Creating Exploits

EEL 6931 – Advanced Ethical Hacking

Group Project – Spring 2014

Mark Bacchus
FIU - Computer Engineering Department
Florida, United States
2394870
Mbacc001@fiu.edu

Maria A. Gutierrez
FIU - Computer Engineering Department
Florida, United States
2227112
mguti023@fiu.edu

Alexander Coronado
FIU - Electrical Engineering Department
Florida, United States
2605672
Acoro012@fiu.edu

***Abstract*— the insights of exploits revealed. This document contains information about exploits, ruby, and details on how to write/create an exploit.**

***Index Items*— Metasploit, exploits, ruby, coding, security, computer security.**

I. Introduction – What is an Exploit?

An exploit is the formula to get access into a system wherever the code or system has a flaw. Next, we are going to present the different types of attackers, the different types of attacks, the different types of exploits, and the terminology.

A. Different types of attackers

1) White hat

The white hats are the good guys. This type of attacker is the ones that run penetration tests or hacks into a system ethically in order to find the flaws the system may have. After, they share their findings with the company or network's owner without using the exploits to harm others.

2) Black hat

The black hats are the bad guys. This type of attacker will find flaws and holes in the system and won't notify anyone to use for their own benefit. They will exploit it themselves to harm others.

3) Red hat

The red hats are those who hack the system, usually Linux/UNIX platforms, in order to find flaws and make it better without any permission from anyone through open source software.

4) Gray hat

The gray hats are in between black hats and white hats. These are the type of attackers who will find a flaw and let the owner of the system know. At the same time, they will let the world know by publicizing it in the web so other attackers would go for an attack to harm others.

B. Different types of attacks

1) ARP Poisoning

The attacker gets in between the computer and the router. This way, having physical access, the attacker continues to run a man in the middle attack from this situation.

2) Brute Force

Brute Force is basically deciphering a password. The process is long depending on the victim's password. This attack is very noticeable because trying out the password attempts the system can tell whether the person trying to log in is actually the victim or not.

3) Dictionary Attack

A dictionary attack is when the attacker uses a text file that contains various common words for passwords and uses it to apply a brute force attack on the system, but this one is faster.

4) DNS Poisoning

The DNS poisoning attack is when the attacker gains physical access to the computer. Once the attacker gained physical access, it can spoof the system.

5) Fuzzing

This attack goes hand in hand with the social engineering attack. First, the attacker uses an exploit and a payload to access gain access into the system. The attacker fools the system because he/she is impersonating the victim once the attacker has gained the access desired.

6) Network Mapping

This attack is basically the attacker tracing the victim's steps. The attacker will scan the network for the entire system including the services the system has. This way, the attacker gains access and commit the harmful attack.

7) Password Cracking

Password cracking is when the attacker cracks the victim's password through rainbow tables.

8) Session Hacking

This attack happens when the attacker uses the victim's credentials to impersonate the victim and gain access to their system. An example of this attack is password hacking. The attacker finds out the password of the victim and logs in as the victim.

9) Social Engineering

This attack happens when the attacker impersonates and convinces the victim of being someone of trust. This way, the attacker gathers important information or credentials from the victim to commit harmful attacks.

10) SQL Injection

This happens when the attacker uses injections of malicious SQL responses or scripts to gain full access to the system.

11) Vulnerability Scanning

The attackers, to gain knowledge about the system, do this type of scanning. They make a penetration test after finding out how many exploits are known to the system. Then, they proceed with the attack to gain full access.

12) Wardriving

Wardriving is an attack where the attacker has access to an encrypted access point. From there, the attacker captures the packets that are encrypted to decrypt them. After decrypting them, they gain access to the network because they find out the key necessary.

13) Zero Day

Zero Day is the term they have chosen to describe a new attack that no one knows about. It presents new attacks without any solution yet.

C. Different types of exploits

1) Arbitrary Code Execution

The name itself explains it, the attacker puts a bug into a software on the machine so he/she can execute the code arbitrarily.

2) Buffer Overflow

This is the most common exploit, also known as buffer overrun. This is the type of exploit that writes data to the buffer, and then the boundaries of the buffer are overrun. Once the boundaries are overrun, the adjacent memory gets overwritten. This exploits creates different problems to the data such as losing it, creating incorrect results, etc.

