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 Walida

Contents

1 Introduction:

IoT devices, mobile apps, and other systems that need super fast responses are everywhere now. That's why edge computing has become such a big deal, a system that moves data processing and storage closer to where the data is generated instead of sending everything to the cloud. This cuts down delays, saves bandwidth, and lets systems make decisions in real time in areas like self-driving cars, smart cities, industrial automation, and healthcare monitoring... But this setup also brings new security challenges since having devices and servers all over the place makes them more vulnerable to attacks, especially since they don't have the same protections as big cloud servers. Edge computing usually works in layers : the devices themselves (Edge Device Layer, EDL), the local servers that support them (Edge Server Layer, ESL), and then the big cloud servers (Cloud Server Layer, CSL). Devices like sensors and controllers interact directly with the real world, handling data locally. That's great for speed, but it also opens up new attack points. Edge servers take on heavier tasks from devices, manage multiple endpoints, and make sure communication stays secure. The cloud handles the big-picture stuff : analytics, AI, and storing data long-term. Every layer helps the system work, but each also has its own weak spots. Security in edge computing is tricky. Attacks like DDoS, side-channel exploits, malware, and stolen credentials are common. Since edge systems are spread out, resource-limited, and varied, old-school cloud security doesn't always cut it. Edge computing needs smarter, lighter, and adaptable defenses , ones that protect the system without slowing it down. In this research, we're looking at the architecture of edge computing, the security risks, and privacy challenges, with a close look at each layer (EDL, ESL, CSL). The goal is to figure out why problems happen, see how they're currently handled, and point out big challenges for the future.

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). From the perspective of computational power, the systems built on the CSL have the most powerful computation capability, followed by the ones on the ESL. The devices at the EDL usually have the lowest computational power.

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

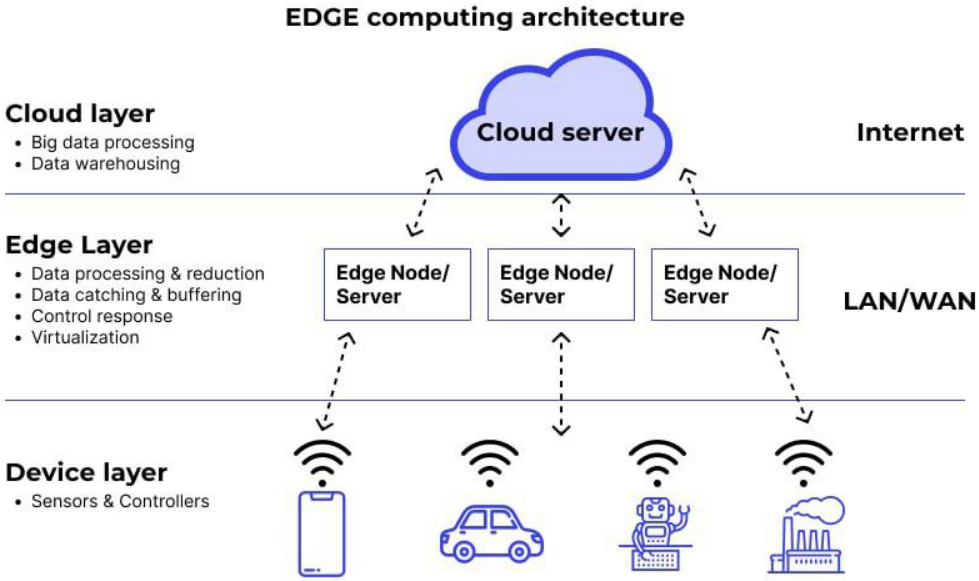

Figure 1: Architecture of Edge Computing

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 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.

2.4 Example

Source

To better understand how communication and processing occur within this architecture, let's take one of the most well-known examples : the self-driving car.

1. Edge Devices Layer In this layer, all sensors inside the self-driving car (such as cameras, radar, LiDAR, and GPS) continuously collect real-time data about the surroundings. The onboard

computer processes this data locally and makes instant decisions, like braking or steering, without sending everything to the cloud. This allows the car to react immediately and avoid accidents by minimizing delay.

