

Writeup Luxembourg Cybersecurity Challenge 2018

Mike Lorang

August 12, 2018

Challenge #0

The first challenge gives me the following statement: *Unfortunately, the user has forgotten his password. You need to analyze the binary file and try to find the password. If you have found the correct password, you can enter it in the online system.* After downloading the Linux binary, and giving it the rights to execute (`chmod +x linux-password.bin`) it is ready to execute. After executing the following prompt appears:

Please enter your password:

Now you can run `strings` or `ltrace` to get the flag respectively the password, namely: `"lon20ecsc18don"`.

Challenge #1

The challenge number 1 gave me the following information: *Michele hides a nasty flag in a nasty picture. Michele doesn't like others to see the stuff so several steps are taken to hide it. You got an image of an 8MB USB stick and have to solve a bunch of easy challenges to get to the result. But are you also able to get the full picture to achieve full score?.*

Additionally I got a hint saying: *Imopray, ometimesay NTFSAY artitionspay oday ot-nay orkway onway Inuxlay. Andway ometimesay Inuxlay artitionspay oday otnay orkway onway Inuxlay. Ecundosay, Astermay Ootbay Ecordsray areway otnay alwaysway onlyway infectedway utbay ancay ontaincay ingsthay orfay aterlay7. Inallyfay icspay andway exifsway areway unfay.*

This is pig latin, which means translated:

Primo, sometimes NTFS partitions do not work on Linux. And sometime Linux partitions do not work on Linux. Secundo, Master Boot Records are not always only infected but can contain things for later7. Finally pics and exifs are fun.

Additionally an .img file was given.

After downloading the image, I analyzed it with *autopsy* on Windows. I found on *vol1* unallocated space. When displaying that with `cat`, the following string was readable `546865416e737765724973`. Converting that from hex to ASCII results in *TheAnswerIs*.

On *vol2* seems to be a Hidden partition.

On *vol3* there is a .7z file. When trying to extract it, it prompts for an password. When using "TheAnswerIs" as password, the file decrypts correctly. As the hint proposed I checked the metadata of the picture. The Flag is in the XMP and is: FLAG: a1d0c6e83f027327d8461063f4ac58a6. Tadaa.

Challenge #2

This challenge was about decrypting a banner. When opening the file with a hex-editor one could see two types of patterns. One pattern was 0xaa 0x5e 0xe6 0xa4 0x12 0xb3 0x3d 0x2b. The other was 0x9 0x38 0x1f 0x6e 0xcd 0x69 0x81 0x36. I proceeded by replacing the first pattern by "-" and the second by "#" (without the quotes). Now I just need to adjust the size of the window to find that the banner is "ABCABC"

Challenge #3

I went to bfy.tw/F5k5 where I was redirected to another site which told me to use Google and download Fabios does Steganography. After having downloaded the steg tool, I used it to extract the information out of the image. I got an archive file which I extracted. In the archive there was the *diary.txt* file, as well as a file called *encrypted.zip*. The diary was saying something about trust, headers and suffixes. Because of the fact that I was not able to extract the encrypted.zip file, I deduced that the suffix ".zip" was not correct. The "file" command and an online file format analyzer did not bring me any further. When I was looking into the file itself it remembered me of a .7z file. I tried to change the header file of it and extracting it, but without success. The header and the footer of the file were certainly broken. That is only as far as I got :(

Challenge #4

I unfortunately did not get very far with this challenge. The hint, that was base64 encoded did not help very much. I read the release notes of the different openssl version but could not find a relevant vulnerability which was patched in 0.9.zh. The vulnerability which was published in *September 6th* 2006 didn't get me far. As the cleartext was only 6 bytes, it could be possible to brute-force it. But this would be almost infeasible because I have no idea what the cleartext looks like.

