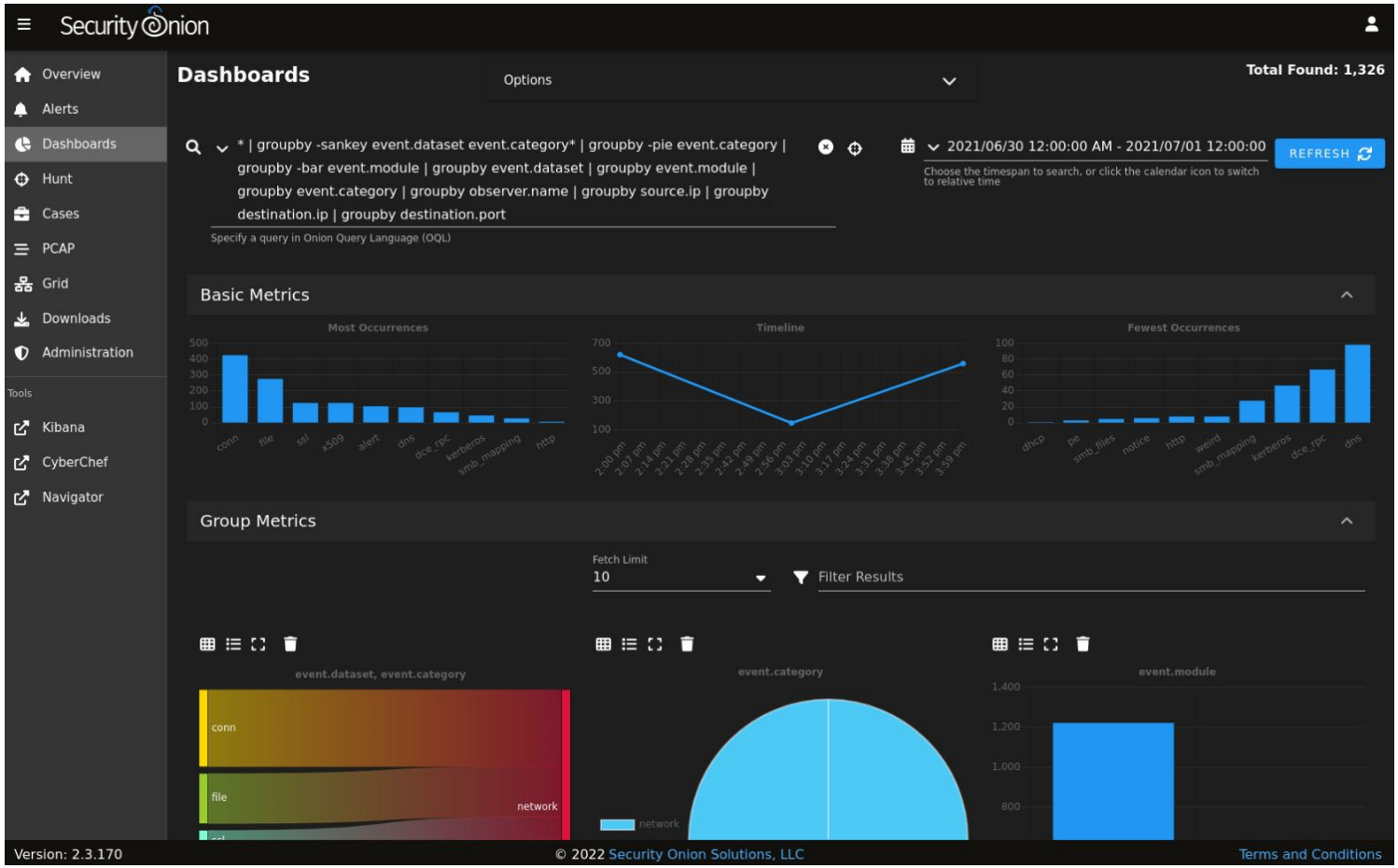
Available Tools - HELK

- **Hunting ELK** (HELK) is an open-source solution for threat hunting and analysis.
- HELK can perform analysis in conjunction with other tools, such as SQL, Jupyter Notebook, and Kibana.