2. Edge Server Layer : Here, nearby edge servers or 5G base stations receive summarized information from several cars in the same area. These servers analyze local traffic conditions, detect incidents, and send quick alerts back to nearby vehicles (for example, warning them about congestion or a roadblock ahead). This layer helps coordinate vehicles within a region efficiently and with low latency.

3. Cloud Server Layer: In the cloud layer, large data centers collect non-urgent information from many vehicles and edge servers. The cloud performs advanced analytics, trains AI models to improve driving algorithms, and sends software updates back to the cars. This global processing ensures continuous learning and long-term improvement of autonomous driving systems.

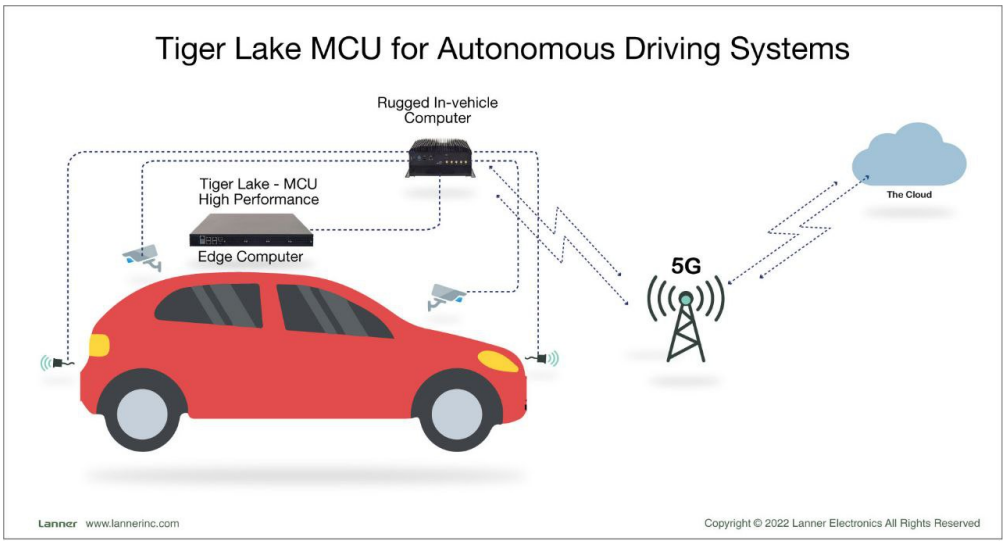

Figure 2: self-driving car

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 cyber attack in which attackers aim to disrupt normal services provided by one or more servers based on distributed resources such as a cluster of compromised edge devices (a.k.a botnet) [18]. It is a powerful attack which aims to prevent the legitimate use of a service. A traditional DDoS attack occurs when an attacker persistently sends streams of packets to a victim from the compromised distributed electronic devices; thus the hardware resources of the victim are quickly exhausted for handling these malicious packets and can no longer process any legitimate request on time. In some other DDoS scenarios, an attacker persistently sends malformed packets that confuse an application or a protocol of the victim to falsely conclude that all channels and resources are occupied.Edge servers are particularly vulnerable because they have lower computational capacity than cloud servers and often serve devices with heterogeneous, insecure firmware. Attackers commonly compromise numerous edge devices (botnets) and coordinate them to target edge infrastructure.

