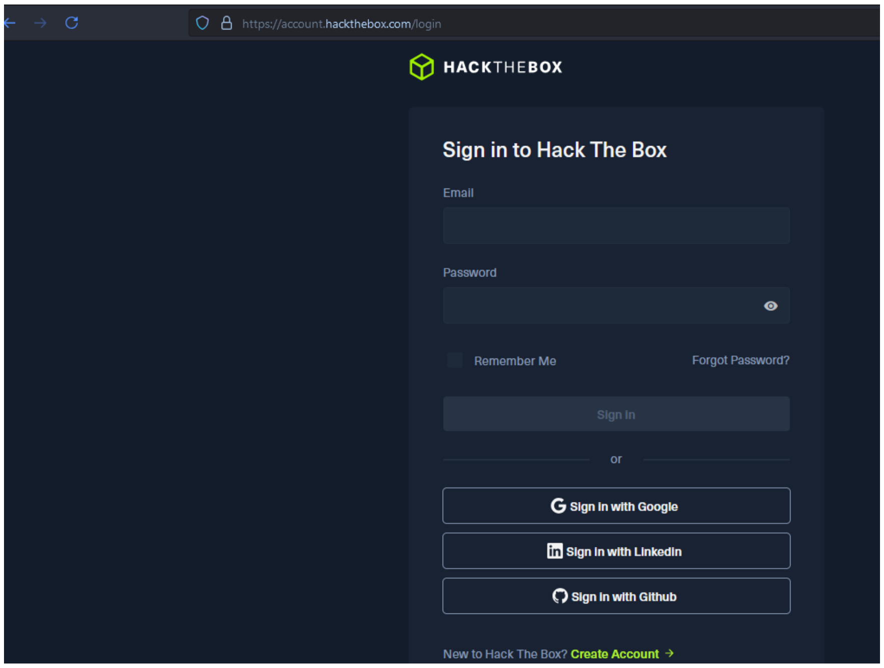
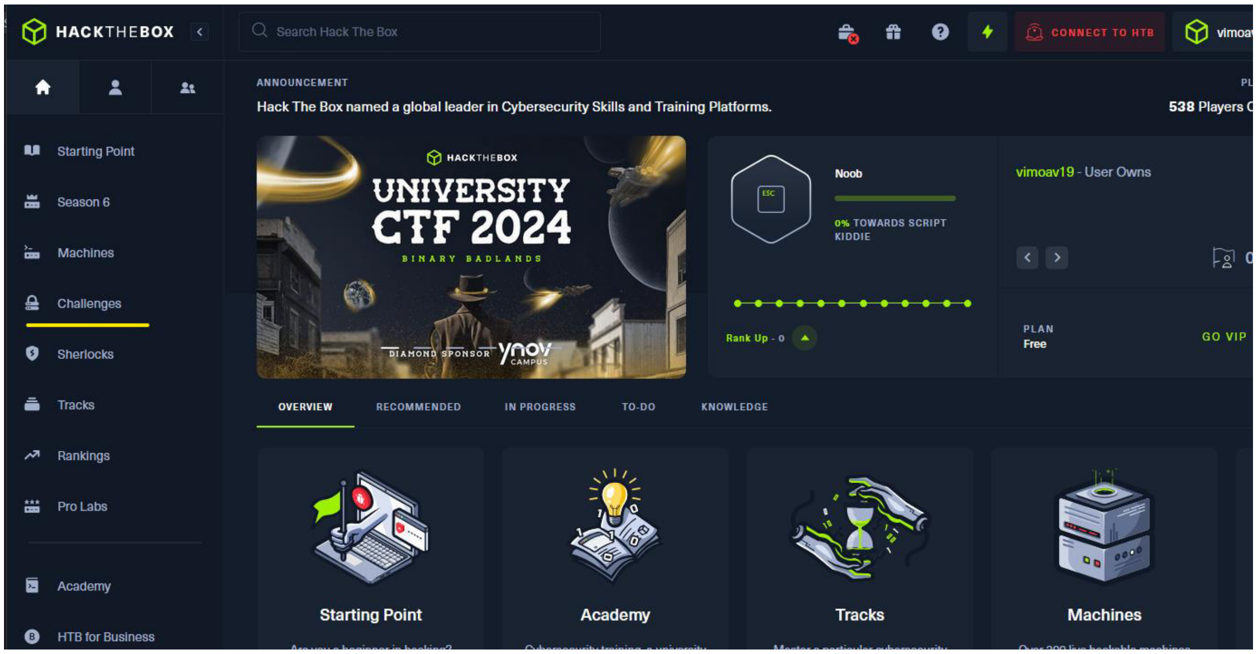
Hack the Box Challenge: Signing Factory
Category: Crypto
Difficulty rating: Medium
Description: A group of researchers has developed a new modern method for signing tokens used for authentication, and they are getting ready for an ultimate security audit before their product is released. We were assigned to perform the security evaluation of their signing mechanism and server. The challenge focuses on the assessment of the new method of signing messages against past known attacks on similar systems (Hack the Box, 2024).

