IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

# AEGIES-AUTO: Autonomous Self-Healing Security System

**Surya Narayana Reddy B G[*1], Manoj Kumar L[*2], Prof. Thejaswini M N[*3]**
Students, Department of Computer Application[1,2]
Assistant Professor, Department of Computer Application[3]
Maharaja Institute of Technology Mysore, Belawadi, , Karnataka, India

**Abstract:** *Cyberattacks are becoming more frequent and sophisticated, and traditional antivirus software is often too slow or ineffective to detect and prevent new threats. Many security setups today still rely on human intervention. The delay this creates gives attackers a window to do serious damage. This shows how important it is to have a system that can detect and neutralize threats automatically, without waiting for human intervention. This paper presents AEGIES-AUTO, an autonomous security system capable of detecting and eliminating cyber threats in real time without any human involvement. The system operates in real time, monitoring all active processes and network connections on a host machine, and responds instantly when it detects suspicious behaviour, such as a reverse shell, a malicious process, or unusual port activity. Its response sequence terminates the offending process, blocks the associated ports, generates a forensic backup of the threat data for subsequent review, captures a screenshot as evidence, and sends instant alerts via email, desktop notifications, and voice messages, all within seconds of detection. AEGIES-AUTO is implemented in Python and provides a web-based dashboard showing live security metrics, threat history, and overall system health. It also features whitelist management to prevent false positives, a quarantine directory for isolating suspicious files and structured report generation for organised documentation. What makes AEGIES-AUTO different from traditional tools is the combination of speed and true autonomy: it doesn't wait for a human response, so threats are neutralised immediately upon detection. The system is realistic security for students, individual users and small organisations that cannot justify the cost of commercial security subscriptions, being lightweight, easy to deploy and free of licensing costs. In summary, AEGIES-AUTO demonstrates that effective, autonomous, real-time threat response can be achieved without the overhead normally associated with enterprise-grade security platforms.*

**Keywords**: *Cyberattacks*.

## 1. INTRODUCTION

The speed and sophistication of cyberattacks have outstripped the ability of traditional, signature-based security tools to keep up. Malware like reverse shells, ransomware, and crypto-mining can compromise a system in seconds, but traditional anti-virus software still relies on periodic signature updates and, in many cases, human review before a response is made. The delay provides attackers with an operational window that is often long enough to establish persistence, escalate privileges or exfiltrate data.

This was the gap that AEGIES-AUTO, from the mythological shield of Zeus, combined with 'AUTO' for full automation, was developed to fill. It acts as a standalone self-healing security system that constantly monitors system processes and network activity, and detects and neutralises anomalous behaviour without waiting for an individual to intervene. It's not like traditional antivirus software, which needs to be updated with new signature databases regularly; instead, it observes process names, command line arguments, port activity, and resource consumption patterns to detect malicious behavior on the fly.


DOI: 10.48175/IJARSCT-37879

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, July 2026 Impact Factor: 8.2

### *A. Statement of the problem*

While antivirus software is widely deployed, cyberattacks are one of the most persistent problems of modern computer systems. Many attacks go undetected or are detected only after damage has been done, primarily because conventional tools depend on databases of existing attack signatures, which are inherently blind to new or modified attacks. A reverse shell or crypto-miner can compromise a system in minutes, but industry reports consistently show the average time to detect a breach is measured in days, sometimes weeks. For the individual user, the student, the small organisation without a dedicated security team, round-the-clock manual monitoring is simply not realistic. What's needed is a smart, automated system that can constantly monitor an environment, identify threats as they occur, and recover the system without waiting for a human to step in.

### *B. Motivation*

The reason for this work is the practical observation that many people, students and small organisations cannot afford enterprise-grade Endpoint Detection and Response (EDR) platforms and cannot staff round-the-clock security operations. Even specialised platforms like Kali Linux, popular with security professionals, are vulnerable during the window between an attack and the trigger of a response. AEGIES-AUTO was developed to provide automated, always-on protection to this very segment of the market, and to mitigate the false-positive fatigue that affects so many tools on the market, through a smart whitelist that can exclude safe, recurring processes from scrutiny.