3.1.2 Attack Specifications:

DDoS attacks may happen when malicious edge devices communicate with the edge servers. In a DDOS attack, an attacker first compromises a cluster of edge devices and takes full control of them; then it commands each device to launch a denial-of-service attack targeting the edge server, causing the shutdown of its services. A typical architecture of a DDoS attack in edge computing is shown in Figure 3. DDoS attacks targeting edge computing can be taxonomized as flooding-based attacks and zero-day attacks. Flooding-Based Attacks. Flooding-based attacks are a type of DDoS attacks aiming to shutdown the normal service of a server based on a large amount of flooded malformed/malicious network packets, and are mainly classified as UDP flooding, ICMP flooding, SYN flooding, ping of death (PoD), HTTP flooding, and Slowloris, according to the attack techniques. In a UDP flooding attack, an attacker continuously sends a large amount of noisy UDP packets to a target edge server, causing the server incapable of handling the benign UDP packets in time and thus interrupting the normal UDP services provided by the edge server . In an ICMP flooding attack, an attacker exploits the ICMP protocol to craft an attack by sending a large number of ICMP Echo Request packets to a target edge server as fast as possible without waiting for the replies. This type of attack consumes both outgoing and incoming throughputs of the victim server since the server returns an ICMP Echo Reply packet upon each receipt of a ping request, resulting in a significant system-wide slowdown . In a SYN flooding attack, an attacker exploits the three-way handshake of the TCP protocol by initiating a huge amount of SYN requests with a spoofed IP address to a target edge server, while the server responds with a SYN-ACK packet to the spoofed IP for each SYN request and waits for the confirmation ACK that never comes . In a HTTP flooding attack, an attacker simply sends a tremendous amount of HTTP GET, POST, PUT, or other legitimate requests to an edge server. Since the edge server is less likely able to handle a huge number of legitimate requests in a timely manner due to the restricted computation power, its normal services can be easily throttled if a large amount of HTTP traffics are received in a short period of time Zero-day attacks: Attackers exploit previously unknown code

or protocol vulnerabilities to trigger memory corruption or crashes. Such attacks require discovery of a zero-day vulnerability in code running on the target (e.g., a heap overflow) and are difficult to predict or filter because they abuse logic or implementation flaws rather than volume.

3.1.3 Defense Strategies Against Flooding:

Defenses against flooding generally adopt a detect-and-filter philosophy and fall into two main categories: Per-packet detection and filtering: Inspect individual packets and drop suspicious ones before they reach the server (e.g., packet filters integrated with congestion control). Techniques include matching packet identifiers, negative selection algorithms for validating source IPs, and path-based identifiers. These can be effective but are vulnerable to evasion through IP/MAC spoofing, crafted headers, or tools that alter identifiers. Statistics-based detection: Analyze traffic aggregates to detect anomalous clusters indicative of DDoS using entropy measures or machine learning. Entropy methods detect distributional changes but require substantial traffic samples and can struggle with encrypted flows. Machine and deep learning (decision trees, Naive Bayes, autoencoders, neural nets) can automate detection, including for encrypted traffic, but need large and representative training data and risk overfitting; they generally become effective only after a significant volume of attack traffic has occurred.

3.1.4 Defense Strategies Against Zero-Day Exploits:

Zero-day defenses address code-level vulnerabilities through memory-safety analysis and protective isolation: Memory analysis and detection: Pointer-tainting, ECC memory, and firmware analysis aim to reveal or mitigate memory corruption. Techniques that analyze firmware or apply deep-learning vulnerability detectors (RNNs, GNNs, NLP) can identify risky code even without source code, but firmware encryption and anti-debug mechanisms limit applicability. Active protection and isolation: In-process memory isolation, SDN-based IoT firewalls, lightweight isolation on access routers, and fuzzing from application sides can reduce attack surface or detect crashes without firmware access. However, isolation and advanced protections often incur overhead that is prohibitive for constrained IoT/edge devices and may not eliminate complex zero-day triggers.

