International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Survey on Metasploit Framework

Manjunatha T N[1], Shashidhar M S[2], Vinay G[3], Vittal S[4]
[1,2,3,4]Assistant Professor, Department of Computer Science, Sri Revana Siddeshwara Institute of Technology, B'lore

Abstract— Penetration testing is a uniquely challenging job. Paid to think like a criminal to use guerilla tactics to advantage, and to find the weakest links in a highly intricate net of defenses. The most widely used tool by the penetration tester is Metasploit Framework. Metasploit Framework is a framework for developing and executing exploit code against a remote target machine. It is highly flexible, efficient and time saving for the Pentesters. It has many numbers of modules which helps a lot for the Pentesting. It can be easily configured with various other tools like set, nmap, etc and even the GUI interface Armitage is widely used. This open source platform provides a consistent, reliable library of constantly updated exploits and offers a complete development environment for building new tools and automating every aspect of a penetration test. These products provide a different perspective on how to conduct and automate large-scale penetration tests. The Metasploit Framework is an infamously volatile the code base is updated dozens of times every day by a core group of developers and submissions from hundreds of community contributors.

Keywords— Metasploit, Pentesters , GUI, Armitage, Framework.

I. INTRODUCTION

Since the late nineties, the Internet has grown at an exponential rate. One of the biggest spurts in growth came between the years of 1995-2000 with the dot-com bubble that prompted the spawn of ecommerce for virtually every facet of society. The success of the Internet has brought great change to the world as we know it however, not all of this growth has been productive. With thousands of sites launching daily and limited resources available to monitor the credibility and/or security of these sites the existence of vulnerabilities was inevitable. Eventually exploits became rampant causing the information security field to step up its game. The result was the pre-emptive existence of vulnerability testers that's sole purpose was to attempt to exploit such software far before others got the opportunity. One of these researchers was a man by the name of H.D. Moore who in the summer of 2003 founded the Metasploit. H.D.'s purpose was to create a penetration testing tool that could be easily utilized by even novice users to perform penetration testing, regression testing, patch verification, and development. As with any other security tool, this software is often described with the proverbial double-edged sword as in the wrong hands it can be used for great harm.

II. DESCRIPTION

A. What it is?

The Metasploit Framework is a tool that collectively combines exploits into one central location ideally for security researchers. Originally developed using the Perl scripting language Metasploit is now currently on its reincarnation. Version 1.0 was written solely by H.D. Moore using Perl sporting curses based front-end. Version 2.0, also written in Perl, and included the help of a few additional developers. For Version 3.0, Metasploit received a complete overhaul. Written in the powerful scripting language Ruby, Metasploit 4.11 now boasts the power of automation due to the nature of Ruby's status as an object-oriented language. Additionally, Metasploit is considered multi-platform running on most variations of Unix and Windows.

B. Who is it for?

The Metasploit Framework was developed with the intentions of making security experts' lives easier. The original primary users were considered to be network security professionals, security administrators, product vendors, and other likeminded security researches. Each would use the tool within the guidelines of their own discipline; network security professionals for penetration testing, security administrators for patch installation verification, product vendors for regression testing, and other security researchers for perhaps development of other exploits.

C. Terminology

The Metasploit Framework can be a bit confusing for novice users as the Linux distribution offers no Graphic User-Interface (GUI). Therefore learning the semantics and syntax of the framework is essential to the effective use of the software. The primary outline of the majority of attacks in Metasploit revolves around the following foundation.

1) Choosing an exploit and configuring it.
2) Checking for susceptibility.
3) Choosing a payload and configuring it.
4) Choosing an encoding system.
5) Executing the exploit.

III. INTERFACES

A. The Console Interface

After you have installed the Framework, you should verify that everything is working properly the easiest way to do this is to execute the msfconsole user interface. If you are using Windows, start the msfgui interface and access the Console link from the Window menu. The console should display an ASCII art logo, print the current version, some module counts, and drop to an "msf" prompt. From this prompt, type help to get a list of valid commands. You are currently in the "main" mode; this allows you to list exploits, list payloads, and configure global options. To list all available exploits, type show exploits. To obtain more information about a given exploit, type info module name. The console interface was designed to be flexible and fast. If you enter a command that is not recognized by the console, it will scan the system path to determine if it is a system command. 1 If it finds a match, that command will be executed with the supplied arguments. This allows you to use your standard set of tools without having to leave the console. The console interface supports tab completion of known commands. The msfweb interface includes tab completion by default, but the msfconsole interface requires that Ruby was built with the Readline library.

