Journal of Electrical Engineering, Electronics, Control and Computer Science – JEEECCS, Volume 11, Issue 39, pages 11-22, 2025

Emerging Cybersecurity Threats in Embedded Systems: A Review of Attack Techniques, Anomaly Detection, and AI-Based Prediction Approaches

1st Drăgușin Sebastian-Alexandru
Department of Electronics, Computers and Electrical Engineering
National University of Science and Technology Polyethnic Bucharest
Pitesti, Romania
dragusin.sebi@yahoo.com

2nd Bizon Nicu
Department of Electronics, Computers and Electrical Engineering
National University of Science and Technology Polyethnic Bucharest
Pitesti, Romania
nicubizon@yahoo.com

Abstract –Embedded systems have become fundamental to modern technological infrastructures, powering applications from smart vehicles and medical devices to critical industrial control. However, their rapid integration into the IoT (Internet of Things) ecosystem has significantly expanded the attack surface, exposing them to a wide range of cybersecurity threats. This paper provides a structured review of attack techniques targeting embedded systems, including fuzzing, reconnaissance, shellcode injection, denial-of-service, and backdoor exploitation. Furthermore, it discusses the role of AI (Artificial Intelligence) in early anomaly detection and predictive threat modeling. By comparing traditional and AI-enhanced security mechanisms, the paper highlights both the advantages and limitations of current defense strategies. The analysis emphasizes the growing importance of adaptive, data-driven models capable of operating under resource-constrained embedded environments, proposing a synthesis of theoretical and practical advances in securing embedded architectures.

Keywords- embedded systems, cybersecurity, attack taxonomy, anomaly detection, artificial intelligence, predictive models, IoT security

I. INTRODUCTION

The security of embedded systems has become a topic of major importance in today's technological context. Embedded systems, defined as microprocessor-controlled devices that perform specific functions in a larger ensemble, are ubiquitous in our daily lives. They are found in medical devices, vehicles, household appliances, industrial equipment and critical infrastructures, thus influencing many aspects of modern society [1], [2].

In recent years, the number of embedded devices connected to global networks has grown exponentially due to the rise of IoT. It is estimated that by the end of 2025 there will be more than 75 billion devices connected to the internet, offering multiple advantages but at the same time expanding the attack surface for cybercriminals. This massive increase in connectivity brings with it significant challenges in terms of the security of these systems [3].

Embedded systems are essential for the operation of critical infrastructures such as power grids, transportation systems, and communications. A cyberattack on these systems can have devastating consequences, from service interruptions to massive economic losses and risks to public safety. These systems are often designed for very specific tasks and operate in resource-constrained environments. This specific complexity makes security difficult to implement and maintain, requiring customized solutions and a deep understanding of the operational context [1], [2].

The security of embedded systems is essential for data protection and privacy. Embedded devices collect and manage large amounts of data, including sensitive personal information. Ensuring the security of this data is crucial for preventing unauthorized access and protecting the confidentiality of information. In many critical applications, such as medical or industrial, the reliability of embedded systems is vital. Cybersecurity helps ensure that these systems operate continuously and without interruption, preventing failures that can have serious consequences [1], [2].

Embedded systems are attractive targets for cyberattackers who can exploit vulnerabilities to take control, steal data, or sabotage the operation of devices. Implementing robust security measures is crucial to

prevent these attacks and protect the integrity of systems. However, implementing security in embedded systems involves several unique challenges. Resource limitations, such as processing capacity and energy consumption, require efficient and optimized security solutions. Also, the rapid evolution of technologies and cyber threats requires constant adaptability and regular updates of protection mechanisms [4].

The use of AI in anomaly detection and forecasting is an innovative and essential approach in the modern context of cybersecurity and embedded systems. Anomalies, defined as deviations from the expected behavior of a system, can indicate the presence of malfunctions, errors or even cyberattacks. Early and accurate detection of these anomalies is crucial for maintaining the integrity and reliability of critical systems [5], [6].

One of the main motivations for adopting AI in this field is its ability to process and analyze large volumes of data in real-time. Embedded systems are constantly generating operational data, and manually analyzing this data would be extremely laborious and inefficient. ML (Machine Learning) algorithms can identify complex patterns and detect anomalies with much greater accuracy and speed than traditional methods [6].

AI also provides flexibility and adaptability in the face of an ever-changing threat landscape. ML models can be trained to recognize new types of attacks or anomalies based on historical data and recent developments. It allows for continuous updating and improvement of detection mechanisms, providing a significant advantage in the fight against emerging threats [6].

Another key aspect of using AI is its ability to forecast anomalies before they fully manifest. Predictive algorithms, such as neural networks and regression models, can predict potential problems based on trends and patterns observed in operating data. This allows for proactive interventions and preventive measures, minimizing the negative impact on the system and reducing downtime [5].

AI is also helping to reduce false alarms, which are a common problem in anomaly detection systems. By continuously refining algorithms and using advanced classification techniques, AI can more effectively differentiate between legitimate activities and suspicious behaviors, thereby improving the accuracy and relevance of the alerts generated [7].

In Figure 1, the number of papers indexed by Scopus in the period 2021-2025 on the topic of anomaly detection in security systems is graphically presented.

TITLE-ABS-KEY (anomaly AND detection AND security AND systems)

In addition to a theoretical introduction, this paper provides in Section II a comprehensive literature review of recent research addressing cyberattack techniques and AI-based anomaly detection in embedded systems. Section III presents the theoretical foundations underlying cybersecurity mechanisms and modeling principles, while Section IV details the concepts of anomalies and malformations within cyberattack models, emphasizing their detection and interpretation in intelligent embedded contexts. Finally, paper outlines the main conclusions and future research directions, summarizing key findings and identifying open challenges in securing embedded architectures.

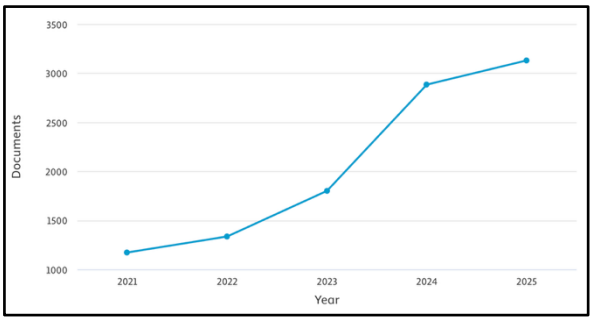

Figure 1. Papers published in 2021-2025 on the detection of anomalies in security systems

II. Literature Review

[8] provides an extensive synthesis of embedded device security analysis, describing the dynamic vs. static landscape (with a focus on QEMU (The Quick Emulator) emulation and web interface fuzzing), the issue of re-hosting monolithic and kernel-level firmware, as well as the current challenges—end-to-end automation, low dynamic analysis throughput, and scaling to large batches of firmware—thus outlining concrete directions for future research.

