[Date]

Sparsh Mishra (TEAM 4 / Project 2)

104202994

**FINAL REFLECTION**

2024-HD06-COS80029 -TECHNOLOGY APPLICATION PROJECT

# ACKNOWLEDGEMENT

We respectfully acknowledge the [Name of Indigenous **Group] people** as the **Traditional Owners and Custodians** of this land. We pay our deepest respects to their **Elders, past, present, and emerging**. We understand and honour their enduring spiritual connection to this country and recognize their unwavering sovereignty.

**I also wish it be noted that the facilitator Dr. Ayesha Ashfaq**, has been an integral part of the journey spanning the development of the product throughout the semester. She has guided myself and the team along and helped us achieve quality in results and resolution in efforts. Her firm approach is compelling and it has been a pleasure pursuing development under her.

And I also wish **to acknowledge the support my team mates have bestowed me with**, and the consistent efforts they have put in, making it convenient for me to keep working away at my own pacing, without bother and with concentration. I wish to work in environments where teams can achieve this trust and efficiency whenever the context of that environment is team-based.

# INTRODUCTION

The following is a structured communication of my efforts, contribution, takeaways and learning along the course of development and research in Technology Application Project in the June Semester, 2024 facilitated by Dr. Ayesha Ashfaq at Swinburne University of Technology, Sydney (Parramatta). The report aims to summarize the above-mentioned components of my journey in the semester along the duration of the project for academic assessment and personal reference.

The report **begins with my reflection on the project**, with its description, requirements, goals and boundaries set by myself and the team with co-ordination with the client and the facilitator. This is followed by the targets that were specifically set for myself and then a self-assessment against those targets, post project completion.

Then I have **followed this with a list of contributions** that summarise significant contributions I have made to the project and evidence to support these claims. This is to distinguish between contributions from those that my mates have made to the project, while also eradicate ambiguity between assumed overlap in between my own contributions. This distinction will help with convenient assessment of my efforts towards the research and development of the project.

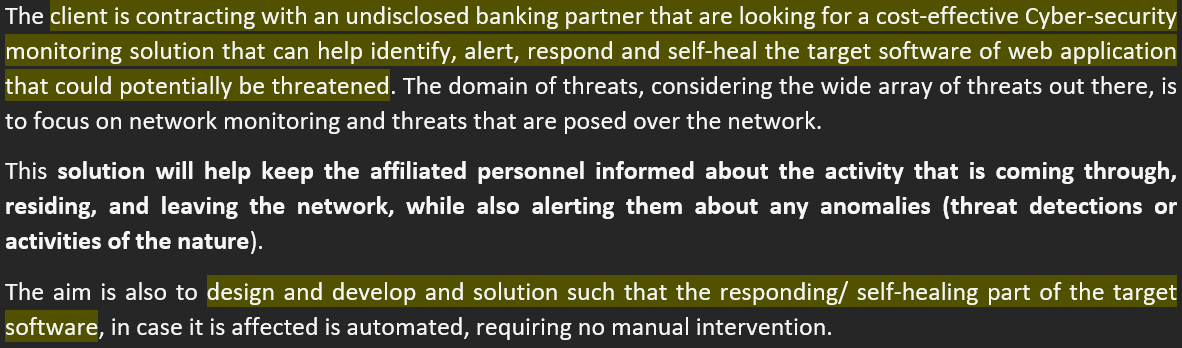
**Then** comes the succeeding section of the report where I contrast my **journey against the pre-defined Unit Learning Outcomes (ULOs)** and **reflective topics** to help provide feedback from my perpective against what the unit expects and requires students recount to give the assessor and other concerned parties a detailed self-assessment. I have provided evidence where possible to substantiate the topics, learning s and reflections I have put forward as my accomplishments. Tere are also some sections that help me put out what I could have done better or different if pursuing the project again (and there isn’t much really, but still).

**And then a conclusion** on this intense journey that I have undertaken with notes and observations that help provide an overall summary spanning everything I have done (and did not do), ending wiyh my farewell and a message to the assessor and any other parties who will review the report.

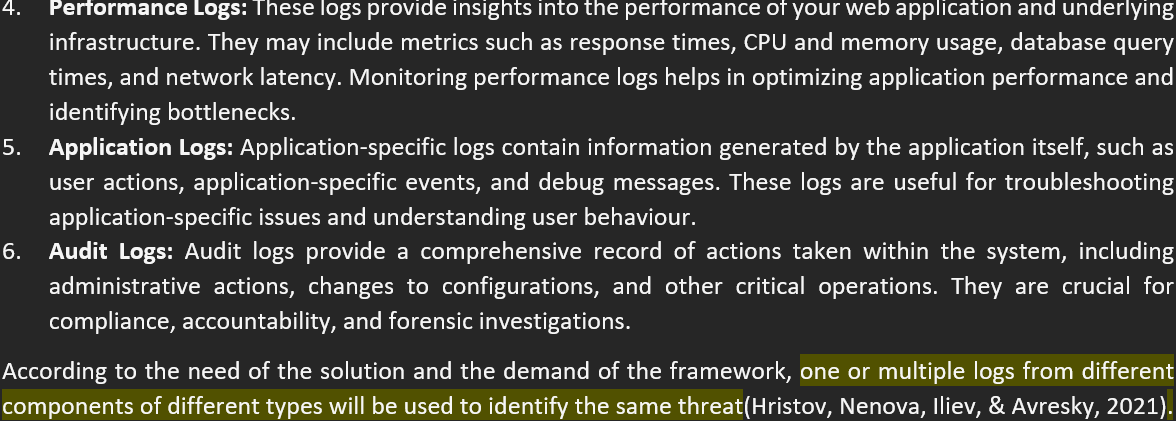
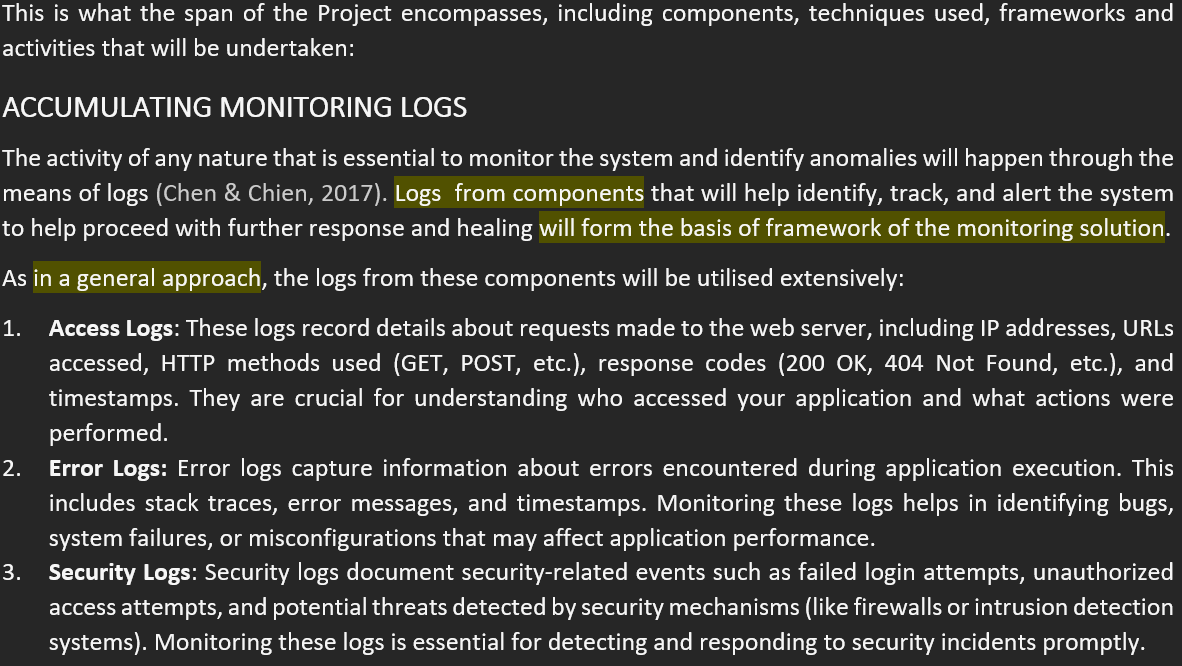
# PROJECT: GOALS, BACKGROUND AND SCOPE