B. Armitage (The GUI Interface)

The armitage component of Metasploit is a fully interactive graphical user interface created by Raphael Mudge. This interface is highly impressive, feature rich, and available for free. We won't be covering armitage in depth, but it is definitely worth mentioning as something to explore. Our goal is to teach the ins and outs of Metasploit, and the GUI is awesome once you understand how the Framework actually operates.

C. The Command Line Interface

If you are looking for a way to automate exploit testing, or simply do not want to use an interactive interface, then msfcli may be the solution. This interface takes a module name as the first parameter, followed by the options in a VAR=VAL format, and finally an action code to specify what should be done.

D. The Web Interface

The msfweb interface is based on Ruby on Rails. To access this interface, execute msfweb to start up the server. The msfweb interface uses the WEBrick web server to handle requests. By default, msfweb will listen on the loopback address (127.0.0.1) on port 55555. A log message should be displayed indicating that the service has started. To access the interface, open your browser to the appropriate URL (http://127.0.0.1:55555/ by default). The main msfweb interface consists of a toolbar containing various icons and a background with the metasploit logo. If you want access to a console, click the Console link. This console interface is nearly identical to the standard msfconsole interface. The Exploits, Auxiliary, and Payloads links will walk you through the process of selecting a module, configuring it, and running it. Once an exploit is run and a session is created, you can access these sessions from the Sessions link. These icons will open up a sub-window within the page. These windows can be moved, minimized, maximized, and closed.

IV. ARCHITECTURE

The MSF file system is laid out in an intuitive manner and is organized by directory.

1) lib: the 'meat' of the framework code base
2) data: editable files used by Metasploit
3) tools: various useful command-line utilities
4) modules: the actual MSF modules
5) plugins: plugins that can be loaded at run-time
6) scripts: Meterpreter and other scripts
7) external: source code and third-party libraries The basic library for most tasks

Volume 4 Issue VIII, August 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

8) Handles sockets, protocols, text transformations, and others.
9) SSL, SMB, HTTP, XOR, Base64, Unicode
10) Intended to be useful outside of the framework

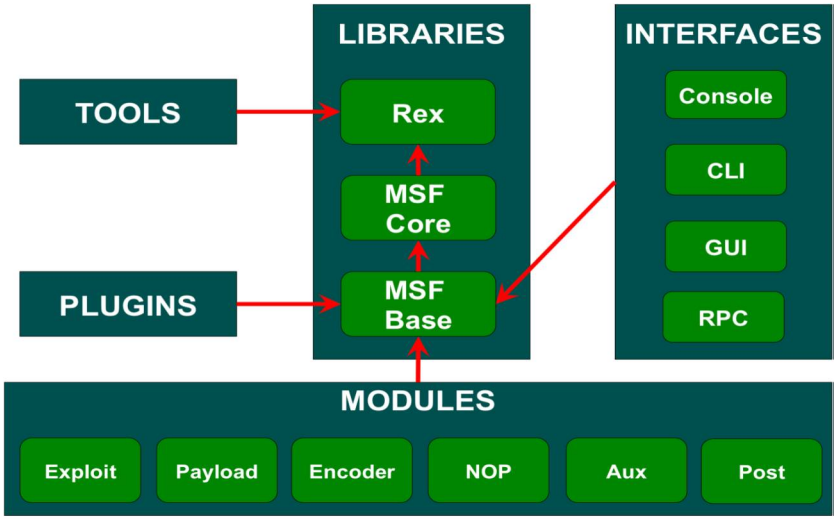

Fig 4.1 Libraries - rex

A. Libraries – MSF Core
1) Provides the 'basic' API
2) Defines the Metasploit Framework

B. Libraries – MSF Base
1) Provides the 'friendly' API
2) Provides simplified APIs for use in the Framework

C. Modules
Metasploit, as presented to the user, is composed of various modules. Exploits and Auxiliary module
1) Defined as modules that use payloads
2) An exploit without a payload is an Auxiliary module
3) Aux makes use of the MSF REX library and other mixins
4) Payloads, Encoders, Nops
5) Payloads consist of code that runs remotely
6) Encoders ensure that payloads make it to their destination
7) Nops keep the payload sizes consistent.

