# Exploitation

Cybersecurity

Penetration Testing Day 3

# Class Objectives

By the end of today's class, you will be able to:

Run scripted Shellshock exploits.

Consult the Exploit-DB database to research publicly disclosed exploits.

Search for exploits and shellcode using SearchSploit.

# This Week

This week, we moved through the first three stages of pen testing:

01

Defining the purpose and scope of the test, and conducting passive and active reconnaissance.

02

Once we have access to the organization's infrastructure, we can perform scanning and enumeration techniques to find valuable targets

03

After scanning networks for vulnerabilities, we can execute the exploits that we know an organization is vulnerable to.

Today, we will exploit a vulnerability called Shellshock, also known as the bash bug.

# Shellshock

Shellshock is a remote code execution (RCE) vulnerability that allows attackers to execute arbitrary Bash code on vulnerable targets.

With Shellshock, attackers can perform almost any Bash command, including:

01 Download sensitive data.

02 Send and receive shells to and from the target.

03 Backdoor the victim.

# Shellshock

Shellshock is made possible due to a vulnerability in the way servers parse HTTP requests. Let's look at that vulnerability:

- One way that websites are served is through **Common Gateway Interface (CGI),** which defines a set of rules that allow web clients to run scripts on a server.
- If a client requests a script, such as `/cgi-bin/status.sh`, the server will run the script `status.sh`. Then it will send the output back in an HTTP response.
- Sometimes, the server needs to use HTTP headers to run CGI scripts.
- Servers will load HTTP request headers as Bash environment variables and run the CGI scripts. The scripts can then access the HTTP headers by reading the variables.

# Shellshock

For example, sending the following request results in the server creating an **`HTTP_USER_AGENT`** variable equal to **`curl`**.

Bash does not sanitize headers before loading them as environment variables.

This means it loads the value sent in the header.

```
GET /index.html HTTP/1.1
Host: example.com
User-Agent: curl
Connection: keep-alive
```

# Shellshock

For example, the request below results in **HTTP_USER_AGENT** being set to `() { :;};`. This is cryptic, but in vulnerable versions of Bash, this creates a code function that does nothing.

```
GET /index.html HTTP/1.1
Host: example.com
User-Agent: () { :;};
Connection: keep-alive
```

Bash interprets the value of this header as code, allowing us to execute arbitrary code on the target.

Shellshock can be exploited when we include malicious code after the seemingly arbitrary `() { :;};` string.

# Shellshock

For example, malicious codes can open TCP connections back to your machine or show user passwords.

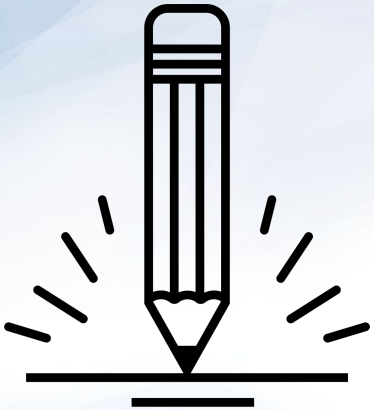
**Note the syntax:**

```
GET /index.html HTTP/1.1
Host: example.com
User-Agent: () { :;}; /bin/bash -c
'cat /etc/passwd'
Connection: keep-alive
```

**`/bin/bash -c 'command'`** runs the string **`'command'`** as a bash command.

Using **`/bin/bash -c`** helps preserve **`stdout`** more consistently.

This is different from passing a command directly because of how Bash handles the output.



# **Activity:** Shellshock Payloads

In this activity, you will customize a script to exploit the Shellshock vulnerability.