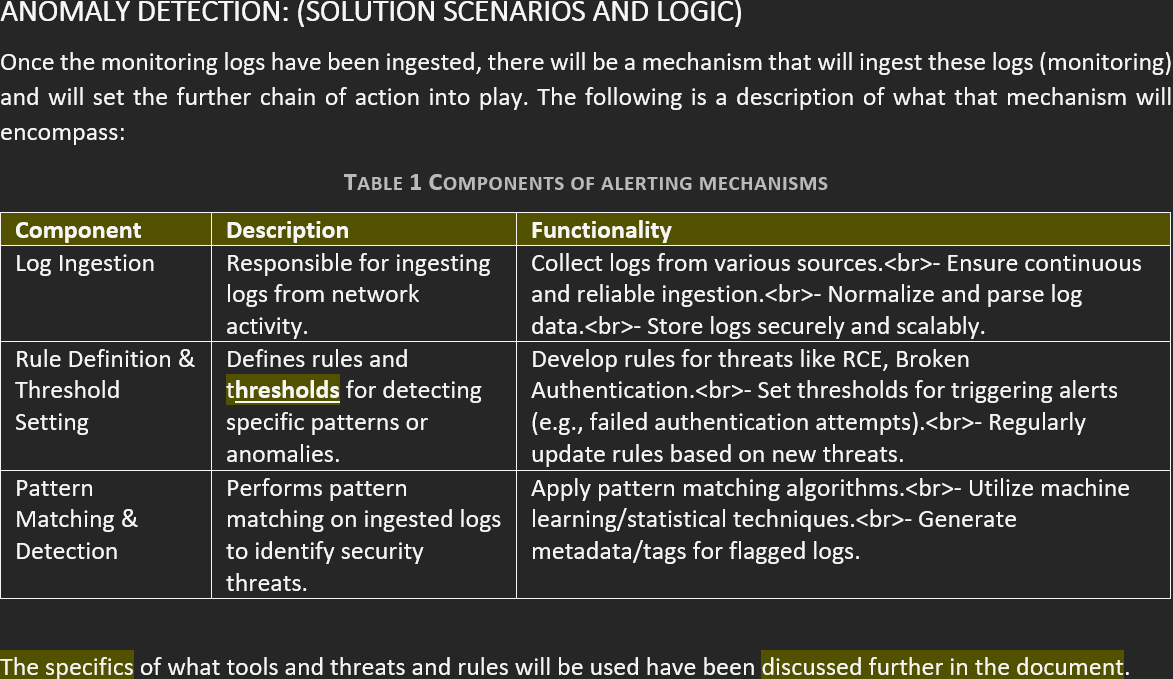
This is what the client came to the team with and a recounting of what was expected including the boundaries, scope and everything else.

