# Technical Architecture and Functional Overview of ARMADA: A Next-Generation SIEM System for Proactive Cyber Defense

**Aleksandar Kotevski[1,2], Ljubica Bila Kotevski[1,2]*, Zoran Vučković[3], Dalibor Dobrilovic[4], Stanislav Cerović[3], Marko Krstić[5] and Rade Dragović[6]**

[1]Megatrend University, Faculty of Computer Science, 11000 Belgrade, Serbia

[2]Advanced Cyber Security DOO, Rudnicka 8, 11000 Belgrade, Serbia

[3]Academy of Hospitality, Tourism, and Wellness, Tošin Bunar 179đ, 11000 Beograd, Serbia

[4]University of Novi Sad, Technical Faculty "Mihajlo Pupin", Djure Djakovic bb, 23000 Zrenjanin, Serbia

[5]Regulatory Authority for Electronic Communications and Postal Services (RATEL), Palmotićeva 2, 11000 Beograd, Serbia

[6]Institute for Standards and Technology, 11000 Belgrade, Serbia

E-mails: aleksandar.kotevski@acs.co.rs , ljubica.bila.kotevski@acs.co.rs , zvuckovic@akademijahtw.bg.ac.rs , dalibor.dobrilovic@uns.ac.rs , stanislav@akademijahtw.bg.ac.rs , marko.krstic@ratel.rs , rade.dragovic@instate.biz

*Corresponding author

**Abstract:** *In the recent decade, we have witnessed rapid developments of Information and Communications Technology (ICT). As a result, we are continuously facing changes in interconnected systems, not only considering the complexity and growth of interconnected devices, but also their functional diversity. These changes affect the impact on the security of these systems as well. That's why it is important to adapt existing SIEM (Security Information and Event Management) systems to tackle these new challenges. This paper presents a novel approach to building SIEM systems. The ARMADA platform represents a cutting-edge, scalable SIEM system that provides real-time cyber defense across diverse infrastructures, including critical government, financial, and industrial IoT sectors. This paper details ARMADA's architecture and core functionalities. ARMADA's combination of a microservices-based design, big data processing, AI-driven capabilities, and automated response provides a robust and adaptable solution that effectively addresses modern cybersecurity challenges. The system's efficiency and functionality are evaluated with the analyses of the system's response to three vector attacks.*

*Keywords: SIEM architecture; SIEM management; microservices; Kubernetes; Elasticsearch*

# 1 Introduction

In today's rapidly evolving threat landscape, traditional cybersecurity approaches are increasingly insufficient. Modern cyber threats are not only more sophisticated but also adapt to conventional defense mechanisms, making it critical for organizations to employ security solutions that are both proactive and highly adaptable. This paper presents ARMADA, a novel approach to building SIEM systems. ARMADA, a next-generation SIEM (Security Information and Event Management) developed by Advanced Cyber Security (ACS), addresses these challenges by providing a comprehensive cybersecurity platform that integrates advanced detection, real-time response, and robust analytics. ARMADA is specifically designed to protect critical infrastructure and manage the complexities of modern network environments. Its architecture is built to scale, handling vast quantities of data with precision and efficiency. This scalability allows ARMADA to serve various sectors, from government and finance to industrial IoT and telecommunications, ensuring that its security capabilities meet the unique requirements of each.

One of ARMADA's defining features is its combination of machine learning, behavioral analytics, and big data processing. These elements enable ARMADA to detect both known and unknown threats with a high degree of accuracy. Machine learning algorithms are continuously trained on large datasets, allowing ARMADA to identify subtle patterns and anomalies that may indicate the presence of emerging threats. Behavioral analytics further enhance detection capabilities by establishing baselines for normal activity, helping ARMADA to identify deviations that could signify malicious actions. Big data processing provides the infrastructure needed to handle extensive datasets, enabling ARMADA to deliver comprehensive insights into network behavior and threat landscapes.

ARMADA represents a unique SIEM architecture based on microservices. It is built upon technologies such as Kubernetes, Docker and Elasticsearch. The integration of microservices-based SIEM with Kubernetes and Elasticsearch offers significant advantages but also introduces challenges, such as ensuring consistent security policies across distributed services and managing the complexity of monitoring in highly dynamic environments. The contribution of this paper addresses the challenge of managing the complexity of SIEM architectures [1], and system behavior during threat detection and response.

To enhance its response capabilities, ARMADA integrates with Security Orchestration, Automation, and Response (SOAR) platforms. SOAR integration enables ARMADA to automate and coordinate response actions across various security tools, providing a streamlined and efficient approach to incident management. Through SOAR, ARMADA can initiate automated workflows, alert relevant stakeholders, and execute containment strategies, all in a matter of seconds. This integration not only improves response times but also ensures consistency in

handling incidents, reducing the risk of human error and enhancing overall security posture.

This paper is structured as follows. After the Introduction and Related Work sections, the ARMADA architecture is explained. The ARMADA system's behavior, threat management, and response are evaluated with the system activity analyses during the three common vector attacks. Finally, the conclusions and the further work are presented.

The focus of this paper is on architectural design and functional behavior analysis rather than performance benchmarking. To avoid exceeding the journal's length constraints, the evaluation is intentionally qualitative, while quantitative performance measurements and large-scale experiments are planned as part of future work.