Project Scope
This project aims to identify, analyze, and exploit any vulnerabilities in the signing mechanisms used for tokens on the assigned server. We will start gathering information to identify vulnerabilities, accessible services, endpoints and exposed APIs. The information obtained may show us the internal architecture or specific implementations related to the signing process. After the analysis we will have a clearer understanding of the signing process, which includes the analysis of the algorithms, cryptographic methods, and protocols and identify potential weaknesses.
https://www.hackthebox.com/



You would need to create an account. After login you can access the challenges in the left side

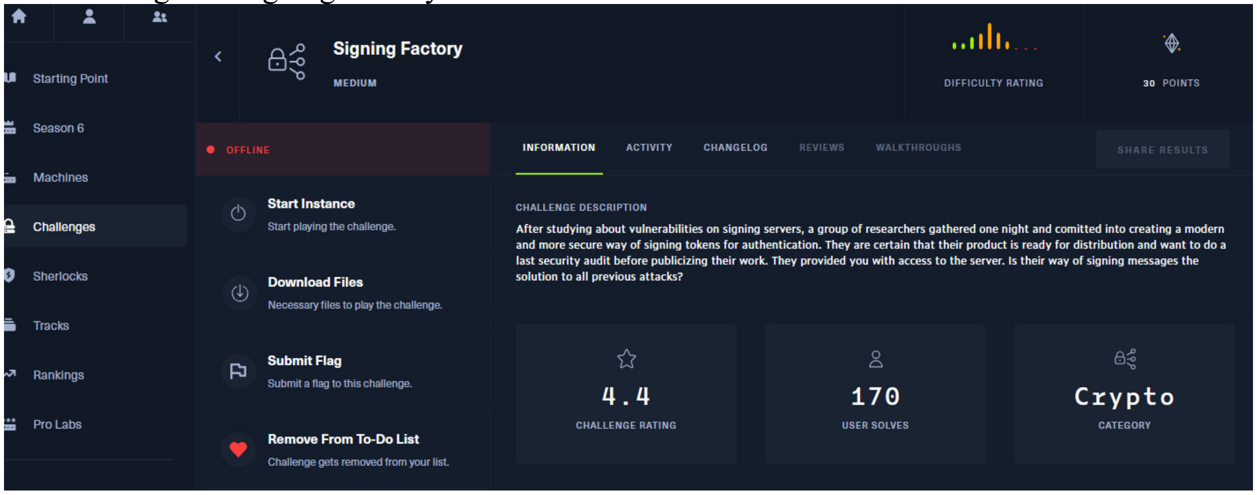


ANNOUNCEMENT

Hack The Box named a global leader in Cybersecurity Skills and Training Platforms.

538 Players

The challenge is “Signing Factory”



