Capture The Flag(CTF)

Due on Wednesday, January 26, 2016

SYSTEM SECURITY FRI

Abraham Adberstein

Level 0

The flag was in main directory.

Steps:

```
level0@hades: $ ls
flag
level0@hades: $ cat flag
836a77c8ef948884ff77492830ece472
```

Level 1

The flag was in main directory. I decided to make a grep command that would look for the flag inside the directories. I did not expect this level to be easier than it seemed.

Pre-steps:
level0@hades: $ su - level1
pass: 836a77c8ef948884ff77492830ece472

Steps:

```
level1@hades: $ grep -rniw '/home' -e "[0-9a-f]{32}"
/home/level1/-:1:fc4e7bbd7a386abda17f74590e583cf0
```

Level 2

The flag was in main directory. I just looked for the hidden files.

Pre-steps:
level1@hades: $ su - level2
pass: fc4e7bbd7a386abda17f74590e583cf0

Steps:

```
level2@hades: $ ls -la
\home\level2\psst_in_here\.flag
level2@hades: $ cat flag
0069d91a41e9b225f19c7d47bec2980b
```

Level 3

The flag was in the main directory encoded in base64, so we can use a binary in linux to decode it. I read somewhere that some passwords in servers where encoded in base64 but turned to be a security vulnerability. I recognized the base64 encoding.

Pre-steps:
level2@hades: $ su - level3
pass: 0069d91a41e9b225f19c7d47bec2980b

Steps:

```
level3@hades: $ cat .flag
NzgwODkxMzg3Mzk3YWY2MzdhMTE5OWY0YTljYmRhNWUK
level3@hades: $ echo -n NzgwODkxMzg3Mzk3YWY2MzdhMTE5OWY0YTljYmRhNWUK | base64 -d
780891387397af637a1199f4a9cbda5e
```

Level 4

The flag has a long list of strings. Using some text binaries I sorted them (sort), displaying the number of repetitions (sort -nr) for each unique line (uniq -c).

When I was thinking what to do for this level, first I came to two conclusions. Either the string has unique characters or it is different to all others. After play first with the sort commands, I discovered there were many duplicates. I supposed the unique string would have no duplicates. I proceeded to improve the sort which would count the repetitions. The unique string would only appear once.

Pre-steps:
level3@hades: $ su - level4
pass: 780891387397af637a1199f4a9cbda5e

Steps:

```
level4@hades: $ cat flags | sort | uniq -c | sort -nr
10 .
10 .
10 .
1 9fb8778d5b6648bd8f80f9a0332db6b8
```

Level 5

A bash script was set to run in the .bashrc when users tried to ssh/change to this account. The solution was to use a new shell that would not load the bash script when it started up. The flag was just in a file in the main directory for this level.

Steps:

```
level4@hades: $ su - level5
9fb8778d5b6648bd8f80f9a0332db6b8
HERE.
BEGONE! I am Anger.
You shall never get past me mortal.
level4@hades: $ su - level4 -s /bin/sh
9fb8778d5b6648bd8f80f9a0332db6b8
level5@hades: $ cat flag
pass: J5lN3Qe4s7ktiwvcCj9ZHWrAJcUWEhUq
```

Level 6

I did not want to look inside every folder for the password so I decided to make a grep line to find the pattern of the MD5 password. A regex expression for recursively finding the pattern inside the different directories was the ideal tool.

Pre-steps:
level5@hades: $ su - level6
pass: J5lN3Qe4s7ktiwvcCj9ZHWrAJcUWEhUq

Steps:

```
level6@hades: $ grep -rnws '/usr/share' -e "[0-9a-f]{32}"
/share/info/you_cant_see_me:1:175636dc12b41e6d222350a2cc7b694f
```

Level 7

For this level I decided to use a regex expression as well for recursively finding the pattern of MD5 passwords on different folders in the main directory.

Pre-steps:
level6@hades: $ su - level7
pass: 175636dc12b41e6d222350a2cc7b694f

Steps:

```
level6@hades: $ grep -rnws '/home/level7/' -e "[0-9a-f]{32}"
/home/level7/-flag19:1:70b043bd272de8172ef990d657d99da3
```

Level 8

The flag was a binary file that when ran showed an interrupt signal. GDB was used to analyze the file, the "file" command , and the "idd" command as well. Object dump (-s) showed the contents of the file and the flag. Later, just by reading the file one could find the flag at plainsight...

Pre-steps:
level7@hades: $ su - level8
pass: 70b043bd272de8172ef990d657d99da3

Steps:

```
level8@hades: $ objdump -s flag"
```

Level 9

This level was the hardest of all, mainly consisting of brute-force approaches. The .ssh file was confusing because the flag might be used as key or file to create a key for ssh. In the end, the file was just not important. Several attempts to find the MD5 password were tried. Dividing the flag file into strings of 32 characters was the first attempt, but did not result. Then the file was divided every 32 characters moving one character at a time, giving more than 8000 passwords.

Some attempts were tried to convert the hex file to a binary and find a hidden binary packet inside. Another attempt was to create a ssh key or find a way related to the different encryptions systems used for ssh.

Pre-steps:
level8@hades: $ su - level9
pass: c20b252dede019e2083053dc89f711c8

Scripts:

A python script was needed to obtain the passwords and then bash script was used to brute force the ssh. It took over 2-3 hours to find the password.

Bash Script, test.sh

```
#!/bin/sh

counter=1
while IFS='' read -r line || [[ -n "$line" ]]; do
  echo "$counter. Attempt: $line"
  sshpass -p "$line"  ssh -T level10@hades
  let counter=counter+1
done < "$1"
```

Level 9 continued on next page. . .

Python Script, finder.py

```
def main():
  filename = "flag"
  file = open(filename, "r+")
  myString =  file.readline().strip()
  passwords = []

  for i in range(0, len(myString)):
    passwords.append(myString[i:i+32])
    if i+32 == len(myString):
      break

  for i in passwords:
    print i

main()
```

Steps

```
level9@hades: $ python finder.py >> flag.txt
level9@hades: $ chmod +x test.sh
level9@hades: $ ./test.sh flag.txt
.4794
.
.
.4795
Permission denied, please try again.
4796. Attempt: d7fce2ffe4adef128e1c26382bdda570
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

Success!

Level 10

For this level, the output was reversed into binary. It was a bit tricky because it wouldn't be converted until the flag "-p" was added to consider the plain hexdump file. Afterwards, the file command was used to see the type of zip file, which was bz2. The file was unzipped showing a gzip file inside. The file was unzipped again and the ASCII text flag was found.

Pre-steps:
level9@hades: $ su - level10
pass: c20b252dede019e2083053dc89f711c8

Steps:

```
level9@hades: $ xxd -r -p flag.txt flag
level9@hades: $ file flag
flag: bzip2 compressed data, block size = 900k
level9@hades: $ mv flag flag.bz2
level9@hades: $ bzip2 -d flag.bz2
level9@hades: $ file flag
flag: gzip compressed data
level9@hades: $mv flag flag.gz
level9@hades: $gzip -d flag.gz
level9@hades: $cat flag
82564ecf01f3bbdc490531bb10df2f94
https://www.youtube.com/watch?v=dQw4w9WgXcQ82564ecf01f3bbdc490531bb10df2f94
```

Never Gonna Give You Up!