# 2 Related Work

Existing SIEM research addresses scalability, correlation, and anomaly detection through various architectural approaches; however, most solutions rely on monolithic or partially distributed designs, with limited focus on Kubernetes-native microservices and automated response orchestration. In [2] the research focuses on measuring the performance of resource consumption (CPU and RAM) of a complex SIEM system. This system is built on various components such as Elastic (ELK) Stack, Sleeps, and Zeek IDS. In [3] authors presented SIEM system capable of dealing with cyberattacks and anomalies against the Smart Grid application layer protocol. The architecture of SPEAR SIEM consists of the following elements: AlienVault OSSIM SIEM, SPEAR SIEM Basis, message bus, Big Data Analytics Component, the Visual-based Intrusion Detection System (VIDS), and Grid Trusted Module (GTM). The approach of SIEM correlation with self-adaptation is in focus in the article [4], where the correlation engine automatically learns and produces correlation rules based on the context for different types of multi-step attacks using genetic programming. This approach significantly reduced the intervention of operators. An open-source SIEM with incorporated technical measures to protect and control personal data thus ensuring compliance with the GDPR is presented in [5]. The authors used the Elastic Stack components (including Elasticsearch, Logstash, and Kibana) and open-source plugins (e.g. ReadonyRest's Elasticsearch). The topic of GDPR-compliant data processing is the focus of the research published in [6]. The paper [7] deals with the implementation of a hybrid kill-chain framework in SIEM software which resulted in a novel log ontology capable of normalizing security sensor data. The effectiveness of the new configuration was tested against a baseline configuration.

Some SIEM software are focused on a particular type of traffic such as in [8] where the encrypted Skype traffic by using an ad-hoc developed enhanced probe (ESkyPRO). Such an enhanced probe is designed by exploiting some machine learning concepts. A rule generation for TCP SYN flood attack is considered in [9] with RETE algorithm. In [10] the authors presented a contribution to the ontological reasoning approach to correlate alerts and events to achieve decidable reasoning and to reduce the number of alerts, in particular false positives. To deal with the increasing threats in heterogeneous and complex networks, the authors present in [11] the approach based on Latent Semantic Analysis (LSA) to reduce the unnecessary noise in huge data generated from devices.

The approach in log mining using the microservices in presented in [12]. The study has been conducted in the context of a Clearwater IP Multimedia Subsystem. The system consists of microservices deployed in Docker containers and it is applied to real-world critical information systems from the Air Traffic Control domain. Dockerized Elastic Stack for SIEM is presented in [13]. This system uses an Elastic stack which consists of Elasticsearch (ES), Logstash, and Kibana. The paper [14] explores the challenges during deploying and maintaining SIEM systems, as well as the integration of emerging technologies like ML and AI with SIEM for advanced threat detection.

Although there is a variety of research focused on improving the efficiency and operations of SIEM systems, there are still reasons for further improvements. The contribution of this paper addresses the challenge of managing the complexity of SIEM architecture, and system behavior during threat detection and response.

Unlike many existing SIEM solutions, ARMADA addresses limitations of log-centric and ontology-based approaches by combining a Kubernetes-orchestrated microservices architecture with AI-assisted analytics and SOAR-driven automated response for cloud-native environments. While existing research focuses on improving SIEM systems through enhanced correlation techniques, machine learning–based detection, and compliance mechanisms, the Armada redefines the role of SIEM as a coordinated participant within a distributed, context-aware, and orchestrated cyber defense ecosystem, supporting scalable, adaptive, and proactive security operations beyond traditional SIEM-centric designs.

# 3 ARMADA's System Architecture

ARMADA's technical architecture is designed to handle high volumes of data with speed, scalability, and precision. Its architecture integrates several key components that enable it to perform rapid data ingestion, process complex analytics, and provide real-time threat detection and response. By utilizing a microservices-based

design, ARMADA achieves a modular and flexible structure that can scale to meet the demands of various environments. This section provides an in-depth look at ARMADA's architecture, focusing on its microservices-based design, real-time data ingestion capabilities, and advanced analytics powered by big data. The key ARMADA features are:

- Microservices-Based Design
- Real-Time Data Ingestion and Processing
- Advanced Analytics and Big Data Capabilities

The overall microservices-based architecture of the ARMADA platform is illustrated in Figure 1. As shown in Figure 1, ARMADA integrates log ingestion, distributed processing, scalable storage, and security analytics components orchestrated through Kubernetes.

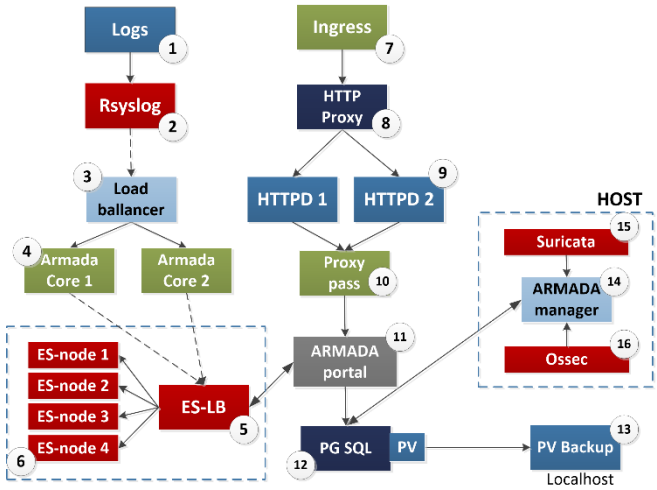


Figure 1
ARMADA system architecture

The system components are as follows. Logs (1) is the receiving interface for logs to be processed within the ARMADA system. Logs from various sources are collected here to ensure all relevant data is captured for security monitoring and analysis. Rsyslog (2) collects logs and forwards them to the Load Balancer for distribution. It serves as a log management tool that gathers log messages from different sources and prepares them for further processing. Load Balancer (3) distributes incoming log traffic between the Armada Core servers. This helps ensure efficient handling of high log volumes and maintains system performance by balancing the load across multiple cores.