**Suggested Time:**
20 Minutes

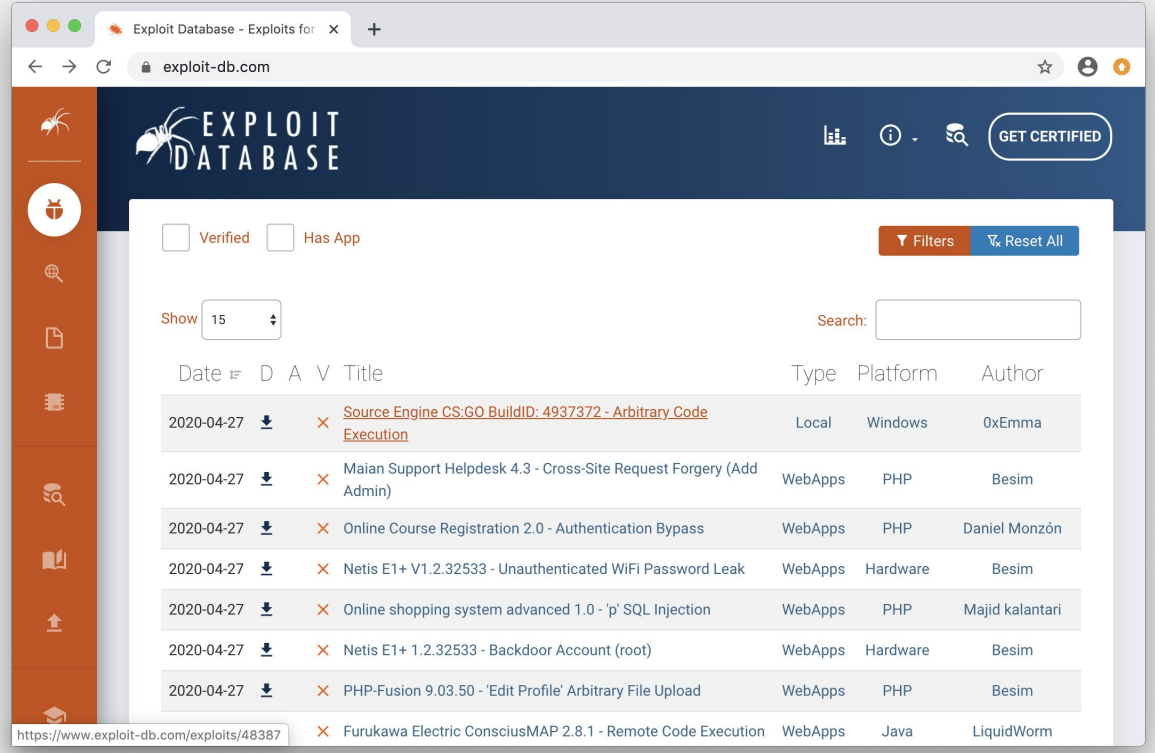
# **Time’s Up!** Let’s Review.

# SearchSploit

Now we'll use a tool called **SearchSploit** to locally store a library of exploit information and the scripts used to execute the exploits.

# SearchSploit and Exploit-DB

**Exploit Database (Exploit-DB)** is a popular online database that contains publicly disclosed exploits, cataloged according to their **Common Vulnerability and Exposure (CVEs)** identifier.



# SearchSploit and Exploit-DB

SearchSploit is a command-line utility that allows you to take an offline copy of the Exploit Database with you wherever you go.

- SearchSploit allows security professionals to perform detailed offline searches of hundreds of exploit scripts through their copy of the repository.
- This capability is useful if you are working on a security assessment with an air-gapped, segregated network that lacks an internet connection.
- SearchSploit comes preinstalled on Kali Linux, but should be updated prior to each use and on a weekly basis.

