الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic and Republic of Algeria

Ministry of Higher Education and ScientificResearch

National HigherSchoolof Advanced Technologies

وزارة التعليم العالي والبحث العلمي

المدرسة الوطنية العليا للتكنولوجيات المتقدمة

1st Year Second Cycle

Introduction to Cyber Security Project

Groupe 07

Edge Computing – Project Report

Group Members:

DOKKAR Chaima
FERRAT Ilham
KELLAKH Ranyme
AIT HOCINE Khadidja
KHEDRI Manel
BOUTRIA Manal
BOUAKKAZ Madjeda
DJEBOURI Lynda
FERKIOUI Akram
HAMMOUTI Walid

Contents

1 Introduction:

Edge computing has emerged as a critical architectural shift aimed at bringing computation and intelligence closer to data sources, especially in environments where latency, bandwidth constraints, and real-time processing requirements dominate system design. The topic was selected because modern distributed infrastructures—from IoT deployments to autonomous services—demand architectures capable of delivering efficiency, resilience, and robust security at the network edge. Throughout this work, we analyze the essential components that define edge computing, including architectural layers, communication models, security threats, and the mechanisms used to mitigate them. Each aspect is examined from both conceptual foundations and contemporary research developments.

The research team organized the work into coordinated sections to provide a structured and comprehensive study. Djebouri Linda introduced the general context and outlined the key security threats that motivate the need for secure edge infrastructures and contributed to the preparation and design of the presentation. Khedri Manal examined the architecture of edge computing by analyzing the edge device, edge server, and cloud server layers. Ait Hocine Khadidja investigated DDoS attacks in edge environments and reviewed existing defense mechanisms. Bouakkaz Madjeda explored side-channel attacks and the protective techniques used to counter them. Boutria Manal analyzed malware-injection attacks that target edge nodes. Ferrat Ilham discussed weaknesses in authentication and authorization processes. Hammouti Walid provided a comparative study between security challenges in edge systems and those found in cloud computing. Kellakh Ranyme identified the root causes underlying persistent security issues. Ferkioui Akram presented the main challenges facing current systems and worked on the project report. Finally, Dokkar Chaima outlined future research directions to support the evolution of secure and scalable edge architectures.

Through this collaborative effort, the group offers a unified and coherent understanding of edge computing and highlights the open problems that continue to drive research in this rapidly developing field.

2 Architecture of Edge Computing:

In this section, we present a general architecture of edge computing shown in Figure 1, which mainly consists of three layers: an edge device layer (EDL), an edge server layer (ESL), and a cloud server layer (CSL).

2.1 Edge Device Layer (EDL):

Edge devices are those low-level electronic devices deployed at EDL which operate in the physical world to complete tasks such as sensing, actuating and controlling. Each edge device is logically controlled by one or more microcontrollers (MCUs), with each being a small computer running on a single integrated circuit . The low-level software interface programmed in the MCUs that provide controls to the device's hardware is known as firmware. All the functions including sensing, controlling, and computing are coded in the firmware and therefore, handled by the MCUs. Edge devices can be further categorized as IoT devices and mobile devices. IoT devices are lightweight electronic devices that are interconnected or connected to the edge servers in ESL through wireless protocols such as 4G/5G, WiFi, and Bluetooth. They usually run on lightweight preemptive /cooperative real-time operating systems (RTOS), e.g., FreeRTOS and RT Thread . Once after a RTOS is burned into the chip of the IoT devices, it usually does not provide further programming interfaces. Some examples of IoT devices include : smart home devices, health monitoring devices, and smart warehouse carts in industrialized IoT (IIoT). Different from IoT devices, mobile devices usually have more advanced and costly preemptive operating systems, e.g., Android and iOS, providing programmable interfaces for developers to code their own applications at the top of the OSes. Some examples of mobile devices include : smartphones, tablets, and central controllers of smart vehicles.