Armada Core 1 and Armada Core 2 (4) process log data and send it to the Elasticsearch Load Balancer for indexing. These core components are responsible for initial data analysis, preparing logs for more in-depth examination and storage. Elasticsearch Load Balancer - ES-LB (5) balances requests between the Elasticsearch nodes to manage data storage and retrieval. It ensures that data requests are handled smoothly and that Elasticsearch nodes aren't overloaded, optimizing response times. ES-nodes 1-4 (6) Elasticsearch nodes that store and index log data for analysis and querying. These nodes form the backbone of ARMADA's data storage, enabling quick searches and in-depth data analysis across the indexed logs.

Ingress (7) acts as an interface for web HTTPS access to the ARMADA portal. This component manages secure entry points, allowing authorized users to connect to the ARMADA system safely over the web. HTTP Proxy (8) manages incoming web requests and routes them to the HTTPD servers. It provides an additional layer of security and load management by directing user traffic efficiently to the backend servers. HTTPD 1 and HTTPD 2 (9) handle web server duties and serve the ARMADA portal to users. These servers ensure that the web interface is accessible and responsive, delivering the ARMADA portal's contents to the end-users. Proxy Pass (10) routes traffic from the HTTPD servers to the ARMADA portal. Acting as a bridge, it connects the HTTPD servers to the ARMADA portal, allowing user requests to reach the main application interface seamlessly. ARMADA Portal (11) is the central interface where users interact with the ARMADA system. This portal provides access to monitoring, alerting, and management tools, making it the primary access point for security operations. PostgreSQL (12) provides database storage for the ARMADA system. It stores critical data for the portal, including user information, configurations, and operational logs that support the system's backend. Additionally, it houses alerts and detected threats identified by the ARMADA Manager, ensuring a centralized record of security incidents for analysis and reporting.

PV Backup (13) Maintains backup storage for persistent volumes, hosted on the localhost. This component ensures data persistence and recovery capabilities, safeguarding critical information in case of data loss or corruption. ARMADA Manager (14) Coordinates security modules like Suricata and Ossec on the host system. This manager facilitates communication between different security tools, optimizing the platform's overall security posture. After detecting cyber threats or incidents, any event classified as a threat or incident with a specific severity level is recorded in the PostgreSQL database, ensuring that critical information is documented for further analysis and response. Suricata (15) Provides network security monitoring by inspecting traffic for potential threats. It identifies malicious patterns in real-time, allowing the ARMADA system to detect and respond to network-based attacks swiftly. Ossec (16) Conducts host-based intrusion detection and monitoring within the ARMADA system. Ossec adds an additional layer of

security by monitoring the integrity of files and identifying suspicious behavior directly on the host machine.

Although the presented architecture is described at a conceptual level, the ARMADA system has been implemented and validated in a controlled real-world environment. The platform is deployed on standard x86-64 server infrastructure and orchestrated using Kubernetes, enabling scalable and modular operation. Typical hardware requirements include multi-core CPUs, 64–128 GB of RAM per node, SSD or NVMe storage for high-throughput log ingestion, and optional GPU acceleration for AI-driven analytics modules.

# 4 System Evaluation

The proposed system is evaluated qualitatively through three representative attack vectors, in order to analyze architectural behavior and response workflows under realistic threat scenarios. Three vector attacks selected for the evaluation are Ransomware/Phishing, Denial of Service (DoS/DDoS) and IoT/IIoT Device Exploitation. The first two vector attacks are selected for frequency as shown in [15][16]. The third attack is selected because of IoT/IIoT growth and as a potentially significant threat in the future caused by that growth.

## 4.1 Ransomware attack

Ransomware represents a significant threat in the cybersecurity landscape, characterized by malicious software that encrypts a user's data and demands payment for the decryption key. This form of cyberattack has escalated in prevalence, targeting individuals, corporations, and even critical infrastructure systems. The typical vectors for ransomware infiltration include phishing emails, malicious hyperlinks, and unsecured network connections. Once inside a network, ransomware swiftly encrypts files, effectively locking users out and disrupting normal operations. Compounding the issue, many ransomware strains possess lateral movement capabilities, enabling them to propagate across networks and infect multiple machines, thereby amplifying their destructive potential.

The rapid encryption process and network propagation pose severe risks to organizations, particularly when critical systems or sensitive data are involved. The consequences extend beyond immediate operational disruption; businesses may suffer substantial financial losses, reputational damage, and the irreversible loss of data, especially if backups are compromised or inadequate. Given these high stakes, it is imperative for organizations to deploy advanced security solutions capable of real-time detection and instantaneous mitigation of ransomware threats.

The high-level ransomware attack flow and its interaction with ARMADA detection and response mechanisms are depicted in Figure 2.