The authors [9] conduct a comprehensive survey of anomaly detection in IoT, proposing a multi-layer taxonomy (types/sources of anomalies, learning frameworks), comparing computing platforms (cloud, fog, edge, hybrid) and methodologies (information-theory, graph-/spectral-based, blockchain, ML/DL (Deep Learning), AE (Autoencoder), RNN (Recurrent Neural Network)/LSTM (Long Short-Term Memory), CNN (Convolutional Neural Network), GAN (Generative Adversarial Network), plus evolutionary, RL (Reinforcement Learning) and ensemble), and systematizing datasets (NSL-KDD (Network Security Laboratory-Knowledge Discovery Dataset), BoT-IoT (Botnet Internet of Things Dataset), CICIDS2017 (Canadian Institute for Cybersecurity Intrusion Detection System 2017), UNSW-NB15 (University of New South Wales Network-Based Dataset 2015), Yahoo Webscope) and metrics (Precision, Detection Rate, FPR (False Positive Rate), ROC-AUC (Receiver Operating Characteristic-Area Under Curve), F1). The paper highlights challenges (reducing complexity, environmental/architectural constraints, privacy and interpretability, lack of adequate datasets) and future directions (XAI (Explainable Artificial Intelligence), edge-intelligence, self/weak supervision, Transformers, digital twin).

The paper [10] conducts an analytical review of the security of embedded systems, identifying 12 factors influencing CSES (Cybersecurity of Embedded Systems) – from features, implementation and connectivity protocols, to attack surfaces, impact and actors – and proposes the 9-layer MuLFESC (Multi-Layered Framework for Embedded Systems Cybersecurity) framework for security-by-design and

new risk metrics, providing a coherent roadmap for assessing and strengthening defense at the hardware–software–network level.

In the paper [11] a systematic literature mapping on AIoT/TinyML for the detection of anomalies on the MCU (Microcontroller Unit) is proposed, synthesizing 18 studies (2021–2023) and providing a taxonomy of algorithms (CNN, Autoencoder, LSTM, GMM (Gaussian Mixture Model), IF (Isolation Forest), TEDA (Typicality and Eccentricity Data Analysis), HyBNN (Hybrid Bayesian Neural Network), LSH (Locality-Sensitive Hashing)), methods/metrics (Accuracy, F1, AUC, RMSE (Root Mean Square Error)), platforms (Raspberry Pi, STM32, ESP32, Arduino, Jetson) and architectures (edge/fog/cloud), with benefits (low latency, increased privacy) and notable gaps (lack of dataset standardization, modest power reporting, absence of LoRaWAN (Long Range Wide Area Network)/5G connections, few real-time deployments).

Paper [12] conducts a comprehensive review of CPS (Cyber-Physical Systems) security in the context of 5G and Beyond 5G technologies, analyzing the architecture, threats, and security solutions associated with authentication, access, and encryption. The paper describes a complete taxonomy of attacks (DoS (Denial of Service), spoofing, data tampering, replay, injection), includes CPS threat models based on ISO (International Organization for Standardization)/IEC (International Electrotechnical Commission) 27001:2013 and presents a synthesis of the most important industrial incidents (Stuxnet, Ukraine Attack, Jeep Hack). The authors propose a multi-dimensional analysis framework: cyber-attack emanation, prototyping, detection and architecture design and correlate these aspects with emerging technologies (AI, blockchain, PUF (Physically Unclonable Function), ML/DL) for securing 5G/6G-enabled CPS applications, outlining future research directions related to physical-layer security, adaptive AI defenses and low-latency protection mechanisms.

A comprehensive survey of deep learning models for anomaly detection is presented in [13], covering architectures (CNN/RNN/LSTM, Autoencoders, GAN), standard datasets, and evaluation metrics, describing IDS (Intrusion Detection System) taxonomies (HIDS (Host-based Intrusion Detection System)/NIDS (Network-based Intrusion Detection System)), adversarial attacks, an end-to-end detection pipeline, and current challenges (generalization, class imbalance, interpretability), with future directions towards hybrid models, hardware acceleration, and XAI.

III. Theoretical Foundations

In this section, the main attack techniques used in embedded systems are presented. These attacks, which include *Fuzzers*, *Analysis*, *Backdoors*, *DoS*, *Exploits*, *Generic*, *Reconnaissance*, *Shellcode*, and *Worms*. Understanding these techniques is essential to ensuring the security and integrity of computer systems. Each of these techniques represent specific methods by which attackers can compromise systems, either by exploiting software vulnerabilities or by infiltrating and manipulating them for malicious purposes. The analysis of these techniques allows the development of more effective defense strategies and the implementation of appropriate preventive measures in the face of cyber threats.

A. Fuzzers Attacks

A "Fuzzing" attack is a method used in cybersecurity to test and discover vulnerabilities in software. This process involves sending randomly generated or deliberately invalid data to applications, protocols, or systems to cause errors or unexpected behaviors that may reveal exploitable bugs [14], [15], [16].

Fuzzing attacks (types) can be classified as follows:

- **Mutational Fuzzing**: It modifies existing data in a random way to create new and unexpected inputs. For example, it can modify data packages or input files to test the robustness of the application [17].
- **Generative Fuzzing**: Builds entirely new inputs based on protocol or data format specifications. Especially useful for testing applications with structured inputs, such as network protocols or file formats [18].
- **Protocols Fuzzing**: Focuses on sending invalid or altered data packets to applications that use network protocols. An example is sending modified TCP/IP (Transmission Control Protocol/Internet Protocol) packets to test network stacks [19].
- **Fuzzing file formats**: Create and open corrupted files to test the parsers and decoders of the applications that process these files. For example, a fuzzer can generate malformed JPEG image files to test the applications that open them [20].

Fuzzing provides a simple and effective test design, requiring no detailed knowledge of the internal behavior of the tested system, which makes it recommended for "black box" testing. The method is particularly useful for identifying unexpected errors and exploitable bugs that can escape manual testing, helping to quickly discover vulnerabilities in firmware and protocol stacks. In contexts where the system is closed or incubated in a heterogeneous environment, fuzzing remains one of the most practical automated techniques for exploring the attack surface [14], [15], [16].