CHALLENGE DESCRIPTION

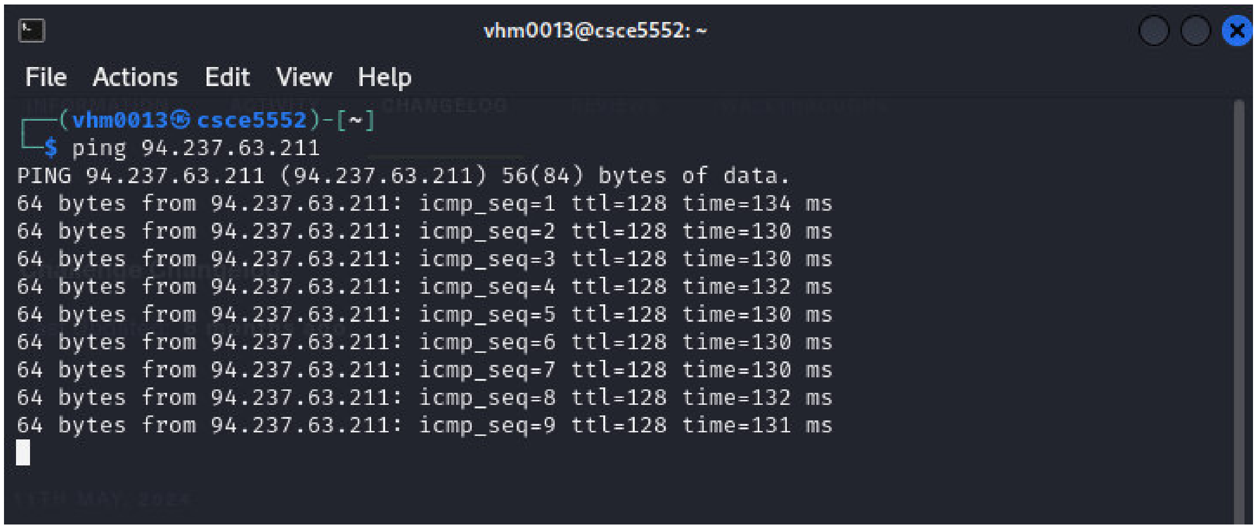
After studying about vulnerabilities on signing servers, a group of researchers gathered one night and comitted into creating a modern and more secure way of signing tokens for authentication. They are certain that their product is ready for distribution and want to do a last security audit before publicizing their work. They provided you with access to the server. Is their way of signing messages the solution to all previous attacks?

Then, you need to establish a connection to HTB servers. I used OpenVPN to connect; there is a file with the server information and some code.

Below, you will see what I have done so far to try to solve the challenge.

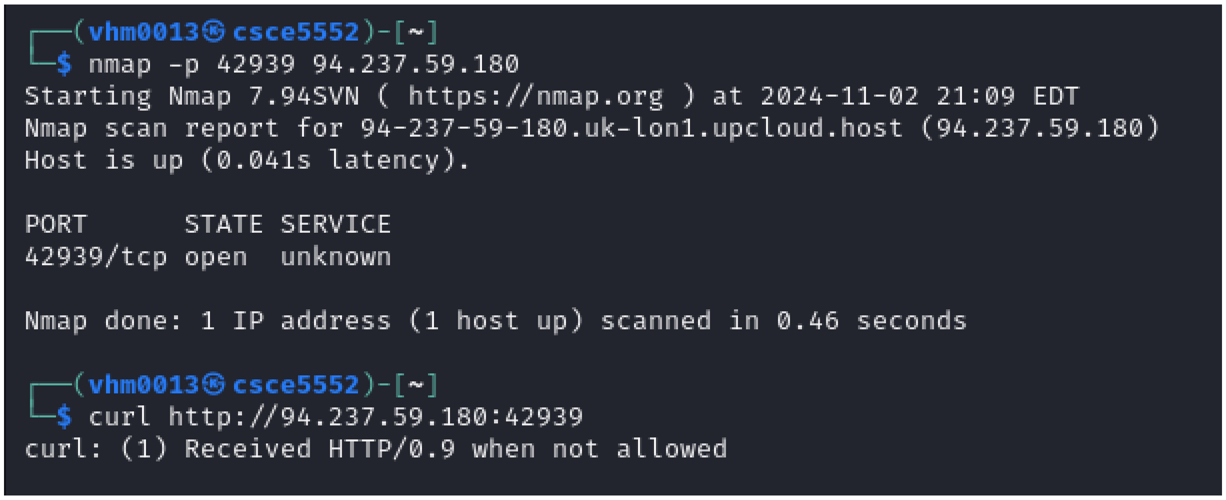
Here is an update of what I attempted to do with the Signing Factory challenge.

I successfully connected through the VPN to HTB using OpenVPN , verifying communication with the server.



