OUR EXTRA FEATURES

1-Animated Web interface using flask, CSS, HTML

2-Downloading Files

3-Firewall Attack Detection

4-Multithreading

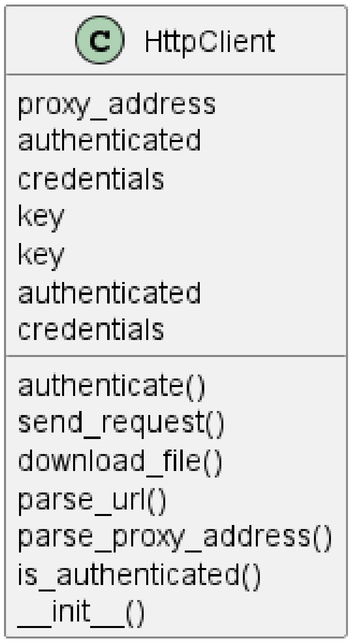
5-Cache Expiry Date

6-SQL injection Pattern Detection

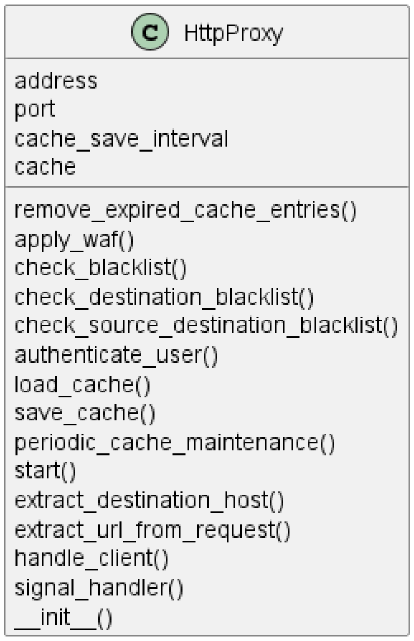
7-Cache Storage Optimization

8-Automated Emails

UML Diagrams



```
HttpClient
---
proxy_address
authenticated
credentials
key
key
authenticated
credentials
---
authenticate()
send_request()
download_file()
parse_url()
parse_proxy_address()
is_authenticated()
__init__()
```



```
HttpProxy
---
address
port
cache_save_interval
cache
---
remove_expired_cache_entries()
apply_waf()
check_blacklist()
check_destination_blacklist()
check_source_destination_blacklist()
authenticate_user()
load_cache()
save_cache()
periodic_cache_maintenance()
start()
extract_destination_host()
extract_url_from_request()
handle_client()
signal_handler()
__init__()
```

Client Code (HttpClient Class):

Initialization:

- proxy_address: Stores proxy server address.

- authenticated: Tracks authentication status.

- credentials, key: Initially set to None.

authenticate(credentials):

- Opens socket connection to the proxy.

- Sends user credentials for authentication.

- Updates key and authentication status upon success.

send_request(url):

- Establishes socket connection to the proxy.

- Validates authentication status.

- Parses URL into hostname and path.

- Constructs and sends an HTTP GET request.

- Receives and returns the response.

download_file(file_url):

- Opens socket connection to proxy.

- Validates authentication.

- Creates an HTTP GET request for the file.

- Attempts to retrieve the file content.

Helper Methods (parse_url, parse_proxy_address, is_authenticated):

- parse_url: Extracts hostname and path from URL.

- parse_proxy_address: Splits proxy address for IP and port.

- is_authenticated(): Checks authentication status.

Proxy Server Code (HttpProxy Class):

Initialization (__init__):

- Defines proxy address and port.

- Sets up authentication credentials.

- Configures cache settings and blacklists.

- Specifies WAF and injection rules.

Cache Management Functions (remove_expired_cache_entries, save_cache, load_cache, periodic_cache_maintenance):

- remove_expired_cache_entries(): Deletes expired cache entries.

- save_cache(), load_cache(): Serialize/deserialize cache data.

- periodic_cache_maintenance(): Periodically saves cache and removes expired entries.

Security Measures (apply_waf, check_blacklist, check_destination_blacklist, check_source_destination_blacklist, authenticate_user):

- apply_waf(data): Applies rules to detect attacks.

- Blacklisting functions for IPs and IP pairs.

- authenticate_user(client_socket): Handles user authentication.

Handling Client Requests (handle_client):

- Processes client connections and requests.

- Authenticates users, detects attacks using WAF.

- Manages file downloads, conditional requests, and cache updates.

- Serves cached content when available.

Signal Handling (signal_handler):

- Handles termination signals, saves cache before exit.

Web Interface:

- Flask Setup and Configuration:

- Imports Flask, render_template, request, redirect.
- Imports HttpClient and emailer modules.
- Initializes Flask app.
- Creates an instance of HttpClient.

- Route Definitions:

- `index()`:

- Redirects to the dashboard if authenticated; otherwise, renders the welcome page.

- `dashboard()`:

- Redirects to the homepage if not authenticated; otherwise, renders the authenticated dashboard page.

- `login()`:

- Redirects to the dashboard if authenticated.
- Handles GET and POST requests for user login.
- Validates user credentials with HttpClient.authenticate().
- Renders login_failed.html upon failed authentication.

- `choose_action()`:

- Redirects to the homepage if not authenticated.
- Processes user choices (1, 2, 3) from the dashboard and renders corresponding action pages (access_website.html, download_file.html, email_file.html).
- Renders invalid_choice.html for any unhandled choices.

- `perform_action()`:

- Redirects to the homepage if not authenticated.
- Processes actions based on user choices (1, 2, 3) from action forms.
- Calls HttpClient methods (send_request, download_file) based on user inputs.
- Renders result pages indicating the success or failure of actions.

- Error Handling:

- Handles exceptions from emailer.send() in the '3' choice action.

- Flask App Execution: - Starts the Flask app in debug mode.