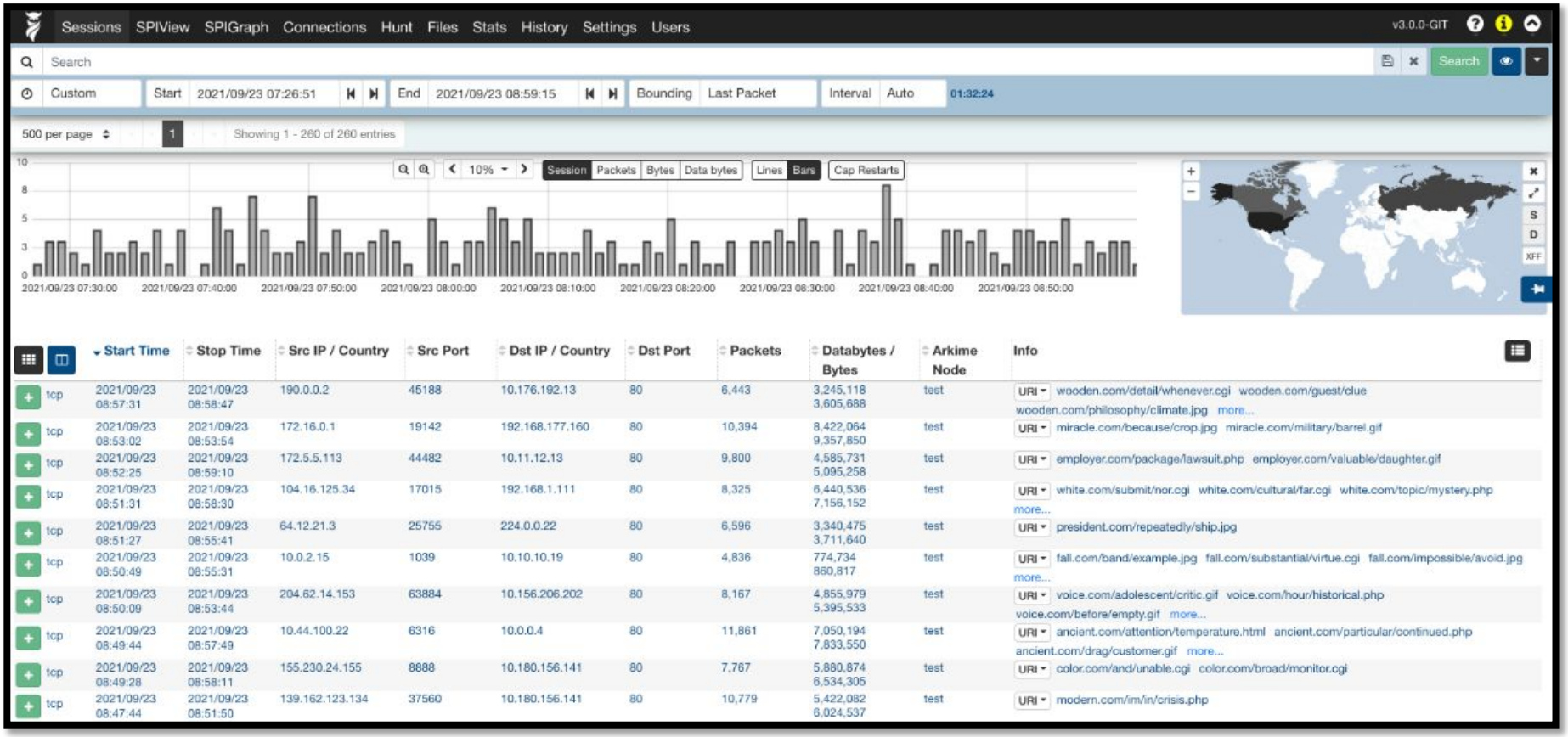
Available Tools – Security Onion

- **Security Onion** is an open-source SIEM that (like Graylog and Splunk) provides the users with an overview of various system logs that can be used in threat detection and network behavior analysis.