3) Code Injection

As the name says, this is the exploit where code gets injected with a bug in order to process invalid data.

4) Cross-Site Scripting

Cross-site scripting, or also known as XSS, is the type of exploit that you can find in web applications. The attackers inject the script on the client's side so it spreads through the web to users who views those infected websites.

5) Denial-of-Service (DoS)

As the name itself says, this exploit will ask the server for some data from many sources at the same time. This creates a lot of traffic and confuses the server's allocated memory making it crash.

6) Heap Spraying

This happens when the attacker uses other exploits such as arbitrary code execution to flood the application and make it crash as soon as it can.

7) HTTP Header Injection

This is a web-based attack where the attacker uses HTTP headers to infect the victim's system. The attacker usually creates a system where the headers are dynamically generated throughout the web.

8) Privilege-confusion bugs

These exploits are bugs that their main focus is to confuse the system by requesting forgery or attacks to the system.

9) Unauthorized Data Access

This type of exploit is just someone having unauthorized access to certain data in a system without wanting it.

D. Terminology

1) Client Side Exploit

This means doing the exploit through social engineering, on the client's side of the system.

2) Exploit

An exploit, as stated earlier, is the access found through the system's flaws

3) Honey Pot

This is an application that will have flaws intentionally to create an easy way to get the exploits into the system to keep track of them and sometimes even prevent attacks.

4) Injection

As the name suggests, is the term used to describe the addition or injection of the payloads into the system.

5) Module

This is referred to a tool.

6) Payload:

It is the information about the exploit that identifies the material's destination and source and it is sent after the exploit has already run.

7) Physical Access:

It is having the computer or system's access in location.

8) Pivoting

Pivoting is the displacement or movement from either an IP range to another or in between a network, machine, or a subnet.

9) Remote Access:

It is not having the physical access to the system so they access it remotely.

10) Scanner

A scanner is a tool that scans or looks up something specific throughout the system.

11) Server Side Exploit

This is the opposite of client side exploit; this is when the exploit is being accessed through the server.

12) Shellcode

The shellcode is the code that is in a payload.

13) Spoof

Spoof is the term used to describe the masking of the attacker's IP address or information.

14) Target

The target as its definition suggest is the machine or system that the attacker is focused on harming.

15) Trigger

Trigger is used in an exploit by the attacker or the user.

16) Vulnerability

Vulnerability is where the system is flawed. It is the location in the system that the attackers exploit.

Having a little bit of background information about the exploits, we now proceed to give information about Ruby.

II. Ruby

A. What is Ruby?

Ruby is an interpreter, general-purpose object-oriented programming language developed by Yukihiro Matsumoto in Japan in the 90's. As an interpreted language the computer read the code line by line in contrast the compile languages, where the computer reads the entire code up front. The first public release of this language was on December 21 of 1995, were Matsumoto announced the Ruby's 0.95 version. According to his creator Matsumoto was inspired in his favorite languages: Perl, Smalltalk, Eiffel, Ada, and Lisp [1]. This language has the difference that everything is an object and every line of code can be given unique properties and actions, also called instance variables and methods correspondingly.

Ruby also present different implementations, which can be appointed the following:

- JRuby is made for Java Virtual Machine, optimizing compiler, concurrent threads, garbage collector, and many collections of libraries.
- Rubinius is a core library built using Ruby, providing a C application-programming interface (API) for running C-extensions to ease migrating from Matz's Ruby Interpreter, also called CRuby.
- MacRuby is a library for Mas OS X that allowed write desktop applications with ease.
- MRuby is an implementation that can be linked and embedded within your application, very useful in devices such as Arduino.
- IronRuby provide the user to connect Ruby with .NET Framework programming.
- MagLev brings a distributed object space to Ruby dedicated for Virtual Machines.
- Cardinal is a "Ruby compiler for Parrot Virtual Machine" [1].