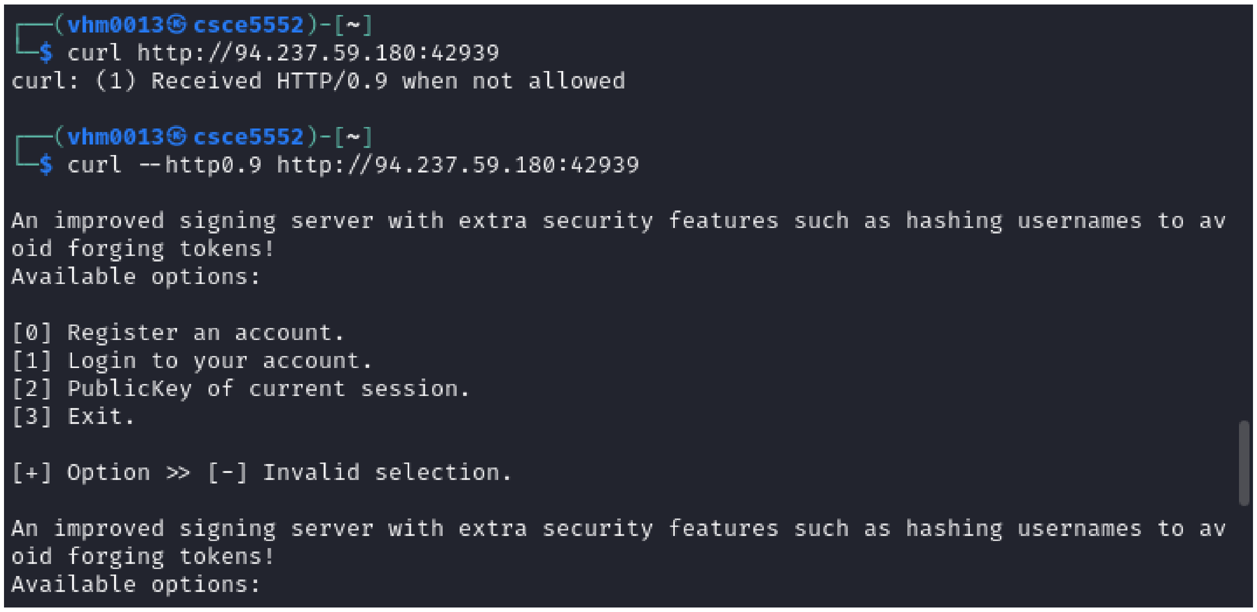
```
┌──(vhm0013㉿csce5552)-[~]
└─$ ping 94.237.63.211
PING 94.237.63.211 (94.237.63.211) 56(84) bytes of data.
64 bytes from 94.237.63.211: icmp_seq=1 ttl=128 time=134 ms
64 bytes from 94.237.63.211: icmp_seq=2 ttl=128 time=130 ms
64 bytes from 94.237.63.211: icmp_seq=3 ttl=128 time=130 ms
64 bytes from 94.237.63.211: icmp_seq=4 ttl=128 time=132 ms
64 bytes from 94.237.63.211: icmp_seq=5 ttl=128 time=130 ms
64 bytes from 94.237.63.211: icmp_seq=6 ttl=128 time=130 ms
64 bytes from 94.237.63.211: icmp_seq=7 ttl=128 time=130 ms
64 bytes from 94.237.63.211: icmp_seq=8 ttl=128 time=132 ms
64 bytes from 94.237.63.211: icmp_seq=9 ttl=128 time=131 ms
```

I also ran Nmap and identified port 42939/tcp is open and classified as an unknown server.