### *C. Contribution of this Work*

A real-time process- and network-monitoring engine that does not rely on a signature database, but rather on behavioral indicators.

A fully autonomous response pipeline that stops malicious processes, blocks suspicious ports and quarantines suspicious files without human intervention.

A forensic backup and structured logging mechanism that captures process metadata, screenshots and timestamps for post-incident analysis.

Multi-channel alerting system across email, desktop notification and voice alert, ensuring that users are always informed even when they are away from the screen.

Lightweight browser-accessible dashboard to visualise security posture in real-time, export to Excel, HTML and CSV reports.

## II. RELATED WORK

The research in autonomous cybersecurity has been accelerated by advances in machine learning and behavioural analysis. Table I outlines the principal sources used in this project and the manner in which each influenced the design of AEGIES-AUTO.

**TABLE I. Summary of Literature Reviewed**

| Source | Key Finding | Relevance to AEGIES-AUTO |
|---|---|---|
| Kumar et al. [6], 2021 | AI-based systems reduce response time; behavioural analysis outperforms signatures, but false positives remain a challenge | Informs autonomous detection and response design |
| Singh et al. [7], 2022 | Behavioural monitoring of processes and network calls detects zero-day attacks; context is essential | Supports the process/network monitoring approach |
| Stallings [8], 2019 | Covers encryption, authentication, malware, and defence-in-depth | Foundational basis for secure credential handling |
| Schneier [9], 2015 | Explains hashing, digital signatures, and secure storage | Guides secure logging and credential protection |
| MITRE | Structured taxonomy of adversary tactics and | Validates the threat categories |

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

| | | |
|---|---|---|
| ATT&CK [14] | techniques | targeted by the detection engine |

### *A. Limitation and Existing Systems*

Traditional antivirus software relies primarily on signature-based detection, which requires frequent updates and still misses new threats, and response is usually dependent on a human acting on an alert, which slows things down considerably. More sophisticated Endpoint Detection and Response (EDR) solutions provide automated response and real-time monitoring, but are often difficult to configure and expensive to license, putting them out of reach for individual users and smaller organizations. Host-based intrusion detection systems (HIDS) are often detection-focused rather than remediation-focused, generating alerts that still require a human to investigate. Network-based security such as firewalls and intrusion prevention systems protect at the perimeter and can miss threats that originate from within the system itself. Another limitation across these categories is the general lack of integrated forensic backup. The lack of an evidence trail makes it hard to understand how an attack unfolded after the fact.

### *B. Research Gap*

This body of work reveals a clear gap: few systems bring together real-time monitoring, autonomous response, forensic evidence capture, structured logging and multi-channel alerting into a single, cohesive and free platform. Commercial EDR products provide similar capability, but require significant expertise and budget to deploy. Host based intrusion detection systems tend to just alert, and leave remediation to a human operator. AEGIES-AUTO directly addresses this gap by making autonomous response the default behaviour rather than optional add-on, and by bundling forensic backup, multi-channel notification and reporting into one modular architecture.

## III. THEMES AND SCOPE

### *A. Goals*

Carry out continuous real-time monitoring of running processes to flag suspicious behaviour based on defined patterns.
Observe active network connections for suspicious ports and reverse-shell-like activity.
Build an auto-healing engine capable of killing hostile processes and blocking affected ports without manual intervention.
Take forensic backups capturing process metadata before remediation for later review.
Provide a browser-based dashboard of security events, system metrics and recent alerts.
Real-time alerts through email, desktop notification and voice to reach users through multiple channels.
Enable export of security reports in Excel, HTML and CSV formats for documentation and review.

### *B. Objectives*

AEGIES-AUTO is designed for Windows platforms, and due to its Python foundation, it has cross-platform support for Kali Linux and Ubuntu. The system covers a range of attack vectors including reverse shells, Meterpreter-style payloads, netcat listeners, and processes running with abnormally high resource usage. Functionally, it covers process monitoring, network connection analysis, automated threat elimination, forensic backup, web dashboard, email notifications, comprehensive logging, desktop and voice alerts, screenshot capture, report export, system-tray integration and auto-start on boot. The system is designed to operate on standard consumer hardware and does not require any special infrastructure, requiring only Python 3.11 or later.