2.2 Edge Server Layer (ESL):

ESL has a hierarchical structure with multiple sub-layers consisting of various edge servers with increasing computational power from bottom to up as shown in Figure *. The edge servers located at the lowest sub-layer include wireless base stations and access points (APs), which are mainly deployed for communication purpose to receive data from the edge devices and send control flows back to them through different wireless interfaces. Upon receiving data from edge devices, base stations/APs forward the data to the edge servers located at the upper sub-layer, which are mainly in charge of handling computation tasks. Upon receiving data passed from base stations/APs or edge servers at the lower sub-layers, the edge servers conduct relevant computation and analysis tasks on their own. If the complexity of a task exceeds the computation limits of the current edge server, it would offload the task to the servers located at the higher sub-layers, which possess more powerful computation capabilities. These servers then conclude with asequence of control flows and pass them back to the base stations/APs, which forward them to the edge devices in the end. Edge servers handle most of the core computing functions such as authentication, authorization, computation, data analytics, task offloading, and data storage for edge computing.Regardless of the server's layer, there are other important components that play a major role in connecting and communicating the sub-layers within this layer : the Edge Gateway collects data from nearby IoT devices, performs basic processing, and forwards only relevant information to upper layers. The

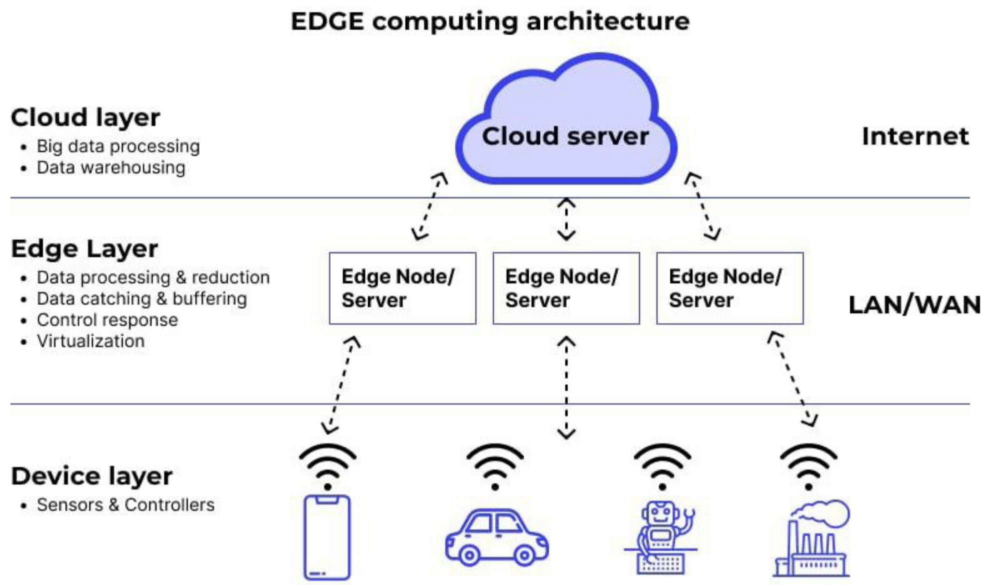

Figure 1: Architecture of Edge Computing

Switch connects local devices (like edge servers and gateways) within the same network, ensuring fast data transfer inside the local area. The Router directs data between different networks (for example, from the edge network to the cloud) managing traffic and choosing the best path for communication. Together, they enable efficient, secure, and low-latency data flow between devices and the cloud.

2.3 Cloud Server Layer (CSL):

Cloud server layer (CSL)hosts center cloud servers and data centers, with the cloud servers responsible for the highest level authentication and authorization, computation, and integration of different tasks offloaded from edge servers, and the data centers in charge of storing vast amount of data generated by the edge devices and edge servers. The state-of-the-art cloud servers and cloud data centers consist of clusters of powerful machines. Because the security of CSL has been extensively studied , in this article, we mainly explore the security issues of EDL and ESL in edge computing.

3 Edge Computing Security and Privacy:

3.1 DDoS Attacks and Defense Mechanisms in Edge Computing:

3.1.1 Introduction:

DDoS refers to a type of active cyber attack in which external attackers aim to disrupt the availability of normal services provided by one or more servers using distributed resources such as a cluster of compromised edge devices (a.k.a. botnet). It mainly targets the Network, Transport, and Application layers of the OSI model. It is a powerful interruption attack that prevents legitimate users from accessing a service, thereby threatening the availability aspect of information security. A traditional DDoS attack occurs when an attacker persistently sends massive streams of packets to a victim from compromised distributed electronic devices; thus, the hardware resources of the victim are quickly exhausted for handling these malicious packets and can no longer process legitimate requests on time. In other cases, attackers send malformed or spoofed packets that confuse the victim's protocol, causing it to falsely conclude that all channels are occupied. Edge servers are particularly vulnerable because they have limited computational capacity, weak authentication, and often operate with heterogeneous and insecure firmware. Attackers commonly compromise numerous edge devices and coordinate them remotely to launch large-scale attacks against edge infrastructures.