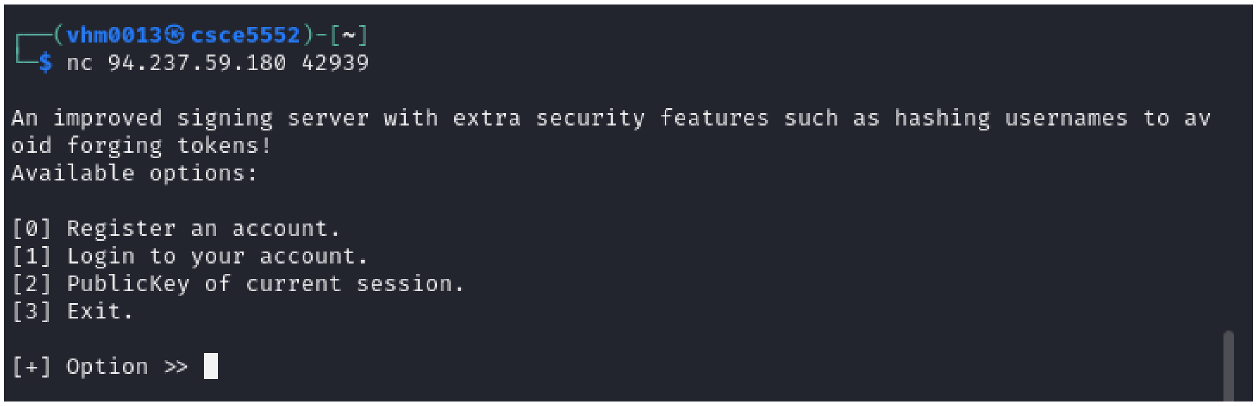
```
┌──(vhm0013㉿csce5552)-[~]
└─$ nmap -p 42939 94.237.59.180
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-02 21:09 EDT
Nmap scan report for 94-237-59-180.uk-lon1.upcloud.host (94.237.59.180)
Host is up (0.041s latency).

PORT      STATE SERVICE
42939/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.46 seconds

┌──(vhm0013㉿csce5552)-[~]
└─$ curl http://94.237.59.180:42939
curl: (1) Received HTTP/0.9 when not allowed
```

Then I made initial attempts to access the service using “curl” but resulted in an HTTP/0.9 error, as you can see in the image below.



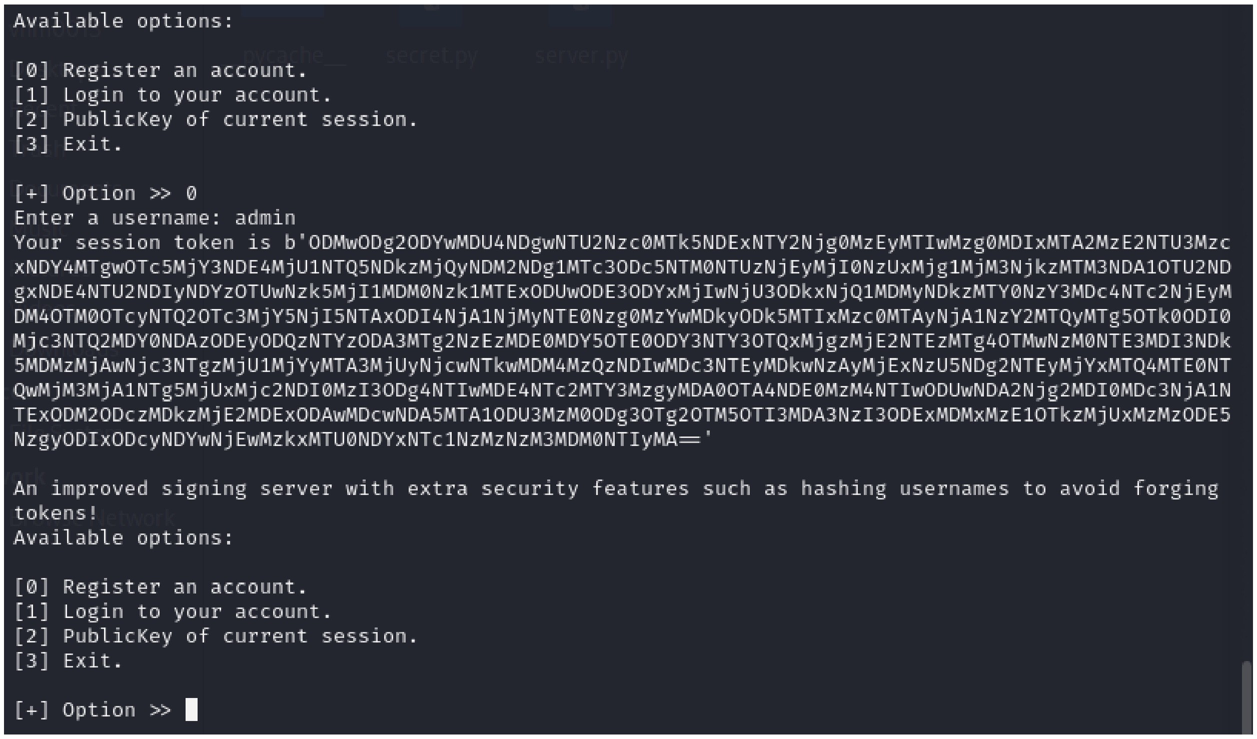
```
┌──(vhm0013㉿csce5552)-[~]
└─$ curl http://94.237.59.180:42939
curl: (1) Received HTTP/0.9 when not allowed

┌──(vhm0013㉿csce5552)-[~]
└─$ curl --http0.9 http://94.237.59.180:42939

An improved signing server with extra security features such as hashing usernames to av
oid forging tokens!
Available options:

[0] Register an account.
[1] Login to your account.
[2] PublicKey of current session.
[3] Exit.

[+] Option >> [-] Invalid selection.

An improved signing server with extra security features such as hashing usernames to av
oid forging tokens!
Available options:
```