(exploit-db.com)

# SearchSploit Demo

We will use the Kali VM to run a Shellshock exploit on the Shellshock VM, which is an intentionally vulnerable machine specifically designed to test the Shellshock exploits. We will:

01

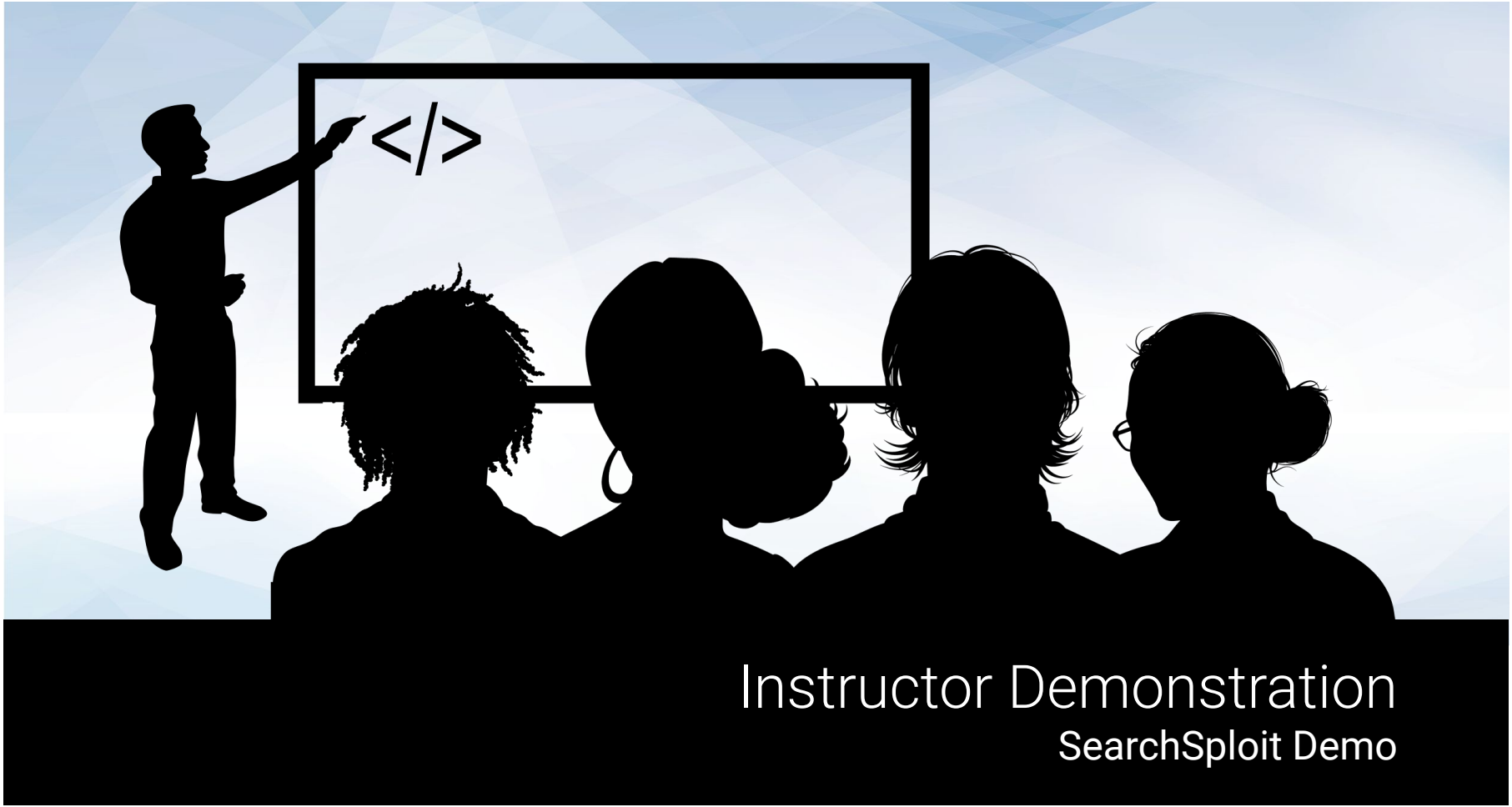
Run the help command in order to assist us in searching for exploit scripts.

02

Run basic `searchsploit` commands and break down their syntax.