3.1.2 Attack Specifications:

In a typical DDoS scenario, the attacker (external) compromises a cluster of edge devices, gains full control, and then commands each to send flooding traffic toward the target edge server, overwhelming it until services shut down. Such flooding-based DDoS attacks include UDP flooding, ICMP flooding, SYN flooding, and HTTP flooding, all relying on saturating the victim's network and processing capacity. Other variants such as zero-day attacks exploit unknown software vulnerabilities, affecting both availability and integrity of systems. Defensive measures include per-packet inspection, traffic filtering, and rate limiting, though detection remains challenging due to IP spoofing and the resemblance between malicious and legitimate traffic. An attacker in a DDoS scenario compromises a cluster of edge devices, takes control, and instructs them to send excessive traffic to a target edge server to overwhelm it and stop its service. Flooding-based attacks rely on high volumes of malformed or malicious packets and include major types: UDP flooding, ICMP flooding, SYN flooding, HTTP flooding, Ping of Death, and Slowloris. Brief mechanisms: UDP flooding: Sends huge volumes of UDP packets to prevent the server from processing legitimate UDP traffic. ICMP flooding: Sends many ICMP Echo Request (ping) packets rapidly; the victim replies to each, consuming both inbound and outbound throughput and slowing the system. SYN flooding: Abuses TCP's three-way handshake by sending many SYNs with spoofed IPs and not completing the handshake, leaving half-open connections on the server. HTTP flooding: Sends a large number of legitimate-looking HTTP requests (GET/POST/PUT, etc.), overwhelming the server's processing capacity. Zero-day attacks: Exploit previously unknown software or protocol vulnerabilities (e.g., memory corruption) to crash or corrupt systems; these are hard to predict or filter because they abuse logic/implementation flaws rather than just volume.

3.1.3 Defense Strategies Against Flooding:

D1. Per-packet inspection and filtering: Drop suspicious packets before they reach the server (e.g., packet filters tied to congestion control). Effective but can be evaded by IP/MAC spoofing, forged headers, or tools that alter identifiers. 2. Statistics-based detection: Analyze traffic aggregates for anomalies (entropy measures, statistical clustering). Requires large traffic samples and struggles with encrypted flows. 3. Machine/deep learning: Use models (decision trees, Naive Bayes, autoencoders, neural nets) to detect attacks, including some encrypted traffic. These require large, representative training datasets, risk overfitting, and typically become effective only after substantial attack traffic has been observed. Limitations: All approaches face practical challenges — IP spoofing, similarity between legitimate and attack traffic, encrypted flows that hinder inspection, and the need for extensive, high-quality training data. Effective defense usually combines real-time packet filtering with aggregate analysis and ML-based detection.

3.1.4 Defense Strategies Against Zero-Day Exploits:

Zero-day defenses focus on mitigating code-level vulnerabilities through memory safety analysis and protective isolation mechanisms. Memory analysis and detection: Techniques like pointer-tainting, ECC memory, and firmware analysis help identify or reduce memory corruption. Deep learning models (RNNs, GNNs, NLP-based analyzers) can detect vulnerable code even without access to the source. However, encrypted firmware and anti-debugging measures often restrict their effectiveness. Active protection and isolation: Methods such as in-process memory isolation, SDN-based IoT firewalls, lightweight isolation at access routers, and application-side fuzzing help minimize the attack surface or detect crashes. Despite their benefits, these methods can impose heavy computational overheads that are unsuitable for resource-constrained IoT or edge devices and may still fail to prevent complex zero-day exploits.