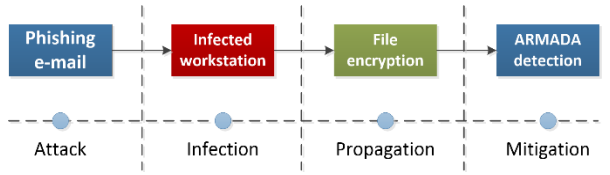


Figure 2
Ransomware attack vector

ARMADA is engineered to counteract ransomware attacks effectively, leveraging its high-speed data processing, minimal latency, and sophisticated analytical capabilities. Its architecture, optimized for sub-three millisecond response times, is crucial for real-time threat identification and classification. Mitigation and enforcement actions are initiated immediately afterward and executed asynchronously, with overall response time depending on the specific containment mechanism.

In the controlled simulation of a ransomware attack within our test environment, ARMADA's advanced capabilities were put to the test. As the ransomware infiltrated the system and initiated its encryption process, ARMADA's real-time monitoring detected an abrupt surge in file access and modification activities. The AI-driven anomaly detection algorithms swiftly recognized these actions as deviations from established normal behavior patterns.

Within milliseconds, ARMADA triggered its automated response protocols:

- **Isolation of Infected Systems:** The affected endpoints were immediately quarantined from the rest of the network, effectively halting the ransomware's ability to propagate to other devices.
- **Termination of Malicious Processes:** Active ransomware processes were identified and terminated, stopping further encryption of files.
- **User Session Management:** Compromised user sessions were suspended to prevent any additional unauthorized activities.

Simultaneously, ARMADA alerted the security operations team with comprehensive incident reports, detailing the nature of the attack, affected systems, and actions taken. This rapid detection and intervention minimized data loss and prevented operational downtime. Critical systems remained unaffected, and backup data integrity was preserved. The successful mitigation of the ransomware attack validated ARMADA's design for real-time threat detection and response, demonstrating its effectiveness in protecting organizational assets against sophisticated cyber threats.

The step-by-step ARMADA response to Ransomware attack is described in Listing 1:

**Listing 1.** ARMADA response to Ransomware Attack

```
Algorithm: ARMADA Response to Ransomware Attack

Input: System and user activity data, network traffic, real-time threat
intelligence
Output: Contained ransomware threat, isolated systems, incident report, and
minimized operational disruption

// Step 1: Anomaly Detection and Machine Learning-Based Pattern Recognition
For each event in system_activity do
    If detect_anomaly(event) then
        log("Anomaly detected in system behavior.")

        // Analyze for ransomware patterns (e.g., sudden surge in file access
or modifications)
        If analyze_for_ransomware_pattern(event) then
            log("Ransomware characteristics identified.")

            // Step 2: Real-Time Data Ingestion and Low-Latency Analysis
            data_stream = ingest_and_analyze_data(system_activity,
network_activity)
            If detect_ransomware_activity(data_stream) then
                log("Confirmed ransomware activity. Initiating response
protocols.")

                // Step 3: Automated Threat Containment and Remediation
                // Quarantining Affected Nodes
                For each node in data_stream.affected_nodes do
                    quarantine_node(node)
                    log("Node quarantined: ", node)
                End For

                // Blocking Malicious IP Addresses or Accounts
                For each malicious_ip in data_stream.malicious_ips do
                    block_ip(malicious_ip)
                    log("Malicious IP blocked: ", malicious_ip)
                End For

                For each compromised_account in
data_stream.suspicious_accounts do
                    suspend_user_session(compromised_account)
                    log("User session suspended: ", compromised_account)
                End For

                // Terminating Malicious Processes
                For each process in data_stream.active_ransomware_processes do
                    terminate_process(process)
                    log("Ransomware process terminated: ", process)
                End For

                // Step 4: Threat Intelligence Integration
                update_threat_intelligence({
                    "new_ransomware_signatures":
detect_new_ransomware_signatures(data_stream),
                    "attack_patterns": data_stream.attack_patterns,
                    "malicious_ips": data_stream.malicious_ips
                })
```

```
                log("Threat intelligence updated with ransomware data.")

                // Step 5: Behavioral Analytics and Baseline Establishment
                update_behavioral_baseline(system_activity)
                log("Behavioral baseline updated for future anomaly
detection.")

                // Step 6: Forensics and Incident Analysis
                forensic_report = generate_forensic_report(
                    incident_details={
                        "origin": data_stream.entry_point,
                        "affected_systems": data_stream.affected_nodes,
                        "attack_path": map_attack_path(data_stream),
                        "compromised_data": data_stream.affected_files
                    },
                    response_actions={
                        "quarantine_nodes": data_stream.affected_nodes,
                        "blocked_ips": data_stream.malicious_ips,
                        "terminated_processes":
data_stream.active_ransomware_processes
                    },
                    recommendations="Strengthen email phishing defenses and
endpoint monitoring"
                )
                log("Forensic report generated for ransomware incident.")

                // Step 7: Reducing Recovery Time and Minimizing Impact
                If recovery_initiated then
                    verify_backup_integrity()
                    restore_affected_files_from_backup()
                    log("System recovery initiated and data restored.")
                End If

                // Security Team Notification
                send_alert_to_security_team(
                    details={
                        "incident_type": "Ransomware",
                        "detected_at": get_current_timestamp(),
                        "affected_nodes": data_stream.affected_nodes,
                        "malicious_ips": data_stream.malicious_ips,
                        "response_actions": ["quarantine",
"terminate_processes", "block_ips"]
                    }
                )
                log("Security team notified with detailed incident report.")
            End If
        End If
    End If
End For
```

## 4.2 Distributed Denial of Service (DDoS) Attack

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service, or network by overwhelming it with a flood of internet traffic. This is achieved by utilizing multiple compromised computer systems as sources of attack traffic, often forming a botnet—a network of

hijacked devices connected to the internet. These devices could range from personal computers to IoT devices like cameras and smart appliances.