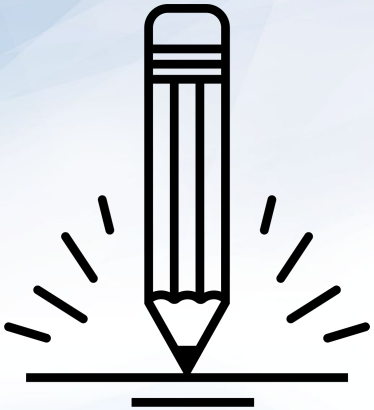
03

Review the various file formats associated with SearchSploit exploit scripts.




# Instructor Demonstration

## SearchSploit Demo



# **Activity:** SearchSploit and Shellshock

In this activity, you will use SearchSploit to identify and use a preconfigured script that will give you a backdoor into the Shellshock machine.

**Suggested Time:**
20 Minutes

# **Time’s Up!** Let’s Review.


Countdown timer

15:00

(with alarm)

# Pen Testing and Network Scanning Review

# Pen Testing and Network Scanning Review

This unit covered the first three stages of a penetration test. In the next unit, we will explore the next steps, additional exploits, and advanced penetration engagement.

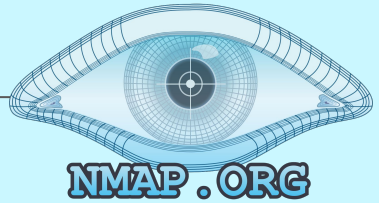
01 Active and Passive Reconnaissance

02 Scanning and Enumeration

03 Gaining Access

Before we move on to those advanced subjects, let’s take some time to review the many concepts, tools, and skills covered throughout this unit.

