**Metasploit: Introduction**

Introduction to Metasploit

Metasploit is the most widely used exploitation framework. Metasploit is a powerful tool that can support all phases of a penetration testing engagement, from information gathering to post-exploitation.

Metasploit has two main versions:

* **Metasploit Pro**: The commercial version that facilitates the automation and management of tasks. This version has a graphical user interface (GUI).
* **Metasploit Framework**: The open-source version that works from the command line. This room will focus on this version, installed on the AttackBox and most commonly used penetration testing Linux distributions.

The Metasploit Framework is a set of tools that allow information gathering, scanning, exploitation, exploit development, post-exploitation, and more. While the primary usage of the Metasploit Framework focuses on the penetration testing domain, it is also useful for vulnerability research and exploit development.

The main components of the Metasploit Framework can be summarized as follows;

* **msfconsole**: The main command-line interface.
* **Modules**: supporting modules such as exploits, scanners, payloads, etc.
* **Tools**: Stand-alone tools that will help vulnerability research, vulnerability assessment, or penetration testing. Some of these tools are msfvenom, pattern\_create and pattern\_offset. We will cover msfvenom within this module, but pattern\_create and pattern\_offset are tools useful in exploit development which is beyond the scope of this module.

This room will cover the main components of Metasploit while providing you with a solid foundation on how to find relevant exploits, set parameters, and exploit vulnerable services on the target system. Once you have completed this room, you will be able to navigate and use the Metasploit command line comfortably.

You can deploy and use the AttackBox to complete tasks and answer the questions.

Main Components of Metasploit

While using the Metasploit Framework, you will primarily interact with the Metasploit console. You can launch it from the AttackBox terminal using the msfconsole command. The console will be your main interface to interact with the different modules of the Metasploit Framework. Modules are small components within the Metasploit framework that are built to perform a specific task, such as exploiting a vulnerability, scanning a target, or performing a brute-force attack.

Before diving into modules, it would be helpful to clarify a few recurring concepts: vulnerability, exploit, and payload.

* **Exploit:** A piece of code that uses a vulnerability present on the target system.
* **Vulnerability:** A design, coding, or logic flaw affecting the target system. The exploitation of a vulnerability can result in disclosing confidential information or allowing the attacker to execute code on the target system.
* **Payload:** An exploit will take advantage of a vulnerability. However, if we want the exploit to have the result we want (gaining access to the target system, read confidential information, etc.), we need to use a payload. Payloads are the code that will run on the target system.

Modules and categories under each one are listed below. These are given for reference purposes, but you will interact with them through the Metasploit console (msfconsole).

Auxiliary

Any supporting module, such as scanners, crawlers and fuzzers, can be found here.

Terminal

root@ip-10-10-135-188:/opt/metasploit-framework/embedded/framework/modules# tree -L 1 auxiliary/

auxiliary/

├── admin

├── analyze

├── bnat

├── client

├── cloud

├── crawler

├── docx

├── dos

├── example.py

├── example.rb

├── fileformat

├── fuzzers

├── gather

├── parser

├── pdf

├── scanner

├── server

├── sniffer

├── spoof

├── sqli

├── voip

└── vsploit

20 directories, 2 files

Encoders

Encoders will allow you to encode the exploit and payload in the hope that a signature-based antivirus solution may miss them.

Signature-based antivirus and security solutions have a database of known threats. They detect threats by comparing suspicious files to this database and raise an alert if there is a match. Thus encoders can have a limited success rate as antivirus solutions can perform additional checks.

Terminal

root@ip-10-10-135-188:/opt/metasploit-framework/embedded/framework/modules# tree -L 1 encoders/

encoders/

├── cmd

├── generic

├── mipsbe

├── mipsle

├── php

├── ppc

├── ruby

├── sparc

├── x64

└── x86

10 directories, 0 files

Evasion

While encoders will encode the payload, they should not be considered a direct attempt to evade antivirus software. On the other hand, “evasion” modules will try that, with more or less success.

Terminal

root@ip-10-10-135-188:/opt/metasploit-framework/embedded/framework/modules# tree -L 2 evasion/

evasion/

└── windows

├── applocker\_evasion\_install\_util.rb

├── applocker\_evasion\_msbuild.rb

├── applocker\_evasion\_presentationhost.rb

├── applocker\_evasion\_regasm\_regsvcs.rb

├── applocker\_evasion\_workflow\_compiler.rb

├── process\_herpaderping.rb

├── syscall\_inject.rb

├── windows\_defender\_exe.rb

└── windows\_defender\_js\_hta.rb

1 directory, 9 files

Exploits

Exploits, neatly organized by target system.

