ELEC 377 Lab 5: Software Security – Testing

Section 003, Group 16 – Thursday Lab

Erhowvosere Otubu, #20293052

Ivan Samardzic, #20296563

Date of Submission: December 5, 2023

Environment Variables

To analyze the teams' ability to obtain environment variables, the following three cases were analyzed.

Case 1 – Sample Shell

Given the objective to obtain environment variables from a server, the first test is a base case to display the sample environment variables of the current shell. The command below is executed in a new shell:

```
printenv > environsample.txt
```

This command obtains all environment variables of the current shell and redirects the output to the file "environsample.txt". Upon execution of this command, several environment variables such as SHELL, LS_COLOURS, and USER are printed, as expected. This test case now confirms the expected output for environment variables within a sample shell.

Case 2 – SelfComp

Upon attacking parameters, writing shell code, and obtaining final return addresses as described in steps 6.1-6.3, the selfcomp program is now executed, as follows:

```
make selfcomp

./selfcomp > selfcomp.txt
```

Visibly, the program's output is redirected to the file "selfcomp.txt", where line 33 indicates the program is executed through selfcomp. Upon inspection of this output, we can see that all environment variables match those obtained from the sample shell in case 1. In addition, an extra line is added to the end of the output to display a sample MD5 hash. Given the nature of the selfcomp script, this preforms as expected, and the program is therefore declared fully functional.

Case 3 – Server Program

According to the guidelines in section 6.4, the same steps are finally repeated for the client and server program. The client is executed through the server program with the following commands:

```
./quoteserv 10000

make client

./client 10000 > server.txt
```

Visibly, quoteserv and client use an arbitrary port number of 10,000 for testing, and the execution output is redirected to the file “server.txt”, where line 33 indicates the program is executed through quoteserv. Upon inspection of this output, we can see that all environment variables match those obtained from the sample shell in case 1, and the selfcomp program in case 2. In addition, an extra environment variable is printed to the end of the output to display the MD5 hash, like in the previous step. However, this MD5 hash is real, as opposed to in selfcomp. In other words, we have now obtained the target MD5 hash to be used for guessing the user’s password. In conclusion, the client script preforms as expected, and the program is therefore declared fully functional.

Password Guessing

Now that the target MD5 hash has been obtained from the client and server program, the Twitter and Facebook accounts of our user, Sam Gualt, can be analyzed to guess the user password. For simplicity, the password was set to be a single word, followed by a number. Given the password specifications, several key words and numbers were extracted from his social media accounts, and used to guess the password corresponding to the MD5 hash obtained earlier.

Target Number

Upon analysis, it was confirmed that only two possible numbers stood out within the social media accounts.

Sam commented on a post that “MmmBop” was the top song on the day he was born. With a simple google search, it is discovered that the release year of this song is 1997. Therefore, the number, 1997 is our first possibility for his password number.

Using the number obtained above, we can also calculate Sam’s current age. Specifically, being born in 1997 indicates that Sam is currently 26 years old. The number, 26, gives us our second number possibility as a result.

Target Word

Once the target numbers were narrowed down to 2 possibilities, Sam’s social media accounts were scavenged for target words.

Specifically, names/places/foods, and any other vital information was extracted from Sam’s account, posts, and comments. Each target word was paired with each of the target numbers, and then trial and error process was then carried out.

Trial and Error

Once all target words and numbers were extracted from Sam’s social media accounts, separate password guesses were executed with the following command:

```
echo -n WordNumber | md5sum >> guesses.txt
```

The above command creates a md5 hash code corresponding the password guess defined in “WordNumber”, which is replaced with a target word and target number. The corresponding md5 code is redirected and appended to the file “guesses.txt” which is used to store the password and respective md5 of every single guess made. Each md5 output is compared to the md5 obtained from environment variables of the of the server program. If an identical match is found, the password guess is declared to be the client’s password.

In our group’s case, the password guess “Modesto1997” produced a hash code identical to the one obtained from the server program. As obtained from his accounts, Modesto is Sam’s hometown, while 1997 is his year of birth. Therefore, “Modesto1997” is the correct password, and the output of the testing is stored in the file “correctpassword.txt”.

Online MD5 Hash Decryption

To further confirm our newly identified password, the hash code obtained from the server program was put through an online tool for md5 hash decryption. This decryption also produced a value of “Modesto1997”, indicating our password is indeed correct.

With all environment variables extracted from both selfcomp, client/server, and the final password identified, the lab is complete.