Despite its usefulness, fuzzing tends to identify mainly simple errors and does not always assess the full impact of discovered vulnerabilities; Many detected cases require further analysis to determine actual exploitability. Also, to achieve adequate coverage of the input space, fuzzing can require significant computational resources and time, while generating a high volume of false alarms that require manual triage and prioritization effort [14], [15], [16].

Among the most popular fuzzing tools are SPIKE and Wireshark, which are used to test network protocols

and applications. SPIKE, for example, is a fuzzer creation kit based on the C programming language, which allows the generation of "fuzzed" messages to induce errors in network services [21], [22], [23].

B. Analysis Attacks

The "Analysis" attack, in the context of cybersecurity, refers to different methods and techniques used to analyze computer systems to identify vulnerabilities, abnormal behavior and potential attacks. This category encompasses various methods, from analyzing source code to monitoring network traffic and evaluating executable behavior [24], [25].

The types of Analysis in cybersecurity are as follows:

- **Static analysis** is the process of examining source code or binaries without executing them, to identify vulnerabilities, logical errors, or deviations from security standards. This method is based on the inspection of syntax, data structures and control flows, and is used in the early stages of development to prevent exploitation in production environments. Through automated static analysis tools, critical vulnerabilities such as buffer overflow, SQL (Structured Query Language) injection or XSS (Cross-Site Scripting) can be detected, making it essential in the security-by-design process and in the evaluation of the code before the final integration of the system [26].
- **Dynamic analysis** consists of examining the behavior of a program during its execution, tracking how it uses system resources and interacts with other hardware or software components. Unlike static analysis, this method allows you to identify vulnerabilities and bugs that only manifest themselves during runtime, such as runtime issues, heap spray, or malicious behavior hidden in executables. By monitoring performance, memory consumption, and execution flow, dynamic analytics provides a practical insight into application stability and security and is essential for validating protection mechanisms in real-world environments [27].
- **Behavioral analysis** is based on observing and interpreting the actions of users, processes, and components of a system, with the aim of identifying unusual or suspicious activity that may indicate a compromise. This method is essential for detecting attacks that cannot be recognized by static signatures, such as zero-day attacks or subtle behavioral exploits. By monitoring access logs, analyzing network traffic, and applying machine learning algorithms to recognize deviant patterns, behavioral analytics provides an additional layer of proactive protection, enabling early identification of emerging threats and continuous adaptation of security policies [28].
- **Forensic analysis** involves the systematic collection, preservation and examination of digital evidence after a security incident, with the aim of reconstructing the sequence of events, identifying attack vectors and determining the extent of the compromise. This includes techniques such as examining volatile memory, analyzing files on disks, investigating network artifacts, and interpreting system logs to reconstruct the attacker's steps and the mechanisms used. The main advantage lies in the ability to provide actionable evidence for remediation and improvement of defensive strategies identifying weaknesses and persistence mechanisms as well as supporting legal procedures where appropriate. Practical examples include extracting memory images for rootkit detection, analyzing system timelines to correlate events, and investigating captured packets to track data exfiltration channels [29].

C. Backdoors Attacks

"Backdoors" attacks are methods by which an attacker creates or uses a hidden way of accessing a computer system, bypassing standard authentication and security mechanisms. These attacks are particularly dangerous because they allow continuous unauthorized access that is difficult to detect and remove [30].

The associated definitions and types of Backdoors are presented as follows:

- **Classic backdoors** are deliberate mechanisms or compromise results that provide hidden access to a system, being integrated either at the software level (source code, modules, components from installation packages) or at the hardware level (microcode, IP (Internet Protocol) compromised blocks). They can be intentionally introduced by malicious developers or later inserted by exploiting vulnerabilities or in the supply chain, ensuring persistence and a discreet path for attackers. Typical examples include commands hidden in the code, unauthorized user accounts, modified binary components, or malicious code distributed through a legitimate update, all of which have the potential effect of remote control and data exfiltration without immediate detection [31].
- **Rootkits** are a specialized class of backdoors designed to hide the attacker's presence in the system, often intervening at the kernel or user-mode level to mask processes, files, network connections, and log entries that could signal compromise. Common techniques include hooks in system calls, DKOM (Direct Kernel Object Manipulation), binary image changes, or kernel module replacements, all to ensure persistence and privilege theft. Rootkits are widely used to maintain privileged access on compromised systems and pose a major challenge to detection, as they can undermine traditional monitoring and antivirus tools, often requiring advanced forensics and hardware-level integrity techniques to identify and eradicate [32].

- **Hardware backdoors** are unauthorized access mechanisms or functionality integrated directly into the physical layer of the device—from IP blocks or microcode to FPGAs (Field-Programmable Gate Arrays), bitstreams, and custom circuits—that allow attackers to bypass security software controls altogether and gain access, persistence, or exfiltration capabilities. These backdoors can occur through deliberate insertions into the supply chain (compromising third-party Ips (Internet Protocols) or the manufacturing process), microcode changes, or unprotected hardware configurations (e.g., accessible JTAG (Joint Test Action Group) debug ports) and are particularly dangerous because they affect the physical integrity of the platform and are extremely difficult to detect and remove. Their detection requires advanced techniques (hardware fingerprinting, side-channel analysis, silicon-level verifications and hardware attestation mechanisms such as PUF (Physically Unclonable Function)/secure boot), and preventive measures are based on robust supply-chain security processes, production integrity checks and security-by-design design [33].

D. DoS Attacks

The "DoS" attack is a method of cyberattack that aims to disrupt or disrupt the services provided by a network, server or other IT infrastructure, making them inaccessible to legitimate users. These attacks are carried out by flooding the target with a large volume of traffic or by exploiting specific vulnerabilities, leading to the exhaustion of the target system's resources. A DoS attack aims to make an IT service unavailable to its legitimate users by excessively consuming the resources of the target system, either through massive traffic (flooding) or by exploiting software vulnerabilities.

Types of DoS Attacks:

- **Flood Attacks**: Flood attacks consist of generating a massive volume of traffic to a target resource (server, gateway, IoT device) to overload its processing and service capacities, which leads to the degradation or unavailability of legitimate services. Representative examples are ICMP (Internet Control Message Protocol) Flood, in which ICMP packets are sent massively to flood the target interface, and SYN (Synchronize) Flood, which exploits the TCP (Transmission Control Protocol) connection establishment mechanism by sending many SYN packets without completing the handshake, thus consuming the server's state resources. In embedded and IoT environments, the effect of a flood can be amplified by hardware limitations (low buffer, weak CPU (Central Processing Unit)) and network topology (low-capacity hubs/gateways), which can trigger functional bottlenecks or severe telemetry losses. Defenses include packet filtering at the edge of the network, rate-limiting, SYN cookies, firewall/ACL (Access Control List) configurations, as well as "scrubbing" solutions at an edge/cloud layer; However, the application of these countermeasures on devices with low resources requires distributed and selective solutions to avoid the increase of false positives or the impact on legitimate functionality [34].
- **Application-Layer DoS**: Exploiting application-specific vulnerabilities to consume server resources. Examples include HTTP (Hypertext Transfer Protocol) GET/POST flood attacks [35].
- **DDoS (Distributed Denial of Service)**: DoS attacks launched from multiple distributed sources, making them even harder to counter. These attacks are usually orchestrated through botnet networks [36].