B. Why Ruby?

Initially Metasploit was a portable tool using Perl, but Metasploit had been rewritten in Ruby on October of 2009 by his developer H.D. Moore. One of the reasons to program in Ruby was because the developers simply enjoy writing in language. The second reason was they found superior the threating model set on the Ruby language compared in other languages such as Python which have the block-indention restriction that helps the developers to identify features in their code.

The most fundamental library used by Metasploit is the Ruby Extension Library (Rex) that provides components including exploitation utility classes, implementations of protocol client and servers among others. Rex it works with the default Ruby components and has no dependencies other than Ruby itself. The Metasploit architecture is illustrated in Figure 1.

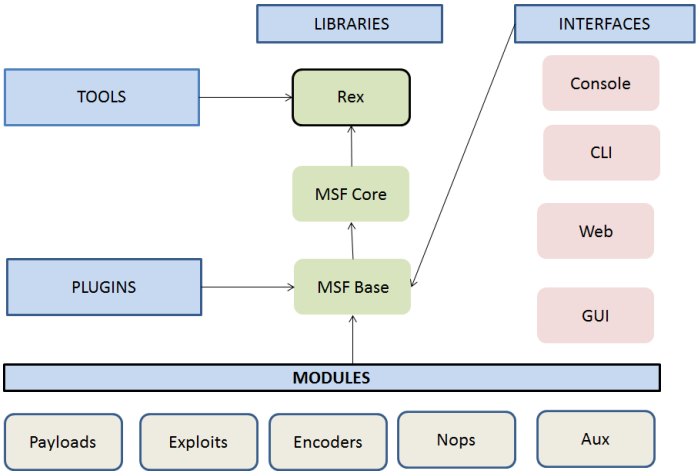

Figure 1. Metasploit Architecture

The Metasploit architecture is divided in the following main blocks: Modules, Interfaces, Libraries, Tools and plugins that will be explained in the next chapter.

III. Writing Exploits in Ruby

A. Where to start

Metasploit interfaces extend the base Ruby library that enables evoking initial utilities of the framework. Commands such as running updates, setting payloads and exploits can be executed. The most important tool from Metasploit is the msfconsole interface in which the user can implement the commands previously appointed.

In order to design an exploit to use in the Metasploit framework should have the following suggestions:

- Make use of the Rex protocol Library
- Randomize nops block
- Randomize all payloads by using encoders
- Avoid complicated code harder to maintain
- Offload all possible work to the Framework.

B. Essential commands and libraries

a) Commands

The commands that will be useful in the Metasploit console are:

- Show exploits and show payloads: These two commands will display all the available exploits and payloads in the Metasploit directory.
- Search exploit: This command will search for a particular exploit.
- We can also use this command to search for any specific search terms.
- Use exploit: This command is used to set any exploit as active and ready to use.
- Show options: This command is used to see the available options or parameters of the exploit in use.
- Set: This command is used to set a value to a parameter in the exploit under use. It is used to set up a payload for a particular exploit in use.
- Show targets: Every exploit is made to attack a particular target service.

b) Libraries

The Ruby Extension Library (Rex) is a collection of modules and classes very useful for develop exploits. For example, invoking classes with the Rex::Arch instruction, allowed the use opcode routines from the Assembly block which will be discussed below.

Assembly: Is the generator of opcode routines necessary for assembly instructions needed to write some exploits. Routines such as adjust the Stack Pointer (SP), call, push, mov and add instructions commonly used in assembly language.

Encoding: class that Encodes algorithms such as XOR with the instruction Rex:Encoding.

Exploitation: There is some common attack vectors that are vulnerable like the Structure Exception Handling (SEH) and overflow presented on windows platforms. Exploitation can be called by the routine Rex::Explotation::"type_of_explotation".

Jobs: Block helpful for task such as break and stop jobs with the class named Rex::JobContainer.

Logging: It will provide information about warning logging, error logging, and information logging with the instruction wlog, elog, ilog respectively.

Protocols: With the instruction Rex::Proto::Type_of_protocol can called some of the more common protocol such as SMB and HTTP.

Sockets: The socket subsystem provides an interface for creating sockets of a given protocol very useful in meterpreter connection. This routine use the TCP socket, SSL socket, subnet walking, notification events and reader/writer permits.