I was not getting a clear output or the service is not HTTP, so I used netcat to send requests and observe the server response manually: nc 94.237.59.180 42939



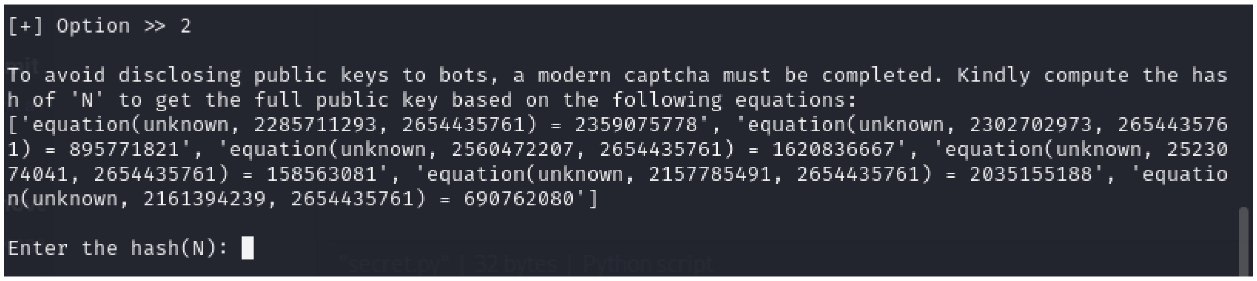
```
┌──(vhm0013㉿csce5552)-[~]
└─$ nc 94.237.59.180 42939

An improved signing server with extra security features such as hashing usernames to av
oid forging tokens!
Available options:

[0] Register an account.
[1] Login to your account.
[2] PublicKey of current session.
[3] Exit.

[+] Option >> █
```

I registered an account to explore the server response and check the session tokens. The server mentions "hashing usernames to avoid forging tokens," hinting at potential token or session manipulation vectors through hash collisions or token modifications.