E. Exploits Attacks

An "Exploit" attack involves the use of vulnerabilities in software or hardware to compromise a computer system. These attacks are designed to exploit weaknesses in a system's security to gain unauthorized access, execute malicious code, or disrupt the normal operation of a device or application an exploit is a sequence of instructions, data, or actions that take advantage of a vulnerability in a computer system to achieve unforeseen or unauthorized behavior [37].

A classification of exploits can be listed:

- **Remote Exploits**: Attacks that are launched remotely without physical access to the target system. These include exploiting vulnerabilities in network services or web applications [38].
- **Local Exploits**: Attacks that require physical access or authenticated access to the target system. Typically, they exploit vulnerabilities to escalate user privileges [39].
- **Zero-Day Exploits**: Attacks that exploit vulnerabilities unknown to the public or software developers at the time of the attack. These vulnerabilities do not yet have a patch available, which makes them particularly dangerous [40].

The exploitation process begins with the discovery of vulnerabilities, which can be achieved through security audits, penetration testing, static/dynamic analysis of the code or by behavioral monitoring of the system to identify anomalies that indicate a potential weakness. Once a vulnerability is identified, the attacker moves on to exploit development — specialized code that leverages that weakness to escalate privileges, bypass control mechanisms, or launch arbitrary execution. This is followed by the launch of the attack, which can occur either through network vectors (malicious packets, HTTP requests, protocol payloads) or through local access (compromised installation environments, physical

devices). Finally, obtaining the results may involve unauthorized access to resources, execution of malicious code, data exfiltration or disruption of the functionality of the system, the concrete effects depending on the nature of the vulnerability and the level of access obtained [37].

Specific exploitation techniques frequently include classic attacks such as buffer overflow, where data written beyond the boundaries of a buffer allows the return area to be overwritten and can lead to arbitrary code execution [41]; SQL injection, whereby malicious SQL commands are injected into queries to databases to obtain or modify unauthorized information [42]; and XSS, which introduces malicious JavaScript code into web pages so that it is executed in the context of legitimate users' browsers, making it easier to steal sessions or manipulate interfaces [43]. Each of these techniques requires specific stages of preparing and adapting the exploit to the target environment and is usually followed by additional steps to maintain persistence and trace coverage (post-exploitation).

F. Generic Attacks

In cybersecurity, the term "Generic" attack is used to describe a class of attacks that are not specific to a particular vulnerability but can be applied in a variety of scenarios to compromise computer systems. These attacks rely on common methods and techniques to exploit weaknesses in systems. A generic attack is a type of cyberattack that does not target a specific vulnerability but uses common techniques and methods to compromise the security of computer systems. These attacks are often automatic and can be tailored to work against a variety of targets [44].

Generic Attacks are defined by the following types:

- **Brute Force Attacks**: These attacks involve trying all possible password combinations until you find the correct one. It is a classic example of a generic attack because it is not based on a specific vulnerability in the system, but on the weakness of user passwords. Studies show that brute force attacks are effective against systems that do not implement adequate protection measures, such as blocking accounts after several failed attempts [45].
- **Phishing**: Phishing involves sending fraudulent messages that appear to come from trusted sources to trick users into divulging sensitive information, such as passwords or bank account details. Research indicates that phishing attacks are extremely common and can be customized to target any group of users, making them a generic type of attack [46].

Generic attacks rely on automation and adaptability, being orchestrated through scripts and kits that continuously scan attack surfaces and adjust payloads to quickly exploit discovered vulnerabilities. Techniques such as spoofing, sniffing and exploiting unsecured services allow actors to gain initial access or sensitive information; Effective countermeasures combine network-level detection (flow-based monitoring), strict hardening policies, and proactive updates to configurations and services to reduce attack windows [44].

G. Reconnaissance Attacks

The "Reconnaissance" attack, also known as "research" or "scanning", is a crucial step in the lifecycle of a cyberattack. It involves collecting information about a target to identify weaknesses that can be exploited later. This preliminary step is essential for planning and executing an effective attack [47].

The stages and techniques of the Reconnaissance Attack are as follows:

- **Passive Reconnaissance**: This involves obtaining information about the target without directly interacting with the system. Methods include internet searches, social media analysis, and study of public documents. The goal is to avoid detection by the target's security systems [48].
- **Active Reconnaissance**: This involves direct interactions with the target system, such as port scanning, pinging, and traceroute. These activities can be detected by security systems but provide more detailed data on network structure and vulnerabilities [48].

Reconnaissance attacks aim to progressively obtain information about a target in order to build an exploitable profile; The typical process includes footprinting (collecting initial data such as domains, IP blocks, and WHOIS records) [49], followed by automated scanning to identify open ports, services, and software versions [50], and then enumeration, which extracts specific details about accounts, policies, and internal resources [51]. These steps, often performed with automated tools, allow attackers to map the attack surface, identify vulnerable vectors, and plan further actions (exploitation, targeted phishing, or lateral movement).

H. Shellcode Attacks

Shellcode is a specialized type of code used in cyberattacks, designed to gain access to a shell (command interface) on a target system. This type of attack is frequently used in exploits of vulnerabilities, especially in buffer overflow, to execute arbitrary commands or gain complete control over the target system. Shellcode is a sequence of instructions assembled to be executed directly by the CPU. The term "shellcode" comes from its original purpose of launching a shell (command-line interpreter) [52].

Types of Shellcode:

- **Local Shellcode**: Used when the attacker already has a certain level of access to the target system and wants to escalate privileges or execute other local commands [53].
- **Remote Shellcode**: Used to compromise a remote system, being injected through the network. Typically, it connects back to the attacker to provide access to a shell or executes predefined commands [53].

- **Staged Shellcode**: It is divided into several stages, so the first stage, called stager, is small and simple, having the role of downloading and executing the second stage, called stage, which contains the complete functionality [53].