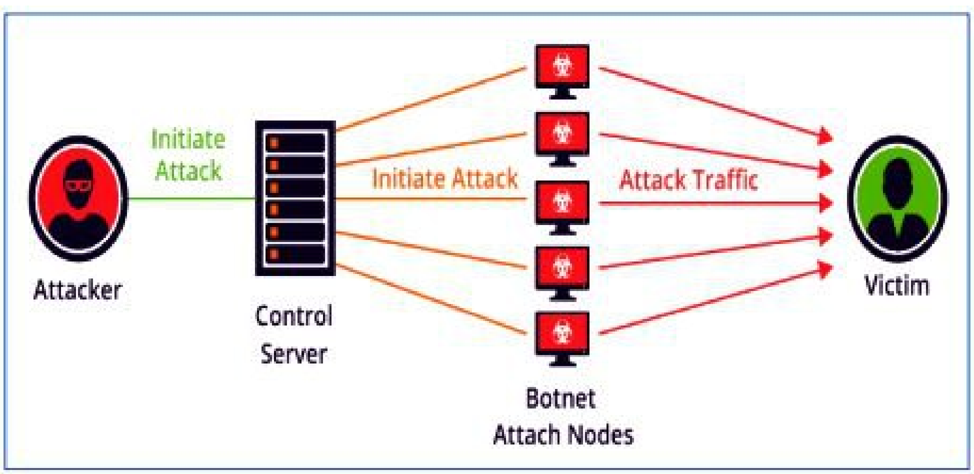

Figure 2: A Typycal Architecture of a DDoS Attack

3.2 Side Channel Attacks and Defenses:

3.2.1 Introduction:

What if I told you hackers could steal your data without ever touching your files, cracking your password, or breaking into your system? Sounds impossible, right? That is exactly what happens in a side-channel attack — a method where attackers exploit indirect physical or system signals, such as power usage, timing, or data from system files like `/proc`, to extract secrets without breaking encryption or accessing the main code.

3.2.2 OSI Layer Concerned:

Side-channel attacks primarily target the **Physical Layer**, since they rely on observable physical behaviors like power consumption and electromagnetic signals. They can also affect the **Application Layer**, where exposed data sources and sensors provide additional leakage paths.

3.2.3 Attack Type and Location:

These attacks are generally **passive**, meaning the attacker observes system behavior rather than altering it. They can be launched by:

- **Internal attackers:** malicious applications reading accessible data or sensors.
- **External attackers:** capturing power traces, electromagnetic emissions, or timing information remotely.

3.2.4 Threat and Damage:

Side-channel attacks primarily compromise **confidentiality**, exposing cryptographic keys, user data, or private operations. The potential damage can range from partial data leakage to a complete compromise of secure systems, depending on the sensitivity of the extracted information.

3.2.5 Detection and Prevention:

Detection of side-channel attacks is extremely difficult because they leave no direct logs or traces. Therefore, prevention focuses on reducing the available leakage sources through:

- **Data Perturbation:** adding noise to timing, power, or communication patterns to hide correlations.
- **Access Control:** restricting access to system interfaces and files like `/proc`.
- **Obfuscation:** randomizing computations and introducing timing variations.
- **Hardware Isolation:** separating sensitive operations using trusted hardware zones.

3.2.6 Attack Basis and Vulnerabilities:

Side-channel attacks exploit indirect signals that correlate with internal system operations, such as:

- Power consumption and timing delays.
- Electromagnetic emissions.
- System and sensor data exposure.

Edge computing devices are especially vulnerable due to their limited resources, high sensor density, and distributed nature, which increase both physical and software side-channel opportunities.

3.2.7 Existing Solutions and Limitations:

- **Differential Privacy:** masks patterns with noise but reduces accuracy and data utility.
- **Access Control:** limits software-based leaks but cannot protect against physical emissions.
- **Obfuscation:** increases latency and energy usage.
- **Hardware Isolation:** enhances protection but cannot completely prevent signal leakage.

3.3 Malware Injection:

3.3.1 Introduction:

A malware injection attack occurs when malicious code is inserted into a legitimate process, service, or device so that it executes within the victim's environment. These attacks aim to compromise confidentiality, integrity, or availability. They are particularly dangerous in edge computing because devices often have limited resources and inconsistent patching mechanisms.

3.3.2 Server-Side Injection Attacks:

1. **SQL Injection (SQLi):** OSI Layer: Application Attack Type: Active Attacker Location: External Attack Threat: Data confidentiality and integrity Damage Level: High Detection Chance: Moderate Possibility of Prevention: High Attacks Based On: Manipulating input fields to alter database queries Vulnerability: Dynamic SQL queries without proper sanitization Existing Solutions and Limitations: Parameterized queries and ORMs mitigate risk but do not prevent logic-level flaws.

2. **Cross-Site Scripting (XSS):** OSI Layers: Presentation and Application Attack Type: Active Attacker Location: External Attack Threat: Session hijacking and data theft Damage Level: Moderate to High Detection Chance: Low Possibility of Pre- vention: Medium Attacks Based On: Injecting malicious JavaScript into web pages Vulnerability: Unsanitized output rendered in browsers Existing Solutions and Limitations: Content Security Policy and escaping user input, but developers often misapply them.