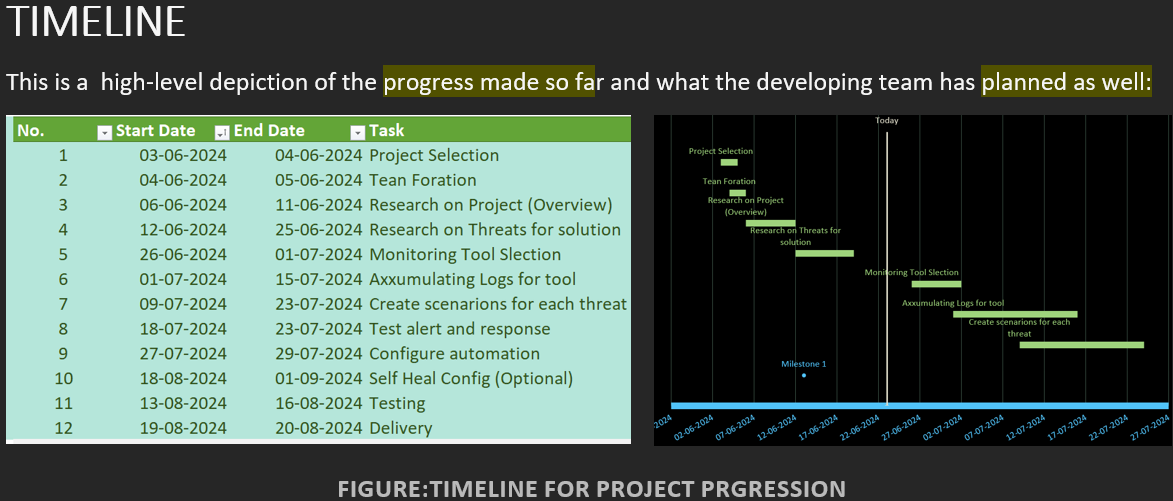
## BACKGROUND

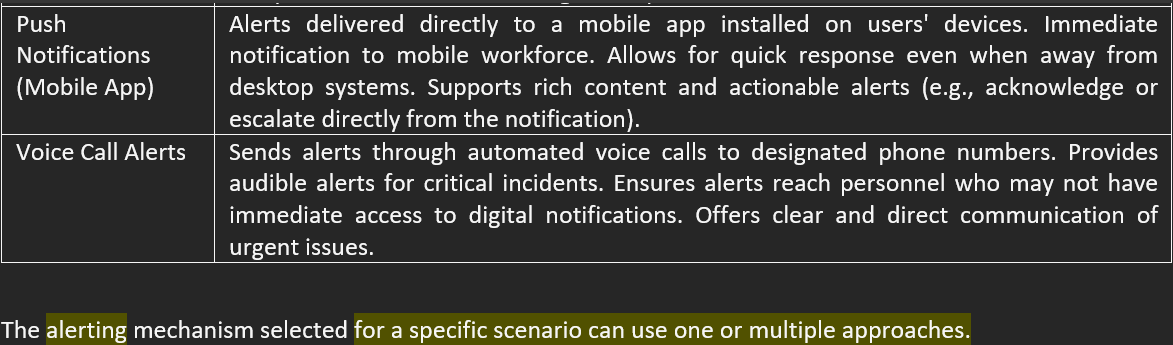
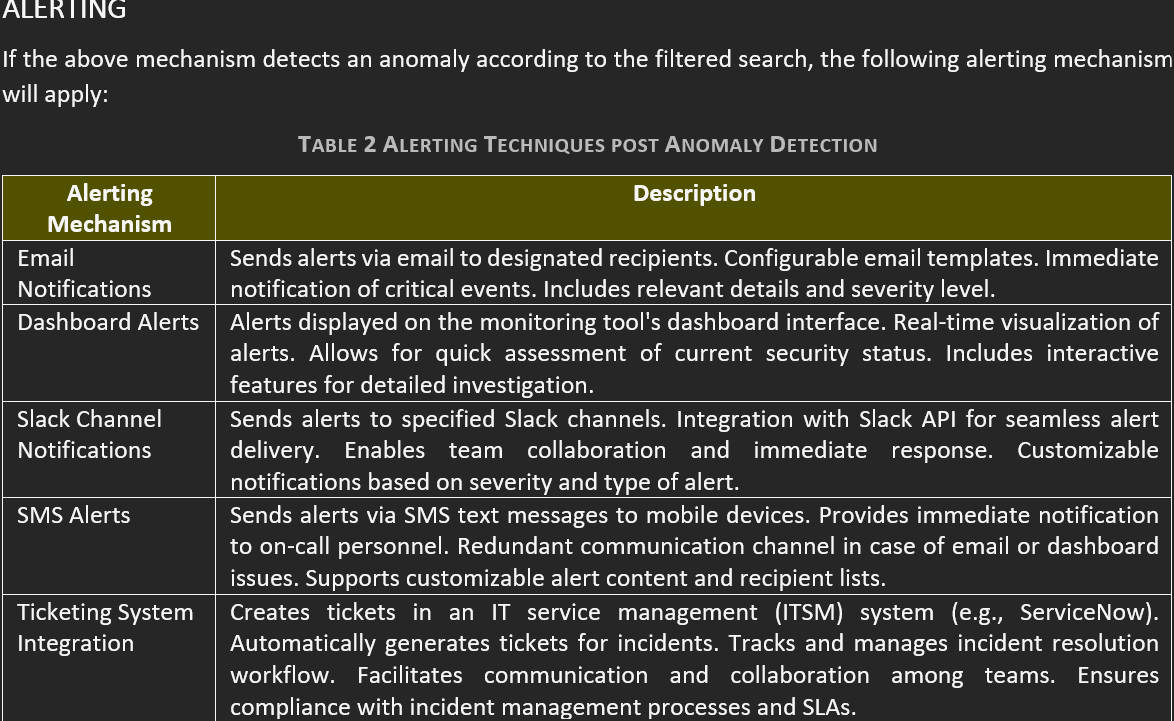


## SCOPE

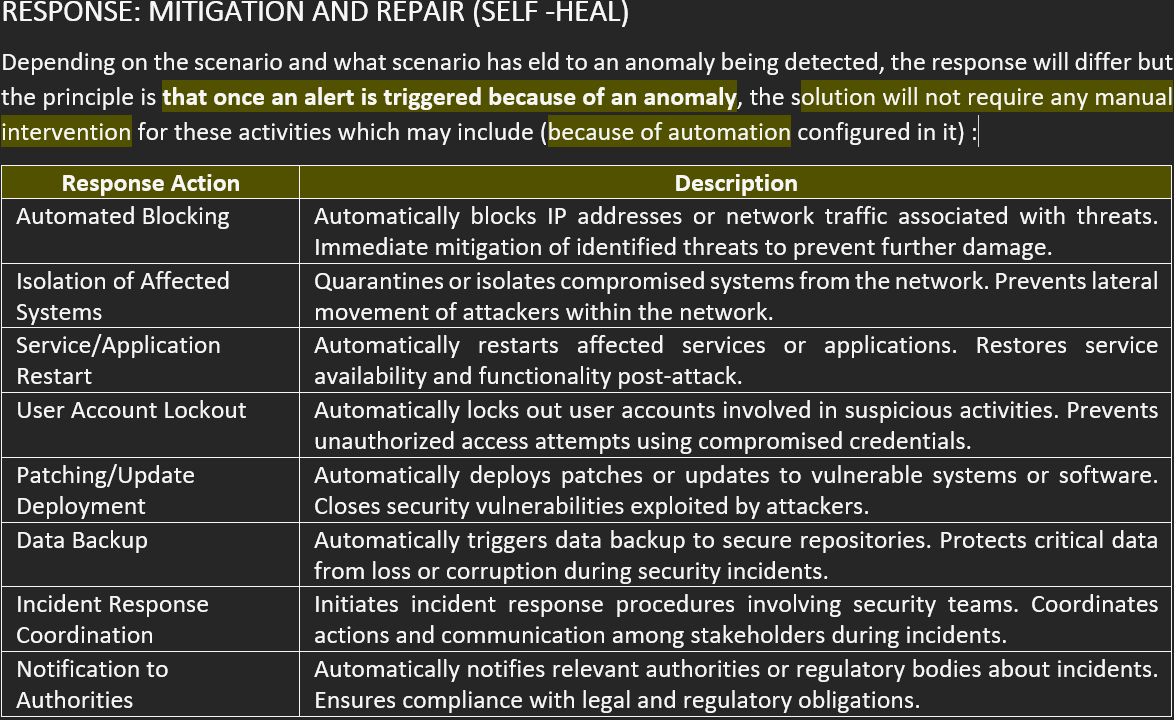












## EXCLUSIONS

The following items have been identified as out of scope by the developers due to either lack of mention or ambiguity in subsequent discussions with the client:

**Monitoring of Non-Security-Related Logs:**

* **Description:** The project does not include the monitoring and analysis of logs unrelated to security threats, such as operational logs, performance metrics, or other non-security event logs.
* **Rationale:** Focusing solely on security-related logs ensures that resources and efforts are concentrated on detecting and responding to potential security incidents promptly. This approach avoids unnecessary processing of non-critical data, optimizing system performance and responsiveness.

**Implementation of Backend Infrastructure Not Directly Related to Threat Detection and Response:**

* **Description:** The project scope excludes the development or deployment of backend infrastructure components that do not directly support threat detection, monitoring, or response.
* **Examples:** This includes general-purpose data storage solutions, networking infrastructure not related to log ingestion, or administrative tools not essential for security operations.
* **Rationale:** Limiting the scope to essential backend components ensures efficient resource allocation toward core functionalities critical to security operations. This prevents scope creep and maintains focus on delivering effective threat detection and response within defined timelines and resource constraints.

**Custom Development of New Features Not Specified in the Initial Scope:**

* **Description:** Custom development efforts outside the predefined features and functionalities specified in the initial project scope are excluded.
* **Clarification:** This includes any additional features, enhancements, or modifications not explicitly documented and approved in the initial scope of work.
* **Rationale:** Adhering strictly to the predefined scope ensures the project remains aligned with stakeholder expectations. It prevents scope expansion without proper evaluation and approval, ensuring project deliverables are completed within the established constraints and timelines.