## IV. SYSTEM ARCHITECTURE PROPOSED

AEGIES-AUTO follows a layered architecture that consists of a Presentation Layer (the Flask-based web dashboard), an Application Layer (the Flask web server and REST endpoints), a Service Layer (detection, isolation and healing engines), and a Data Layer (SQLite database plus JSON and text logs), as shown in Fig. 1. This separation of concerns helps keep the codebase modular, makes individual components easier to test and extend, and limits the blast radius of future changes.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

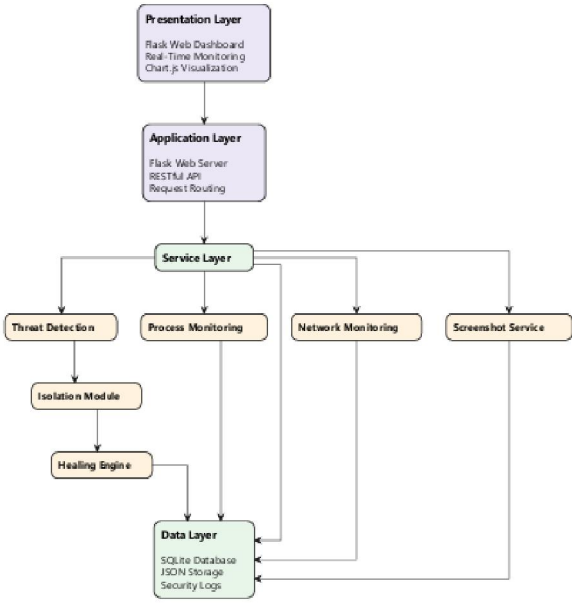


Fig. 1. Layered system architecture of AEGIES-AUTO.

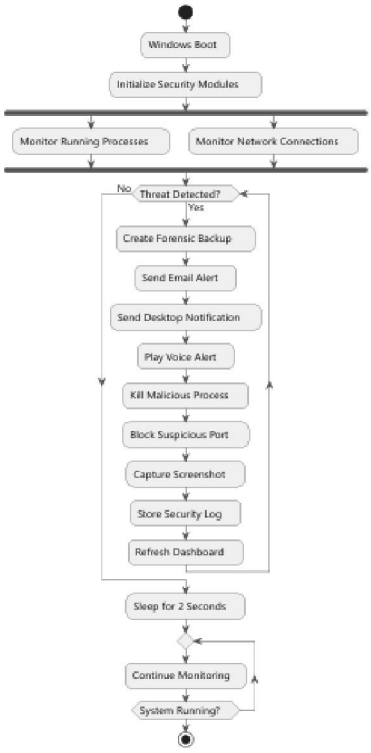


Fig. 2. System flowchart showing the continuous monitor–detect–respond loop.

### *A. Detection Logic*

The detection engine constantly scans running processes via the psutil library and checks them against known-suspicious process names, command-line patterns, and port activity (including commonly abused ports like 4444, 1337,

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, July 2026 Impact Factor: 8.2

and 31337, which are often used for reverse-shell listeners). Anomalies in resource usage (for example a process with a high CPU or memory consumption that is otherwise unremarkable) are considered as an additional signal.

***B. Self Response Pipeline***

After the threat is confirmed, the system executes a predefined sequence (depicted as the operational loop in Fig. 2): a forensic backup is created (tracking the process ID, command line arguments, and timestamp), alerts are sent (email, desktop notification, and voice channel), the malicious process is killed (using system kill signals), the network port used is blocked (at the firewall), a screenshot is taken (visual proof), and the event is logged (structured JSON and plain-text logs) and the dashboard is refreshed.