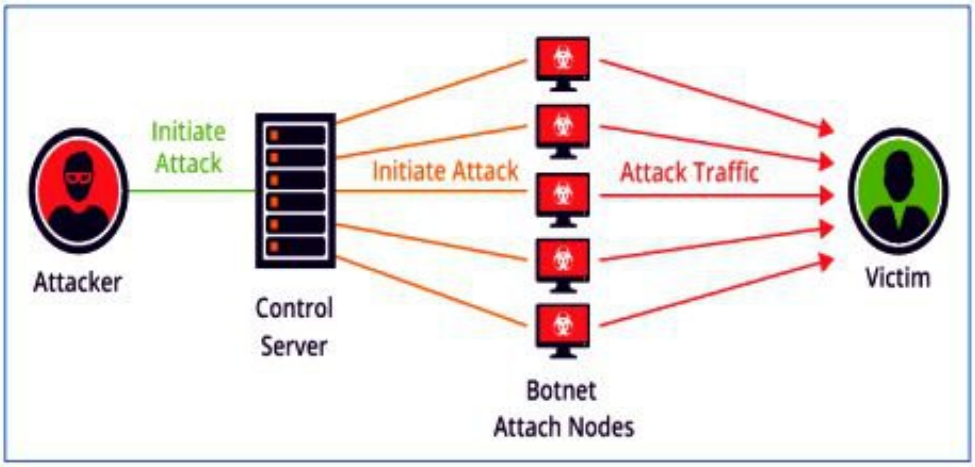

Figure 3: A Typycal Architecture of a DDoS Attack

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: Overpermissive 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, ensuring that it acts within its allowed limits. 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.
- 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 and presentation layers through encryption protocols and secure sessions. Finally, they reach the network layer, especially in 4G/5G, through interception or spoofing between edge nodes.

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 30originating from employees or entities already within the network, and are more dangerous due to their legitimate access.

3.4.6 Attack threat:

Authentification and authorization attacks threaten several security properties:

- **Authentification:** The attacker can impersonate a legitimate user (identity spoofing).

- **Confidentiality:** The attacker gains access to sensitive information
- **Integrity:** Once unauthorized access is obtained, they can modify or delete data.
- **Availability:** In some cases, unauthorized access can lead to service disruption.

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 of authentication and authorization attacks in edge computing:

- Attackers use legitimate credentials or sessions, blending with normal activity.
- Conventional monitoring cannot detect actions within granted privileges or low- profile behavior.
- Effective detection requires advanced techniques (UEBA, AI-driven anomaly detection); traditional tools miss most breaches.

3.4.9 Preventive measures for authentication and authorization attacks in edge computing:

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

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

- Exposed APIs without protection
- Compromised endpoints/nodes

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.

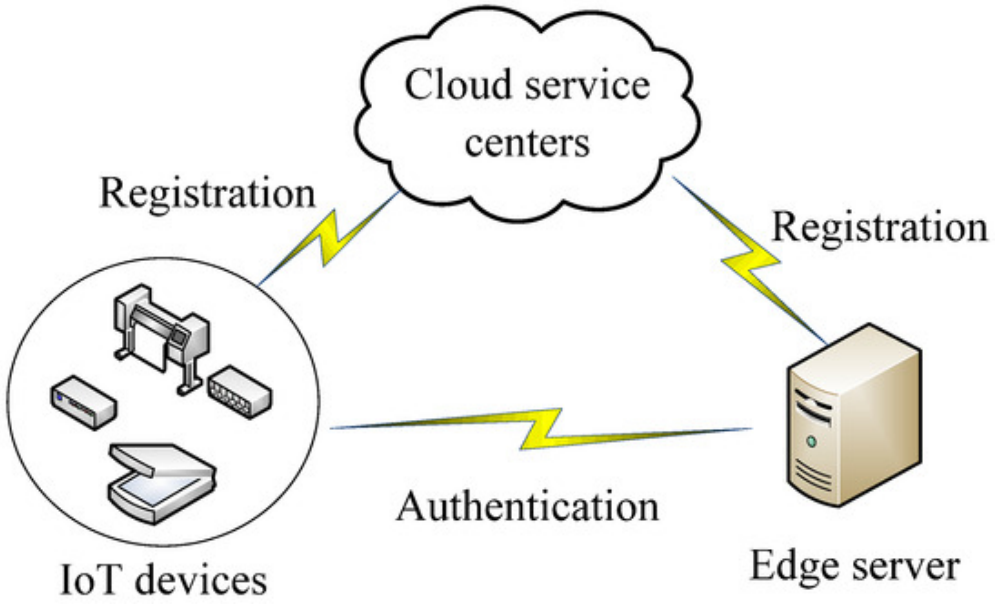

Figure 4: Authentification

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) → harder to attack. Strong firewalls reduce impact. Still vulnerable to zero-day DDoS due to software flaws.	Uses lightweight protocols (CoAP, MQTT, UDP) → 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 → simpler authorization scenarios.	More weak passwords on IoT/mobile devices. Vulnerable to WPA/WPA2, Bluetooth, 4G/5G weaknesses. Decentralized → 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.

