Memory Loss – Find your way back (Workflow/WriteUps for all CTFs)

The order of CTF listed in this document corresponds to the order of the CTFs in the story and in the CTF overview.

7) CTF - Web Exploitation (2):

- The user must find the flag in the html source.
- Right click on the challenge page --> open the inspector view --> Element tab.
- To find the flag you can either look manually through the HTML source code or: CTRL+F and search for "puzzlePiece"
- The flag is located in the code right after the five shelfs in a hidden
- Flag: puzzlePiece{og9XEhgLE2}

4) XSS (1):

- In this challenge, the player should inject an “alert(“XSS”)” function using a script tag.
- This script will be executed from the local client. One ways to inject the script is: “<script> alert("XSS");</script>”
- It is also possible to use <script alert("XSS")/>, but keep in mind, that this is an HTML one line tag.
- This CTF does not require searching for a flag
- The execution of this XSS is simulated because the browser will not execute it automatically.

1) CTF - Web Exploitation (1):

- The goal here is for the player to find their login data in the source code. This is possible because the logins are hardcoded in the source code with base64 encoding.
- To solve the challenge, open the inspector --> search for ("Incorrect Username or Password")
- In this function you should find the base64 encoded login data. The player should now use an online base64 decoder to get the usable login data (for example.: https://www.base64decode.org/)

Solution:

- Username (deocded): www-admin
- Password: puzzlePiece{DhFpMXHyX5}

- The password is also the flag the needs to be entered in order to resume to the next page.

11) CTF - Cryptography (3):

- Copy the text
- Go to an encryption identifier to identify the encryption method. If the player has some knowledge regarding encryption methods, they might be able to recognize a pattern in the encrypted text. The encrypted text has clearly separated world with shifted characters and a key.
- The encryption method used here is Vigenère. Using the online tool https://gc.de/gc/vigenere/ the text can be easily decrypted.
- The flag is written in the decrypted text.
- The player must enter the flag in the following format: puzzlePiece{}

Solution:

- puzzlePiece{Amj8u4y2Ka}

5) CTF - XSS (2):

- Here the goal is to teach the player that other HTML tags can be used in XSS. The image in the browser should be a hint for using the image tag.
- For that the player must define an arbitrary location for the image and use the attribute onerror to call the alert()-function.
- Here is a sample XSS code: ""
- The JavaScript function alert(1) will be executed since the program is not able to load the image after we replaced the correct path with an invalid one.

3) CTF - Forensics (2):

- For this challenge you should copy the text. And then go to an online Base64 converter. (Example: Base64 to Image Decoder / Converter (codebeautify.org))
- The flag is on the image you generate form converting the string.

Solution:

- puzzlePiece{TJeqNhQ5C9}

10) CTF - Cryptography (2):

- This challenge can be done manually using pen and paper, by using online tools such as https://www.rentfort.de/ or by writing a small program.
- The goal is to map every number to character following this pattern (A=1, B=2, C=3, Z=26 etc.)

Solution:

- puzzlePiece{FHZJDFGETR}

2) CTF - Forensics (1):

- To pass this challenge, you will need to download the image.
- Right click on the file --> properties --> Details. The flag is in the title field.

Solution:

- Flag: puzzlePiece{gx6wQudvZb}

8) CTF - Cryptography (1):

To complete this challenge the user has to write a small program that does the following:

- Takes each number from the list an calculates it to mod 42
- Takes the results and maps it to the pattern defined in the challenge
- The output should be inserted into puzzlePiece{insertOutputHere}
- Example Python code that can be compiled online (https://www.online-python.com/):

```
#!/bin/pyhton
list = [181, 152, 103, 335, 147, 256, 101, 192, 125,
396, 172, 170, 104, 269, 214, 377, 288, 324, 401,
142, 163]
char = ['#', '!', '"', '{', '}']

for i in list:
        x = i % 42
        if 0 <= x <= 25:
                print(chr(65+x), end='')
        if 26 <= x <= 35:
                print(chr(48+x-26), end='')
        if 36 <= x <= 40:
                print(char[x-36], end='')
        if x == 41:
                print("_", end='')
```

Solution:

- Flag: puzzlePiece{N0T_VERY_SECURE_#4XQ!}

9) CTF - Reverse Engineering:

- For this challenge you can either manually sort the array characters from 0 to 9
- Or the faster way is to copy the code to your programming environment of choice and change the method `encryptSecret()` to create a string from the char. The main method should then call the function `encryptSecret()`
- Example code:

```
public class ctf9 {
    static char [] theSecret;
    public static String encryptSecret() {
        theSecret = new char [10];
        theSecret [2] = 'G';
        theSecret [5] = 'J';
        theSecret [0] = '7';
        theSecret [6] = '5';
        theSecret [8] = 'e';
        theSecret [3] = '8';
        theSecret [9] = 'Y';
        theSecret [1] = 't';
        theSecret [4] = 'e';
        theSecret [7] = '8';
        return new String(theSecret);
    }

public static String conclusion() {
    String conc = "This case is unsolvable. Give  u
    p!";
    return conc;
}

    public static void main(String[] args) {
System.out.println(encryptSecret());
    }
}
```

Solution:

- Flag: puzzlePiece{7tG8eJ58eY}

6) CTF - XSS (3):

- Here the player should notice that he/she can edit the URL. Another thing we point the players attention to is the code that can be accessed through the light blue icon on the right.
- The player should examine the section of the code in which the result HTML is generated. He/She will notice that user input is used without any validation, which makes it possible to inject an image tag with an on-error attribute.

Here's a possible solution:

"https://twine.ctf/ctf6.html#3' onerror='alert();//"

- With this solution, the program will try to load the image and then fail. Therefore, the code on error with be executed.
- No flag required