IV. Detailed process of writing exploits

Metasploit is a great tool that you can use to take advantage of a lot of well-known security exploits out there. There are tons of exploits to execute and work with available online. But say you would like to build one for yourself, or say you would just like to fix one that already exists and just may not be working correctly. One of the simplest exploits out there would be a buffer over flow exploit, here we will show you how to write and execute it as a means of getting started in in exploit writing and execution.

A. Stack based buffer overflows

Usually with buffer overflows it can lead to the particular program you are exploiting to crash in most cases. When you "exploit" a program you intend to make the application behave in a way it was not intended to. You can do this easily by having control over the instruction pointer/program counter which is the register that tells the program where the next instruction for execution will be located. Usually before a program executes a function it will save a restore point for it to return to after it is done with the particular instruction. Editing this will allow for exploitation, and the ability to point to an attackers shellcode (a small set of code that is used as a payload in an attack, can be locally or remotely located).

There are a lot of different ways to force a program into executing some shell code you might have. The first of which is a "jump" which is also referred to as a "call". If a register is loaded with an address that directs to some type of shell code, then you are able to jump directly to shell code using a jump command. You can rewrite the contents of the EIP register and direct it to whatever shell code you wish to execute. Another command used is a "pop ret". With this you commonly find an address pointing to some shellcode somewhere on a stack. Once you find this you can use a "pop ret" command to delete addresses (skip certain things) on a stack, or go straight to the address that would bring you straight to the shellcode. The only catch with the "pop ret" method is it is only usable when the ESP (the register that points to the top of the stack at any time) contains an address that directs to the shellcode. Along with these there are multiple other methods to force the execution of shell code such as "push return", "jmp", "blind return", etc.

B. Structured exception handling

Another way to execute a successful exploit is through using SEH (Structured exception handling). This is a feature used for handling a lot of hardware and software exceptions. It's used by programmers and debuggers across multitude of programming languages and is found within the application.

```
try
{
  //run stuff.  If an exception occurs, go to <catch> code
}
catch
{
  // run stuff when exception occurs
}
```

Figure 2. Example of exception handling

Windows by default has a built in SHE but is really only used as a last resort with most applications. Usually most

applications have built in SEH that are specific to the program and language it is written in. Usually within code, when something executes that results in an error, the application will rely on its exception handler to catch it, and do something with it. That can include canceling it and returning or trying to correct the error. In order for this to be possible the pointer to where the exception handler is must remain on the stack at all times for each block of code executed. Below is how the basic structure and components of an SHE looks after execution.

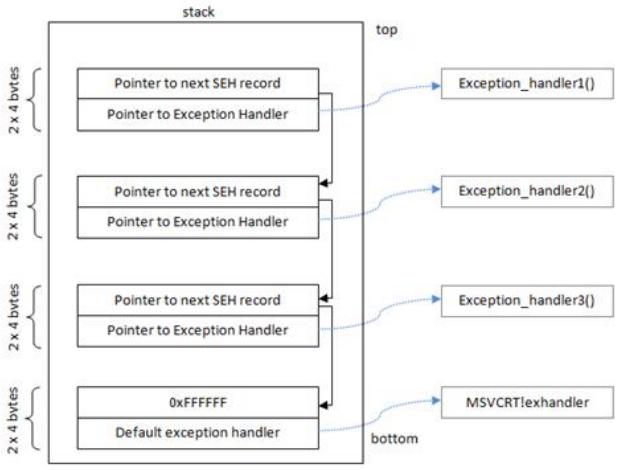

Figure 3. SEH Structure

We can use the SHE handler to our advantage most times by forcing the program to throw an exception, then gain control of it by forcing it to jump to our shell code. A common way to get your OS to jump to the next SHE is by using the instructions/jump code "POP, POP, RET". The OS understands this by jumping to the next HER. So to summarize the steps, first you must trigger an exception, overwrite the pointer to the net SHE with the jump code, overwrite the SE handler with a new pointer that links to an instruction that will return you to the next SHE and execute the jump code. Make sure the shell code is directly after the SE handler that you have just overwritten with the jump code.

