Shi Su
shis

Problem #1 : Running Kerberos and Wireshark

1. Follow the white rabbit

Script available in **shis_hw2/shell.c**

2. Take the red pill

2.1 Analyze the vulnerability

(1) Where is the vulnerability?

The faulty buffer used is **newname** in function convert_fileName, and global variable **target**.
Newname's offset to the value contained in ebp is **-4124**.
Address of target is **0x0804d180**

(2) Where is/are the insecure buffer fill(s) in the source code, operating on the buffer in (1)

The buffer fills used in the exploit:
ploymorph.c: 118 in grok_commandLine strcpy(target, optarg);
ploymorph.c: 200 in convert_fileName strcpy(newname, original);

Other issues:
ploymorph.c: 191-198 in convert_fileName

```
for(i=0;i<strlen(original);i++){
        if( isupper( original[i] ) ){
                newname[i] = tolower( original[i] );
                continue;
        }
        newname[i] = original[i];
}
newname[i] = '\0';
```

2.2 Exploit the vulnerability

Script available in **shis_hw2/rtl_exploit.c**
In this attack, the overflowed buffer change the stack of program from
[...][ebp][return address to main][arg 1 of convert_fileName][...]
to
[a...a][aaa][entry of __libc_system][arg 1 of convert_fileName][address of /bin/sh][/bin/sh]

3. Escape from the Agent Smith

Script available in **shis_hw2/new_exploit.c**
In this attack used the gadgets in polymorph as well as libc, to create a chain of instruction:
pop %ebx ret; # pop out the arg1 for convert_fileName
pop %eax ret; # put the entry of __libc_system on stack, this instruction pop into eax
inc %eax ; pop %edi ; pop %esi ; ret; # increase the value of eax by 1
call %eax # call the function whose address stored in eax

Problem #2 : Side Channel

1. How can you attack this?

Because the MAC verification process validates the MAC string character by character, and terminate on first incorrect character, plus there're some slow operation involved during the validation of each byte. A MAC starts with more number of correct characters takes longer to validate.

So for a certain ticket, we can start with an arbitrary 32 bytes hex string MAC. From first character of MAC string we change it from 0 to F to see which particular value cause slower response time, that means the character is correct, and server proceeds to next character. To deal with the response time variation caused by network or other random issues, we should repeat the process for each character large number of times, and find the value that has longest response time statistically. Then apply this method to the following characters, eventually we will be able to deduce the correct MAC.

2. Golden ticket

Golden ticket : {"username":"shis","is_admin":"true","expired":"2022-01-31"}
MAC : 99c847dc21362f3fa5ccda858ddbbd7a
Script available in **shis_hw2/shis_goldenticket.py**

Below is the list of supported argument, when running without arguments, the script generates MAC for the ticket above.

Use "-u" to provide a different username, e.g. "instructor"
Use "-a" to provide admin setting, "true" or "false"
Use "-e" to provide expiration date, e.g. "2016-01-31"
Use "-h" for more information