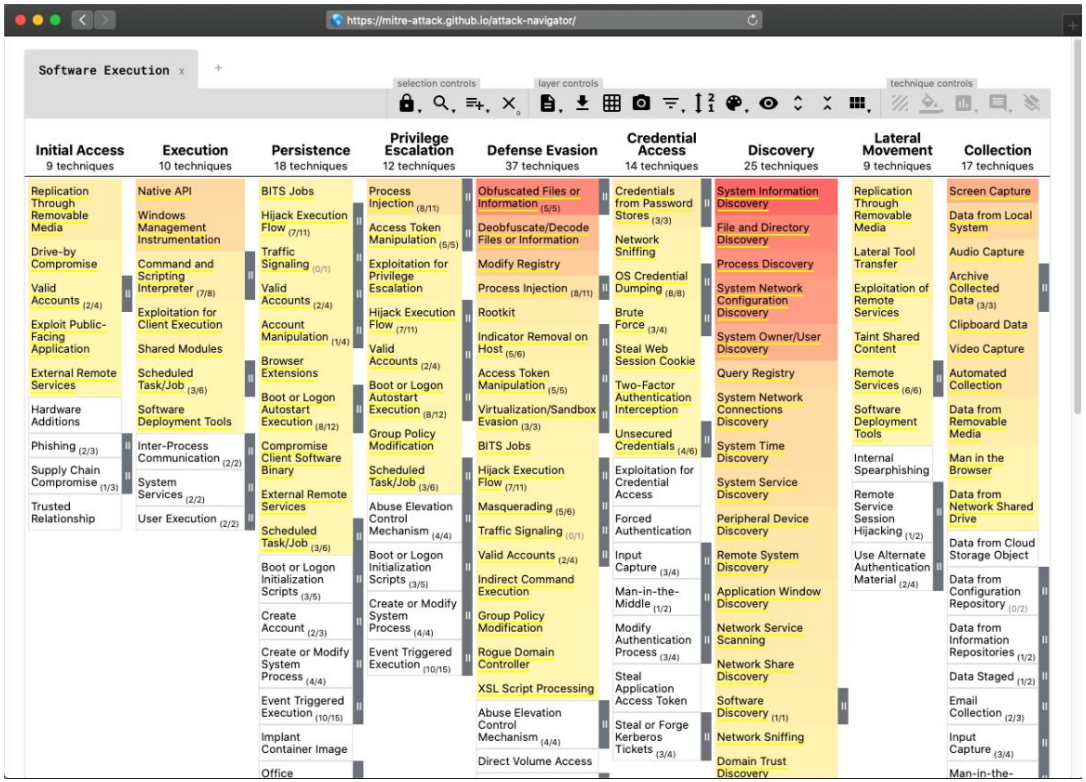
Available Tools – Arkime

- **Arkime** is a tool used for capture and analysis of network packets.
- Allows you to study the information related to any specific packets that may have been flagged by other logging solutions.