Shellcode is the binary payload used to take control of the execution of a vulnerable program and is typically injected by exploiting memory vulnerabilities (e.g., buffer overflow), at which point the return pointers or jump table are manipulated to redirect the execution flow to the malicious code. Structurally, shellcode is often written in assembly language, very compact and self-sufficient (it does not depend on external libraries), a necessary condition because the execution environment can be unpredictable [52]; To avoid detection, attackers frequently apply encoding/obfuscation techniques and polymorphic variants. Defenses include Execution Controls (DEP (Data Execution Prevention)/NX (No-eXecute)), Address Randomization (ASLR (Address Space Layout Randomization)), Control-Flow Integrity, and Behavioral Detection Mechanisms that track anomalies in the execution flow [54], [55].

I. Worms Attacks

Worms are autonomous malware that self-propagates through computer networks without requiring user intervention. Unlike viruses, which attach themselves to existing programs, worms are self-contained entities that multiply and spread rapidly by exploiting network and software vulnerabilities. Worms are autonomous malware programs that propagate through networks, infecting other systems to multiply [56].

They do not require a host program to spread but rather use networks to spread from one computer to another, and they can exploit security vulnerabilities in software or misconfigurations to copy themselves to other systems [56].

Worms can be classified as follows:

- **Email Worms**: They are spread by sending infected emails to contacts in the victim's address list. Often, emails contain malicious attachments or links [57].
- **Internet Worms**: They spread over the Internet, exploiting vulnerabilities in network services to copy themselves to other systems connected to the network [56].
- **File-sharing Worms**: They propagate through P2P (Peer-to-Peer) networks infecting shared files and thus spreading to other users who download those files [58].

The prevention and detection of worm attacks is based on the constant updating of systems through security patches, the use of firewalls and network segmentation to limit the spread, and the identification of propagation is carried out through IDS/NIDS systems and monitoring of network traffic, which can signal anomalous communication patterns specific to these attacks [59].

To highlight the differences between the main types of attacks analyzed in this paper, Table I summarizes the defining characteristics, mechanisms of action, impact on embedded systems and the main prevention measures identified in the literature [16-59].

TABLE I. COMPARISON OF THE MAIN TYPES OF CYBER-ATTACKS ANALYZED

Type of attack	Main mechanism	Target / Vector	Impact on the system	Difficulty of detection	Recommended prevention measures
Fuzzing	Sending invalid/random data to cause errors	Protocols, Files, Firmware	Discovering vulnerabilities; Crashes	Average – detectable by logs and IDS	Regular patching, input validation, and internal fuzz testing
Analysis	Static/dynamic/ forensic inspection of code or traffic	Software, Binary Code, Network	Vulnerability identification, post-attack analysis	Reduced (non-invasive, passive)	Periodic auditing, behavioral monitoring
Backdoor	Introducing a hidden access channel	Software, firmware, hardware	Persistent unauthorized access	High – hides from protection systems	Supply chain integrity, secure boot, firmware verification
DoS	Overload due to excessive traffic or abnormal packages	Servers, gateways, and IoT devices	Service unavailability, network congestion	Medium–high	Firewall/ACL, rate limiting, network segmentation
Exploit	Exploiting a known or zero-day vulnerability	Software / OS (Operating System)	Arbitrary code execution, privilege escalation	High	Patch management, sandboxing, verify integrity cod
Generic	Generic techniques (brute force, phishing)	User accounts, services	Compromise of credentials, unauthorized access	Low–medium	MFA (Multi-Factor Authentication), Strong Passwords, User Education
Reconnaissance	Collecting Target Information	Networks, servers, applications	Exposure of topology and vulnerabilities	Reduced (log-detectable)	IDS, Privacy Policies, Restriction of Public Information
Shellcode	Inserting and executing malicious code at the CPU level	Memory, exploitable applications	Taking control of execution	High	DEP/NX, ASLR, execution flow control, behavioral monitoring
Worms	Self-propagation through the network, without human intervention	Networks, connected systems	Rapid spread, infrastructure damage	Medium	Patches, firewall, IDS/NIDS, network segmentation

IV. Anomalies and Malformations in Cyberattack Patterns

Anomalies and malformations in the context of cybersecurity refer to deviations from the normal behavior of a system or network and are often used to identify potential cyberattacks. In cyberattack models, understanding and detecting these concepts is essential for preventing and responding effectively to threats [60].

A. Anomalies in Cyber Attack Patterns

Anomalies are any behavior or dataset that deviates from expected or normal patterns. In the context of cybersecurity, these can include unusual user activity, unusual network traffic, or unexpected changes to files and systems [60].

The anomalies are classically as follows:

- **Point anomalies**: These are individual data that deviate significantly from the rest of the dataset. For example, a single high-volume data packet in an otherwise constant data stream can indicate a DoS attack [61].
- **Contextual anomalies**: The data is abnormal in the context of a specific time frame or other contextual conditions. For example, a high volume of traffic in a business network may be normal during business hours, but abnormal during the night [62].
- **Collective anomalies**: Refers to a dataset that is abnormal when considered together, but whose individual components may appear normal. For example, a series of small changes in system files that individually seem harmless but together indicate a stealth attack [63].

Malformation detection involves the use of advanced analysis techniques, such as packet analysis, which uses specialized tools to inspect network traffic details and identify inconsistencies, monitoring file integrity by checking checksums (hashes) of critical files to detect unauthorized changes, and implementing intrusion detection systems, which can recognize signatures and behaviors associated with malformations and other malicious activities [64].

Anomaly detection involves the use of various techniques and algorithms, such as:

- **Statistical methods**: It is based on defining a probabilistic model of normal behavior and identifying significant deviations [65].
- **Machine Learning**: ML algorithms such as clustering (K-means) and classification (SVM (Support Vector Machines)) are used to learn normal patterns of behavior and identify deviations [66].
- **Rule-based methods**: Manual definition of rules based on expert knowledge to detect abnormal behavior [67].

B. Malformations in Cyber Attack Patterns

Malformations refer to intentional and malicious changes to the structure or content of data to exploit vulnerabilities in information systems. They can take many forms, including corrupted network packets, altered configuration files, or injected code [60].

Types of Malformations:

- **Malformed network packets**: Attackers can create network packets that do not comply with protocol specifications to cause errors in network devices. For example, fragmented IP attacks use malformed IP packets to bypass firewalls [68].
- **Injected code**: Injecting malicious code into an application or system to take control. Examples include SQL injection and XSS [69].
- **Modified configuration files**: Unauthorized modification of system configuration files to allow unauthorized access or install backdoors [70].

CONCLUSION

The present paper aimed to carry out a systematic analysis of the methods, mechanisms and classifications used in the identification and description of cyber-attacks on embedded systems and critical infrastructures. By correlating the information from the literature, the fundamental concepts regarding cybersecurity, the types of attack (fuzzing, analysis, backdoors, DoS, exploit, worms, reconnaissance, generic, shellcode, etc.) and the principles of anomaly detection were defined.

