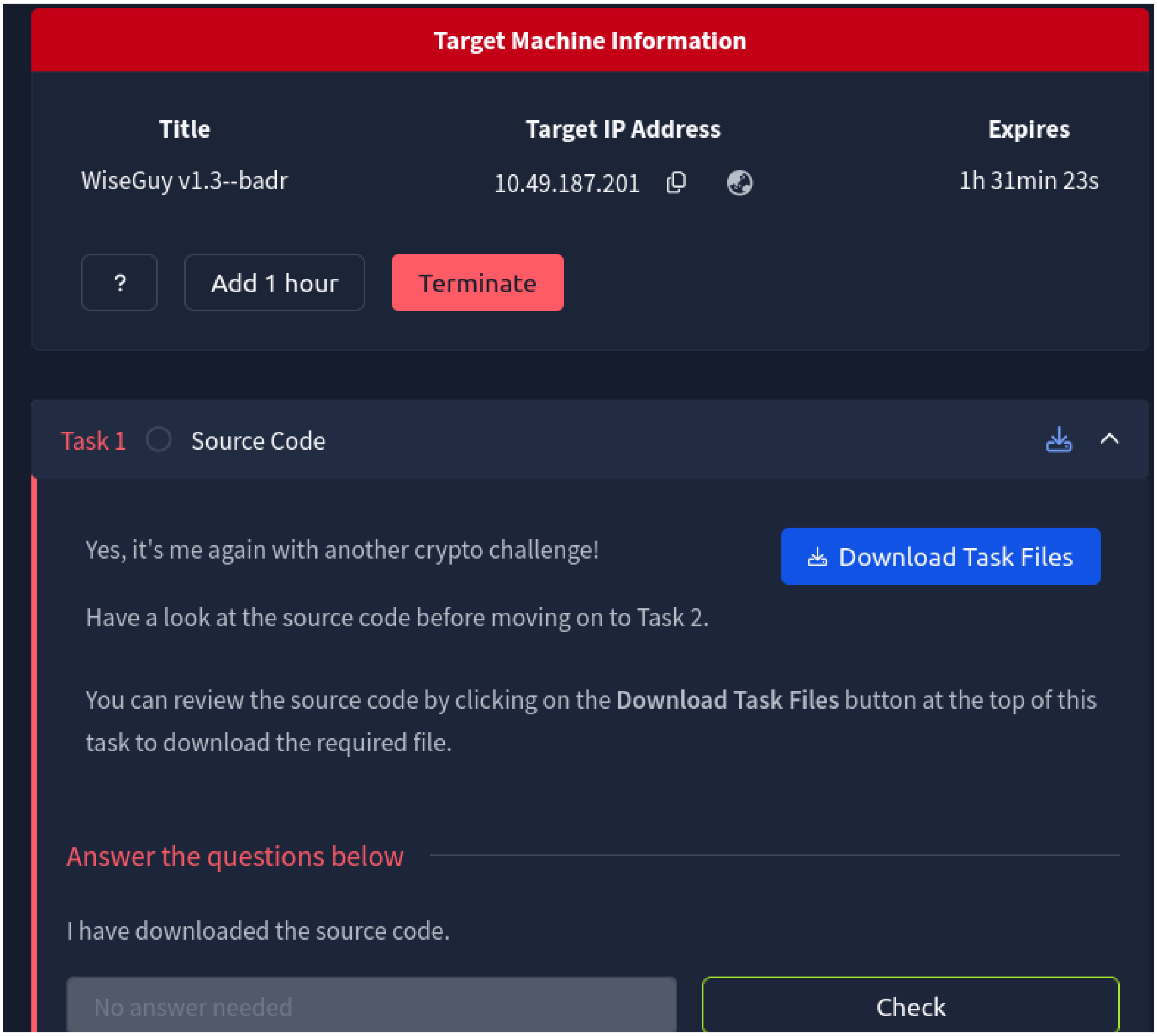
# TryHackMe – W1seGuy Write-Up

## Overview

This room focuses on breaking a weak **XOR encryption** using a **known-plaintext attack**. The service provides an XOR-encrypted flag and asks for the encryption key. By understanding how XOR works and using basic Python scripting in Kali Linux, we can recover the key and decrypt the flag.



Target Machine Information

Title	Target IP Address	Expires
WiseGuy v1.3--badr	10.49.187.201	1h 31min 23s

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.

Answer the questions below

I have downloaded the source code.