D. Metasploit Object Model
In the Metasploit Framework, all modules are Ruby classes.
1) Modules inherit from the type-specific class
2) The type-specific class inherits from the Msf::Module class
3) There is a shared common API between modules Payloads are slightly different.
4) Payloads are created at runtime from various components
5) Glue together stagers with stages

E. Mixins and Plugins
1) Mixins 'include' one class into another
2) This is both different and similar to inheritance
3) Mixins can override a class' methods
4) Mixins can add new features and allows modules to have different 'flavors'.

5) Protocol-specific (i.e.: HTTP, SMB)
6) Behavior-specific (i.e.: brute force)
7) connect() is implemented by the TCP mixin
8) connect() is then overloaded by FTP, SMB, and others.
9) Mixins can change behavior.
10) The Scanner mixin overloads run()
11) Scanner changes run() for run_host() and run_range()
12) It calls these in parallel based on the THREADS setting
13) Metasploit Plugins work directly with the API.
14) They manipulate the framework as a whole
15) Plugins hook into the event subsystem
16) They automate specific tasks which would be tedious to do manually Plugins only work in the msfconsole.
17) Plugins can add new console commands
18) They extend the overall Framework functionality

V. USING THE FRAMEWORK

A. Choosing a Module

From the msfconsole interface, you can view the list of modules that are available for you to interact with. You can see all available modules through the show all command. To see the list of modules of a particular type you can use the show moduletype command, where moduletype is any one of exploits, encoders, payloads, and so on. You can select a module with the use command by specifying the module's name as the argument. The info command can be used to view information about a module without using it. Unlike Metasploit 2.x, the new version of Metasploit supports interacting with each different module types through the use command. In Metasploit 2.x, only exploit modules could be interacted with.

B. Exploit Modules

Exploit modules are the defacto module in Metasploit which are used to encapsulate an exploit.

C. Configuring the Active Exploit

Once you have selected an exploit with the use command, the next step is to determine what options it requires. This can be accomplished with the show options command. Most exploits use RHOST to specify the target address and RPORT to set the target port. Use the set command to configure the appropriate values for all required options. If you have any questions about what a given option does, refer to the module source code. Advanced options are available with some exploit modules, these can be viewed with the show advanced command. Options useful for IDS and IPS evasion can be viewed with the show evasion command.

D. Verifying the Exploit Options

The check command can be used to determine whether the target system is vulnerable to the active exploit module. This is a quick way to verify that all options have been correctly set and that the target is actually vulnerable to exploitation. Not all exploit modules have implemented the check functionality. In many cases it is nearly impossible to determine whether a service is vulnerable without actually exploiting it. A check command should never result in the target system crashing or becoming unavailable. Many modules display version information and expect you to analyze it before proceeding.

E. Selecting a Target

Many exploits will require the TARGET environment variable to be set to the index number of the desired target. The show targets command will list all targets provided by the exploit module. Many exploits will default to a bruteforce target type; this may not be desirable in all situations.

F. Selecting the Payload

The payload is the actual code that will run on the target system after a successful exploit attempt. Use the show payloads command to list all payloads compatible with the current exploit. If you are behind a firewall, you may want to use a bind shell payload, if your target is behind one and you are not, you would use a reverse connect payload. You can use the info payload name command to view detailed information about a given payload. Once you have decided on a payload, use the set command to specify the

payload module name as the value for the PAYLOAD environment variable. Once the payload has been set, use the show options command to display all available payload options. Most payloads have at least one required option. Advanced options are provided by a handful of payload options; use the show advanced command to view these. Please keep in mind that you will be allowed to select any payload compatible with that exploit, even if it not compatible with your currently selected TARGET. For example, if you select a Linux target, yet choose a BSD payload, you should not expect the exploit to work.