# Nmap Review: Host Discovery and Ping Sweep

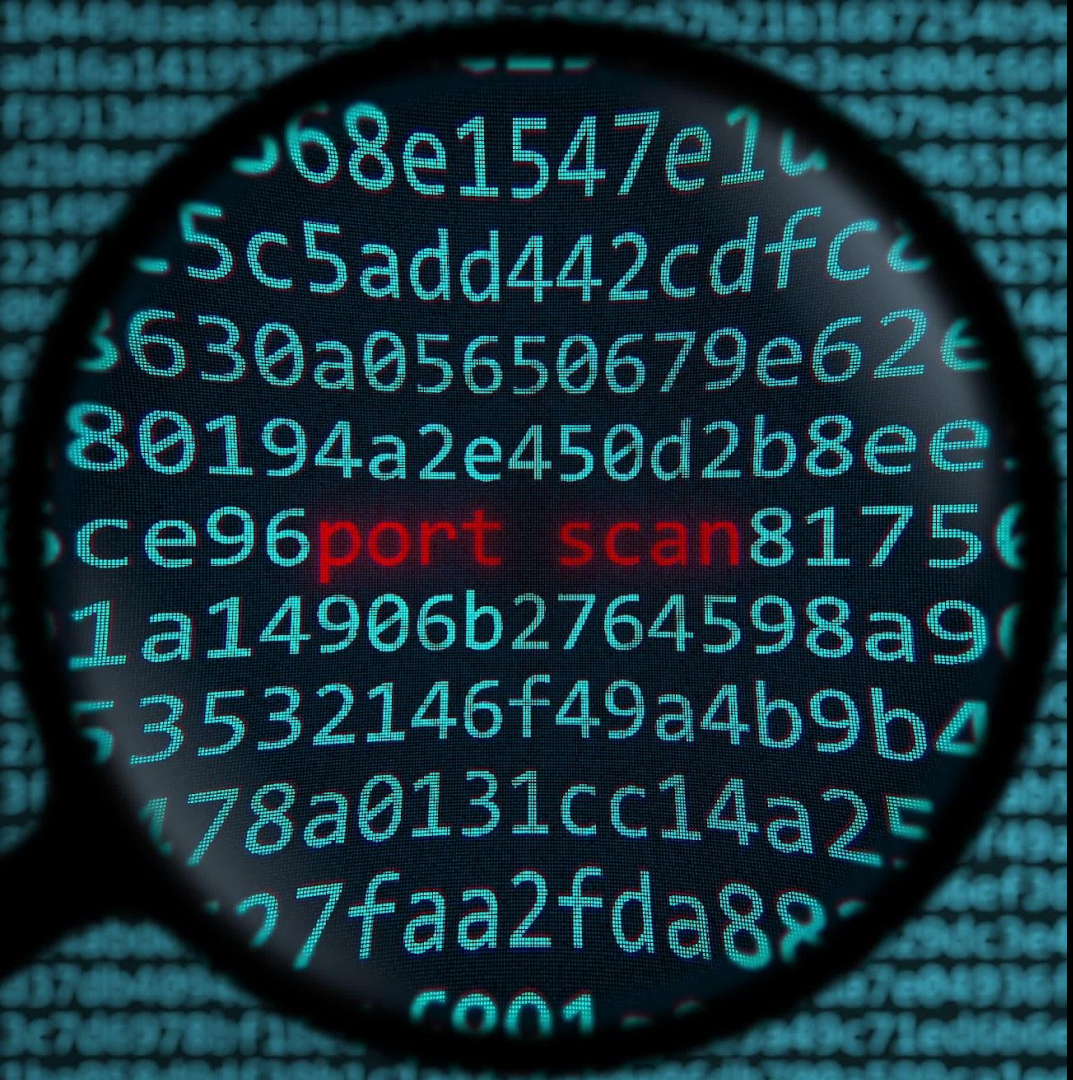



This week, we used Nmap to perform port scans on targets. Attackers will often scan the network for other live hosts in a process called **host discovery.**

One of the most important methods for host discovery is a **ping sweep**.
These send a ping request to every IP address in the target range.
Only live hosts will respond.

```
nmap -sn 192.168.12.0/24.
```

The `-sn` **flag** tells Nmap to ping every machine in `192.168.12.0/24`.
This command does not perform a port scan.




# Nmap Review: Port Scanning

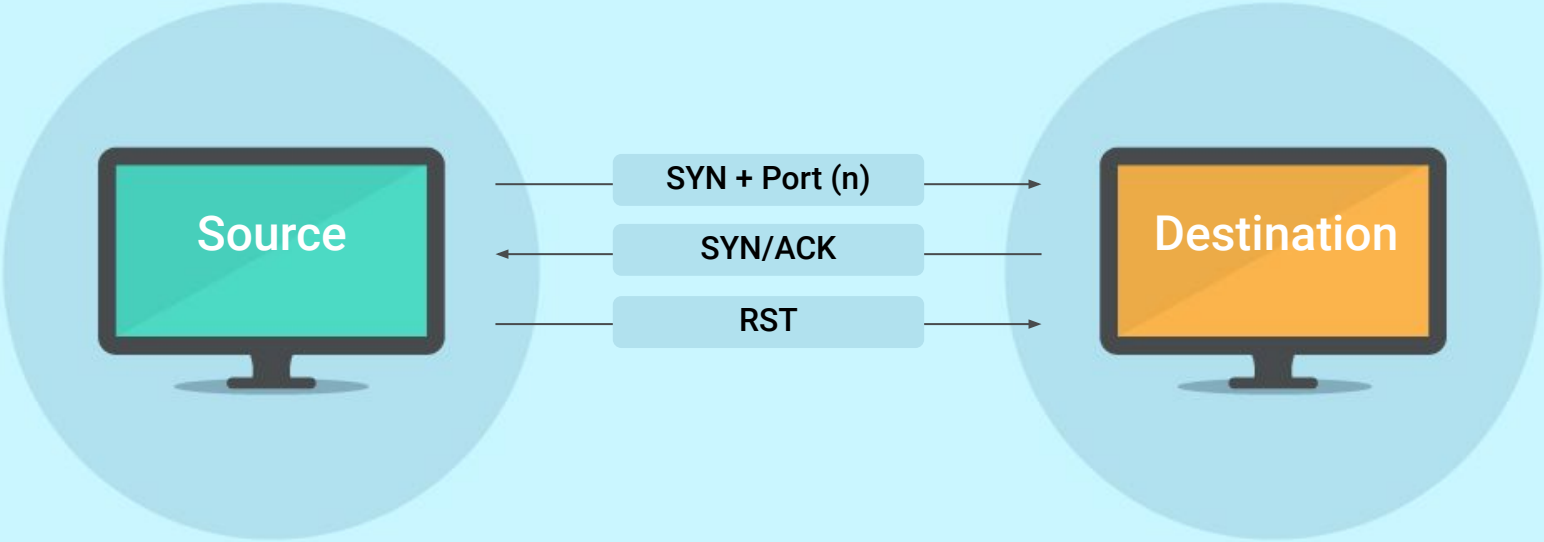
**Port scanning follows host discovery.**

