TEAM II – PROXY SERVER
EECE 351 - Project

By Nadine Mcheik & Najla Sadek

CONTENT

Required Features

A-Proxy

b- Caching

c- Security

Additional Features

A-PROXY

Listening on a port for incoming connections

```
C:\Users\User\Desktop\AUB\4-Fall 23-24\EECE 351\final proje
[*] Proxy Server listening on 127.0.0.1:8000
```

a message describing this request with the IP and exact time of the request

print a message that the response was received with the exact time

print a message with exact time of the actual request

print a message that the response was sent with the exact time

```
ustrative examples in documents. You may use this\n    domain in literature without prior coordination or asking for per
mission.</p>\n    <p><a href="https://www.iana.org/domains/example">More information...</a></p>\n</div>\n</body>\n</html
>\n'
[*] Accepted connection from 127.0.0.1:55734
[1] Received request from ('127.0.0.1', 55734) at 2023-11-27 16:49:15.016897
b'GET / HTTP/1.1\r\nHost: www.example.com\r\n\r\n:68b87b78-76b8-40e3-9dd0-e744c89d90d8'
b'GET / HTTP/1.1\r\nHost: www.example.com\r\n\r\n:68b87b78-76b8-40e3-9dd0-e744c89d90d8' HEYEYEYEYEY
[2] Proxy sending request to www.example.com:80 at 2023-11-27 16:49:15.020248
[*] Destination Host: www.example.com, Destination Port: 80
Uno responsia b'HTTP/1.1 304 Not Modified\r\nAge: 570418\r\nCache-Control: max-age=604800\r\nDate: Mon, 27 Nov 2023 14:4
9:15 GMT\r\nEtag: "3147526947+ident"\r\nExpires: Mon, 04 Dec 2023 14:49:15 GMT\r\nLast-Modified: Thu, 17 Oct 2019 07:18:
26 GMT\r\nServer: ECS (nyb/1D2B)\r\nVary: Accept-Encoding\r\nX-Cache: HIT\r\n\r\n'
[*] Serving from Cache for www.example.com:80
Send cachedd
b'HTTP/1.1 200 OK\r\nAccept-Ranges: bytes\r\nAge: 130774\r\nCache-Control: max-age=604800\r\nContent-Type: text/html; ch
arset=UTF-8\r\nDate: Wed, 22 Nov 2023 12:41:51 GMT\r\nEtag: "3147526947"\r\nExpires: Wed, 29 Nov 2023 12:41:51 GMT\r\nLa
st-Modified: Thu, 17 Oct 2019 07:18:26 GMT\r\nServer: ECS (dce/26D9)\r\nVary: Accept-Encoding\r\nX-Cache: HIT\r\nContent
-Length: 1256\r\n\r\n<!doctype html>\n<html>\n<head>\n    <title>Example Domain</title>\n\n    <meta charset="utf-8" />\
n    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />\n    <meta name="viewport" content="width=dev
ice-width, initial-scale=1" />\n    <style type="text/css">\n    body {\n        background-color: #f0f0f2;\n        mar
gin: 0;\n        padding: 0;\n        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans"
, "Helvetica Neue", Helvetica, Arial, sans-serif;\n        \n    }\n    div {\n        width: 600px;\n        margin: 5e
m auto;\n        padding: 2em;\n        background-color: #fdfdff;\n        border-radius: 0.5em;\n        box-shadow: 2
px 3px 7px 2px rgba(0,0,0,0.02);\n    }\n    a:link, a:visited {\n        color: #38488f;\n        text-decoration: none
;\n    }\n    @media (max-width: 700px) {\n        div {\n            margin: 0 auto;\n            width: auto;\n
 }\n    }\n    </style>    \n</head>\n\n<body>\n<div>\n    <h1>Example Domain</h1>\n    <p>This domain is for use in ill
ustrative examples in documents. You may use this\n    domain in literature without prior coordination or asking for per
mission.</p>\n    <p><a href="https://www.iana.org/domains/example">More information...</a></p>\n</div>\n</body>\n</html
>\n'
```