No answer needed

## Tools Used

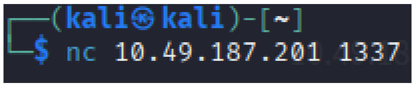
* Kali Linux

* Netcat (`nc`)

* Python 3

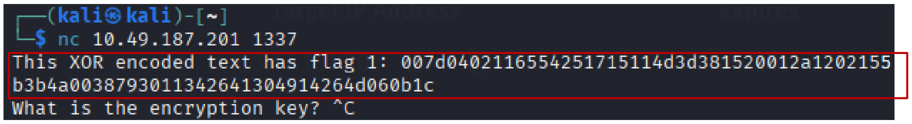
## Step 1: Connecting to the Service

First, connect to the target machine using netcat on port `1337`.



```
┌──(kali㉿kali)-[~]
└─$ nc 10.49.187.201 1337
```

After connecting, the service displays an XOR-encrypted hex string and asks for the encryption key.



```
┌──(kali㉿kali)-[~]
└─$ nc 10.49.187.201 1337
This XOR encoded text has flag 1: 007d0402116554251715114d3d381520012a1202155
b3b4a00387930113426413049142 64d060b1c
What is the encryption key? ^C
```

## Step 2: Understanding the Encryption

Important observations:

* The encryption method is **XOR**

* The key length is **5 characters**

 TryHackMe flags always start with `THM{` and end with `}`

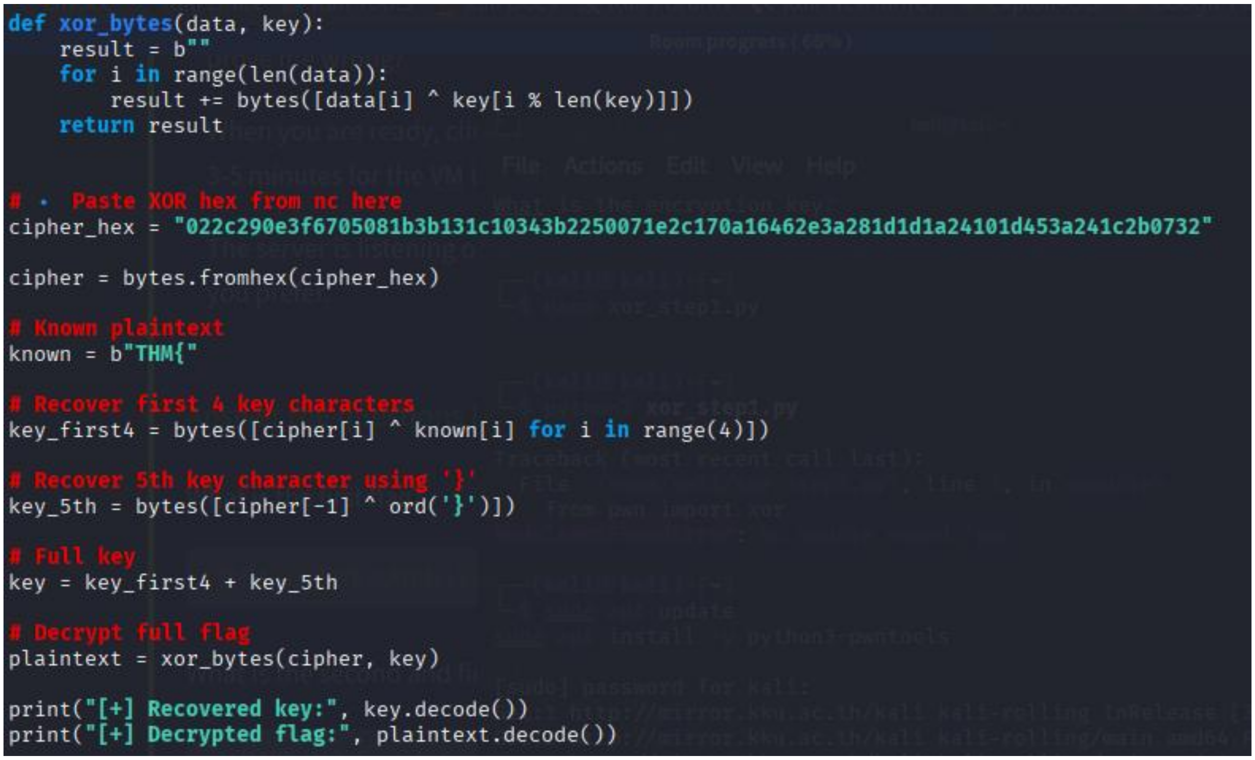
This allows us to perform a **known-plaintext attack**