3. **CSRF / SSRF:** OSI Layers: Network and Application Attack Type: Active Attacker Location: External Attack Threat: Unau- thorized requests or internal resource access Damage Level: Moderate Detection Chance: Low Possibility of Prevention: High Attacks Based On: Trick authenticated users or services into sending crafted requests Vul- nerability: Missing CSRF tokens or improper request validation Existing Solutions and Limitations: SameSite cookies and allowlists reduce exposure but may break service communication.

4. **XML Signature Wrapping:** OSI Layers: Presentation and Application Attack Type: Active Attacker Location: External or Man-in-the- Middle Attack Threat: Message integrity and authenticity Damage Level: High Detection Chance: Low Pos- sibility of Prevention: High Attacks Based On: Wrapping signed XML nodes with new malicious elements Vulnerability: Insecure XML parsing and ID binding Existing Solutions and Limitations: Schema validation and correct parser configuration; however, legacy systems often skip updates.

3.3.3 Device-Side Injection Attacks:

1. **Firmware Injection:** OSI Layers: Physical to Application Attack Type: Active Attacker Location: Internal or Supply-Chain Attack Threat: System takeover and persistent malware Damage Level: Critical Detection Chance: Very Low Possi- bility of Prevention: Medium Attacks Based On: Modifying firmware updates or flashing malicious images Vulnerability: Unsigned or unverified firmware Existing Solutions and Limitations: Secure boot and digital signatures protect new devices but may not apply to legacy hardware.

2. **Remote Code Execution (RCE):** OSI Layers: Transport and Application Attack Type: Active Attacker Location: External Attack Threat: Com- plete device control Damage Level: Critical Detection Chance: Moderate Possibility of Prevention: High At- tacks Based On: Exploiting vulnerabilities in running services Vulnerability: Outdated software or insecure service exposure Existing Solutions and Limitations: Patching and sandboxing help but rely on consistent main- tenance.

3. **Supply-Chain Injection:** OSI Layer: Application Attack Type: Passive to Active Attacker Location: Internal / Third-Party Attack Threat: Malicious dependency integration Damage Level: High Detection Chance: Low Possibility of Pre- vention: Medium Attacks Based On: Inserting malware into libraries or software updates Vulnerability: Unver- ified dependencies and unsigned packages Existing Solutions and Limitations: Code-signing and supply-chain monitoring, but trust in third parties remains a weak link.

4. **WebView / Mobile Injection:** OSI Layers: Presentation and Application Attack Type: Active Attacker Location: External Attack Threat: Session hijacking and data manipulation Damage Level: Moderate Detection Chance: Low Possibility of Pre- vention: Medium Attacks Based On: Abusing WebView or native-JS bridges in mobile apps Vulnerability: Over-permissive WebView settings Existing Solutions and Limitations: Restrict bridge interfaces and sandbox web content, though many apps neglect these controls.

3.3.4 Conclusion:

Each malware injection vector targets different OSI layers and system components. Understanding which layer is affected helps prioritize security controls and apply layered defenses across both server and device sides.

3.4 Authentication and Authorization Attacks and Defense Mechanisms:

3.4.1 Introduction:

Authentification is the process of verifying the identity of users who request certain services. Authorization is the process that determines an entity's access rights and privileges.In edge computing, authentication often takes place between edge devices and edge servers, but it can also be decentralized among devices.

3.4.2 The four main types of authentification and authorisation attacks are:

Brute-force attacks (guessing passwords to gain access), Authentication protocol flaws (exploiting weaknesses), Authorization protocol flaws (bypassing access controls to gain privileges) and Over-privilege attack (abusing excessive permissions to perform unauthorized actions).

3.4.3 OSI layer targeted by the attack:

In edge computing, authentication and authorization attacks mainly target the application layer (identity, access, and API management). They also affect the session ,presentation and network layers .

3.4.4 Attack type:

Authentication and authorization attacks are generally active attacks, because the attacker must interact directly with the system. They are rarely passive.

3.4.5 Attack location:

In edge computing, about 70% of authentication and authorization attacks come from external sources (through public interfaces), while around 30% are internal.