- Port scanning tries to connect to a specified port on a target machine and keeps track of which connection attempts are successful. By default, nmap will scan the top 1000 ports.
- Nmap is the most widely used tool for scanning ports.

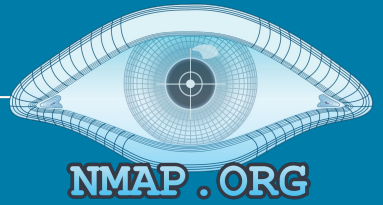
# Nmap Review: Port Scanning

The default scan type is a SYN Scan.

- In a SYN scan, Nmap sends TCP flags to a target port with the SYN flag set.
- If the target port is open, it responds with a packet with the SYN/ACK flags set.
- Nmap responds with an RST, thus closing the Half-Connect scan connection.



# Nmap Review: Port Scanning




## Port scan examples:

| | |
|---|---|
| `nmap -sS -p 445 192.168.12.50` | performs a **SYN scan** against port 445 on `192.169.12.50.` |
| `nmap -sT -p 445 192.168.12.50` | performs a **TCP connect scan** against port 445 on `192.169.12.50.` |
| `nmap -sU -p 53 192.168.12.50` | performs a **UDP scan** of port 53 on `192.168.12.50.` |
| `nmap -sS -p U:53,T:53 192.168.12.50` | scans **UDP** port 53 and **TCP** port 53. Use a **SYN scan** on the **TCP** port. |