VI. APPLICATION

Now that we understand the dictation and origins of Metasploit we can move on to the applications of the product. We will begin with a brief synopsis of a few common attacks followed by some distinguishing characteristics of the Metasploit Framework. While Metasploit is primarily identified as a one-stop exploitation application its sole purpose revolves not just around the exploitation of remote systems, but also around the development of new ones as well. With the 3.0 iteration of Metasploit, near complete automation is possible as scanning, fingerprinting, identifying vulnerabilities, exploiting, and reporting can be configured with some degree of work.

A. Example Attacks

It is without a doubt that the field of computer security is an on-going battlefield where the victor is solely based on time. On the second Tuesday of every month Microsoft releases patches for its many applications and operating system variations to what has affectionately become known in the information security field as “Patch Tuesday”. Upon their release, black hatters of all likes begin the reverse engineering process to discover the original vulnerability. Toping the list of commonly exploited programs include third-party applications such as Adobe Flash, Adobe Reader, JavaScript, and QuickTime. For the direct purpose of this paper though we’ll address exploits pertaining to apache web server and oracle.

Considering one of the most commonly attacked sources are web servers it seems scary to think that Metasploit houses an exploit that within eight commands can compromise an Apache Web Server. The attack we’re referring to utilizes chunked encoding to specially craft an invalid request on the server causing at the bare minimum a Denial-of-Service attack; though with some OSes remote code execution is possible. This is instigated by a stack overflow that is controlled on 64-bit OSes where return addresses are stored on the stack heap. In the collaborative experiment of Rajani, Mohamed, and Stansbury, the results indicated a successful breach with remote code execution being successful in the form of adding users with full permissions and writing files to the root directory on the web server. In the days of prevalent E-commerce some are considering these additions to the Metasploit program as haphazard. Big target for exploits reside in database servers that house large amounts of data ranging from social security numbers to financial data. One of the leaders in database management is Oracle with a n approximate market share of 40 percent. The majority of databases use the same language, Structured Query Language (SQL), thus often exploits targeting databases are based off the use of this language. Because of this no free penetration testing software currently offers an independent direct exploit to the system.

As we mentioned previously, a key feature of the Metasploit Framework is development. At the Black Hat USA Conference in 2009 Chris Gates and Mario Ceballos, presented a method for exploiting Oracle through SQL injection techniques utilizing custom built auxiliary modules. Their attacks consisted of seven steps that make up what they considered the basis of pen testing;

1) Locate a system running Oracle.
2) Determine Oracle Version.
3) Determine Oracle SID.
4) Guess/Bruteforce Username/Password.
5) Privilege Escalation via SQL Injection.
6) Manipulate Data/Post Exploitation
7) Cover Tracks

To do this a separate auxiliary module was required for each step. To locate a system the inclusive NMap was used to direct a port scan searching for commonly used Oracle ports, 1521 1540. A homemade TNS mix-in was added to the Metasploit trunk allowing it to craft TNS packets to determine the Oracle version. In order to guess the Oracle SID a SID enumerator was used as subsequent to version 9.2.0.8 Oracle no longer freely gives out this information. Brute forcing the username/password combination was done by using the pre-existing auxiliary module for Brute force logins using a dictionary list by Pete Finnigan. Privilege escalation of the username gathered in step four was accomplished with a SQL injection vulnerability in the DBMS_EXPORT_EXTENSION package. For post exploitation the win32exec module was used to execute a remote command on the machine to create a user on the

system for future access.