3.4.6 Attack threat:

Authentification and authorization attacks threaten several security properties: authentification , confidentiality , integrity and availability .

3.4.7 Demage level:

Authentication and authorization attacks can have high to critical impact, allowing unauthorized access, data modification, and disruption of system operations. They also undermine trust and accountability, making detection and response difficult.

3.4.8 Reasons for low to moderate detection rate :

Attackers use legitimate credentials or sessions, blending with normal activity, Conventional monitoring cannot detect actions within granted privileges and also Effective detection requires advanced techniques (UEBA, AI-driven anomaly detection).

3.4.9 Preventive measures :

Robust authentication: Hardware-bound MFA, no password-only schemes , **Granular authorization:** Least-privilege via dynamic RBAC/ABAC , **Real-time monitoring:** UEBA, AI-driven anomaly detection, distributed SIEM and **Zero-trust posture:** Verify all requests using mutual TLS and workload identity.

3.4.10 The authentification and authorisation attacks based on:

The attack consists of impersonating a legitimate identity to perform unauthorized actions in real time directly on an edge node. It allows the attacker to steal, modify, redirect, or block critical flows (data, commands) before they reach the cloud.

3.4.11 The exploited vulnerabilities:

Weak identity, Poorly validated permissions , Implicit network trust and Exposed APIs without protection

3.4.12 Exesting solutions and their limitations:

Existing solutions: MFA (confirmation of user identity), RBAC/ABAC, end-to-end encryption (mTLS) end API security.
Limitations: Vulnerable if credentials or tokens are stolen, nodes are compromised, or policies misconfigured; they can add complexity, overhead, and scalability challenges.

3.5 A Comparison Study on Security Issues in Edge Computing and Cloud Computing

3.5.1 Architectural Differences Between Cloud and Edge Computing

Aspect	**Cloud Computing**	**Edge Computing**
Architecture	Centralized — uses remote servers and large data centers on the Internet.	Decentralized — hierarchical structure with distributed edge servers.
End Devices	Fully-fledged computers (PCs, servers).	IoT and mobile devices with limited resources.
Connectivity	Mainly wired Internet connections.	Primarily wireless connections (Wi-Fi, 4G/5G).
Resource Capability	High — powerful centralized infrastructure.	Limited — low-profile edge devices and servers.
Data Processing Location	Centralized data processing in cloud data centers.	Localized processing near data sources (edge nodes).
Service Characteristics	High latency, not location-aware, large-scale centralized control.	Low latency, location-aware, real-time, privacy-enhanced, and bandwidth-efficient.
Attack Surface	Concentrated — fewer but high-value centralized targets.	Distributed — many small edge nodes vulnerable to local and physical attacks.
Security Challenges	Cloud provider security breaches, data leaks, insider threats, DDoS on central servers.	Node tampering, device hijacking, insecure wireless communication, local denial-of-service attacks.
Defense Focus	Strong centralized authentication, encryption, and access control.	Lightweight security, trust management, secure device authentication, decentralized intrusion detection.
Privacy	Centralized data storage — higher privacy risk.	Local data processing — improved privacy control.

3.5.2 Comparison of Security Attacks in Cloud and Edge Computing

Attack Type	**Cloud Computing**	**Edge Computing**
DDoS Attacks	Uses heavyweight protocols (HTTP/HTTPS, FTP, SMTP) $\rightarrow$ harder to attack. Strong firewalls reduce impact. Still vulnerable to zero-day DDoS due to software flaws.	Uses lightweight protocols (CoAP, MQTT, UDP) $\rightarrow$ easier to attack. Low-end devices can be easily compromised (e.g., Mirai botnet). Flooding attacks still practically effective.
Side Channel Attacks	Wired connections limit attack surface. Attacks mostly through packet eavesdropping or crafted queries.	Wireless connectivity increases attack surface. Attackers can exploit wireless signals, power consumption, or proximity-based leaks.
Malware Injection Attacks	Attackers usually compromise the cloud server first, then infect clients. Harder to compromise cloud devices.	Attackers compromise edge devices first, then target servers. Interconnected IoT devices make propagation easier.
Authentication & Authorization Attacks	Vulnerable to dictionary attacks and SSL/TLS weaknesses. Centralized $\rightarrow$ simpler authorization scenarios.	More weak passwords on IoT/mobile devices. Vulnerable to WPA/WPA2, Bluetooth, 4G/5G weaknesses. Decentralized $\rightarrow$ complex authorization, more prone to overprivilege attacks.