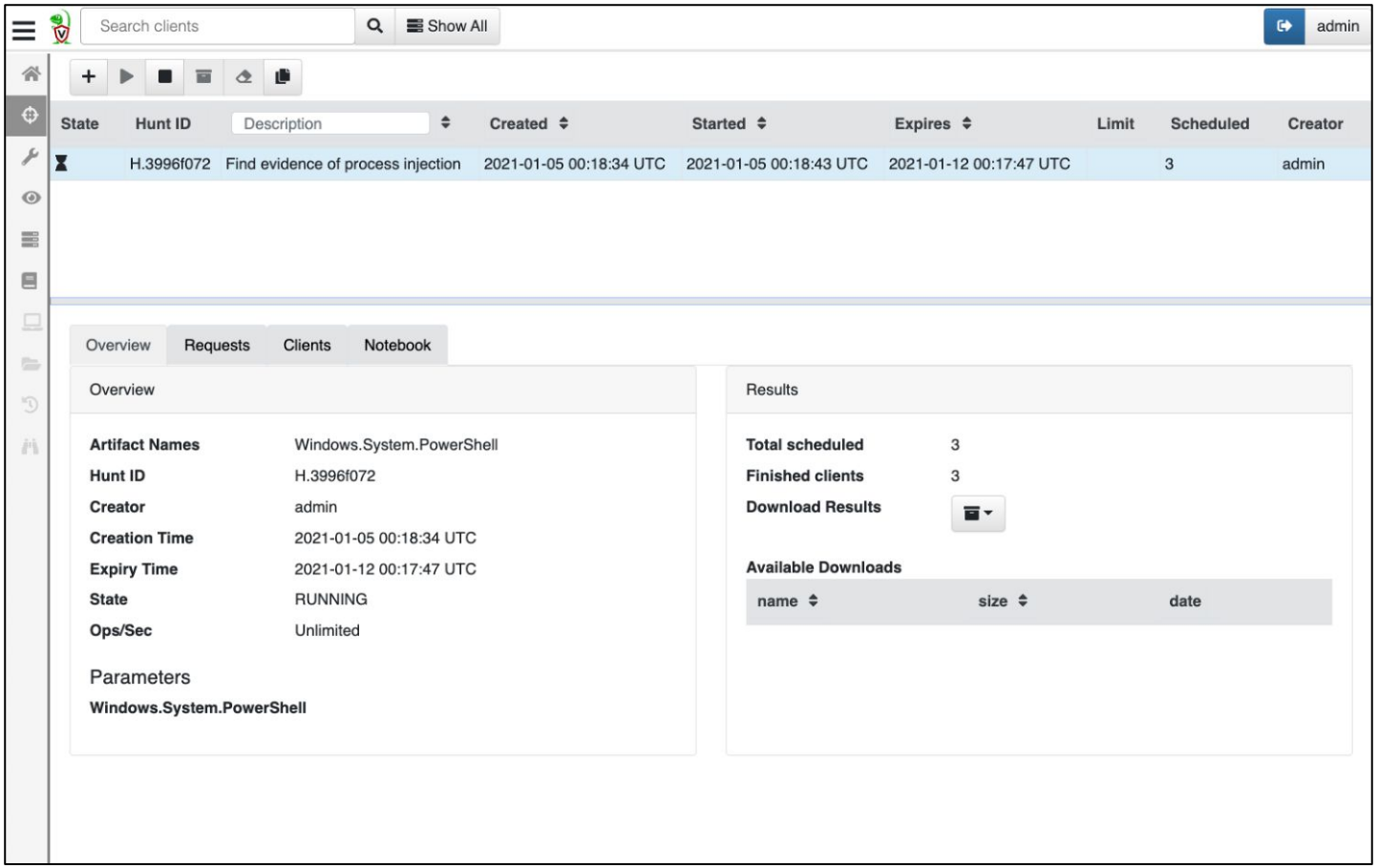
Available Tools – MITRE ATT&CK

- The ATT&CK framework is used by MITRE to categorize the various techniques that a threat actor may attempt to use when compromising a secure system.
- The ATT&CK Navigator tool allows you to view this framework and annotate it as you discover more information about the attack or threat indicator that you are attempting to analyze.



Available Tools – Velociraptor

- **Velociraptor** is an open-source Endpoint Detection and Response (EDR) solution that can be used to monitor and collect data from a variety of devices throughout your network.



Networking

Networking

- Overall networking is usually governed by `/etc/rc.d/init.d/network`
- Network device/interface configurations are either in `/etc/sysconfig/networking` or in `/etc/sysconfig/network-scripts`
- Can either edit manually or use utilities to manage.

- As with most things, GUI tools are available.
- Similar to TCP/IP configuration in Windows.
- More advanced operations (bridging, NAT, advanced routing) take a little more configuration.
- Default firewall software is **firewalld** or **iptables.**

ifconfig:

- Displays or alters network device configs.
- With no options, shows interface's config.
- If interface is omitted as well, show all configs.
- Options include flags, IP address, subnet mask, etc.