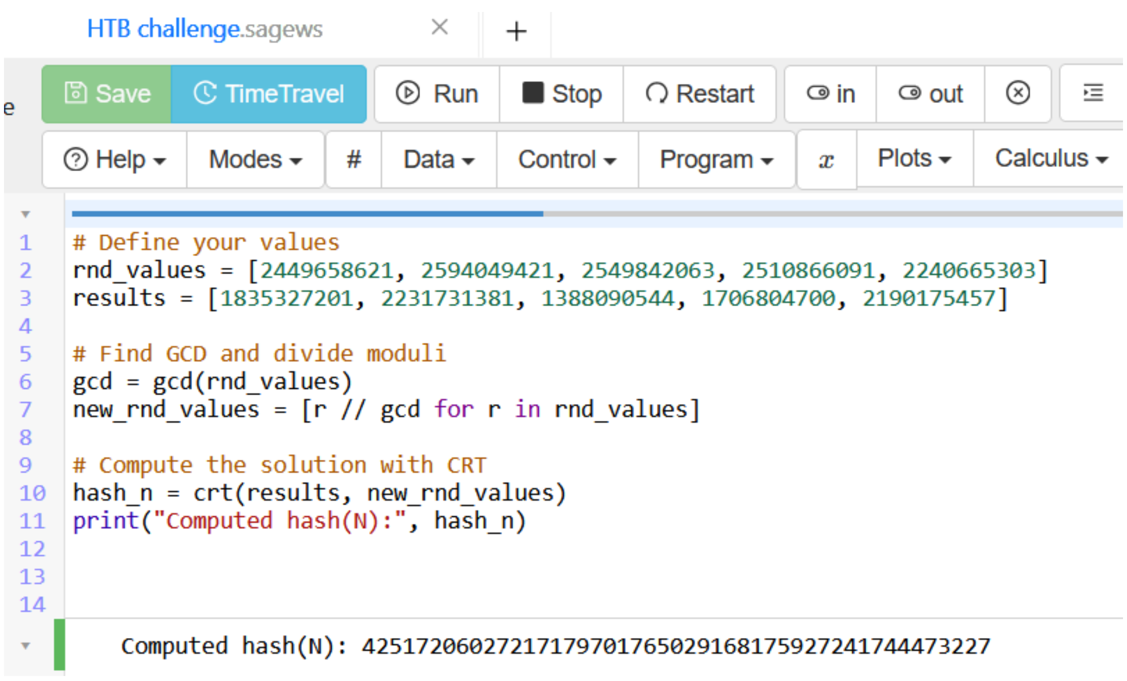
```
Available options:

[0] Register an account.
[1] Login to your account.
[2] PublicKey of current session.
[3] Exit.

[+] Option >> 0
Enter a username: admin
Your session token is b'ODMwODg2ODYwMDU4NDgwNTU2Nzc0MTk5NDExNTY2Njg0MzEyMTIwMzg0MDIxMTA2MzE2NTU3Mzc
xNDY4MTgwOTc5MjY3NDE4MjU1NTQ5NDkzMjQyNDM2NDg1MTc3ODc5NTM0NTUzNjEyMjI0NzUxMjg1MjM3NjkzMTM3NDA1OTU2ND
gxNDE4NTU2NDIyNDYzOTUwNzk5MjI1MDM0Nzk1MTExODUwODE3ODYxMjIwNjU3ODkxNjQ1MDMyNDkzMTY0NzY3MDc4NTc2NjEyM
DM4OTM0OTcyNTQ2OTc3MjY5NjI5NTAxODI4NjA1NjMyNTE0Nzg0MzYwMDkyODk5MTIxMzc0MTAyNjA1NzY2MTQyMTg5OTk0ODI0
Mjc3NTQ2MDY0NDAzODEyODQzNTYzODA3MTg2NzEzMDE0MDY5OTE0ODY3NTY3OTQxMjgzMjE2NTEzMTg4OTMwNzM0NTE3MDI3NDk
5MDMzMjAwNjc3NTgzMjU1MjYyMTA3MjUyNjcwNTkwMDM4MzQzNDIwMDc3NTEyMDkwNzAyMjExNzU5NDg2NTEyMjYxMTQ4MTE0NT
QwMjM3MjA1NTg5MjUxMjc2NDI0MzI3ODg4NTIwMDE4NTc2MTY3MzgyMDA0OTA4NDE0MzM4NTIwODUwNDA2Njg2MDI0MDc3NjA1N
TExODM2ODczMDkzMjE2MDExODAwMDcwNDA5MTA1ODU3MzM0ODg3OTg2OTM5OTI3MDA3NzI3ODExMDMxMzE1OTkzMjUxMzMzODE5
NzgyODIxODcyNDYwNjEwMzkxMTU0NDYxNTc1NzMzNzM3MDM0NTIyMA=='

An improved signing server with extra security features such as hashing usernames to avoid forging
tokens!
Available options:

[0] Register an account.
[1] Login to your account.
[2] PublicKey of current session.
[3] Exit.

[+] Option >> 
```

I continued testing each menu option, focusing on capturing any tokens or session data, especially the public key from option 2.



```
[+] Option >> 2

To avoid disclosing public keys to bots, a modern captcha must be completed. Kindly compute the has
h of 'N' to get the full public key based on the following equations:
['equation(unknown, 2285711293, 2654435761) = 2359075778', 'equation(unknown, 2302702973, 265443576
1) = 895771821', 'equation(unknown, 2560472207, 2654435761) = 1620836667', 'equation(unknown, 25230
74041, 2654435761) = 158563081', 'equation(unknown, 2157785491, 2654435761) = 2035155188', 'equatio
n(unknown, 2161394239, 2654435761) = 690762080']

Enter the hash(N): 
```