## REQUIREMENTS DESCRIPTION

### FUNCTIONAL REQUIREMENTS:

1. **Identify and Document Monitoring Software:** Define and specify the software or web URL that requires monitoring, and outline the necessary monitoring requirements.
2. **Define and Document Security Threats:** Identify and document at least 10 common security threat scenarios relevant to the designated software.
3. **Document Logging and Monitoring Requirements:** Create a detailed document specifying at least 20 requirements for logging and monitoring. This should include specific DVWA logs related to user behavior, such as user input validation, failed login attempts, file access requests, and bandwidth usage.
4. **Scope Additional Security Scenarios:** Document specific requirements for an additional 5 security scenarios relevant to the monitored software.

* **Scenario-Based Monitoring:** Develop and implement monitoring scenarios tailored to each identified threat.
* **File Inclusion Attacks:** Monitor for suspicious file path requests within user input or parameters.
* **DoS Attacks:** Watch for high request rates from a single IP address and excessive bandwidth usage.
* **Identity-Driven Attacks:** Track failed login attempts that exceed a certain threshold and monitor unusual login attempts involving multiple usernames from a single source.
* **Directory Traversal Attacks:** Monitor attempts to access files or directories outside the web root.

1. **Analyse Logs and Set Alerting Thresholds:**
2. **Analyse Logs:** Review existing logs to establish appropriate thresholds for critical metrics.
3. **Define Alert Thresholds:** Set thresholds that trigger alerts based on monitoring scenarios, such as the number of file inclusion attempts, request rate limits per IP, bandwidth usage limits, and thresholds for failed login attempts and suspicious login patterns.
4. **Select and Configure Monitoring Tools:** Implement the monitoring scenarios using selected tools to meet all defined criteria.
5. **Document Security Monitoring and Incident Response Process:** Develop a structured incident response process to effectively handle alerts, including procedures for investigating potential attacks, mitigating threats, and reporting incidents.
6. **Implement Self-Healing Mechanism:** Integrate a self-healing mechanism to automatically respond to critical alerts.

### NON-FUNCTIONAL REQUIREMENTS:

1. **Performance:** Ensure monitoring tools provide real-time monitoring with minimal latency.
2. **Reliability:** Maintain system uptime in accordance with business requirements for availability.
3. **Security:** Ensure all communications and stored data comply with AES-256 encryption standards.
4. **Scalability:** Design the monitoring solution to seamlessly scale with a 50% increase in monitored resources within six months.
5. **Usability:** Document incident response procedures in a user-friendly format accessible to both technical and non-technical staff.
6. **Compliance:** Ensure that monitoring and response processes comply with GDPR and CCPA regulations.
7. **Maintainability:** Ensure that monitoring tools allow for easy configuration updates and troubleshooting by the IT operations team.
8. **Integration:** Ensure seamless integration of monitoring tools with the existing IT infrastructure and alerting systems.

# SELF-ASSESSMENT: TARGETS AND RESULTS (Responsibilities and Roles)

## ROLES / RESPONSIBILITIES

The following are the roles, responsibilities that I was entrusted with during the initial weeks of the project, during the pre-development and research phase. This waws in accordance with the team and facilitator.

|  |  |
| --- | --- |
| **Role/ Responsibility** | **Description / Expectation** |
|  |  |
| SOFTWARE DESIGN | Design an efficient, secure and fit-for-use solution for security monitoring using logs, including all components required for proper functioning post installation. |
| SOFTWARE DEVELOPMENT | Develop the proposed design solution, including preparing the set-up, environment, constant scrum development cycles, culminating inot the final solution. (4 ATTACK CATEGORIES) |
| SOFTWARE TESTING | Test the solution post development thoroughly (after each component) and post final solution dev completion to remove bugs, defects and incorporate any updates or upgrades required by the client. |
| SOFTWARE INTEGRATION | I volunteered to integrate the software for other threats developed by team mates to end with the final deliverable for the client. |
| DOCUMENTATION (TEAM) | I had also volunteered to personally design, develop and finish the documentation for team submissions, given my interest and inclination towards the impact effective documentation can make. |
| COMMUNICATE TEAM PROGRESS | Owing to my keen interest in communication of ideas and notions, I had also planned to convey the team’s progress along with my own, to ensure direction and pace of progress was according to requirements. This included constant participation in discussions with the facilitator, other batch mates and also strike newer conversation constantly, online and otherwise. |

Top of Form

## ACHIEVEMENTS (General)

These are achievements that I believe I have accomplished along the project, and a brief explanation of why I think so:

|  |  |
| --- | --- |
| **Achievements** | **Description / Explanation** |
|  |  |
| Research Cybersecurity, Design and other concepts | It is imperative that prior to development, a thorough research process using proven methodologies is carried out. I did so and my Project Deign Concept is a firm example of this approach. I designed and developed not only the part of the report for my threats but also the general project details and all of that required thorough research, not to mention the constant research that was required during all phases of development. This research gave me ample content to support my decisions and development, proving increasingly crucial as the project progressed. (Refer To Project Design Concept in Canvas for Team 4) |
| Developing Integrated Solution with multiple components | During the development of the monitoring solution that the client required, I was **constantly engaged** in programming endeavours with **PHP, JavaScript, Bash scripts,** and other components. This specific project required the integration of multiple components, such as **Python scripts** working in sync with the **Swatch log monitoring system**, and automating them to run scripts that would trigger alerting and self-healing. This was challenging, but ultimately, the delivery of my solution is a testament to my battle and subsequent victory over this challenge. |
| Volunteering For participation, discussions and development | I have **constantly volunteered** to design my team's documents for submission, help my team with **issues of both technical and non-technical** nature, and advise my batchmates on all sorts of issues. I also **actively participate** in conversations that concern the project, and I always help out whenever and wherever I can. This has helped me become **strikingly confident** in my own abilities to lead, assist, and most importantly, progress through an **all-around methodology** that not only involves the development of the technical aspects of the project but also the **non-technical components** of the journey that the project has come coupled with. |
| Problem Solving of all sorts | All projects require **problem-solving** regardless of their nature, but a **cybersecurity project** demands problem-solving of a more **complex and intricate nature**. This project has pushed my problem-solving limits to the extreme, starting with how to **parse logs**, **analyse** them to determine if an attack has occurred, and use this **timestamp** to trigger a response. Additionally, **automating** this entire solution for **four kinds of threats**, each with different **mitigation strategies**, was just one component of the project. To say the least, it has been an intense puzzle to solve. However, beyond the obvious technical challenges, there was also the question of how to **effectively present** these ideas without burdening people with the actual code—a problem that I believe I have successfully solved with my **presentation**. I am certain that my problem-solving skills have taken a **significant leap** and have improved **markedly**. |
| Trusing the Process and the Team | Development is a process and takes **time, effort,** and most importantly, **coordination** of all parties involved, including the team, the facilitator, and myself. **I have always had a tendency to burden myself with responsibilities that don’t necessarily have to be undertaken by me alone, driven by an anxious approach—not just towards the team or the process**, which are naturally required by a project. But this time around, I have tried **morphing my approach** and becoming more comfortable and trusting of my teammates and the process, which meant giving people time and confidence to allow for **growth and development**, like a seed underneath the soil trying to sprout. This gives me hope that, even professionally, I will be able to **build on this approach** rather than suffocating myself. |