3.5.3 Conclusion

Cloud computing and edge computing offer similar services and functionalities, but their architectural differences lead to significant variations in security risks. Cloud computing, with its centralized design, powerful servers, and wired connections, is generally more resilient to traditional attacks such as DDoS and side channel exploits, though it remains vulnerable to zero-day attacks at the software level.

Edge computing, by contrast, relies on decentralized servers and resource-constrained IoT/mobile devices connected via wireless networks, which exposes it to a wider range of attacks, including device-level DDoS, side channel attacks, malware injection, and authentication/authorization exploits.

Overall, while cloud systems benefit from stronger protections and centralized management, edge systems trade off some security for proximity, real-time performance, and low-cost services, making security a more critical and complex challenge in edge computing environments.

4 Root Causes, Grand Challenges , and Furure Research:

4.1 Root Causes of Security Issues:

Current edge computing infrastructures suffer from serious cyber-attacks, which may result in huge financial loss. The leading root causes of these security threats lie in the unavoidable flaws or vulnerabilities in protocol and code designs as well as their implementations, the concealed correlations between the public and the protected private/secure data, and the lack of fine-grained access control.

4.1.1 Protocol-Level Design Flaws:

Protocol is the fundamental support for basic functions such as communications and data processing in edge computing from the theoretical perspective.Unfortunately, many of the current protocols adopted by edge computing systems have design flaws because their designers mainly focus on utility and user experience, treating security as something unimportant if not unnecessary. By exploiting these flaws, an attacker can achieve attack goals ranging from simply shutting down a normal service (e.g. DDoS) to fully controlling an edge device or server (e.g., malware injection attacks).

4.1.2 Implementation-Level Flaws:

Implementation is the process of practically realizing the edge computing functionalities. Even if a protocol may be proven secure mathematically, it does not necessarily mean that its implementation is secure. Logic flaws in an implementation can neutralize the security strength of a protocol which has been proven strictly secure. There are two main reasons leading to these flaws:

- Developers may misunderstand the foundations of the protocol .
- Migrating a protocol from other platforms to the edge computing platform may cause adaptivity in consistencies.

By exploiting implementation flaws, an attacker can bypass the security features provided by the protocol (e.g., authentication attacks).

4.1.3 Code-Level Vulnerabilities:

Code is the basic unit of a program, which defines the exact execution flow a processor should follow. Many attacks are, in fact, the results of the code-level vulnerabilities. Note that implementation-level flaws and code-level vulnerabilities are two different concepts, with the former mainly referring to the logic flaws in a practical realization while the latter mainly referring to the system bugs that can cause memory failures/corruptions. Such vulnerabilities include stack/heap overflow, use-after-free dangling pointer, format string, and so on. By exploiting these vulnerabilities, an attacker can achieve attack goals ranging from simply shutting down a normal service to fully controlling an edge device or server (e.g., malware injection attacks).

4.1.4 Data Correlations:

In an edge computing system, tons of data would be generated from the edge infrastructures constantly, with some of them being sensitive and placed under strict protections while others less sensitive and exposed to the public without protection. However, there may exist hidden correlations between the sensitive data and insensitive data, which may not be straightforward. Nevertheless, by exploiting these correlations and leveraging various attack models, an attacker can infer the sensitive data (e.g., side channel attacks) or even tamper them (e.g., bad data injection attacks) based on the insensitive data.

4.1.5 Lacking Fine-Grained Access Controls:

Access control is one of the most important security measures, via which an entity is permitted to access only the least resources it needs. Such a measure has a good reputation in protecting traditional general-purpose computing systems for a long time but it cannot be directly adapted to edge computing due to the more complex and fine-grained permission scenarios. Many current edge computing systems implement only coarse grained access control mechanisms, or even do not employ any access control mechanism. Lacking a fine-grained access control specifically designed for edge computing applications may significantly lower the bar of launching an attack (e.g., authorization attacks and MITM attacks).

4.2 Status Quo and Grand Challenges:

4.2.1 Introduction:

Edge computing has emerged to overcome the limitations of traditional cloud computing by providing low latency and localized processing. However, its rapid growth has outpaced the development of robust security mechanisms. Current systems face challenges in four key areas: design philosophy, framework adaptability, access control, and defensive capabilities.