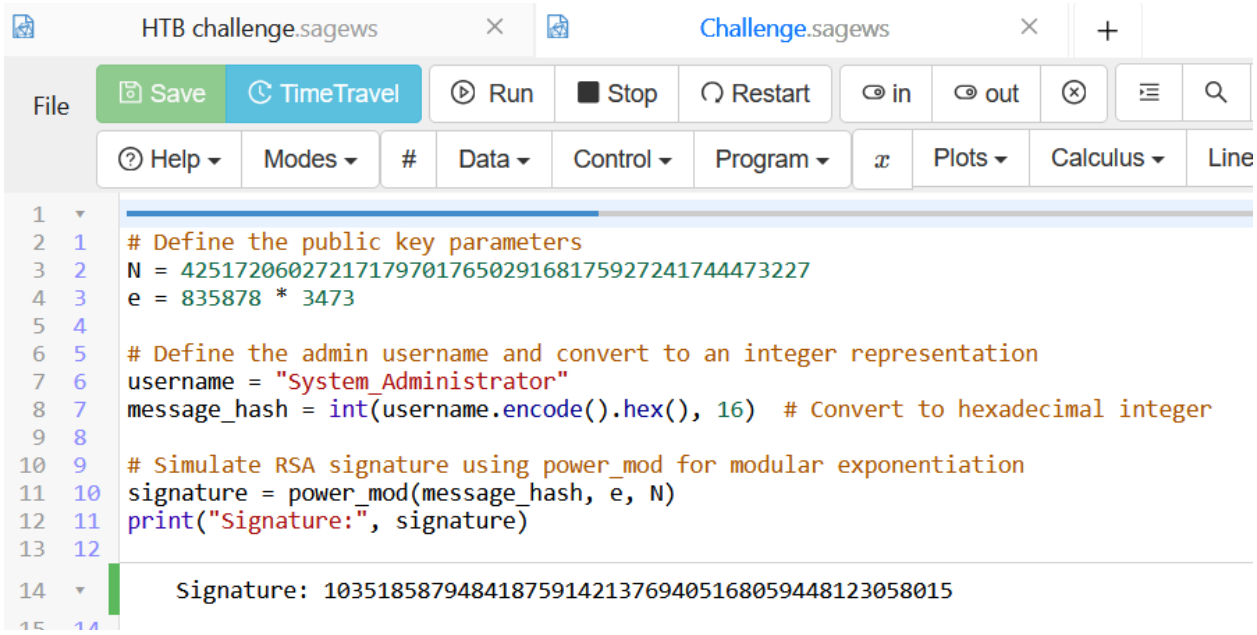
I used CoCalc (SageMatch) https://cocalc.com/ for modular equations and RSA simulation. I was able to calculate the hash(N) using the Chinese Remainder Theorem (CRT) with the equations provided in option 2.



```
# Define your values
rnd_values = [2449658621, 2594049421, 2549842063, 2510866091, 2240665303]
results = [1835327201, 2231731381, 1388090544, 1706804700, 2190175457]

# Find GCD and divide moduli
gcd = gcd(rnd_values)
new_rnd_values = [r // gcd for r in rnd_values]

# Compute the solution with CRT
hash_n = crt(results, new_rnd_values)
print("Computed hash(N):", hash_n)
```

```
Computed hash(N): 42517206027217179701765029168175927241744473227
```

After getting the public key, I simulated RSA signature in SageMath, defining “System_Administrator” as a target username, converted it to an integer using hexadecimal encoding, and used power_mod in SageMath to simulate the RSA signature process



```
# Define the public key parameters
N = 42517206027217179701765029168175927241744473227
e = 835878 * 3473

# Define the admin username and convert to an integer representation
username = "System_Administrator"
message_hash = int(username.encode().hex(), 16)  # Convert to hexadecimal integer

# Simulate RSA signature using power_mod for modular exponentiation
signature = power_mod(message_hash, e, N)
print("Signature:", signature)
```

```
Signature: 10351858794841875914213769405168059448123058015
```

I will continue exploring and testing each menu option to capture any token or session data. I will research whether tokens can be forged by analyzing the hashing or signature mechanism and whether we can create a token that goes through the admin check.