# CONTRIBUTION: DELIVERABLES, SOFTWARE AND MORE

The following is a retelling of the contributions I have made in software development, integration, research, design, documentation and other significant accomplishments that have contributed to the successful delivery of the product to the client, as well as the deliverables expected by the facilitator:

## DESIGN: MECHANSIM FOR OVERALL SOLUTION

I developed component for detecting 4 scenarios of execution attacks using the following chronology:

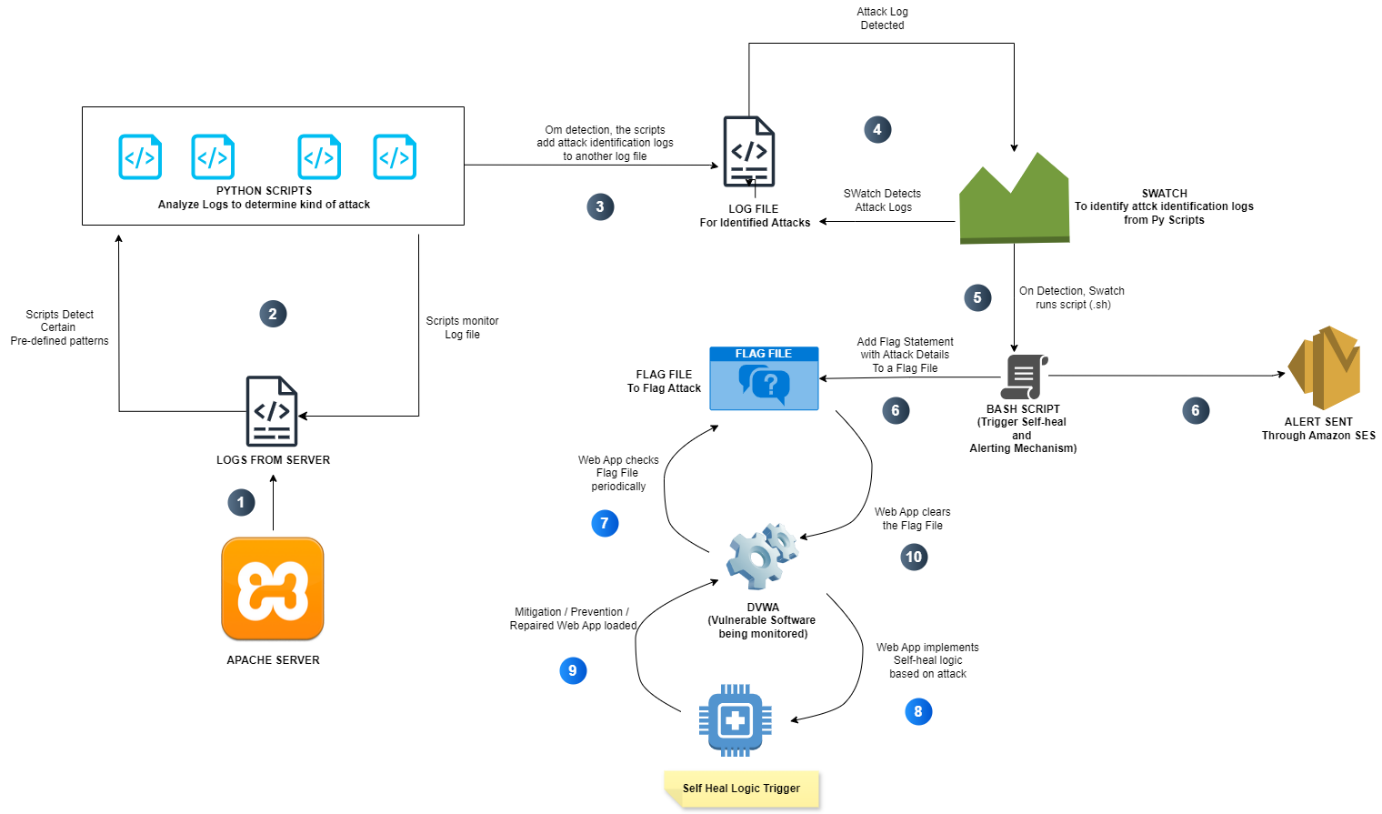


Figure 1 component order of mechanism

1. **Logs from Server:** The process begins with logs generated from the **Apache server**. These logs contain information about activities on the server, including potential attacks.
2. **Python Scripts Monitor Log File:** **Python scripts** continuously monitor the log file from the Apache server. These scripts are designed to detect specific patterns indicative of predefined attacks.
3. **Attack Detected:** When the Python scripts identify an attack pattern within the log file, an attack is considered **detected**.
4. **Python Scripts Add Identification Logs:** Upon detecting an attack, the Python scripts add detailed information about the identified attack to a separate **log file**. This log file serves as a record of detected attacks.
5. **Swatch Detects Attack Logs:** **Swatch**, another tool or process, monitors the log file created by the Python scripts, specifically looking for entries indicating identified attacks.
6. **On Detection, Swatch Runs Script (sh):** When Swatch detects an attack log entry, it executes a **shell script (sh)**, which likely initiates further actions in response to the detected attack.
7. **Web App Checks Flag File Periodically:** A **web application** runs in the background, periodically checking the contents of a **flag file**. This file stores information about active attacks.
8. **Flag File to Flag Attack:** If the web application finds an entry in the flag file indicating an active attack, it considers that attack as **flagged**.
9. **Add Flag Statement with Attack Details to Flag File:** For each flagged attack, the web application adds a **statement** with detailed information about the attack to the flag file. This provides a record of active attacks and their specifics.
10. **Bash Script (Trigger Self-heal and Alerting Mechanism):** A **Bash script** is responsible for triggering the self-healing mechanism and sending alerts. This script is executed as part of the response to a flagged attack.
11. **Alert Sent Through Amazon SES:** The Bash script utilizes **Amazon SES (Simple Email Service)** to send out alerts regarding the detected and flagged attack.
12. **Web App Clears the Flag File:** After a certain period or upon successful mitigation of an attack, the web application **clears** the corresponding entry from the flag file, indicating that the attack has been addressed.
13. **DVWA (Vulnerable Software Being Monitored):** The process involves **DVWA (Damn Vulnerable Web Application)**, suggesting that this vulnerable software is part of the system being monitored for attacks.
14. **Web App Implements Self-heal Logic Based on Attack:** The web application implements **self-healing logic** tailored to specific types of attacks, allowing for automated responses to counteract detected threats.
15. **Self-heal Logic Trigger:** When the self-healing logic is activated based on the detected attack, it takes appropriate actions to **mitigate** the attack and restore normal system operation.

