

CN ASSIGNMENT 1

SUBMITTED TO: SIR USMAN

Arham Khan

22K-4080 BAI-6B

Proxy Server Setup (ZeroOmega Extension):

The screenshot shows the ZeroOmega extension interface. On the left, there's a sidebar with 'SETTINGS' (Interface, General, Import/Export, Theme), 'PROFILES' (Builtin, my proxy selected, proxy, auto switch, New profile...), and 'ACTIONS' (Apply changes, Discard changes). The main area is titled 'Proxy servers' and shows a table with one row: Scheme (default), Protocol (HTTP), Server (localhost), and Port (8888). Below it is a 'Bypass List' section with a text input containing '127.0.0.1 ::1 localhost'. At the top right are 'Export PAC', 'Rename', and 'Delete' buttons.

```
Loading personal and system profiles took 1764ms.
(base) (.venv) PS C:\Users\worka\Downloads> python proxy.py 8888
Simple Proxy started on port 8888...
New connection from ('127.0.0.1', 57251)
New connection from ('127.0.0.1', 57311)
Blocked non-GET request: CONNECT www.facebook.com:443
```

Port 8080 was busy.

HTTP GET Request working perfectly:

The screenshot shows a browser window for 'rest.vulnweb.com' and a NetworkMiner tool. The browser page displays the 'Invicti Vulnerable REST API' homepage. The NetworkMiner tool shows a captured packet for a GET request to 'http://rest.vulnweb.com/'. The details pane shows the following information:

- Request URL: http://rest.vulnweb.com/
- Request Method: GET
- Status Code: 200 OK
- Remote Address: 127.0.0.1:8888
- Referrer Policy: strict-origin-when-cross-origin
- Response Headers:
 - Content-Length: 3556
 - Content-Type: text/html; charset=UTF-8
 - Date: Sat, 08 Mar 2025 20:52:05 GMT
 - Server: Apache/2.4.25 (Debian)
 - Vary: Accept-Encoding

HTTP Request other than GET Method (Not working):

The method is not allowed for the requested URL.

Method Not Allowed

Request URL: http://httpbin.org/post
Request Method: GET
Status Code: 405 METHOD NOT ALLOWED
Remote Address: 127.0.0.1:8888
Referrer Policy: strict-origin-when-cross-origin

Response Headers

- Access-Control-Allow-Origin: *
- Allow: OPTIONS, POST
- Content-Length: 178
- Content-Type: application/json; charset=utf-8
- Date: Mon, 12 Jun 2023 10:30:20 GMT
- Server: gunicorn/20.1.0
- Via: 1.1 varnish
- X-Content-Type-Options: nosniff
- X-Powered-By: Python/3.11.0
- X-Varnish: 600000000000000000
- X-XSS-Protection: 1; mode=block

Blocked non-GET request: CONNECT ssl.gstatic.com:443

Testing using Curl:

```
(base) (.venv) PS C:\Users\warka\Downloads> curl.exe -X POST -x http://localhost:8888 http://example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
```

```
(base) (.venv) PS C:\Users\warka\Downloads> curl.exe -X POST -x http://localhost:8888 http://ptsv3.com/t/your-test-id/post
(base) (.venv) PS C:\Users\warka\Downloads> curl.exe -X POST -x http://localhost:8888 http://jsonplaceholder.typicode.com/posts
(base) (.venv) PS C:\Users\warka\Downloads>
```

Code:

```
import sys
import socket as sk
import threading

def http_client(client_socket):
    request = client_socket.recv(1024) #receiving data in small chunks
    if not request:
        client_socket.close()
    return

# Split request into lines
```

```
headers = request.decode().split('\r\n')
if not headers:
    client_socket.send(b'HTTP/1.0 400 Bad Request\r\n\r\n')
    client_socket.close()
    return

first_line = headers[0].split()
if len(first_line) < 3:
    client_socket.send(b'HTTP/1.0 400 Bad Request\r\n\r\n')
    client_socket.close()
    return

method, full_url, _ = first_line
if method != 'GET':
    print(f"Blocked non-GET request: {method} {full_url}")
    client_socket.send(b'HTTP/1.0 501 Not Implemented\r\n\r\n')
    client_socket.close()
    return

# splitting url
if not full_url.startswith('http://'):
    client_socket.send(b'HTTP/1.0 400 Bad Request\r\n\r\n')
    client_socket.close()
    return

clean_url = full_url[7:]
server_part = clean_url.split('/', 1)
host_port = server_part[0].split(':', 1)
target_host = host_port[0]
target_port = int(host_port[1]) if len(host_port) > 1 else 80
path = '/' + server_part[1] if len(server_part) > 1 else '/'

# Connecting to target server
server_socket = sk.socket()
if server_socket.connect_ex((target_host, target_port)) != 0:
    client_socket.send(b'HTTP/1.0 502 Bad Gateway\r\n\r\n')
    client_socket.close()
    return

# Forward request
server_socket.send(f"GET {path} HTTP/1.0\r\nHost:
```

```
{target_host}\r\n\r\n".encode()

# Relay response
response = server_socket.recv(4096)
while response:
    client_socket.send(response)
    response = server_socket.recv(4096)

# Cleanup
server_socket.close()
client_socket.close()

def myproxy_server(port):
    server = sk.socket(sk.AF_INET, sk.SOCK_STREAM)
    server.setsockopt(sk.SOL_SOCKET, sk.SO_REUSEADDR, 1)
    server.bind(("", port))
    server.listen(10)
    print(f"Simple Proxy started on port {port}...")

    while True:
        client_socket, addr = server.accept()
        threading.Thread(target=http_client, args=(client_socket,)).start()

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python proxy.py <PORT>")
        sys.exit()
    myproxy_server(int(sys.argv[1]))
```