Terminal

root@ip-10-10-135-188:/opt/metasploit-framework/embedded/framework/modules# tree -L 1 exploits/

exploits/

├── aix

├── android

├── apple\_ios

├── bsd

├── bsdi

├── dialup

├── example\_linux\_priv\_esc.rb

├── example.py

├── example.rb

├── example\_webapp.rb

├── firefox

├── freebsd

├── hpux

├── irix

├── linux

├── mainframe

├── multi

├── netware

├── openbsd

├── osx

├── qnx

├── solaris

├── unix

└── windows

20 directories, 4 files

NOPs

NOPs (No OPeration) do nothing, literally. They are represented in the Intel x86 CPU family they are represented with 0x90, following which the CPU will do nothing for one cycle. They are often used as a buffer to achieve consistent payload sizes.

Terminal

root@ip-10-10-135-188:/opt/metasploit-framework/embedded/framework/modules# tree -L 1 nops/

nops/

├── aarch64

├── armle

├── cmd

├── mipsbe

├── php

├── ppc

├── sparc

├── tty

├── x64

└── x86

10 directories, 0 files

Payloads

Payloads are codes that will run on the target system.

Exploits will leverage a vulnerability on the target system, but to achieve the desired result, we will need a payload. Examples could be; getting a shell, loading a malware or backdoor to the target system, running a command, or launching calc.exe as a proof of concept to add to the penetration test report. Starting the calculator on the target system remotely by launching the calc.exe application is a benign way to show that we can run commands on the target system.

Running command on the target system is already an important step but having an interactive connection that allows you to type commands that will be executed on the target system is better. Such an interactive command line is called a "shell". Metasploit offers the ability to send different payloads that can open shells on the target system.

Terminal

root@ip-10-10-135-188:/opt/metasploit-framework/embedded/framework/modules# tree -L 1 payloads/

payloads/

├── adapters

├── singles

├── stagers

└── stages

4 directories, 0 files

You will see four different directories under payloads: adapters, singles, stagers and stages.

* **Adapters:** An adapter wraps single payloads to convert them into different formats. For example, a normal single payload can be wrapped inside a Powershell adapter, which will make a single powershell command that will execute the payload.
* **Singles:** Self-contained payloads (add user, launch notepad.exe, etc.) that do not need to download an additional component to run.
* **Stagers:** Responsible for setting up a connection channel between Metasploit and the target system. Useful when working with staged payloads. “Staged payloads” will first upload a stager on the target system then download the rest of the payload (stage). This provides some advantages as the initial size of the payload will be relatively small compared to the full payload sent at once.
* **Stages:**Downloaded by the stager. This will allow you to use larger sized payloads.

Metasploit has a subtle way to help you identify single (also called “inline”) payloads and staged payloads.

* generic/shell\_reverse\_tcp
* windows/x64/shell/reverse\_tcp

Both are reverse Windows shells. The former is an inline (or single) payload, as indicated by the “\_” between “shell” and “reverse”. While the latter is a staged payload, as indicated by the “/” between “shell” and “reverse”.

Post

Post modules will be useful on the final stage of the penetration testing process listed above, post-exploitation.

Terminal

root@ip-10-10-135-188:/opt/metasploit-framework/embedded/framework/modules# tree -L 1 post/

post/

├── aix

├── android

├── apple\_ios

├── bsd

├── firefox

├── hardware

├── linux

├── multi

├── networking

├── osx

├── solaris

└── windows

12 directories, 0 files

If you wish to familiarize yourself further with these modules, you can find them under the modules folder of your Metasploit installation. For the AttackBox these are under /opt/metasploit-framework/embedded/framework/modules

***Answer the questions below***

What is the name of the code taking advantage of a flaw on the target system?



Correct Answer

What is the name of the code that runs on the target system to achieve the attacker's goal?



Correct Answer

What are self-contained payloads called?



Correct Answer

Is "windows/x64/pingback\_reverse\_tcp" among singles or staged payload?



Correct Answer