The sheer volume of incoming messages, connection requests, or malformed packets sent to the target system forces it to slow down or crash, denying service to legitimate users. DDoS attacks can have profound impacts on organizations, leading to prolonged downtime, loss of revenue, and damage to reputation. In some cases, they are used as a smokescreen for more insidious activities, such as breaching security perimeters to extract sensitive data.

Traditional defense mechanisms often struggle to cope with the scale and complexity of modern DDoS attacks. Attackers continuously evolve their strategies, employing techniques like amplification and reflection to magnify the impact. This necessitates a dynamic and robust defense system capable of real-time detection and mitigation.

Figure 3 presents the conceptual attack vector of a distributed denial-of-service (DDoS) attack used in the evaluation scenario.

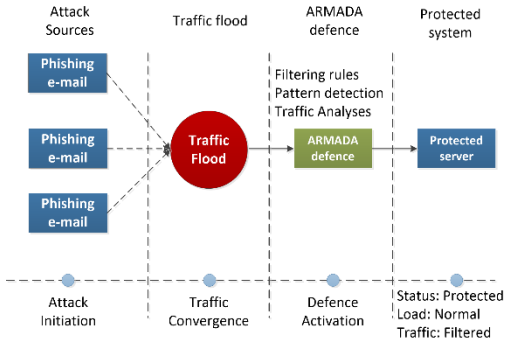


Figure 3
Distributed Denial of Service (DDoS) attack vector

The ARMADA system is architected to effectively counter DDoS attacks through its scalable infrastructure, advanced analytics, and automated response capabilities. Its design ensures that service availability is maintained even under the stress of a large-scale attack.

**Scalability and Resilience through Kubernetes:**

ARMADA leverages Kubernetes for orchestration, enabling it to scale resources dynamically in response to fluctuating network demands. During a DDoS attack, the system automatically provisions additional resources to handle the surge in traffic. This elasticity ensures that critical services remain operational, and legitimate user access is preserved.

Kubernetes also aids in load balancing, distributing network traffic efficiently across multiple nodes. By preventing any single node from becoming a bottleneck, ARMADA maintains optimal performance levels. This horizontal scaling capability is crucial in absorbing the impact of volumetric DDoS attacks that aim to saturate network bandwidth.

**Real-Time Monitoring and Machine Learning Algorithms:**

At the heart of ARMADA's defense strategy is its real-time monitoring system, which continuously analyzes network traffic patterns. Using sophisticated machine learning algorithms, ARMADA establishes a baseline of normal network behavior, including typical traffic volumes, user access patterns, and data flow characteristics.

When a DDoS attack commences, the sudden spike in traffic deviates significantly from the established baseline. ARMADA's anomaly detection algorithms quickly identify these aberrations. The machine learning models are trained to distinguish between legitimate traffic bursts—such as those occurring during peak business hours or promotional events—and malicious traffic surges indicative of a DDoS attack.

**Automated Response with SOAR Integration:**

Upon detection of abnormal traffic patterns, ARMADA integrates with Security Orchestration, Automation, and Response (SOAR) systems to initiate an automated defense. The response protocols are designed to mitigate the attack swiftly while minimizing the impact on legitimate users.

Key automated responses include:

- **Traffic Filtering and Rate Limiting:** ARMADA implements filters to scrutinize incoming traffic, identifying and discarding malicious packets. Rate limiting is applied to suspicious IP addresses, throttling the flow of requests to manageable levels without affecting genuine traffic.
- **Blacklisting Malicious IPs:** The system maintains and updates a blacklist of IP addresses known to be sources of malicious activity. ARMADA dynamically adds offending IPs detected during the attack to this list, blocking further traffic from these sources.
- **Geo-Blocking and IP Reputation Analysis:** If the attack originates from specific geographic regions known for malicious activity, ARMADA can enforce geo-blocking measures. Additionally, it assesses IP reputation scores to make informed decisions about allowing or denying traffic.
- **Traffic Diversion and Sinkholing:** In some scenarios, ARMADA can redirect malicious traffic to sinkholes—servers designed to absorb and analyze attack traffic. This not only protects the network but also gathers valuable intelligence on the attack vectors and methods used.