Networking

ifconfig – used to view and configure IP and subnets on the local Linux machine.

```
[user01@localhost ~]$ ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.88.147  netmask 255.255.255.0  broadcast 192.168.88.255
        inet6 fe80::20c:29ff:fe0e:6824  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:0e:68:24  txqueuelen 1000  (Ethernet)
        RX packets 1482  bytes 2030120 (1.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 464  bytes 33136 (32.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2  bytes 140 (140.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 140 (140.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
[root@localhost user01]# ifconfig ens160 192.168.88.148 netmask 255.255.255.0
```

Networking

netstat – lets you discover which sockets are connected and which sockets are listening.

```
[root@localhost user01]# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp       32      0 localhost.localdo:54016 oscp-router03.gno:https CLOSE_WAIT
udp        0      0 localhost.locald:bootpc 192.168.88.254:bootps   ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    29194    /var/run/chrony/chronyd.sock
unix  3      [ ]         DGRAM                    13183    /run/systemd/notify
unix  2      [ ]         DGRAM                    13185    /run/systemd/cgroups-agent
unix  2      [ ]         DGRAM                    44164    /run/user/1000/systemd/notify
unix  7      [ ]         DGRAM                    13199    /run/systemd/journal/socket
unix  24     [ ]         DGRAM                    13223    /run/systemd/journal/dev-log
unix  3      [ ]         STREAM     CONNECTED     45304    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     132437   /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     48991    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     48056    /run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     40288
```

ping – verifies IP-level connectivity to another TCP/IP computer by sending **Internet Control Message Protocol (ICMP)** Echo Request messages.

```
[user01@workstation01 ~]$ ping -c 4 192.168.88.148
PING 192.168.88.148 (192.168.88.148) 56(84) bytes of data.
64 bytes from 192.168.88.148: icmp_seq=1 ttl=64 time=0.208 ms
64 bytes from 192.168.88.148: icmp_seq=2 ttl=64 time=0.197 ms
64 bytes from 192.168.88.148: icmp_seq=3 ttl=64 time=0.238 ms
64 bytes from 192.168.88.148: icmp_seq=4 ttl=64 time=0.206 ms

--- 192.168.88.148 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3065ms
rtt min/avg/max/mdev = 0.197/0.212/0.238/0.018 ms
[user01@workstation01 ~]$
```

Networking

traceroute – shows the IP addresses used to transfer a packet between the user's computer and the specified host when attempting to ping (or establish a normal connection)

```
prabhakar@Inspiron-3542:~$ traceroute google.com
traceroute to google.com (172.217.26.206), 30 hops max, 60 byte packets
 1  192.168.43.45 (192.168.43.45)  2.014 ms  2.313 ms  2.588 ms
 2  * * *
 3  10.45.1.230 (10.45.1.230)  75.449 ms  115.244 ms  115.224 ms
 4  10.45.8.178 (10.45.8.178)  93.856 ms  115.138 ms  93.822 ms
 5  10.45.8.187 (10.45.8.187)  115.116 ms  115.106 ms  115.070 ms
 6  * * *
 7  218.248.235.141 (218.248.235.141)  120.589 ms  108.033 ms  106.962 ms
 8  218.248.235.142 (218.248.235.142)  114.489 ms * *
 9  72.14.211.114 (72.14.211.114)  98.076 ms  93.232 ms  93.781 ms
10  108.170.253.113 (108.170.253.113)  98.688 ms  91.388 ms 108.170.253.97 (108.170.253.97)  107.241 ms
11  74.125.253.69 (74.125.253.69)  95.120 ms 72.14.237.165 (72.14.237.165)  102.594 ms  103.137 ms
12  maa03s23-in-f14.1e100.net (172.217.26.206)  101.794 ms  97.987 ms  97.165 ms
```