B-CACHING

If Modified Since

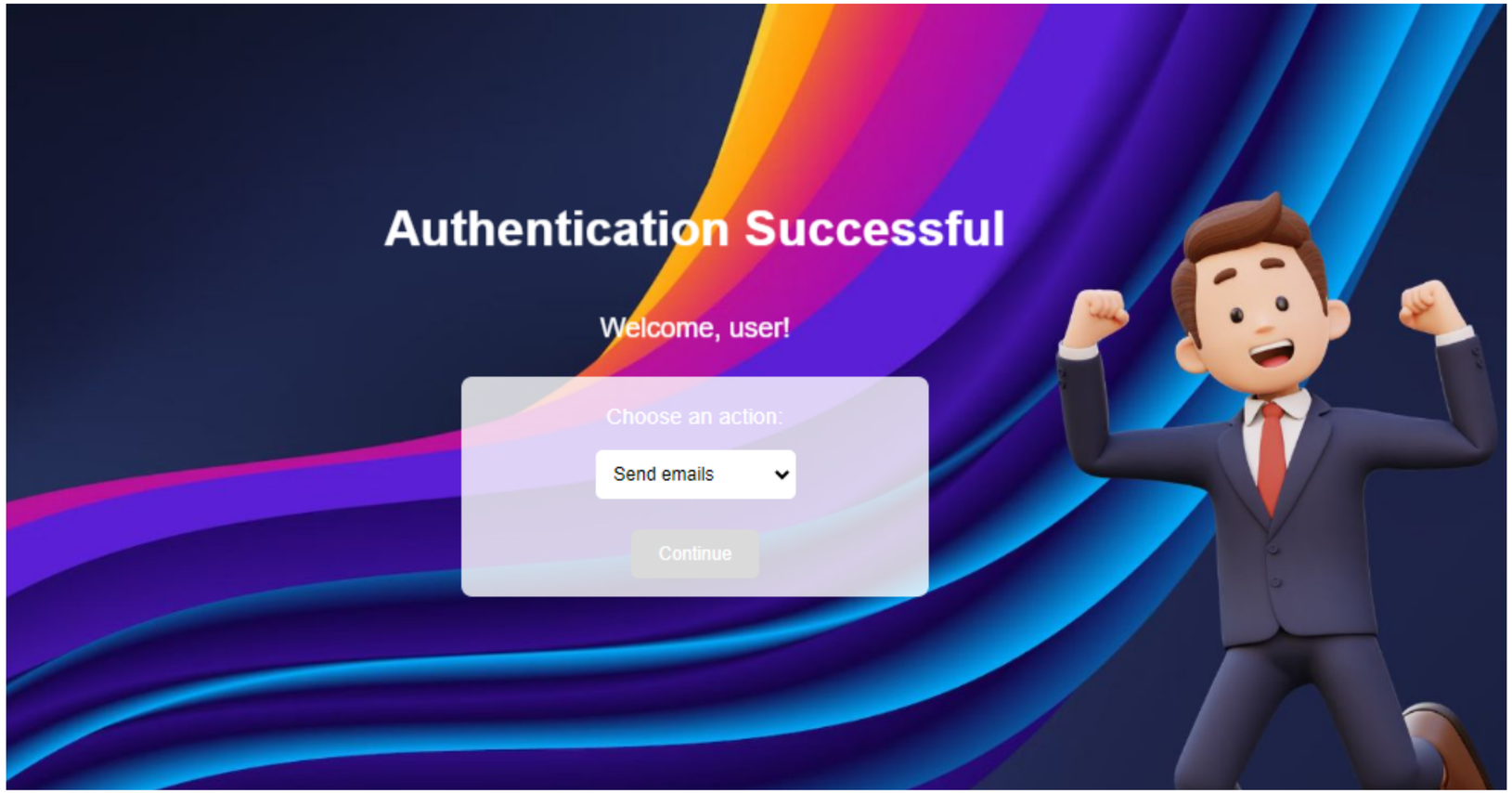
```
else:
    last_modified_timestamp = self.cache[cache_key].get('last_modified', None)
    conditional_request = f"GET / HTTP/1.1\r\nHost: {destination_host}\r\n"
if last_modified_timestamp:
    conditional_request += f"If-Modified-Since: {formatdate(timeval=last_modified_timestamp.timestamp(), localtime=False, usegmt=True)

conditional_request += "\r\n"
# Forward a conditional request to the destination server

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.connect((destination_host, destination_port))
server_socket.send(conditional_request.encode())
```