***C. Whitelist and Quarantine Management***

To reduce false positives, administrators can whitelist certain processes that are known to be safe, and these will not be automatically terminated, keeping the system usable in development or general purpose environments where legitimate tools may otherwise look suspicious. Suspicious files that are not immediately killed as processes are placed in a quarantine directory where they are neutralized but can still be examined, restored or deleted by an administrator at a later time.

## V. DETAILED DESIGN

The static structure of AEGIES-AUTO is shown in the class diagram of Fig. 3 where responsibilities are grouped in a Detection Layer (ThreatDetector, ProcessMonitor, NetworkMonitor), a Response Layer (Alert, Healer) and a Data Layer (Whitelist, Threat). The ThreatDetector class is the orchestrator, calling the process and network monitors and passing on the confirmed threats to the Alert and Healer classes for notification and remediation respectively.

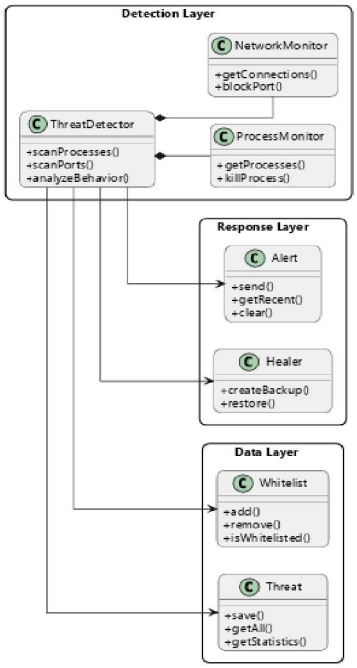


Fig. 3. Class diagram showing the static structure of the detection, response, and data layers.

The entity-relationship diagram of Fig. 4 shows the relational structure of the underlying SQLite database. Core entities are Users, Threats, Alerts, Forensic_Backups, Screenshots, Quarantine, Whitelist, Reports, Security_Logs Multiple detected threats can be associated with a single user session. Each threat can generate one or more alerts, a forensic backup record and associated screenshot, providing investigators with a complete evidentiary trail for each incident.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

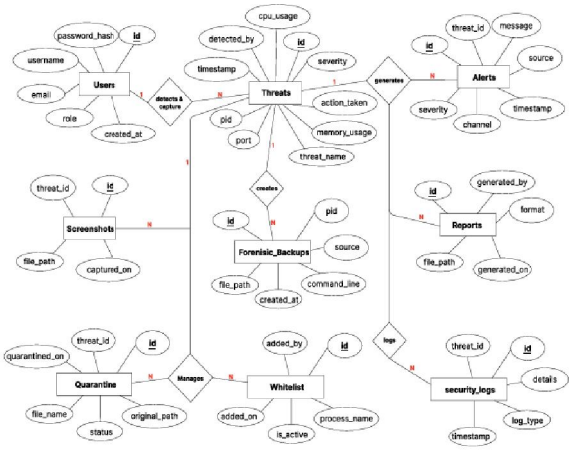


Fig. 4. Entity-relationship diagram of the AEGIES-AUTO database schema.

## VI. IMPLEMENTATION

The system is implemented in Python 3.11 to allow cross-platform compatibility, using psutil for process and network introspection. Threat trends and category breakdowns are visualized in near real time using Chart.js on a web dashboard served by Flask. SQLite provides embedded, serverless storage for threat records, alerts and whitelist configuration, so there is no need for a separate database service.

### *A. Evidence capture and notification*

Notifications are delivered over three separate channels: smtplib for email alerts, plyer/win10toast for Windows desktop notifications, and pyttsx3 for text-to-speech voice alerts, allowing users to be notified even when they are not looking at a screen. pyautogui takes forensic screenshots at the time of detection and pystray with Pillow provides a system tray icon for easy access to the dashboard and controls without switching windows.