Netcat

Netcat is a networking tool that can be used for a number of things, including sending files between remote systems, scanning a target's ports to see which (if any) are open and available for connection, and sending HTTP requests to websites.

```
manav@manav-VirtualBox:~$ nc -z -v 127.0.0.1 1233-1240
nc: connect to 127.0.0.1 port 1233 (tcp) failed: Connection refused
Connection to 127.0.0.1 1234 port [tcp/*] succeeded!
Connection to 127.0.0.1 1235 port [tcp/*] succeeded!
nc: connect to 127.0.0.1 port 1236 (tcp) failed: Connection refused
Connection to 127.0.0.1 1237 port [tcp/*] succeeded!
nc: connect to 127.0.0.1 port 1238 (tcp) failed: Connection refused
nc: connect to 127.0.0.1 port 1239 (tcp) failed: Connection refused
nc: connect to 127.0.0.1 port 1240 (tcp) failed: Connection refused
```

```
manav@manav-VirtualBox:~$ printf "GET /nc.1 HTTPs/1.1\r\nHost: www.geeksforgeeks
.org\r\n\r\n" | nc www.geeksforgeeks.org 80
HTTP/1.0 400 Bad Request
Server: AkamaiGHost
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 216
Expires: Sat, 25 Apr 2020 00:12:15 GMT
Date: Sat, 25 Apr 2020 00:12:15 GMT
Connection: close

<HTML><HEAD>
<TITLE>Bad Request</TITLE>
</HEAD><BODY>
<H1>Bad Request</H1>
Your browser sent a request that this server could not understand.<P>
Reference&#32;&#35;7&#46;2daef82d&#46;1587773535&#46;0
</BODY>
</HTML>
```

Nmap Discovery Scan

- To run an **nmap** discovery scan, open a terminal and run:
  - **nmap -O {ip address/subnet mask} (e.g., nmap -O 192.168.1.0/24)**

- The **-O** flag enables OS detection.

- After running an **nmap** discovery scan, the next step is to document the inventory found and conduct a physical inventory.

- A physical inventory is walking through the building and matching your discovery scan results to devices and locations and getting information such as serial numbers. This will help find any devices that may be unplugged from the network and any devices that do not belong.

Nmap Port and Service Scan

- The other part of knowing what is on your network is knowing what ports, protocols, and services are exposed to the outside world from your system.

- Nmap can provide what ports are open, closed, or filtered. To conduct an nmap port and service scan on a single host, open a terminal and run:
- **nmap –sV {ip address}** **(e.g. nmap 192.168.1.1)**

- Results will display closed ports, open or filtered ports, and services that are detected.

- To run the same scan on a full subnet, use the same nmap command with IP address/subnet mask similar to the discover scan.

Common tcpdump options

- **-i:** identify interface on which to capture traffic
- **-r:** read packets from an existing PCAP file
- **-n:** suppress name resolution for host address and port # – **ALWAYS USE THIS OPTION**
- **-A:** print packet in ASCII
- **-s:** packet snap length, number of bytes to capture from each packet
- **-c:** specify number of packets to capture
- **-C:** Specify packet capture file write size
- **-G:** rotate PCAP files in number of seconds; requires timestamp format in filename
- -**w:** write to PCAP to a file
- **-F:** load BPF filters from a file

tcpdump Examples

- Capture on interface **eth0**, suppress name resolution of IP address and port number, write to **stdout**

```
tcpdump -n -n -i eth0
```

- Capture on interface **eth0**, suppress name resolution of IP address and port number, write to file named **output.pcap**

```
tcpdump -n -n -i eth0 -w output.pcap
```

- Read from output.pcap, filter on TCP/UDP port 53, and write those packets to **dns.pcap**

```
tcpdump -n -n -r output.pcap -w dns.pcap (tcp or udp) and port 53
```

- Capture on interface **eth0**, rotate pcap after 150MB, and write to file **output.pcap**; each subsequent filename will be appended with a number sequentially (output.pcap1, output.pcap2, etc.)

```
tcpdump -n -n -i eth0 -C 150 -w output.pcap
```

Review 3

Which command would you use to view and configure IP and subnets on the local Linux machine?

What is the difference between **ping** and **traceroute**?