**Listing 2.** ARMADA Response to Distributed Denial of Service (DDoS) Attack

```
Algorithm: ARMADA Response to Distributed Denial of Service (DDoS) Attack

Input: Network traffic data, system activity logs, threat intelligence feeds
Output: Mitigated DDoS threat, preserved service continuity, incident report

// Step 1: Scalability and Resilience through Kubernetes
If detect_ddos_attack(network_traffic) then
    log("DDoS attack detected. Initiating response protocols.")

    // Dynamic Resource Scaling
    scale_resources_dynamically()
    log("Resources scaled up to handle increased traffic load.")

    // Load Balancing Across Nodes
    balance_network_traffic()
    log("Network traffic load balanced across nodes.")

    // Step 2: Real-Time Monitoring and Machine Learning Algorithms
    For each traffic_event in network_traffic do
        If detect_anomalous_traffic(traffic_event) then
            log("Anomalous traffic detected.")

            If classify_as_ddos(traffic_event) then
                log("Traffic classified as DDoS. Initiating mitigation
steps.")

                // Step 3: Automated Response with SOAR Integration

                // Traffic Filtering and Rate Limiting
                For each ip in detected_suspicious_ips do
                    apply_rate_limiting(ip)
                    log("Rate limiting applied to IP: ", ip)
                End For

                // Blacklisting Malicious IPs
                For each ip in detected_malicious_ips do
                    add_to_blacklist(ip)
                    block_ip(ip)
                    log("Malicious IP blocked and blacklisted: ", ip)
                End For

                // Geo-Blocking and IP Reputation Analysis
                For each region in
identify_suspicious_geographies(network_traffic) do
                    If region_reputation_score(region) < threshold then
                        apply_geo_blocking(region)
                        log("Geo-blocking enforced for region: ", region)
                    End If
                End For

                // Traffic Diversion and Sinkholing
                If diversion_required then
                    divert_to_sinkhole(network_traffic)
                    log("Malicious traffic diverted to sinkhole.")
                End If

                // Step 4: Threat Intelligence and Collaborative Defense
                update_threat_intelligence({
                    "blacklisted_ips": detected_malicious_ips,
```

```
                    "attack_patterns":
analyze_attack_patterns(network_traffic)
                })
                log("Threat intelligence updated with DDoS attack data.")

                // Step 5: Behavioral Analytics and Adaptive Learning
                update_network_baseline(traffic_patterns)
                retrain_ml_models()
                log("Behavioral baseline and ML models updated for adaptive
learning.")

                // Step 6: Forensic Analysis and Post-Attack Assessment
                forensic_report = generate_forensic_report(
                    incident_details={
                        "attack_type": "DDoS",
                        "detected_at": get_current_timestamp(),
                        "source_ips": detected_malicious_ips,
                        "traffic_patterns":
analyze_attack_patterns(network_traffic)
                    },
                    response_measures={
                        "rate_limiting_ips": detected_suspicious_ips,
                        "blacklisted_ips": detected_malicious_ips,
                        "geo_blocked_regions":
identify_suspicious_geographies(network_traffic)
                    },
                    recommendations="Enhance load balancing and traffic
filtering capabilities"
                )
                log("Forensic report generated with post-attack analysis.")

                // Step 7: Security Team Notification
                send_alert_to_security_team(
                    details={
                        "incident_type": "DDoS",
                        "detected_at": get_current_timestamp(),
                        "affected_services":
identify_affected_services(network_traffic),
                        "response_actions": ["scale_resources",
"rate_limiting", "blacklist_ips", "sinkholing"]
                    }
                )
                log("Security team notified with detailed incident report.")

            Else
                log("Traffic does not match DDoS characteristics.")
            End If
        Else
            log("No anomaly detected in network traffic.")
        End If
    End For
Else
    log("No DDoS attack detected.")
End If
```

The functions add_to_blacklist(ip) and block_ip(ip) represent abstracted automated response actions. In practical deployments, these functions are implemented using standard host-level and network-level security mechanisms, such as dynamic firewall rule updates (e.g., iptables or nftables on Linux-based systems), Kubernetes network policies for containerized environments, or SOAR-integrated perimeter

security devices. The exact implementation is deployment-specific but relies on widely adopted and well-established technologies.

## 4.3 IoT/IIoT Device Exploitation

The advent of the Internet of Things (IoT) and Industrial Internet of Things (IIoT) has interconnected a vast array of devices, from smart sensors to industrial control systems. While this connectivity offers numerous operational benefits, it also introduces significant security vulnerabilities. Many IoT/IIoT devices are designed with minimal security features, often lacking robust authentication mechanisms or relying on default credentials. This makes them attractive targets for attackers seeking to exploit these weaknesses to gain unauthorized access, disrupt services, or use the devices as entry points into larger networks.

In our test scenario, we simulated an attack where a threat actor targets a vulnerable IIoT sensor within an industrial control system. The attacker exploits a known firmware vulnerability to gain control over the device. Once compromised, the device is used to intercept sensitive data, manipulate operational parameters, and establish a foothold for lateral movement within the network. Such exploitation could lead to significant operational disruptions, safety hazards, and financial losses, particularly in critical infrastructure environments.

The IoT/IIoT device exploitation scenario and the corresponding attack vector are illustrated in Figure 4.

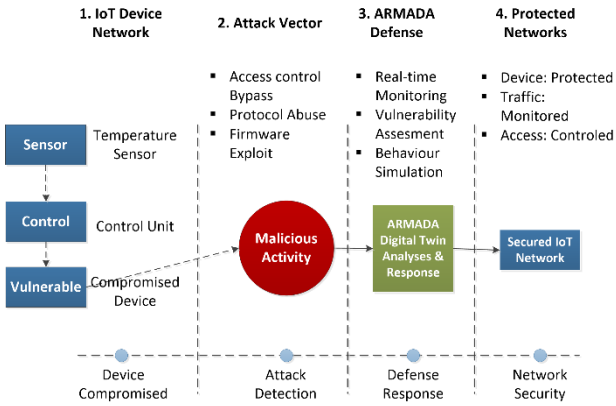


Figure 4
IoT/IIoT device exploitation vector attack

ARMADA employed its innovative Digital Cloning feature to proactively defend against the simulated IoT/IIoT exploitation. Digital Cloning involves creating virtual replicas of physical devices within a controlled environment. These clones

emulate the behavior and responses of the actual devices, allowing ARMADA to safely simulate potential attacks and analyze their impact without risking real assets.

As the attacker attempted to exploit the device vulnerability, ARMADA's real-time monitoring detected unusual activity. The system's predictive analytics, powered by advanced machine learning algorithms, identified anomalies in the device's behavior. Indicators such as unexpected firmware modifications, irregular communication patterns, and unauthorized command executions were flagged immediately.