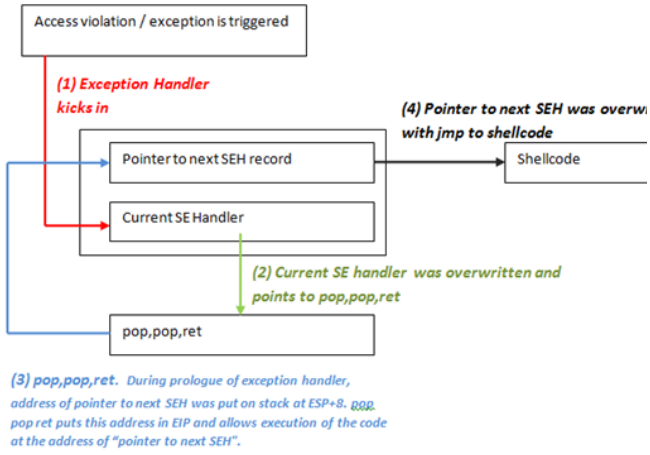

Figure 4. SHE Execution

V. Conclusion

In conclusion, Metasploit is in fact a very powerful tool. There are many packages and exploits available to help you take advantage of all known vulnerabilities know. However sometimes you might not have an exploit available that you need or you might have problems with current exploits that you are attempting to use. In this case it is very handy to learn how exactly exploits are written and how they work. You need a good understanding of register movements and how stacks and buffer operates, as well a strong programming background. It is recommended that since Metasploit does primarily use ruby, that you should also. However that is not necessarily true, exploits can be written in almost any language with little knowledge of ruby. Once you get started writing exploits it really becomes an easy process that allows you to take advantage of many security flaws that come up.

References

[1] Corlean Team, "Exploit Writing Tutorial part 1: Stack Based Overflows", July 2009. HTTP: https://www.corelan.be/index.php/2009/07/19/exploit-writing-tutorial-part-1-stack-based-overflows/

[2] Corlean Team, "Exploit writing tutorial part 4: From Exploit to Metasploit – The basics", August 2009. HTTP: https://www.corelan.be/index.php/2009/08/12/exploit-writing-tutorials-part-4-from-exploit-to-metasploit-the-basics/

[3] nanoquetz9l, "Metasploit exploit development - The series Part 1", July 2012. HTTP: https://securitystreet.jive-mobile.com/#jive-document?content=%2Fapi%2Fcore%2Fv2%2Fposts%2F5785

[4] Mati Aharoni, William Coppola, Paul Hand, Alain Hernandez, Devon Kearns, David Kennedy, Steven McElrea, Matteo Memelli, Jim O'Gorman, David Ovitz, Carlos Perez, "Exploit writing tutorial part 4: From Exploit to Metasploit – The basics", 2014. HTTP: http://www.offensive-security.com/metasploit-unleashed/Writing_An_Exploit

[5] Brett Leahy, "Pentesters Thought of Attack!", August 2010. HTTP: http://tysonmax20042003.wordpress.com/tag/types-of-exploits/

[6] sec00rit3y, "Exploit writing - Stack based Buffer overflow", August 2009. HTTP: http://blog.pusheax.com/2013/03/exploit-writing-stack-based-buffer.html?m=1

[7] Metasploit Framework: Metasploit Developer's Guide TTP: https://www.ruby-lang.org/

[8] Ruby: A Programmer's Best Friend HTTP: https://www.ruby-lang.org/en/about/

[9] M. Bachle and P. Kirchberg, "Ruby on Rails", IEEE Intelligent Systems, Ravensburg, Germany, 2007

[10] M. Sheeran and S. Singh, “Ruby as a Basis for Hardware/Software Codesign”, Chalmers Technical University, Sweden, 2000.

[11] A. Singh, “Metasploi Penetration Testing Cookbook”, Packt Publishing, Birmingham, Mumbai, 2012.

[12] D. Thomas, “Programming Ruby 1.9 & 2.0: The Pragmatic Programmers’ Guide”, The Pragmatic Programmers, Dallas, Texas, 2012.

[13] P. Cooper, “Beginning Ruby from Novice to Professional”, Apress, USA, 2007.