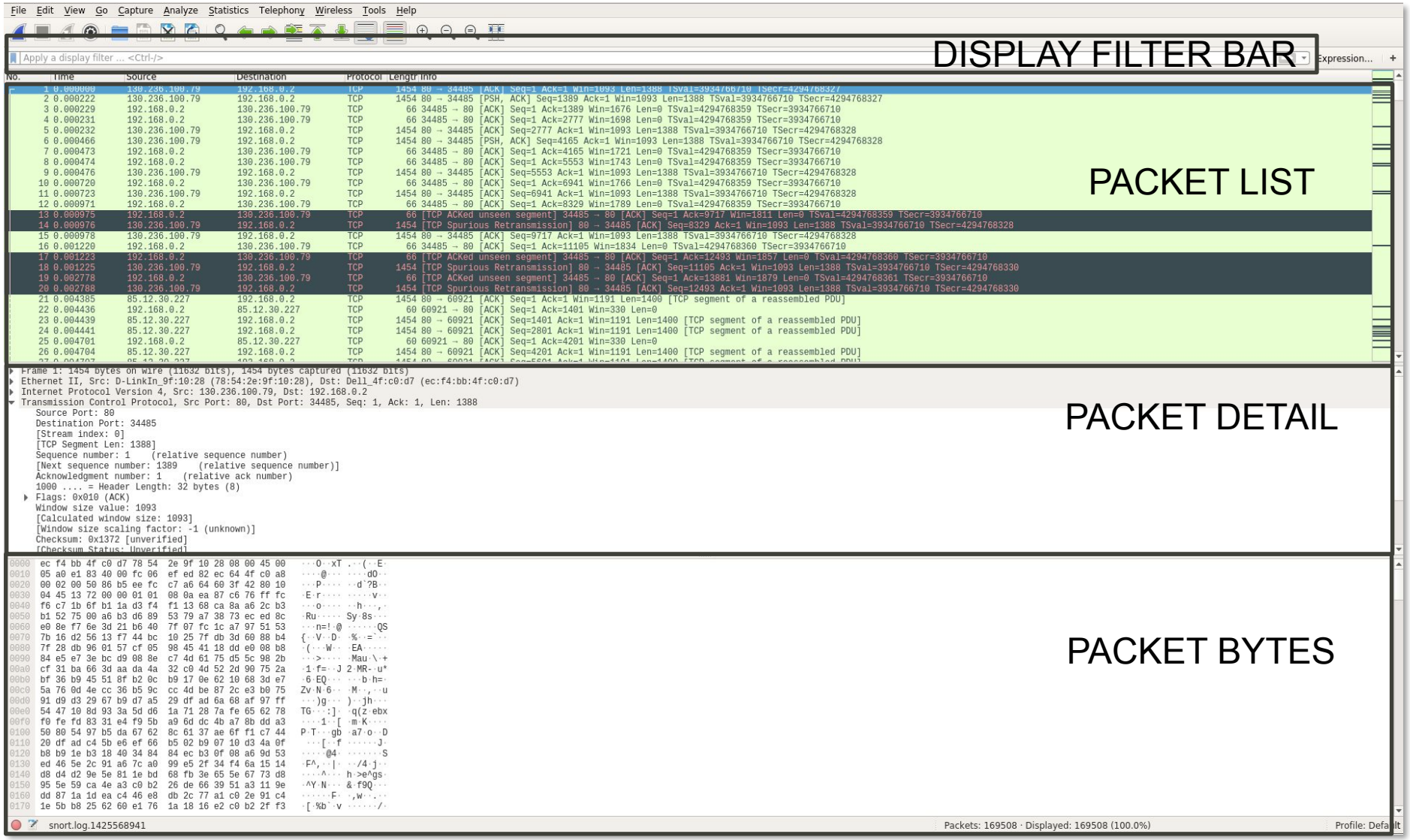
What are two networking tools that can be used to scan for open ports?

Wireshark

- Wireshark is perhaps the most well-known network protocol analyzer.
  - Robust GUI for analysis
  - Provides deep inspection of hundreds of protocols
  - Can perform live capture and offline analysis
  - Cross-platform, compatible with many operating systems
  - Decryption support for several protocols, HTTPS, with key
  - Supports several capture formats (libpcap, Pcap, PcapNG, etc.)