## SOFTWARE COMPONENT: EXECTION ATTACK DETECTION , ALERTING AND SELF-HEALING

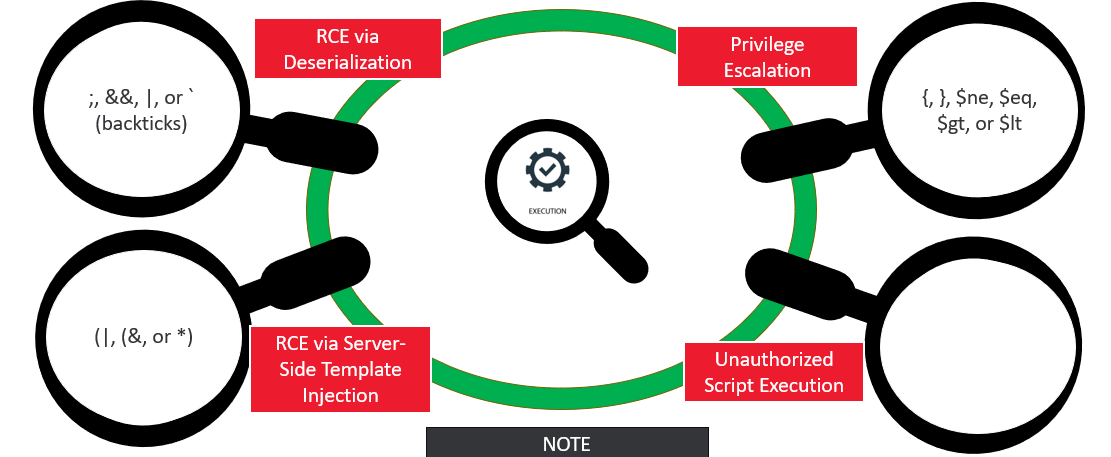


Figure 2 patterns for execution attack detection

### Python Script for Log Monitoring

**Contribution:** I developed a Python script to monitor and analyse log files for suspicious activities. This script is designed to detect various types of execution attacks, including Remote Code Execution (RCE) and Command Injection.

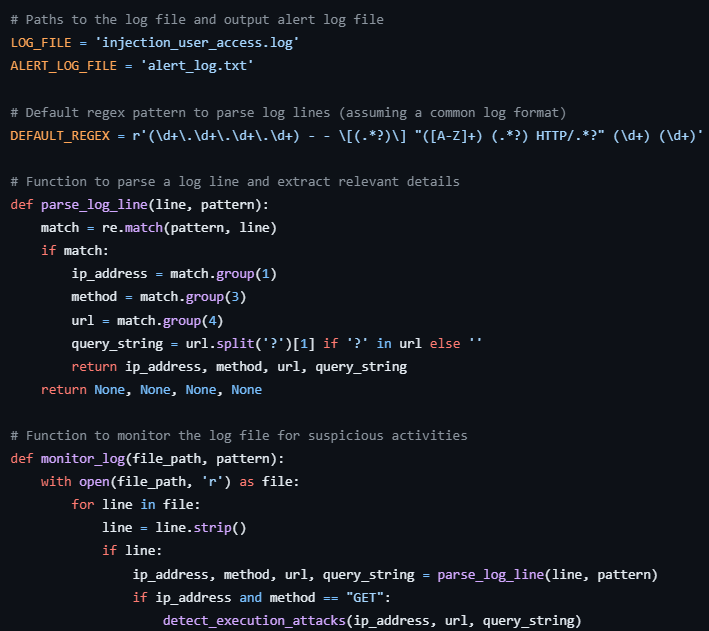


Figure 3 EXECUTION ATTACK PYTHON SCRIPT (1)

### Detection and Alert System

**Contribution:** I integrated functions to detect specific attack patterns and handle alerts. These functions assess if an attack has occurred based on predefined criteria and then log detailed alert messages.

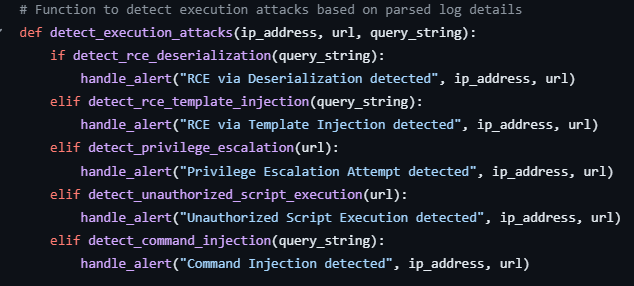


Figure 4 EXECUTION ATTACK PYTHON SCRIPT (2)

### Swatch Configuration for Alert Handling

**Contribution:** I set up a Swatch configuration file to monitor the alert log generated by the Python script. Swatch processes alerts related to RCE and executes actions such as writing to a flag file and running a notification script.

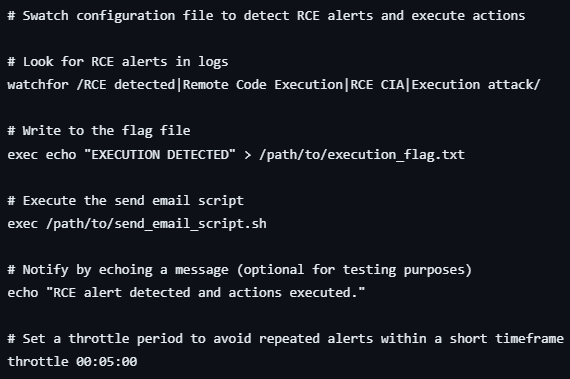


Figure 5 EEXECUTION ATTACK SWATCH CONFIG FILE

### Integration with DVWA

**Contribution:** I developed a PHP file for DVWA that continuously checks for the flag file created by Swatch. Upon detecting the flag, it adjusts the security level, reverses the attack effects, and blocks the attacking IP, thus integrating the entire system to ensure automated responses to threats.



Figure 6 SELF-HEAL MECHANISM (DVWA EXECUTION)

By developing the Python script, configuring Swatch, and integrating the DVWA PHP file, I created a robust system for detecting and handling injection attacks. This comprehensive approach not only logs and alerts on potential threats but also automates response actions to mitigate the impact of detected attacks.

## SOFTWARE COMPONENT: INJECTION ATTACK DETECTION , ALERTING AND SELF-HEALING

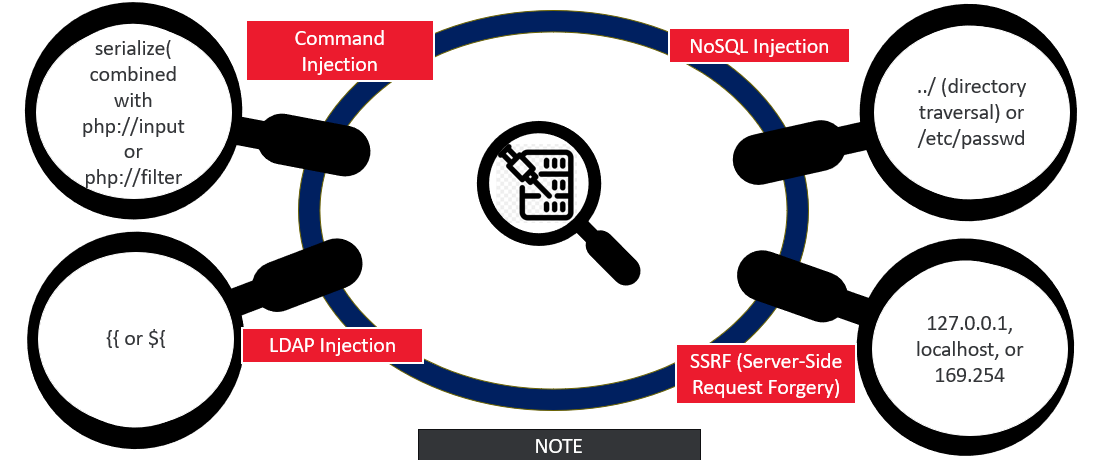


Figure 7 PATTERNS FOR DETECTING INJECTION ATTACKS

**NOTE: The code structure for the python file and Swatch config file is similar to execution attacks and can be found in my repo, link to which is at the end of this section.**

### Python Script for Log Monitoring

**Contribution:** I developed a Python script to monitor log files specifically for injection attacks. This script identifies various types of injection attempts by parsing log entries and extracting relevant details.