Upon detection, ARMADA initiated a series of automated response actions:

- **Isolation of the Compromised Device:** The affected device was promptly isolated from the network to prevent the attacker from accessing other systems or exfiltrating data.
- **Firmware Integrity Verification:** ARMADA conducted an integrity check on the device's firmware, confirming unauthorized alterations indicative of a compromise.
- **Threat Intelligence Integration:** Information about the attack vector and the exploited vulnerability was fed into ARMADA's threat intelligence database, enhancing its ability to recognize and respond to similar threats in the future.
- **Security Team Notification:** Detailed alerts were sent to the security operations team, providing comprehensive insights into the attack, including timestamps, the nature of the exploitation, and recommended remediation steps.

The ARMADA response to the IoT/IIoT is presented in Listing 3.

**Listing 3.** ARMADA Response to IoT/IIoT Device Exploitation

```
Algorithm: ARMADA Response to IoT/IIoT Device Exploitation

Input: Device activity data, network activity, firmware version
Output: Mitigated threat, isolated device, firmware verification, threat
intelligence update

// Step 1: Digital Cloning for Proactive Defense
create_digital_clone(vulnerable_device)
log("Digital clone created for proactive threat simulation.")

simulate_attack_on_clone()
If detect_exploitation_in_clone() then
    log("Exploitation detected in digital clone. Monitoring real device
closely.")

    // Step 2: Real-Time Monitoring and Anomaly Detection
    For each activity in device_activity do
```

```
        If detect_anomalous_behavior(activity) then
            log("Anomaly detected in IoT/IIoT device activity.")

            If detect_firmware_modification(activity) or
detect_unauthorized_commands(activity) then
                flag_for_investigation(activity)
                log("Anomaly flagged: unauthorized firmware modification or
command execution.")

                // Step 3: Automated Response Actions
                // Isolation of the Compromised Device
                isolate_device_from_network(vulnerable_device)
                log("Compromised device isolated from network.")

                // Firmware Integrity Verification
                firmware_status = verify_firmware_integrity(vulnerable_device)
                If firmware_status == "compromised" then
                    log("Firmware integrity compromised on device.")
                Else
                    log("Firmware integrity verified as intact.")
                End If

                // Threat Intelligence Integration
                update_threat_intelligence_database({
                    "attack_vector": "IoT/IIoT Device Exploitation",
                    "vulnerability": exploited_vulnerability,
                    "indicators": get_attack_indicators(activity)
                })
                log("Threat intelligence database updated with IoT/IIoT attack
data.")

                // Security Team Notification
                send_alert_to_security_team(
                    details={
                        "incident_type": "IoT/IIoT Device Exploitation",
                        "detected_at": get_current_timestamp(),
                        "device": vulnerable_device,
                        "exploit_details": activity,
                        "recommended_action": "Patch device firmware and reset
configurations"
                    }
                )
                log("Security team notified with incident details.")

            Else
                log("No unauthorized modifications detected in device
behavior.")
            End If
        Else
            log("No anomaly detected in device behavior.")
        End If
    End For
Else
    log("No exploitation detected in digital clone. Device secure.")
End If

// Step 4: Post-incident Analysis and Reporting
If incident_resolved then
    forensic_report = generate_forensic_report(
        incident_details={
            "device": vulnerable_device,
            "exploit_type": "IoT/IIoT Device Exploitation",
```

```
            "response_actions": ["isolation", "firmware verification", "threat
intelligence update"]
        },
        recommendations="Strengthen authentication mechanisms and monitor
firmware updates."
    )
    log("Forensic report generated with post-incident analysis.")
End If
```

# 5 Conclusions and Future Work

ARMADA represents a significant advancement in SIEM (Security Information and Event Management) technology, providing a robust and scalable solution that is capable of addressing both current and emerging cyber threats. By combining a modular architecture with AI-driven analytics, ARMADA delivers real-time threat detection, response, and mitigation across a wide range of industries, from finance and telecommunications to government and critical infrastructure. ARMADA's adaptability, speed, and comprehensive security features make it an invaluable tool for organizations that face complex and evolving cybersecurity challenges.

ARMADA's proactive approach to cybersecurity, which emphasizes early detection and immediate response, sets it apart from traditional, reactive SIEM solutions. The platform's modular design not only allows it to scale seamlessly but also enables organizations to deploy specific functions as needed, providing flexibility that is essential in today's dynamic threat landscape. ARMADA's integration of machine learning, behavioral analytics, and big data processing empowers it to detect both known and unknown threats with high precision, positioning it as a next-generation solution in cybersecurity.

The complex and robust architecture of ARMADA is presented in this paper. ARMADA's complexity is present in its elements (Microservices, Elasticsearch, Kubernetes, Docker containers) and its integration with IDS. The approach to their interconnection and cooperation is explained in detail in this paper as a proposal for efficient management of multi-component and complex SIEM systems designed to monitor complex environments. The efficiency of the system and the component coordination are evaluated using three common attack vectors. The vector attack and system's response are described in detail and prove the system's efficiency.

Looking forward, future developments for ARMADA will focus on expanding its automated response capabilities. By enhancing automation, ARMADA aims to reduce the time between detection and response further, minimizing human intervention and enabling organizations to handle large-scale threats efficiently. ARMADA's automated response features will continue to integrate with SOAR (Security Orchestration, Automation, and Response) platforms, supporting complex, multi-step workflows that enhance incident management and streamline security operations.