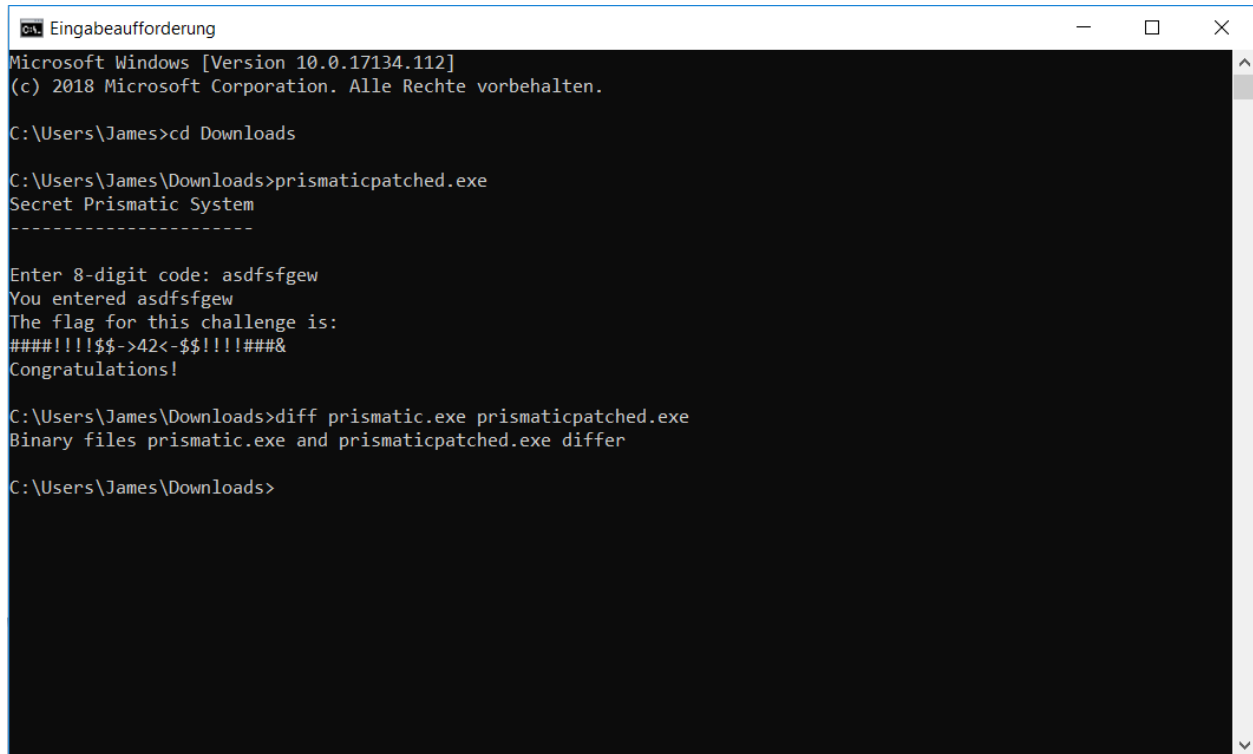
Challenge #5

In this challenge we got an executable, *prismatic.exe*. When executing, we get prompted for an 8 digit code. When entering it incorrectly it says that it is wrong and we do not get the flag. When executing *strings.exe*, we get the strings of the program. Unfortunately the flag seems to be encrypted somehow. I then had a look at it in the *ollydbg*. After several runs

I figured that I could negate a jump condition, such that it does jumps to the clause giving me the flag.

The Link to the patched code is:

”<https://github.com/punshLineLama/ctf/tree/master/ecsc/Challenge5>”.



```
Eingabeaufforderung
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\James>cd Downloads

C:\Users\James\Downloads>prismaticpatched.exe
Secret Prismatic System
-----

Enter 8-digit code: asdfsfgew
You entered asdfsfgew
The flag for this challenge is:
####!!!$$->42<-$$!!!!###&
Congratulations!

C:\Users\James\Downloads>diff prismatic.exe prismaticpatched.exe
Binary files prismatic.exe and prismaticpatched.exe differ

C:\Users\James\Downloads>
```

Figure 1: Terminal showing the flag when executing the patched program

The flag is: `####!!!$$->42<-$$!!!!###&`

Challenge #6

The hint was base64 encrypted and results in: ”-tttt is the winner, precision does matter, but some times, precision is suspicious.” After some research it turns out that -tttt is a tcpdump flag.

The part about ”precision is suspicious” may be referred to that the first 16 packets arrive in a very short period of time.

When inspecting the milliseconds at which the packets arrived, one can see that when converting the milliseconds of the arrival time of the first and last 16 packets, from decimal to ASCII, that they are all within the hexadecimal range. Hmm dis is suspicious.

Simply converting the hexadecimal numbers to ASCII does not result in any readable ASCII signs. So maybe we have to apply some operations on them.

I wrote a little script which was doing pairwise operations on the integer values of the milliseconds. The result was not satisfying thought. I did not manage to recognize any pattern

or message in it. I uploaded the script to my Github anyway. Here is the link to it:
<https://github.com/punshLineLama/ctf/tree/master/ecsc/challenge6>