Chapters III and IV presented the theoretical foundations and categories of attack analyzed, highlighting the mechanisms by which they affect the confidentiality, integrity and availability of the systems. This stage allowed the knowledge to be structured in a unitary way, facilitating the identification of vulnerable areas and the appropriate defense methods.

The paper contributes to an in-depth understanding of the relationship between attack vectors and protection principles, providing a solid foundation for the development of intelligent detection and prevention solutions. In the next stage of the research, the author aims to develop his own attack detection system based on the analysis of anomalies and malformations, using artificial intelligence techniques and statistical processing.

Future directions include expanding the database with real samples of traffic and security events, standardizing the testing process, as well as applying XAI methods to increase decision-making transparency. Thus, this paper not only synthesizes the existing literature, but also substantiates the practical approach of designing an autonomous system, capable of detecting abnormal behaviors early and reducing the exposure of embedded systems to complex cyberattacks.

REFERENCES

[1] P. Koopman, "Embedded system security," *Computer (Long Beach Calif)*, vol. 37, no. 7, pp. 95–97, Jul. 2004, doi: 10.1109/MC.2004.52.

[2] A. I. Molcut, S. Lica, and I. Lie, "Cybersecurity for Embedded Systems: A review," *2022 15th International Symposium on Electronics and Telecommunications, ISETC 2022 - Conference Proceedings*, 2022, doi: 10.1109/ISETC56213.2022.10009944.

[3] "IoT devices installed base worldwide 2015-2025| Statista." Accessed: Oct. 23, 2025. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[4] S. A. Dragusin, N. Bizon, and R. N. Bostinaru, "Comprehensive Analysis of Cyber-Attack Techniques and Vulnerabilities in Communication Channels of Embedded Systems," *Proceedings of the 16th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2024*, 2024, doi: 10.1109/ECAI61503.2024.10607432.

[5] M. Alabadi and Y. Celik, "Anomaly Detection for Cyber-Security Based on Convolution Neural Network: A survey," *HORA 2020 - 2nd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings*, Jun. 2020, doi: 10.1109/HORA49412.2020.9152899.

[6] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of Machine Learning Algorithms for Anomaly Detection," *International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2020*, Jun. 2020, doi: 10.1109/CYBERSECURITY49315.2020.9138871.

[7] G. D. Apostolidis, I. Kalouptsoglou, M. Siavvas, D. Kehagias, and D. Tzovaras, "AI-Enhanced Static Analysis: Reducing False Alarms Using Large Language Models," *Proceedings - 2025 IEEE International Conference on Smart Computing, SMARTCOMP 2025*, pp. 288–293, 2025, doi: 10.1109/SMARTCOMP65954.2025.00088.

[8] X. Zhou, P. Wang, L. Zhou, P. Xun, and K. Lu, "A Survey of the Security Analysis of Embedded Devices," *Sensors 2023, Vol. 23, Page 9221*, vol. 23, no. 22, p. 9221, Nov. 2023, doi: 10.3390/S23229221.

[9] D. Adhikari, W. Jiang, J. Zhan, D. B. Rawat, and A. Bhattarai, "Recent advances in anomaly detection in Internet of Things: Status, challenges, and perspectives," *Comput Sci Rev*, vol. 54, p. 100665, Nov. 2024, doi: 10.1016/J.COSREV.2024.100665.

[10] A. Aloseel, H. He, C. Shaw, and M. A. Khan, "Analytical Review of Cybersecurity for Embedded Systems," *IEEE Access*, vol. 9, pp. 961–982, 2021, doi: 10.1109/ACCESS.2020.3045972.

[11] S. Trilles, S. S. Hammad, and D. Iskandaryan, "Anomaly detection based on Artificial Intelligence of Things: A Systematic Literature Mapping," *Internet of Things*, vol. 25, p. 101063, Apr. 2024, doi: 10.1016/J.IOT.2024.101063.

[12] S. Rani, A. Kataria, S. Kumar, and V. Karar, "A new generation cyber-physical system: A comprehensive review from security perspective," *Comput Secur*, vol. 148, p. 104095, Jan. 2025, doi: 10.1016/J.COSE.2024.104095.

[13] M. Kohli and I. Chhabra, "A comprehensive survey on techniques, challenges, evaluation metrics and applications of deep learning models for anomaly detection," *Discover Applied Sciences*, vol. 7, no. 7, pp. 1–29, Jul. 2025, doi: 10.1007/S42452-025-07312-7/TABLES/7.

[14] H. Liang, X. Pei, X. Jia, W. Shen, and J. Zhang, "Fuzzing: State of the Art," *IEEE Trans Reliab*, vol. 67, no. 3, pp. 1199–1218, Sep. 2018, doi: 10.1109/TR.2018.2834476.

[15] H. Dai, C. Murphy, and G. Kaiser, "Configuration fuzzing for software vulnerability detection," *ARES 2010 - 5th International Conference on Availability, Reliability, and Security*, pp. 525–530, 2010, doi: 10.1109/ARES.2010.22.

[16] X. Deng, Y. Duan, and K. Deng, "A Fuzzing Method for Embedded Software," *IEEE Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2021*, pp. 1735–1738, Oct. 2021, doi: 10.1109/ITNEC52019.2021.9587220.

[17] S. K. Cha, M. Woo, and D. Brumley, "Program-adaptive mutational fuzzing," *Proc IEEE Symp Secur Priv*, vol. 2015-July, pp. 725–741, Jul. 2015, doi: 10.1109/SP.2015.50.

[18] P. Zhang, B. Ren, H. Dong, and Q. Dai, "CAGFuzz: Coverage-Guided Adversarial Generative Fuzzing Testing for Image-Based Deep Learning Systems," *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4630–4646, Nov. 2022, doi: 10.1109/TSE.2021.3124006.

[19] Z. Hu and Z. Pan, "A Systematic Review of Network Protocol Fuzzing Techniques," *IMCEC 2021 - IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference*, pp. 1000–1005, Jun. 2021, doi: 10.1109/IMCEC51613.2021.9482063.

[20] D. Kengo Oka, " Automating File Fuzzing over USB for Automotive Systems ," *Building Secure Cars*, pp. 211–239, May 2021, doi: 10.1002/9781119710783.CH10.

[21] V. Herdt, D. Große, H. M. Le, and R. Drechsler, "Verifying Instruction Set Simulators using Coverage-guided Fuzzing," *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019*, pp. 360–365, May 2019, doi: 10.23919/DATE.2019.8714912.