Another area of development is ARMADA's AI capabilities. The platform will continue to leverage advanced machine learning algorithms, focusing on predictive analytics to anticipate potential threats before they manifest. By incorporating predictive models, ARMADA will enable organizations to adopt a more preventive security stance, reducing the likelihood of incidents and strengthening overall resilience. The continued integration of AI will enhance ARMADA's threat prediction and mitigation abilities, ensuring that it remains at the forefront of cybersecurity technology.

# References

[1] David R. Miller, Shon Harris, Allen Harper, Stephen VanDyke, Chris Blask, Security Information and Event Management (SIEM) Implementation, 1st ed, The Mcgraw-Hill Education, 2010.

[2] Adabi Raihan Muhammad, Parman Sukarno, Aulia Arif Wardana, Integrated Security Information and Event Management (SIEM) with Intrusion Detection System (IDS) for Live Analysis based on Machine Learning, Procedia Computer Science, Vol. 217, 2023, pp 1406-1415, https://doi.org/10.1016/j.procs.2022.12.339.

[3] Panagiotis Radoglou-Grammatikis, Panagiotis Sarigiannidis, Eider Iturbe, Erkuden Rios, Saturnino Martinez, Antonios Sarigiannidis, Georgios Eftathopoulos, Yannis Spyridis, Achilleas Sesis, Nikolaos Vakakis, Dimitrios Tzovaras, Emmanouil Kafetzakis, Ioannis Giannoulakis, Michalis Tzifas, Alkiviadis Giannakoulias, Michail Angelopoulos, Francisco Ramos, SPEAR SIEM: A Security Information and Event Management system for the Smart Grid, Computer Networks, Vol. 193, 2021, 108008, https://doi.org/10.1016/j.comnet.2021.108008.

[4] Guillermo Suarez-Tangil, Esther Palomar, Arturo Ribagorda, Ivan Sanz, Providing SIEM systems with self-adaptation, Information Fusion, Vol. 21, 2015, pp 145-158, https://doi.org/10.1016/j.inffus.2013.04.009.

[5] Ana Paula Vazão, Leonel Santos, Rogério Luís de C. Costa, Carlos Rabadão, Implementing and evaluating a GDPR-compliant open-source SIEM solution, Journal of Information Security and Applications, Vol. 75, 2023, 103509, https://doi.org/10.1016/j.jisa.2023.103509.

[6] Florian Menges, Tobias Latzo, Manfred Vielberth, Sabine Sobola, Henrich C. Pöhls, Benjamin Taubmann, Johannes Köstler, Alexander Puchta, Felix Freiling, Hans P. Reiser, Günther Pernul, Towards GDPR-compliant data processing in modern SIEM systems, Computers & Security, Vol. 103, 2021, 102165, https://doi.org/10.1016/j.cose.2020.102165.

[7] Blake D. Bryant, Hossein Saiedian, Improving SIEM alert metadata aggregation with a novel kill-chain based classification model, Computers & Security, Vol. 94, 2020, 101817, https://doi.org/10.1016/j.cose.2020.101817.

[8] Mario Di Mauro, Cesario Di Sarno, Improving SIEM capabilities through an enhanced probe for encrypted Skype traffic detection, Journal of Information Security and Applications, Vol. 38, 2018, Pages 85-95, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2017.12.001.

[9] M. Siva Niranjan Raja, A.R. Vasudevan, Rule Generation for TCP SYN Flood attack in SIEM Environment, Procedia Computer Science, Vol. 115, 2017, Pages 580-587, https://doi.org/10.1016/j.procs.2017.09.117.

[10] Tayeb Kenaza, Mahdi Aiash, Toward an Efficient Ontology-Based Event Correlation in SIEM, Procedia Computer Science, Vol. 83, 2016, pp 139-146, https://doi.org/10.1016/j.procs.2016.04.109.

[11] P. Dairinram, D. Wongsawang and P. Pengsart, "SIEM with LSA technique for Threat identification," 2013 19th IEEE International Conference on Networks (ICON), Singapore, 2013, pp. 1-6, doi: 10.1109/ICON.2013.6781951.

[12] Marcello Cinque, Raffaele Della Corte, Antonio Pecchia, Micro2vec: Anomaly detection in microservices systems by mining numeric representations of computer logs, Journal of Network and Computer Applications, Vol. 208, 2022, 103515,https://doi.org/10.1016/j.jnca.2022.103515.

[13] F. Mulyadi, L. A. Annam, R. Promya and C. Charnsripinyo, "Implementing Dockerized Elastic Stack for Security Information and Event Management," 2020 - 5th International Conference on Information Technology (InCIT), Chonburi, Thailand, 2020, pp. 243-248, doi: 10.1109/InCIT50588.2020.9310950.

[14] Noyan Tendikov, Leila Rzayeva, Bilal Saoud, Ibraheem Shayea, Marwan Hadri Azmi, Ali Myrzatay, Mohammad Alnakhli, Security Information Event Management data acquisition and analysis methods with machine learning principles, Results in Engineering, Vol. 22, 2024, 102254, https://doi.org/10.1016/j.rineng.2024.102254.

[15] Europol (2023), Cyber-attacks: the apex of crime-as-a-service, Europol Spotlight Report series, Publications Office of the European Union, Luxembourg.

[16] Yuchong Li, Qinghui Liu, A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments, Energy Reports, Vol. 7, 2021, pp 8176-8186, https://doi.org/10.1016/j.egyr.2021.08.126.