B. Meterpreter

One of the more powerful payloads discussed above, meterpreter, originally emerged in Metasploit version 2.2. What makes meterpreter so powerful is its elusiveness in being detected by even the most knowledgeable security analysts. Meterpreter is an advanced dynamically integrated payload that resides entirely in-memory by injecting DLL stagers. Once a compromised system is discovered and exploited the meterpreter payload establishes a client side command interpreter with which to communicate. This allows the attacker to remotely interact with the host system without having to establish separate connections. Under normal circumstances, once a system is exploited a single payload is delivered that is only able to execute one command and then it is done. This one command could be something as simple as adding a user or opening a remote shell with which to communicate. In doing this a resulting cmd.exe process would be created in the process list with SYSTEM rights. Immediately, this would raise red flags. However, with meterpreter DLL injection is used to upload the meterpreter process into the compromised processes' heap. Normally, an uploaded DLL would be written to the disc, yet meterpreter alters the way the Load Library utilizes core API calls redirecting them to the memory location of the meterpreter DLL.

The truly beautiful thing about meterpreter is its ability to remain undetectable by most commonly implemented host based IDSes. By embedding itself into preexisting processes and not altering system files on the hard-drive, the HIDS is never made any wiser. Not to mention, the process in which meterpreter hides can be changed at a whim so tracking it and stopping the process becomes rather difficult even to the trained eye.

C. Penetration Testing Automation

For years the dream of automating penetration testing, often abbreviated as pentesting, has been considered just that, a dream. The challenges facing post-exploitation automation include; visible processes, file transfer capabilities, and exploit expiration. The problem of readily visible processes with suspicious user rights. The only way around executing a separate visible process would be to install a root kit or backdoor, though this requires the transfer and installation of additional malware. This brings up the capability of a payload to transfer/install files to the remote system. To do this requires an advanced payload that will most likely be compromised in its writing of data to the remote system. Lastly, exploit expiration refers to the acknowledgement that some exploits can only be run once. Without separate sessions to enlist multiple tasks the process can become time consuming if not impossible. This would require the use of another exploit to complete other tasks. According to Irani and Weippl's research on post exploitation automation, meterpreter in conjunction with commonly installed scripting languages provides just the right tools for a solution to these problems. Meterpreter's innate ability to remain hidden in current processes through DLL injection allows a method around the visible process problem. Additionally, because the process is not blatantly listed the use of a root kit or backdoor is not necessary, thereby, voiding the need for file uploading and the risk of running into anti -virus scans. Though, on that note, an analysis conducted by Mark Baggett found that only 3 out of 32 reputable antivirus programs interpreted a standalone meterpreter payload as a security risk. Lastly, meterpreter also provides users with the ability to open independent sessions allowing for efficient multitasking. With the help of meterpreter post-exploitation scripts can be ran with the capability of not only further exploiting the current system, but previously non-exploitable machines as well. This technique of using an exploited machine to exploit a previously safe-guarded machine is known as pivoting. Not to go into too much detail, but the port forwarding service of meterpreter provides this capability allowing for a connection back to the initial attacker. The point is that the automation of post-exploitation tasks extends far beyond just automating the initial system. With the right script automation can be implemented to scan an entire network for vulnerabilities.

VII. CONCLUSION

The purpose of this paper was to provide the reader with an understanding of Metasploit such that he/she may use it himself. Metasploit is a powerful tool that like we said time and time again, in the wrong hands can be used for great harm. It provides an abundance of resources for legitimate network security professionals, security administrators, product vendors, and developers to use in a variety of ways. In fact, in its diversity lays the key to its success. However, the only real way to fully understand the intricate design of the Metasploit Framework is to use it. Hopefully, this paper helps others to understand the capabilities of Metasploit and utilize it as a tool for themselves.

VIII. ACKNOWLEDGMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the

people who made it possible, whose constant guidance and encouragement crowned the efforts with success. I would like to profoundly thank SRS Institute of Technology for providing such an environment for the successful completion of work. I would like to precise my thankfulness to Shashidhar M S & Vinay G many peoples who has assisted during the publication.

REFERENCES

[1] https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/
[2] http://www.metasploit.com/
[3] http://www.rapid7.com/products/
[4] http://www.qatar.cmu.edu/iliano/courses/06S -GMU -ISA767/project/papers/mohamed rajani -stansbury.pdf
[5] http://www.offensive security.com/metasploit unleashed/
[6] METASPLOIT the Penetration Tester's Guide by David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni. No Starch Press example of a book in