### Detection and Alert System

**Contribution:** I implemented functions to detect various types of injection attacks, including Command Injection, LDAP Injection, NoSQL Injection, SSRF, and XSS. If an attack is detected, the script logs an alert message with details.

### Swatch Configuration for Alert Handling

**Contribution:** I configured Swatch to monitor the alert log generated by the Python script. Swatch detects any injection-related alerts, writes to a flag file, and executes an email notification script.

### Integration with DVWA

**Contribution:** I developed a PHP file for DVWA that checks the flag file created by Swatch. Upon detecting the flag, the PHP script adjusts the security settings, reverses the effects of the injection attack, and blocks the IP involved.

By developing the Python script, configuring Swatch, and integrating the DVWA PHP file, I created a robust system for detecting and handling injection attacks. This comprehensive approach not only logs and alerts on potential threats but also automates response actions to mitigate the impact of detected attacks.

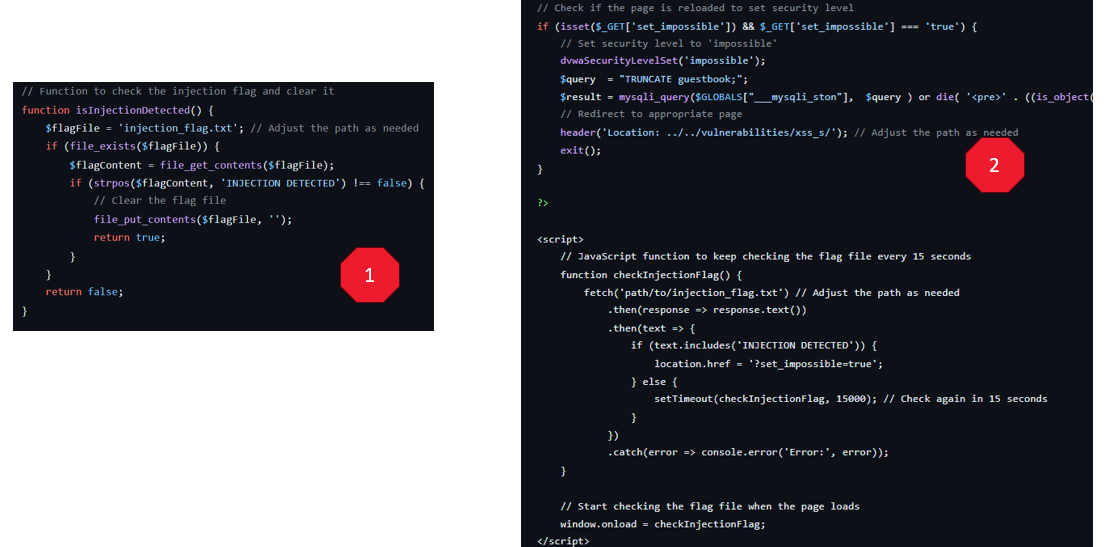


Figure 8 self-heal mechanism for injection attack (dvwa)

## SOFTWARE COMPONENT: SESSION ATTACK DETECTION , ALERTING AND SELF-HEALING

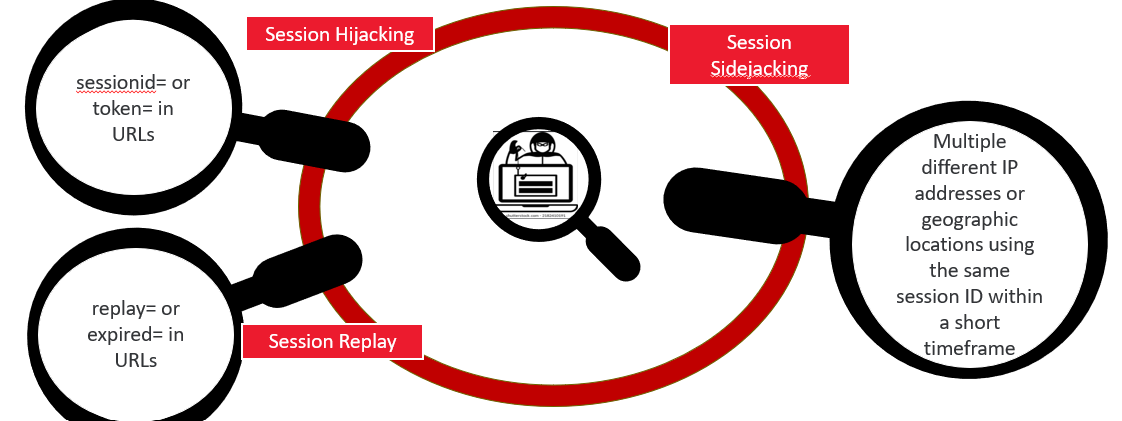


Figure 9 PATTERNS FOR SESSION ATTACK DETECTION

**NOTE: The code structure for the python file and Swatch config file is similar to execution attacks and can be found in my repo, link to which is at the end of this section.**

### Python Script for Log Monitoring

**Contribution:** I developed a Python script to monitor log files for session attacks. This script parses log entries to identify suspicious activities related to session management and handles alerts accordingly.

### Detection and Alert System

**Contribution:** I implemented functions to detect various types of session attacks, including Session Hijacking, Session Fixation, and Session Side jacking. When an attack is detected, the script logs an alert message detailing the suspicious activity.

### Swatch Configuration for Alert Handling

**Contribution:** I configured Swatch to monitor the alert log generated by the Python script. Swatch detects any session-related alerts, writes to a flag file, and executes an email notification script to inform about the detected session attacks.

### Integration with DVWA

**Contribution:** I developed a PHP file for DVWA that checks the flag file created by Swatch. Upon detecting the flag, the PHP script adjusts the security settings to mitigate session-related attacks, such as regenerating session IDs and invalidating affected sessions. The script also blocks suspicious IP addresses involved in the attacks.