### *B. Core Monitoring Algorithm*

The basic monitoring loop, shown in pseudocode below, runs forever scanning the running processes, comparing each one to the known threat indicators and – if there is a match – executing the full alert-and-remediation flow before pausing briefly and starting again.

```
BEGIN
  INITIALISE detection modules, database, notifiers
  SET threat_indicators = {process_names, ports, patterns}
  WHILE system is running DO
     processes = GET running processes
     connections = GET active network connections
     FOR each entry in processes + connections DO
        IF entry matches threat_indicators
          AND entry NOT IN whitelist THEN
           CREATE forensic_backup(entry)
           SEND email_alert, desktop_alert, voice_alert
           KILL process(entry)
           BLOCK port(entry)
           CAPTURE screenshot()
           LOG event TO database AND JSON/text logs
```

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, July 2026 Impact Factor: 8.2

```
        REFRESH dashboard
      END IF
    END FOR
    WAIT scan_interval (2 seconds)
  END WHILE
END
```

### *C. Web Dashboard*

The dashboard is a Flask app, listening on port 5000. It refreshes its statistics via a background thread every two seconds, querying the database for the latest counts of threats detected, processes killed and recent alerts. Figure 5 shows the live dashboard, which includes a chart for the seven-day threat trend, a breakdown of threats by process type, and a scrolling log of the most recent events. It also includes export controls for Excel, HTML and CSV reports.

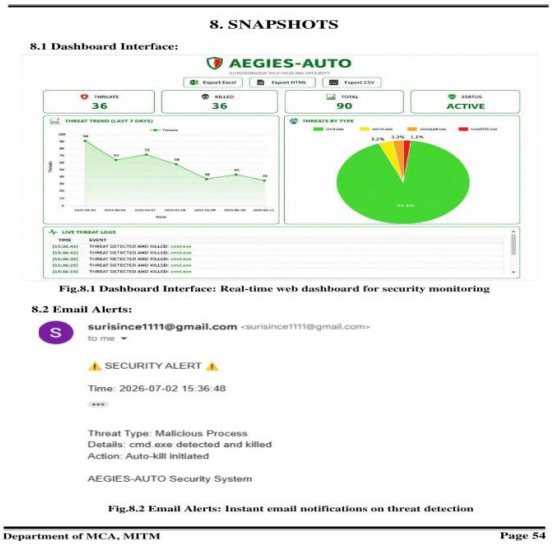


Fig. 5. AEGIES-AUTO web dashboard showing live threat statistics and event logs.

## VII. TEST AND VALIDATION

Black-box testing (equivalence partitioning, boundary value analysis, and error guessing) and white-box testing (statement, branch, and path coverage) were applied on the detection, isolation, backup, notification and dashboard modules to validate the system.

### *Black-Box Testing*

**TABLE II. Representative Black-Box Test Cases**

| ID | Scenario | Expected Result |
|---|---|---|
| BB01 | Detect notepad.exe as threat | Threat detected |
| BB05 | Kill notepad.exe after detection | Process terminated |
| BB06 | Send email alert on detection | Email received |
| BB07 | Capture screenshot on threat | Screenshot saved |
| BB08 | Generate Excel report | Excel file created |
| BB09 | Add process to whitelist | Process whitelisted |

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, July 2026
Impact Factor: 8.2

### *B. White-Box Testing*

**TABLE III. Representative White-Box Test Cases**

| ID | Code Section | Test Scenario | Result |
|---|---|---|---|
| WB-01 | scan_processes() | Mock process list | Correct threat list returned |
| WB-03 | kill_process() | Valid PID | Process terminated |
| WB-04 | kill_process() | Invalid PID | Error handled gracefully |
| WB-05 | create_backup() | Backup creation | Backup file created |
| WB-09 | send_email() | Invalid SMTP config | Error handled gracefully |

### *C. Functionality and Performance Results*

All functional test cases - process detection, process termination, forensic backup creation, whitelist protection, report export in Excel, HTML and CSV formats - were passed under repeated execution. The system's practical responsiveness was measured by performance testing, summarised in table IV, rather than merely its correctness.