## Step 3: Writing the XOR Solver Script

To automate the process, a Python script was written to:

1. Convert hex to bytes
2. Recover the XOR key using known plaintext
3. Decrypt the full ciphertext

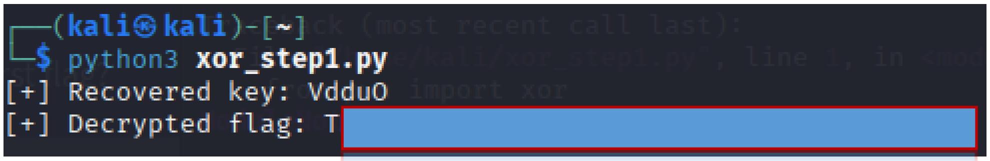
Python script



```python
def xor_bytes(data, key):
    result = b""
    for i in range(len(data)):
        result += bytes([data[i] ^ key[i % len(key)]])
    return result


# ·  Paste XOR hex from nc here
cipher_hex = "022c290e3f6705081b3b131c10343b2250071e2c170a16462e3a281d1d1a24101d453a241c2b0732"

cipher = bytes.fromhex(cipher_hex)

# Known plaintext
known = b"THM{"

# Recover first 4 key characters
key_first4 = bytes([cipher[i] ^ known[i] for i in range(4)])

# Recover 5th key character using '}'
key_5th = bytes([cipher[-1] ^ ord('}')])

# Full key
key = key_first4 + key_5th

# Decrypt full flag
plaintext = xor_bytes(cipher, key)

print("[+] Recovered key:", key.decode())
print("[+] Decrypted flag:", plaintext.decode())
```

## Step 4: Running the Script

The encrypted hex from the netcat session is pasted into the script and executed:

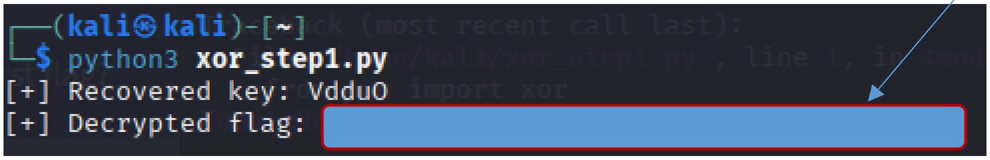


```
┌──(kali㉿kali)-[~]
└─$ python3 xor_step1.py
[+] Recovered key: Vddu0
[+] Decrypted flag: [redacted]
```

The script outputs:

* The recovered 5-character XOR key

* The decrypted flag



```
┌──(kali㉿kali)-[~]
└─$ python3 xor_step1.py
[+] Recovered key: Vddu0
[+] Decrypted flag: [redacted]
```

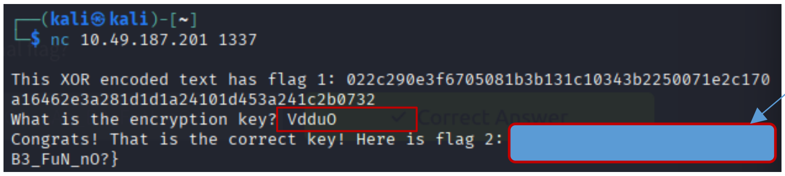
Flag 1

## Step 5: Submitting the Key

Without closing the netcat session, the recovered key is entered when prompted:

What is the encryption key?

If the key is correct, the service responds with the next flag.



```
┌──(kali㉿kali)-[~]
└─$ nc 10.49.187.201 1337

This XOR encoded text has flag 1: 022c290e3f6705081b3b131c10343b2250071e2c170
a16462e3a281d1d1a24101d453a241c2b0732
What is the encryption key? Vddu0
Congrats! That is the correct key! Here is flag 2: [redacted]
B3_FuN_n0?}
```

Flag 2

## Important Notes

* The encrypted hex **changes every session**

* Reconnecting requires repeating the process

* The key must be submitted in the **same session**

## Conclusion

This room demonstrates why XOR encryption is insecure when used improperly. By leveraging known plaintext and basic scripting, the encryption key can be recovered easily. The challenge reinforces core concepts in cryptography, scripting, and CTF workflow.

## Learning Outcomes

* Understanding XOR encryption weaknesses

* Performing a known-plaintext attack

* Writing reusable Python scripts for CTFs

* Managing session-based challenges

**Room: ** TryHackMe – W1seGuy

Category: Cryptography

Difficulty: Easy / Medium