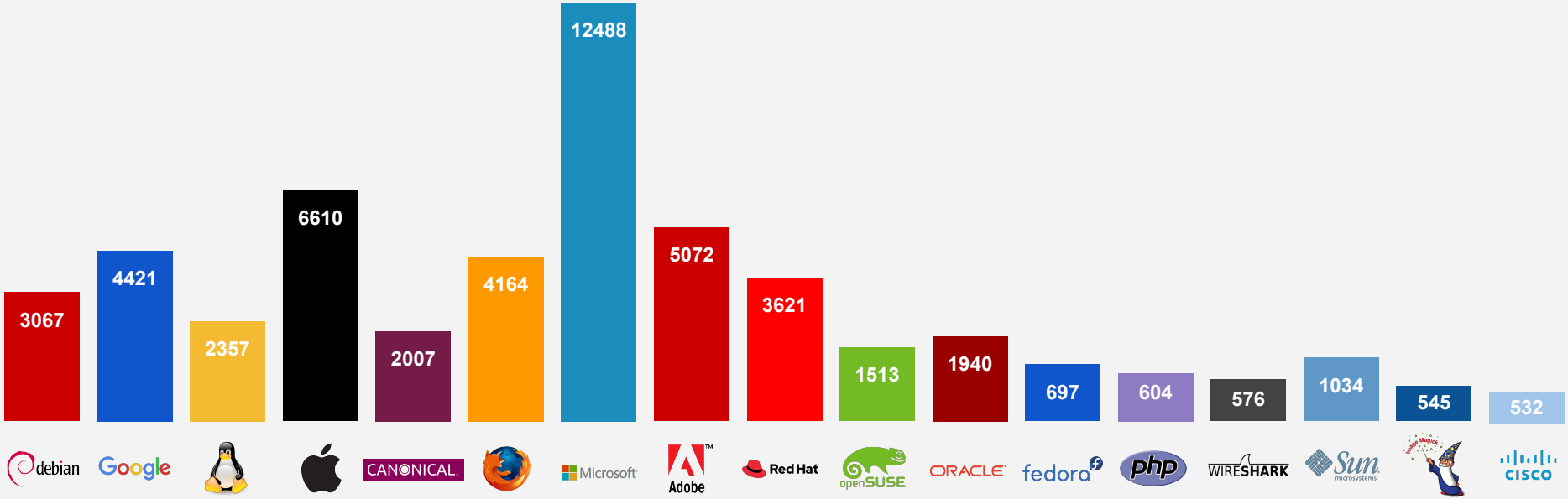
# OS and Service Enumeration

After identifying open ports on a target machine, we need to determine which operating system is running on each target and which services are running on each open port.

# OS and Service Enumeration

While many protocols are typically associated with certain ports, such as HTTP/**80** or HTTPS/**443**, in reality, any service can run on any port. This is why it's crucial to verify which services are running on all open ports, even the common ones.

**Total number of vulnerabilities of top 50 products (by vendor):**




(Source)

# OS and Service Enumeration

Nmap flags allow us to profile both operating systems and services.

The `-sV` flag enables **service and version detection**.

- This causes Nmap not only to determine if a target port is open, but also to determine which service is running on that port.

The `-O` flag enables passive OS detection.

- Nmap attempts to determine the target machine's operating system based only on the data it collects during a normal port scan. In other words, it does not send specially-crafted packets to determine the target operating system.

The `-A` flag enables active OS detection.

- Nmap sends additional packets meant to trigger responses revealing the target's OS. This method is much noisier than the -O flag and more likely to be detected by an IDS.

# OS and Service Enumeration

Nmap flags allow us to profile both operating systems and services.

The `-sV` flag enables service and version detection.

- This causes Nmap not only to determine if a target port is open, but also to determine which service is running on that port.

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

- Nmap attempts to determine the target machine's operating system based only on the data it collects during a normal port scan. In other words, it does not send specially-crafted packets to determine the target operating system.

The `-A` flag enables active OS detection.

- Nmap sends additional packets meant to trigger responses revealing the target's OS. This method is much noisier than the -O flag and more likely to be detected by an IDS.

# OS and Service Enumeration

Nmap flags allow us to profile both operating systems and services.

The `-sV` flag enables service and version detection.

- This causes Nmap not only to determine if a target port is open, but also to determine which service is running on that port.

The `-O` flag enables passive OS detection.

- Nmap attempts to determine the target machine's operating system based only on the data it collects during a normal port scan. In other words, it does not send specially-crafted packets to determine the target operating system.

The `-A` flag enables **active OS detection**.

- Nmap sends additional packets meant to trigger responses revealing the target's OS. This method is much noisier than the -O flag and more likely to be detected by an IDS.

# OS and Service Enumeration

## Detection command examples:

`nmap -sV -p 80,443 192.168.12.50` performs a service scan of ports `80` and `443` on `92.168.12.50.` Does not determine the OS.

`nmap -O -sV -p 80,443 192.168.12.50` performs a service scan of ports `80` and `443` on `192.168.12.50` and determines the OS through passive methods.

`nmap -A -sV -p 80,443 192.168.12.50` performs a service scan of ports `80` and `443` on `192.168.12.50` and determines the OS through active methods.