**TABLE IV. Performance Test Results**

| Metric | Target | Observed |
|---|---|---|
| Threat detection time | < 3 s | 1.8 s |
| Process kill time | < 1 s | 0.5 s |
| Dashboard load time | < 3 s | 2.1 s |
| Email alert delivery | < 10 s | 5 s |
| CPU usage | < 20% | 15% |
| Memory usage | < 200 MB | 156 MB |

This means that the autonomous response pipeline (detection, backup, alerting and termination) is effective enough to work within the time window in which most fast-moving threats such as reverse shells or crypto-miners could otherwise take hold. Resource overhead was modest during testing, supporting the suitability of the system for continuous background operation on standard consumer hardware.

## VIII. USES

AEGIES-AUTO has practical applications for a number of user groups. It can serve as a practical teaching tool for students and security enthusiasts to gain insight into threat detection and autonomous response, similar to a guided sandbox with active defence for training programmes and course work. Basic but effective protection of individual users against common threats. Added benefit of protection even when users are not actively monitoring their system. Educational institutions can use it in their computer labs in order to stop students from running malicious programs intentionally or unintentionally, thus maintaining a stable learning environment.

For smaller organizations that don't have security staff, the automated response capability means IT staff don't have to spend time on routine disruptions and can instead focus on larger initiatives, while still delivering a meaningful layer of protection. Developers and IT professionals can use it to protect their development workstations from malicious background processes. Developers and IT professionals can focus on their development work without second guessing every background action.

## IX. CONCLUSION AND FUTURE SCOPE

AEGIES-AUTO proves that a self-healing, autonomous security system can be built using open-source components without sacrificing responsiveness or forensic depth. The system closes much of the detection-to-response gap that leaves conventional tools – and their users – exposed, by combining real-time process and network monitoring with automatic remediation, forensic backup, structured logging and multi-channel notifications. Its small resource footprint

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, July 2026 Impact Factor: 8.2

and absence of licensing costs, make it a realistic option for individuals, students, educational laboratories and small organisations that cannot justify the expense of commercial EDR.

Future work will include incorporating machine-learning-based anomaly detection to detect threats falling outside known behavioural patterns, extending support to macOS and Linux for mixed-environment deployments, adding centralised management for monitoring multiple instances across a network, integrating external threat-intelligence feeds to keep detection logic current against emerging attack techniques, building a companion mobile application for remote monitoring, and implementing automated remediation that can roll back or repair compromised files and configurations to further strengthen the system's self-healing behaviour.

## REFERENCES

[1] Python Software Foundation, “Python 3.11 Documentation,” Available online.
[2] Flask Documentation, “Flask Web Framework,” [Online]. Available: https://flask.palletsprojects.com/
[3] psutil Documentation, “Cross-platform process and system monitoring,” [Online]. Available: https://docs.python.org/3/
[4] SQLite Documentation, “SQLite Database Engine,” [Online]. Available: https://psutil.readthedocs.io/ Available: https://sqlite.org/docs.html
[5] Chart.js Documentation, "Simple HTML5 Charts," [Online]. Available: https://www.chartjs.org/docs/
[6] S. Kumar, R. Patel, and D. Shah, "AI-Based Autonomous Security Systems," International Journal of Cybersecurity Research, 2021.
[7] A. Singh, P. Verma, and R. Gupta, “Real-Time Threat Detection Using Behavioural Analysis,” Journal of Information Security, 2022.
[8] W. Stallings, Cryptography and Network Security: Principles and Practice, 4th ed. Pearson Education, 2019.
Wiley. [9] B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2015.
[10] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modelling Language User Guide. Addison-Wesley, 1999.
[11] R. S. Pressman, B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th ed., 2014. McGraw-Hill Education. 2019.
[12] I. Sommerville, Software Engineering, 10th ed. Pearson Education. 2016.
[13] National Institute of Standards and Technology. NIST Special Publication 800-63, “Digital Identity Guidelines”.
[14] MITRE ATT&CK Framework, “Adversarial Tactics, Techniques, and Common Knowledge.” [Online]. Available: https://pages.nist.gov/800-63-3/ [15] OWASP Foundation, "OWASP Top 10. [Online]. Available: https://attack.mitre.org/ [Online]. Available at: https://owasp.org/www-project-top-ten/ [Accessed 20 July 2020].