4.2.2 Lack of Security-by-Design:

Edge systems are primarily designed for performance and low latency, often neglecting security considerations in the early stages. This results in built-in vulnerabilities that make edge nodes susceptible to attacks such as malware injection, DDoS, and unauthorized access. The distributed and heterogeneous nature of edge environments further complicates the later integration of security measures, highlighting the need for a security-by-design approach.

4.2.3 Non-Migratability of Traditional Security Frameworks:

Security solutions created for centralized cloud environments are not directly applicable to edge systems due to differences in computational capabilities, architectures, and communication protocols. The diversity of edge devices and lack of standardization cause fragmented security management and hinder the development of universal frameworks.

4.2.4 Fragmented and Coarse-Grained Access Control:

Access control in edge environments is inconsistent and lacks granularity. Different platforms use incompatible models, and permissions are often too broad to handle dynamic interactions between devices. Developing fine-grained, context-aware access control policies remains a significant research challenge.

4.2.5 Isolated and Passive Defense Mechanisms:

Current defense strategies are largely reactive and siloed, offering limited protection against advanced or coordinated attacks. Most rely on detection and patching rather than proactive threat prevention. A key challenge is to design adaptive and collaborative defense systems capable of anticipating and responding dynamically to evolving threats.

4.3 Future Research Directions:

4.3.1 Introduction:

Recent analyses of Distributed Denial-of-Service (DDoS) attacks, malware injection, and authentication breaches reveal major limitations in current edge computing security. These threats exploit edge-specific characteristics such as limited resources, protocol diversity, and physical exposure. To strengthen resilience, future research must explore advanced, adaptive defense mechanisms that evolve with emerging attack techniques.

4.3.2 Multi-Layer Integrated Security Frameworks:

Future studies should focus on designing comprehensive multi-layer architectures addressing vulnerabilities across the edge stack. A promising direction is a three-layer framework combining hardware-rooted trust, intelligent orchestration, and decentralized management.

The outer layer enforces fine-grained, context-aware access control and dynamic authorization, evaluating authentication based on user identity, location, and operational context.

The middleware layer integrates software-defined networking (SDN) and network function virtualization (NFV) enhanced with autonomous threat response. Using federated learning, edge nodes collaboratively detect DDoS or anomaly patterns without centralizing data, preserving bandwidth and privacy. Lightweight, unsupervised anomaly detection models must be optimized for constrained edge resources while ensuring accuracy against evolving malware.

The inner layer establishes hardware-based trust through Trusted Execution Environments (TEEs), Physically Unclonable Functions (PUFs), and Memory Protection Units (MPUs). These mechanisms safeguard sensitive operations in secure enclaves and create isolation boundaries that resist firmware tampering and zero-day attacks.

4.3.3 Cross-Cutting Security Imperatives:

Privacy and cryptographic resilience are key challenges. Homomorphic encryption and differential privacy must be adapted for edge analytics to enable secure computation on encrypted data without revealing sensitive information. With quantum computing on the horizon, research should advance post-quantum cryptography and crypto-agility frameworks, ensuring smooth algorithm migration across heterogeneous devices.Another priority is cross-layer vulnerability management—using formal verification to assess risks from task offloading and maintaining consistent protection through unified security policies. Scalable, automated patch management must ensure timely updates across diverse edge ecosystems to prevent widespread exploitation.

4.3.4 Implementation Challenges and Research Trajectories:

Achieving these paradigms requires solving real-world challenges. The performance-security tradeoff must be quantified, ensuring minimal latency impact for real-time systems such as autonomous vehicles and industrial IoT. Research should also emphasize interoperability standards, enabling consistent security across different edge platforms, and energy-efficient protection mechanisms to suit battery-constrained devices.

4.3.5 Conclusion: Toward Comprehensive Edge Security:

The evolution of edge computing security demands a shift from isolated countermeasures to integrated, intelligent, and adaptive defense frameworks. Advancing research in hardware-rooted trust, autonomous orchestration, privacy-preserving computation, and cross-layer resilience will enable trustworthy, large-scale edge ecosystems. These innovations are essential for the secure adoption of edge computing in critical domains such as autonomous systems, industrial IoT, and smart cities.

5 References

- IEEE Xplore: Edge Computing Security: State of the Art and Challenges
- YouTube:
  - Video 1
  - Video 2
  - Video 3
  - Video 4
  - Video 5
- What is Edge Architecture? - https://www.geeksforgeeks.org