- Can struggle to load large PCAP files
- Not suitable for live capture over long periods of time; will eventually crash

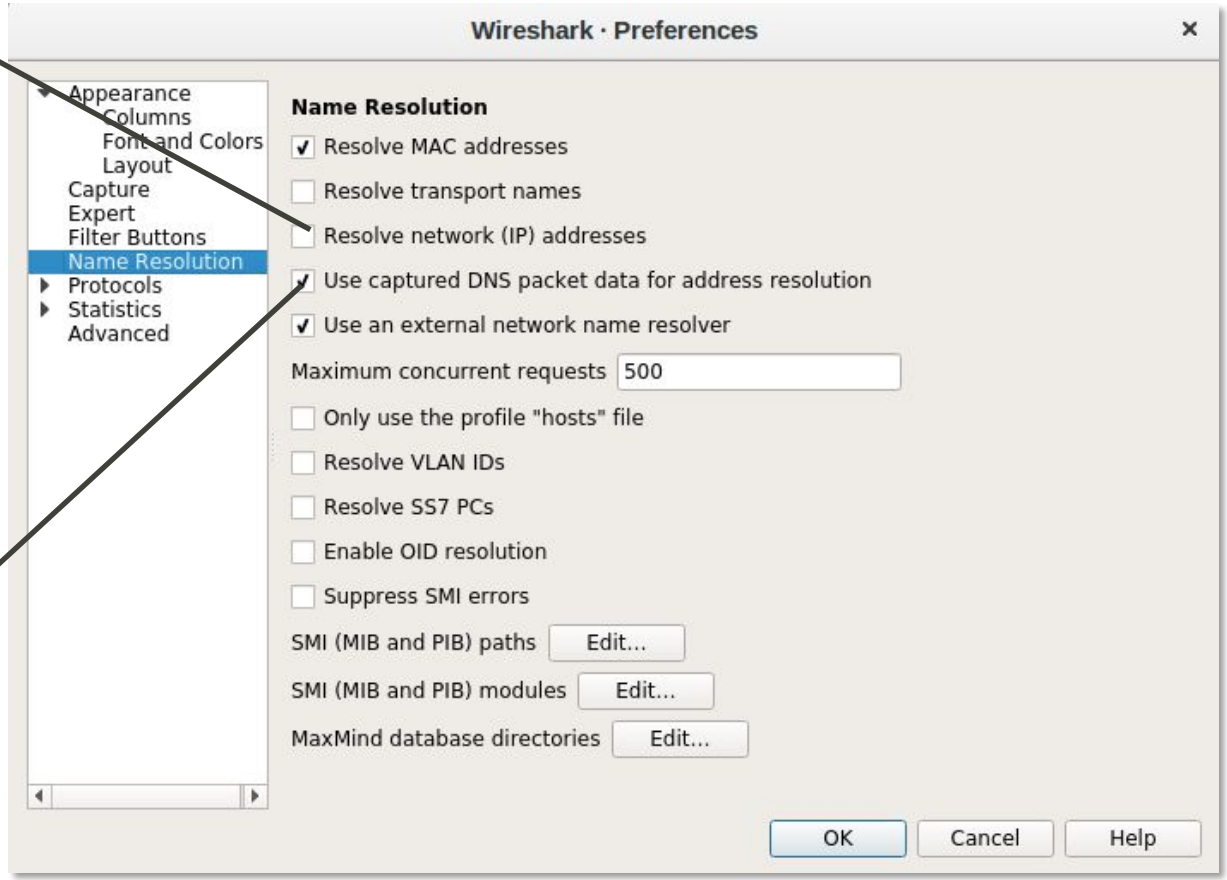
Wireshark Interface



Wireshark Name Resolution

Will query host's DNS server for every IP address in the capture; BAD OPSEC

Will use DNS requests in the same packet capture to resolve IP addresses; does not use external DNS server; name resolution is relevant to the time of the capture; IP addresses/hostnames change over time



Lab 2 – Wireshark Analysis

Lab 2 – Wireshark Analysis

- For this lab, we will be using Wireshark to analyze packet information related to the WannaCry ransomware.

- Using the instructions provided to you by the instructor, you will be able to complete various exercises that teach you how to use the Wireshark tool.