[22] J. Wang, B. Chen, L. Wei, and Y. Liu, "Skyfire: Data-Driven Seed Generation for Fuzzing," *Proc IEEE Symp Secur Priv*, pp. 579–594, Jun. 2017, doi: 10.1109/SP.2017.23.

[23] H. ; Li *et al.*, "A Novel Network Protocol Syntax Extracting Method for Grammar-Based Fuzzing," *Applied Sciences 2024, Vol. 14, Page 2409*, vol. 14, no. 6, p. 2409, Mar. 2024, doi: 10.3390/APP14062409.

[24] M. Safta, P. Svasta, M. Dima, A. Marghescu, and M. N. Costiuc, "Design and setup of Power Analysis attacks," *2016 IEEE 22nd International Symposium for Design and Technology in Electronic Packaging, SIITME 2016*, pp. 110–113, Dec. 2016, doi: 10.1109/SIITME.2016.7777256.

[25] S. R. Kavya Rani, B. C. Soundarya, H. L. Gururaj, and V. Janhavi, "Comprehensive Analysis of Various Cyber Attacks," *2021 IEEE Mysore Sub Section International Conference, MysuruCon 2021*, pp. 255–262, 2021, doi: 10.1109/MYSURUCON52639.2021.9641089.

[26] M. Nachtigall, L. Nguyen Quang Do, and E. Bodden, "Explaining static analysis-a perspective," *Proceedings - 2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASEW 2019*, pp. 29–32, Nov. 2019, doi: 10.1109/ASEW.2019.00023.

[27] V. Anandhi *et al.*, "Malware Detection using Dynamic Analysis," *2023 International Conference on Advances in Intelligent Computing and Applications, AICAPS 2023*, 2023, doi: 10.1109/AICAPS57044.2023.10074588.

[28] M. A. Salitin and A. H. Zolait, "The role of user entity behavior analytics to detect network attacks in real time," *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2018*, Nov. 2018, doi: 10.1109/3ICT.2018.8855782.

[29] Q. R. McCluskey, M. M. Chowdhury, S. Latif, and K. Kambhampaty, "Computer Forensics: Complementing Cyber Security," *IEEE International Conference on Electro Information Technology*, vol. 2022-May, pp. 507–512, 2022, doi: 10.1109/EIT53891.2022.9813886.

[30] Y. Li, Y. Jiang, Z. Li, and S. T. Xia, "Backdoor Learning: A Survey," *IEEE Trans Neural Netw Learn Syst*, vol. 35, no. 1, pp. 5–22, Jan. 2024, doi: 10.1109/TNNLS.2022.3182979.

[31] T. Hemmert, A. May, J. Mittmann, and C. R. T. Schneider, "How to Backdoor (Classic) McEliece and How to Guard Against Backdoors," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial*

Intelligence and Lecture Notes in Bioinformatics), vol. 13512 LNCS, pp. 24–44, 2022, doi: 10.1007/978-3-031-17234-2_2.

[32] S. Suresh Kumar, A. Ponni Valavan, and V. Prathiksha, "Prevention of Kernel Rootkit in Cloud Computing," *Proceedings of the 7th International Conference on Intelligent Computing and Control Systems, ICICCS 2023*, pp. 732–739, 2023, doi: 10.1109/ICICCS56967.2023.10142886.

[33] A. Waksman and S. Sethumadhavan, "Silencing hardware backdoors," *Proc IEEE Symp Secur Priv*, pp. 49–63, 2011, doi: 10.1109/SP.2011.27.

[34] H. M. Albadi, M. A. Khder, S. W. Fujo, and T. M. Yousif, "A Literature Review of the Seriousness of Flooding-based DoS Attack," *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2022*, pp. 463–469, 2022, doi: 10.1109/3ICT56508.2022.9990774.

[35] Y. Feng, J. Li, and T. Nguyen, "Application-Layer DDoS Defense with Reinforcement Learning," *2020 IEEE/ACM 28th International Symposium on Quality of Service, IWQoS 2020*, Jun. 2020, doi: 10.1109/IWQOS49365.2020.9213026.

[36] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: A classification," *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2003*, pp. 190–193, 2003, doi: 10.1109/ISSPIT.2003.1341092.

[37] P. Kijsanayothin and R. Hewett, "Exploit-based analysis of attack models," *Proceedings - IEEE 12th International Symposium on Network Computing and Applications, NCA 2013*, pp. 183–186, 2013, doi: 10.1109/NCA.2013.18.

[38] T. Bao, R. Wang, Y. Shoshitaishvili, and D. Brumley, "Your Exploit is Mine: Automatic Shellcode Transplant for Remote Exploits," *Proc IEEE Symp Secur Priv*, pp. 824–839, Jun. 2017, doi: 10.1109/SP.2017.67.

[39] "What is an Exploit? Exploit Prevention - Bitdefender." Accessed: Oct. 25, 2025. [Online]. Available: https://www.bitdefender.com/consumer/support/answer/10556/

[40] S. Regi, G. Arora, R. Gangadharan, R. Bathla, and N. Pandey, "Case Study on Detection and Prevention Methods in Zero Day Attacks," *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), ICRITO 2022*, 2022, doi: 10.1109/ICRITO56286.2022.9964873.

[41] C. Cowan, F. Wagle, Calton Pu, S. Beattie, and J. Walpole, "Buffer overflows: attacks and defenses for the vulnerability of the decade," pp. 119–129, Nov. 2002, doi: 10.1109/DISCEX.2000.821514.

[42] A. Rai, M. M. I. Miraz, D. Das, H. Kaur, and Swati, "SQL Injection: Classification and Prevention," *Proceedings of 2021 2nd International Conference on Intelligent Engineering and Management, ICIEM 2021*, pp. 367–372, Apr. 2021, doi: 10.1109/ICIEM51511.2021.9445347.

[43] H. C. Chen, A. Nshimiyimana, C. Damarjati, and P. H. Chang, "Detection and prevention of cross-site scripting attack with combined approaches," *2021 International Conference on Electronics, Information, and Communication, ICEIC 2021*, Jan. 2021, doi: 10.1109/ICEIC51217.2021.9369796.

[44] S. Souissi and A. Serhrouchni, "AIDD: A novel generic attack modeling approach," *Proceedings of the 2014 International Conference on High Performance Computing and Simulation, HPCS 2014*, pp. 580–583, Sep. 2014, doi: 10.1109/HPCSIM.2014.6903738.

[45] L. Bosnjak, J. Sres, and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, pp. 1161–1166, Jun. 2018, doi: 10.23919/MIPRO.2018.8400211.