**NOTE: For demonstrating detection of Self-healing for Session attack, I have configured the Brute Force vulnerability exploitation page to respond to the self-healing mechanism and perform certain actions when flag file statement is detected on Session attack detection by Python script.**

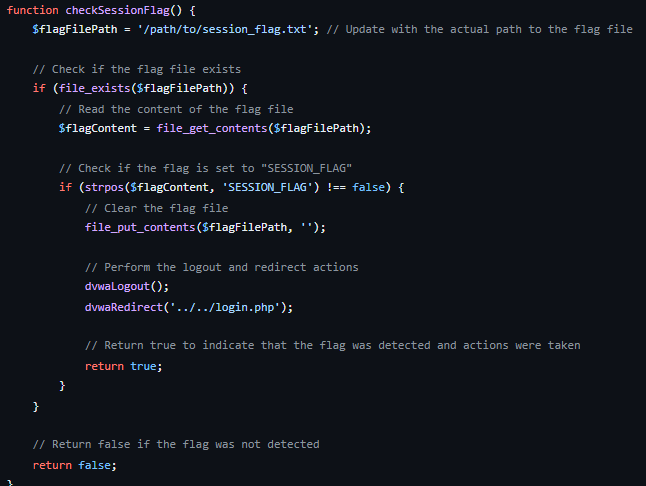


Figure 10 SELF-HEAL MECHANISM FOR SESSION ATTACK (DVWA BRUTE FORCE APGE)

By developing the Python script, configuring Swatch, and integrating the DVWA PHP file, I established a comprehensive system for detecting and handling session attacks. This approach ensures effective monitoring, alerts, and automated responses to mitigate the impact of session-related threats.

## SOFTWARE COMPONENT: PHISHING ATTACK DETECTION , ALERTING AND SELF-HEALING

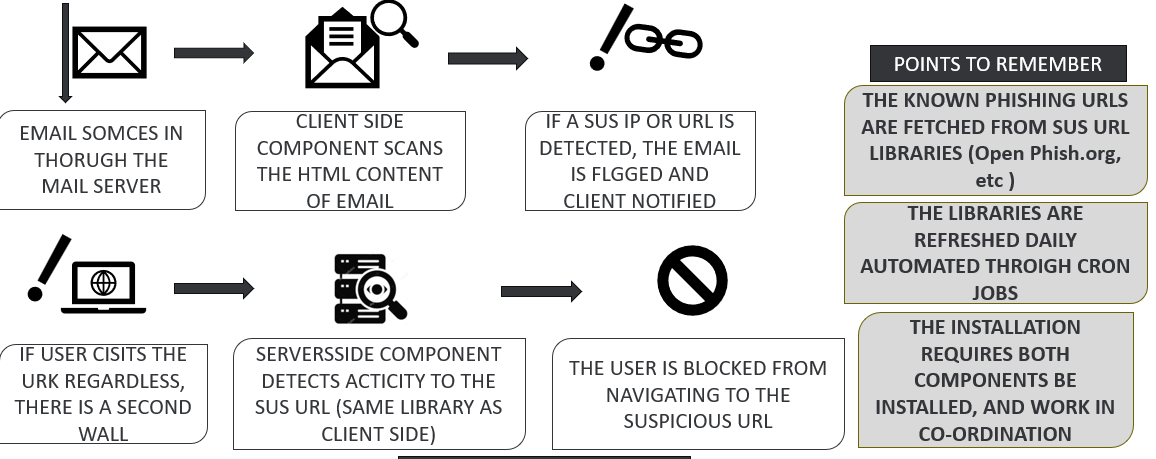


Figure 11 methodology for phishing email detection

For Phishing attacks, I have made assumption and morphed techniques that I have thoroughly discussed in the User Manual (Project Report).

This is the **design / methodology for the mechanism**:

**1. Email Sources Through Mail Server**

Emails are collected from various sources and processed through the mail server.

**2. Client-Side Component Scan**

The HTML content of an incoming email is scanned by a client-side component.

**3. Suspicious IP or URL Detection**

The client-side component checks the email for any suspicious IPs or URLs against a database of known phishing URLs (fetched from libraries like Open Phish).

**4. Email Flagging and Notification**

If a suspicious IP or URL is found, the email is flagged as potentially harmful, and the user is notified.

**5. Server-Side Component Detection**

Simultaneously, a server-side component also scans for suspicious activity related to the same URLs.

**6. User Blocking (Optional)**

If a user attempts to visit a flagged suspicious URL, both client-side and server-side components work together to block the user from accessing the website.

**7. Library Refresh**

The database of known phishing URLs is updated regularly through automated cron jobs to ensure up-to-date protection.

**8. Demonstration**

For testing purposes, a dummy email can be scanned using the client-side script, and its URLs can be checked against the phishing URL library.

### Email Sources Through Mail Server

I developed a system where emails are collected from various sources and processed through a mail server. This system ensures that all incoming emails are efficiently managed and examined.

### Client-Side Component Scan

I created a client-side component to scan the HTML content of incoming emails. This component plays a crucial role in analysing the email's content to identify any potential threats.

### Suspicious IP or URL Detection

In this phase, I implemented functionality to detect suspicious IP addresses or URLs. This detection is performed against a database of known phishing URLs, which I integrated from libraries such as Open Phish. This step is vital in identifying potential phishing attempts based on known threats.

### Email Flagging and Notification

When the client-side component identifies a suspicious IP or URL, I designed the system to flag the email as potentially harmful. Additionally, I ensured that users receive notifications about these flagged emails to keep them informed and alert.

### Server-Side Component Detection

Alongside the client-side scan, I developed a server-side component that scans for suspicious activity related to the same URLs. This dual approach enhances the system's ability to detect and respond to potential threats more effectively.

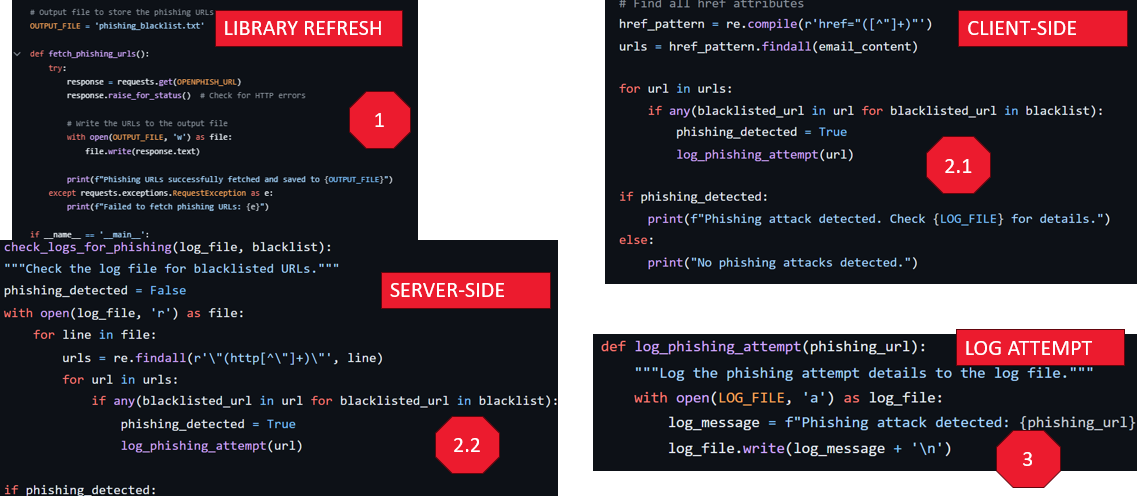


Figure 12 phishing attack logic

### User Blocking (Optional)

For added security, I included an optional feature where users attempting to visit flagged suspicious URLs are blocked. This mechanism, which involves coordination between the client-side and server-side components, helps prevent users from accessing potentially harmful websites.

### Library Refresh

To ensure the system remains effective against new threats, I set up an automated process to refresh the database of known phishing URLs regularly. This is managed through cron jobs, keeping the threat database current and relevant