TryHackMe Room Write-up

Title: W1seGuy

Description: A w1se guy 0nce said, the anwser is usually plain as day.

Author: Igor Buszta

Source code

Description: *Yes, it's me again with another crypto challenge! Have a look at the source code before moving on to Task 2. You can review the source code by clicking on the Download Task Files button at the top of this task to download the required file.*

Objective: *Analize the downloaded code.*

Tools: *macOS, VS Code*

Let's focus on the important elements of the code. It contains variable *flag*, which is later XORed (⊕) with a randomly generated key which we need to guess. The formula is:

$$\text{RealFlag} \oplus \text{key} = \text{encodedText}$$

```
flag = 'THM{thisisafakeflag}'
```

Picture 1: Fake flag variable.

```
flag = open('flag.txt','r').read().strip()
```

Picture 2: Real flag, hidden in the server.

We are in fact sure, that the flag begins with THM{ (XORed *t* gives backslash '\' but the key is only digits and characters, the real flag is in flag.txt loaded at the beginning).

The fifth character remains unknown and we need to bruteforce it.

```
for i in range(0,len(flag)):
    xored += chr(ord(flag[i]) ^ ord(key[i%len(key)]))
```

Picture 3: Flag XORing.

```
res = ''.join(random.choices(string.ascii_letters + string.digits, k=5))
key = str(res)
```

Picture 4: Generating random key. k=5 tells us the length of it.

Get those flags!

Description: *Your friend told me you were wise, but I don't believe them. Can you prove me wrong?*

When you are ready, click the Start Machine button to fire up the Virtual Machine. Please allow 3-5 minutes for the VM to start fully.

The server is listening on port 1337 via TCP. You can connect to it using Netcat or any other tool you prefer.

Objective: What is the first flag? What is the second and final flag?

Tools: AttackBox, macOS, netcat (nc), Python (VS Code)

The code is available in the github repo but below you can find the vital fragments of it.

First thing we do is do XOR operation on inputed HEX string and know prefix, which is *THM{*.

```
known_prefix = "THM{"
partial_key = ""

# First 4 digits of key
for i in range(4):
    partial_key += chr(ct[i] ^ ord(known_prefix[i]))
```

Picture 5: XOR operation in our solution code.

```
    # Decyrpting message
    for i in range(len(ct)):
        decrypted_text += chr(ct[i] ^ ord(candidate_key[i % 5]))

    is_printable = all(32 <= ord(c) <= 126 for c in decrypted_text)
    ends_correctly = decrypted_text.endswith('}')

    if is_printable and ends_correctly:
        valid_solutions.append((candidate_key, decrypted_text))

return valid_solutions
```

Picture 6: Brute forcing the key and seeing if decrypted text ends with '}'.

There's also another important information – the flag ends with '}'. We use that to narrow our output of bruteforce to flags starting with 'THM{' and ending with '}'. We login into the server via AttackBox using netcat and IP given at the time.

```
> /opt/homebrew/bin/python3 /Users/igorrob/Downloads/decryptor_w1seguy.py
HEX: 3e187e4a185b315f5f1c2f2847701c1e64505a0b2b3e410209061c4a593d18244a011d18287c4315
KEY: jP31h
FLAG: THM{p1alntExtAtt4ckcAnr3alLyhUrty0urxOr}
```

Picture 7: Output of the code in VS Code terminal.

```
root@ip-10-80-129-178:~# nc 10.80.146.149 1337
This XOR encoded text has flag 1: 3e187e4a185b315f5f1c2f2847701c1e64505a0b2b3e41
0209061c4a593d18244a011d18287c4315
What is the encryption key? jP31h
Congrats! That is the correct key! Here is flag 2: THM{BrUt3_ForC1nG_XOR_cAn_B3_
FuN_nO?}
```

Picture 8: AttackBox terminal with inputed key.

As visible in Picture 7, there's only one candidate for a valid key, giving us the flag which meets our requirements. We copy the first flag from VS Code terminal and second and final flag from AttackBox terminal. We solved the room!