[46] K. D. Tandale and S. N. Pawar, "Different Types of Phishing Attacks and Detection Techniques: A Review," *Proceedings of the 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing, ICSIDEMPC 2020*, pp. 295–299, Oct. 2020, doi: 10.1109/ICSIDEMPC49020.2020.9299624.

[47] A. Alshamrani, "Reconnaissance Attack in SDN based Environments," pp. 1–5, Oct. 2020, doi: 10.1109/ICT49546.2020.9239510.

[48] S. N. Hidayah Zulkiffli, M. N. Ahmad Zawawi, and F. A. Rahim, "Passive and Active Reconnaissance: A Social Engineering Case Study," *2020 8th International Conference on Information Technology and Multimedia, ICIMU 2020*, pp. 138–143, Aug. 2020, doi: 10.1109/ICIMU49871.2020.9243402.

[49] K. Shye Lianq and V. Selvarajah, "Footprinting and Reconnaissance: Impact and Risks," *IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics, ICDCECE 2022*, 2022, doi: 10.1109/ICDCECE53908.2022.9793237.

[50] D. Theodorus, M. S. Nabi, and Q. Al-Maatouk, "Web-based Reconnaissance and Vulnerability Scanner: A Review and Proposed Solution," *2021 International Conference on Data Analytics for Business and Industry, ICDABI 2021*, pp. 666–670, 2021, doi: 10.1109/ICDABI53623.2021.9655963.

[51] M. M. ; Alani, E. Damiani, M. M. Alani, and E. Damiani, "XRecon: An Explainbale IoT Reconnaissance Attack Detection System Based on Ensemble Learning," *Sensors 2023, Vol. 23, Page 5298*, vol. 23, no. 11, p. 5298, Jun. 2023, doi: 10.3390/S23115298.

[52] G. Yang, X. Chen, Y. Zhou, and C. Yu, "DualSC: Automatic Generation and Summarization of Shellcode via Transformer and Dual Learning," *Proceedings - 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022*, pp. 361–372, 2022, doi: 10.1109/SANER53432.2022.00052.

[53] "Types of Shellcode | eCPPTv2 Notes." Accessed: Oct. 25, 2025. [Online]. Available: https://zer0verflow.gitbook.io/ecpptv2-notes/system-security/shellcoding/types-of-shellcode

[54] J. Ganz and S. Peisert, "ASLR: How Robust Is the Randomness?," *Proceedings - 2017 IEEE Cybersecurity Development Conference, SecDev 2017*, pp. 34–41, Oct. 2017, doi: 10.1109/SECDEV.2017.19.

[55] Y. Lin, "Novel Techniques in Recovering, Embedding, and Enforcing Policies for Control-Flow Integrity," 2021, doi: 10.1007/978-3-030-73141-0.

[56] V. S. Koganti, L. K. Galla, and N. Nuthalapati, "Internet worms and its detection," *2016 International Conference on Control Instrumentation Communication and Computational Technologies, ICCICCT 2016*, pp. 64–73, Jul. 2017, doi: 10.1109/ICCICCT.2016.7987920.

[57] C. C. Zou, D. Towsley, and W. Gong, "Email worm modeling and defense," *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, pp. 409–414, 2004, doi: 10.1109/ICCCN.2004.1401687.

[58] M. E. Johnson, D. McGuire, and N. D. Willey, "The evolution of the peer-to-peer file sharing industry and the security risks for users," *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2008, doi: 10.1109/HICSS.2008.436.

[59] N. Wattanapongsakorn, E. Wonghirunsombat, T. Assawaniwed, V. Hanchana, S. Srakaew, and C. Charnsripinyo, "A network-based internet worm intrusion detection and prevention system," *2013 International Conference on IT Convergence and Security, ICITCS 2013*, 2013, doi: 10.1109/ICITCS.2013.6717779.

[60] M. Ravinder and V. Kulkarni, "A Review on Cyber Security and Anomaly Detection Perspectives of Smart Grid," *Proceedings - 5th International Conference on Smart Systems and Inventive Technology, ICSSIT 2023*, pp. 692–697, 2023, doi: 10.1109/ICSSIT55814.2023.10060871.

[61] M. Zhao and J. Chen, "A Review of Methods for Detecting Point Anomalies on Numerical Dataset," *Proceedings of 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2020*, pp. 559–565, Jun. 2020, doi: 10.1109/ITNEC48623.2020.9085206.

[62] M. Farshchi *et al.*, "Contextual Anomaly Detection for a Critical Industrial System Based on Logs and Metrics," *Proceedings - 2018 14th European Dependable Computing Conference, EDCC 2018*, pp. 140–143, Nov. 2018, doi: 10.1109/EDCC.2018.00033.

[63] J. Tao *et al.*, "Visual Analysis of Collective Anomalies Through High-Order Correlation Graph," *IEEE Pacific Visualization Symposium*, vol. 2018-April, pp. 150–159, May 2018, doi: 10.1109/PACIFICVIS.2018.00027.

[64] G. Muruti, F. A. Rahim, and Z.-A. bin Ibrahim, "A Survey on Anomalies Detection Techniques and Measurement Methods," pp. 81–86, Feb. 2019, doi: 10.1109/AINS.2018.8631436.

[65] S. A. Dragusin, N. Bizon, R. N. Bostinaru, F. M. Enescu, R. M. Teodorescu, and C. Savulescu, "Analysis of Vulnerabilities in Communication Channels Using An Integrated Approach Based on Machine Learning and Statistical Methods," *Proceedings of the 16th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2024*, 2024, doi: 10.1109/ECAI61503.2024.10607483.

[66] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine Learning for Anomaly Detection: A Systematic Review," *IEEE Access*, vol. 9, pp. 78658–78700, 2021, doi: 10.1109/ACCESS.2021.3083060.

[67] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on IP flows," *Proceedings - IEEE INFOCOM*, pp. 424–432, 2009, doi: 10.1109/INFCOM.2009.5061947.

[68] M. A. Al Naeem, A. Abubakar, and M. M. H. Rahman, "Dealing with well-formed and malformed packets, associated with point of failure that cause network security breach," *IEEE Access*, vol. 8, pp. 197554–197566, 2020, doi: 10.1109/ACCESS.2020.3034383.

[69] R. Riley, X. Jiang, and D. Xu, "An architectural approach to preventing code injection attacks," *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 30–39, 2007, doi: 10.1109/DSN.2007.13.

[70] "IEEE Standard Classification for Software Anomalies.," Dec. 1993, doi: 10.1109/IEEESTD.1994.121429.