C-SECURITY

User Authentication



Client Filtering

```
def check_blacklist(self, client_address):
    # Check if the client's IP address is in the blacklist
    return client_address[0] in self.blacklist
```

Webpage Filtering

```
def check_destination_blacklist(self, destination_address):
    # Check if the destination IP address is in the blacklist
    return destination_address[0] in self.destination_blacklist
```

Client/Web pair filtering

```
def check_source_destination_blacklist(self, source_address, destination_address):
    # Check if the source-destination pair is in the blacklist
    return {"source": source_address[0], "destination": destination_address[0]} in self.source_destination_blacklist
```

ADDITIONAL FEATURES

AUTOMATED EMAIL SYSTEM

```
        """
            # add the components to the email
            html_part = MIMEText(content, 'html')
            email.attach(html_part)

            smtp = smtplib.SMTP("smtp-mail.outlook.com", port=587)
            smtp.starttls()
            smtp.login(sender, "proxypro2023") #add the password of the sender's account
            # r[3] is the email address of the recipient. we add to it the ccd addresses.
            print(r)
            recipient = [r[3]]
            r3 = recipient+cc
            email = email.as_string()
            smtp.sendmail(sender,r3, email) # the line that sends the email
            smtp.quit()

    except Exception as e:
        print(e)
        raise Exception(e)
```

WEB INTERFACE

DOWNLOADING FILES

```
approved_extensions = ['xlsx', 'ppt', 'pdf']
if any(ext in request_data.decode() for ext in approved_extensions):
    url = self.extract_url_from_request(request_data)
    response = requests.get(url)
    response.raise_for_status()

    file_path = "C:/Users/User/Documents/project.xlsx"


    # Write only the content to the file
    with open(file_path, 'wb') as file:
        file.write(response.content)
    #with open(file_path, 'wb') as file:
     #  file.write(content)
    print("File downloaded and saved at:", file_path)
    client_socket.send("HTTP/1.1 200 OK\n\nFile downloaded successfully".encode())
    return
```

FIREWALL ATTACK DETECTION

```
Rules for attacks
elf.waf_rules = [
    r"(?i)<script.*?>.*?<\/script.*?>",  # Example rule for detecting script tags
    r"(?i)<.*?on[a-z]+\s*=\s*\"[^\"]+\".*?>",  # Example rule for detecting event attributes
    # Add more rules as needed
```

MULTITHREADING

CACHE EXPIRY DATE

```python
if b"304 Not Modified" not in response_data:
    print(f"[*] Updating Cache for {cache_key}")

    # Update cache with the new response and timestamp + expiration time if present
    last_modified_header = re.search(rb'Last-Modified: ([^\r\n]+)', response_data)
    expiration_header = re.search(rb'Expires: ([^\r\n]+)', response_data)
    last_modified_timestamp = parsedate_to_datetime(last_modified_header.group(1).decode('utf-8')) if last_modified_header else
    expiration_time = parsedate_to_datetime(expiration_header.group(1).decode('utf-8')) if expiration_header else None

    self.cache[cache_key] = {'response': response_data, 'last_modified': last_modified_timestamp, 'expiration_time': expiration
    self.save_cache()
```

SQL INJECTION PATTERN DETECTION

```
elf.sql_injection_patterns= [
'\b(?:select|union|insert|update|delete|from|where)\b',
'\b(?:exec|sp_executesql|xp_cmdshell)\b',
'\b(?:alter|create|drop)\b\s+(?:table|database|procedure)',
'\b(?:--|#|\/\*)[^\n]*\b'  # Detect comments for comment-based injection
```

CACHE STORAGE OPTIMIZATION

```python
def remove_expired_cache_entries(self):
    current_time = datetime.now()

    # Create a list of cache keys to remove
    keys_to_remove = [key for key, entry in self.cache.items() if 'expiration_time' in entry and entry[

    # Remove the expired entries from the cache
    for key in keys_to_remove:
        print(f"[*] Removing expired cache entry for {key}")
        del self.cache[key]

    # Save the updated cache
    self.save_cache()
```

THANK YOU!