Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Metasploit

Aleksander Mundjar

11.18.2020

Outline

- Dictionary of terms
- About
- Different Interfaces
- Payloads
- Msfvenom (demo)
- Evading detection,
- Veil framework (demo)
- Exploits vs Auxiliary
- Exploits and Meterpreter (demo)
- Usage in Pentesting and automation
- Homework

About Metasploit

- **Metasploit** is an open source penetration testing tool commonly used in red teaming and vulnerability assessment, developed by **Moore**. **H. D.** in 2003, it was bought by **Rapid7** in 2007.
  Think of it as a giant database of exploits and payloads that can be used to attack a vulnerable machine.

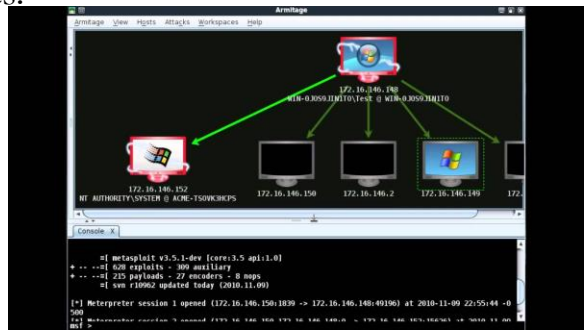
- It is most useful when, an attacker knows a machine is vulnerable and wants an easy way to exploit it. Manual exploitation using **Metasploit** usually consists of gathering information about the target, then searching every service version in the Ms database to see if it is vulnerable. Because of that, Metasploit is usually automated and takes care of the "low hanging fruit" when doing a pentest (on every up to date network it should find nothing).

- There are multiple ways one can use Metasploit in this presentation I will be mostly using the msfconsole which is the most widely used interface for **Metasploit**, but there are also others like **MsfGUI**, **Msfcli**, **Msfweb**, **Metasploit Pro** and **Armitage**.

Different interfaces

- As I have mentioned before, **Metasploit** has a few different interfaces.
- **Armitage**:

  Exploitation still happens in console but who doesn't like a nice visual representation.



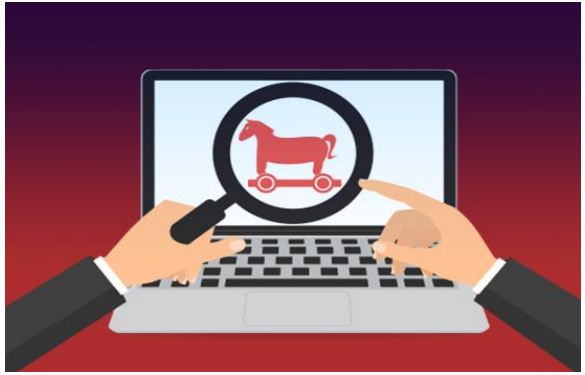
- **Metasploit Pro**:

  This one is not free, with addition of automated phishing and the ability to create reports, on top of what a normal Metasploit has to offer, it comes at a price of $15,000 per user, per year.

Payloads

- **Payloads** are pieces of malicious code that intend to harm the system in different ways.
- Usually when we talk about **malware**, **payloads** need some kind of user interaction in order to trigger, clicking a link, running some software...

  Let us say that we create a game of tic tac toe. But we sneak a **payload** in the source of the game that, when triggered, opens a **shell** to the system that is running it. In this case user needs to actually run our game in order to trigger the payload.



Payloads

- On the other hand, when we talk about payloads being part of an **exploit**, that usually works in a different way.

- When payload is part of an **exploit** that targets a **vulnerability** in the system, then **minimal** user interaction is needed in order to get the **payload** to run.
  Since we are exploiting a vulnerability there is no need for trojans(our tic tac toe game).

- We will generate a few different **payloads** during demo sections of this presentation to get better understanding of how they work.

Msfvenom

- One way to generate malicious payloads is with a framework called **Msfvenom**.
- **Msfvenom** is a combination of **Msfpayload** and **Msfencode**, putting both of these tools into a single Framework instance.
- **Msfvenom** can generate malicious payloads in all kinds of different formats (exe, dll, apk...) but more about that in the demo.
- We will use **Msfvenom** in order to create payloads/malware that we will then run on the victim machine, simulating a phishing attack in which the victim runs our payload voluntarily.
- One key weakness of **Msfvenom** is that it is widely known, which means, most up to date anti-virus software will flag payloads created by **Msfvenom** as malicious, and remove them.

Msfvenom

Screencast demo

Evading detection

- The biggest problem with creating and using payloads like what we saw in the demo, is that they are easily detected by most modern anti-malware software.
- Thus we need a way for our payloads not to be detected and mitigated.
  Most common way Antivirus detects a piece of malware are signatures.
- Since payloads generated using Msfvenom are well known, their signatures were seen before and are thus recognised.
- One way to avoid being detected is to obfuscate our payload (Veil framework)
- Does it really make a difference? Lets take a look.

Veil framework

- The **Veil Framework** is a collection of tools designed for use during offensive security testing.
- Its main traits are **Evasion** and **Ordnance**, we will be using the Evasion part in order to obfuscate our payload to make It harder for Antivirus to detect it
- We will also look at source code produced in order to get a feeling for how the obfuscation looks like.

```
=========================================================================
Veil-Evasion | [Version]: 2.28.2
=========================================================================
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=========================================================================

Main Menu

       51 payloads loaded

Available Commands:

       use             Use a specific payload
       info            Information on a specific payload
       list            List available payloads
       update          Update Veil-Evasion to the latest version
       clean           Clean out payload folders
       checkvt         Check payload hashes vs. VirusTotal
       exit            Exit Veil-Evasion

[menu>>]:
```

Veil – Framework

Veil framework

Screencast demo

Exploits vs Auxiliary

- There are a lot of modules inbuild in Metasploit. But every one of them is either an **Exploit** or an **Auxiliary**

- One distinct difference between this two is, that **Exploits** generally take advantage of an existing vulnerability in a system, and thus provide us with access to the system, and to do so, **Exploit** has to execute some kind of payload.

- in contrast, **Auxiliary** modules do not carry payload and are mostly used in recon phase of an attack (enumeration, scanning)

Exploits and Meterpreter

- **Meterpreter** is an advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime.
- We will be using Meterpreter in the next screencast demo in order to exploit a vulnerable box using Metasploit framework
- We will **Exploit** vulnerable instance of a FTP server that is running on the vulnerable machine.

Exploits and Meterpreter

Screencast demo

Usage in Pentesting and automation

- As I have mentioned somewhere at the beginning of this presentation, manual exploitation using **Metasploit** is quite unproductive. For that reason most companies automate the process of penetration testing with **Metasploit.**

- Because Metasploit is an open-sourced framework automation and customization is possible in different ways:
  - **Resource Scripts**
  - **Plugins**
  - **Auxiliary Module Custom Commands**
  - **Custom Auxiliary Modules**
  - **Metasploit Remote API**
  - **Ruby Programming**

Resources

- The main resource for this presentation was a book from Mike O'Leary: **Cyber Operations; Building, Defending and Attacking Modern Computer Networks Second Editon(2019)**.

- And of course some googling and reading all kinds of articles online

Homework

- So, I was told that I must give Homework :)
- **Part 1**: Get root access to a vulnerable machine running at karna.crq.systems(**only accessible from jagababa**), there will be a POW text file waiting. Use **Metasploit** to run your exploits.

- **Part 2**: Try to create payload that gives you a reverse shell on a system that runs it. Your payload should have as low **Virustotal** score as possible.