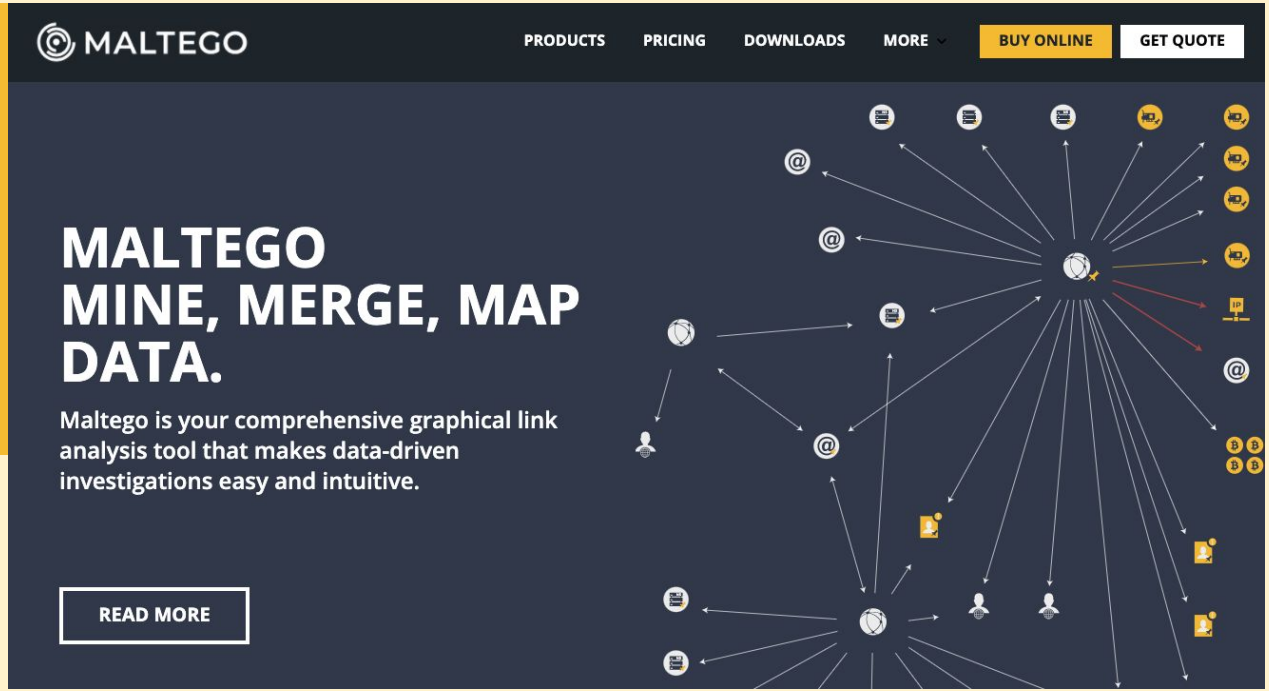
Noisier than the `-O` option.

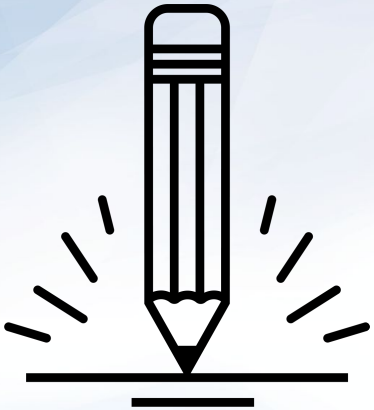
Network mapping is the process of collecting all the information gathered into a single document, known as a network map.

# OS and Service Enumeration

You can use tools for generating graphical network maps, such as Maltego. But maps drawn by hand often work just as well.

Drawing a map yourself helps clarify attack tactics and asks you to use all the information collected during the intelligence gathering phase.





# **Activity:** Nmap and Scanning Review

In this activity, you'll use the Nmap documentation to learn useful new flags. Once you complete the tasks, you'll review other pen testing and scanning concepts with a brief quiz.

**Suggested Time:**
20 minutes

# **Time’s Up!** Let’s Review.


The End