4.2 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).

4.3 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).

4.4 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).

4.5 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.

4.6 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).

4.6.1 Root Causes of Security Issues (A Summary):

ROOT CAUSE	PROBLEM	POSSIBLE ATTACKS
Protocol-Level Design Flaws	Edge protocol designers mainly focus on utility and user experience, treating security as something unimportant.	Ranging from simply shutting down a normal service (e.g., DDoS) to fully controlling an edge device or server (e.g., malware injection attacks).
Implementation-Level Flaws	• Developers may misunderstand the foundations of the protocol. • Migrating a protocol from other platforms to the edge computing platform may cause adaptivity inconsistencies.	Bypass the security features provided by the protocol (e.g., authentication attacks).
Code-Level Vulnerabilities	Stack/heap overflow, use-after-free, dangling pointer, format string, and so on.	Ranging from simply shutting down a normal service to fully controlling an edge device or server (e.g., malware injection attacks).
Data Correlations	There may exist hidden correlations between sensitive data and insensitive data, which may not be straightforward.	Infer the sensitive data (e.g., side-channel attacks) or tamper with them (e.g., bad data injection attacks) based on the insensitive data.
Lacking Fine-Grained Access Controls	Implement only coarse-grained access control mechanisms, or do not employ any access control mechanism at all. (Lacking a fine-grained access control specifically designed for edge computing applications.)	E.g., authorization attacks and MITM attacks.

4.7 Status Quo and Grand Challenges:

4.7.1 Introduction:

Edge computing has rapidly evolved as a response to the limitations of traditional cloud computing, offering low latency, location awareness, and efficient data processing near the source. However, this fast growth has outpaced the development of robust security mechanisms. Current edge infrastructures face several significant challenges that hinder the establishment of a secure and resilient ecosystem. These challenges can be grouped under four main aspects: design philosophy, framework adaptability, access control, and defensive capabilities.

4.7.2 Lack of Security-by-Design:

Most edge systems are developed with a strong emphasis on performance optimization and low latency to meet the demands of IoT, smart city, and mobile applications. Security is often treated as an afterthought rather than an integral part of the design process. This lack of a security-by-design approach results in inherent vulnerabilities, exposing edge nodes and devices to a wide range of attacks such as malware injection, DDoS, and unauthorized access. Because the architecture of edge computing is distributed and heterogeneous, adding security later becomes complex and inefficient. Integrating security considerations from the early design stages is thus a fundamental but still largely unmet challenge.

4.7.3 Non-Migratability of Traditional Security Frameworks:

Existing security mechanisms built for centralized systems, such as those used in cloud environments, cannot be directly applied to edge computing. The differences in computational power, operating systems, communication protocols, and network hierarchies make direct migration impractical. Moreover, the diversity of devices—from sensors to mobile phones to microservers—creates compatibility issues that prevent unified implementation. Even within the same domain, a framework developed for one edge computing scenario (e.g., industrial IoT) may not function effectively in another (e.g., vehicular networks). This lack of standardization leads to fragmented security management and complicates the creation of universally applicable solutions.

4.7.4 Fragmented and Coarse-Grained Access Control:

Access control in edge systems remains one of the weakest links. Current models are fragmented because different vendors and platforms use different access strategies, making it difficult to design a global or interoperable security policy. They are also coarse-grained, meaning that permissions are typically broad (e.g., read/write access) instead of being fine-tuned to specific operations, contexts, or devices. In complex edge environments where multiple devices interact dynamically, this limited granularity makes it difficult to define "who can do what, where, and when." Developing fine-grained, context-aware access control remains an open research problem.

4.7.5 Isolated and Passive Defense Mechanisms:

Most existing defense mechanisms for edge computing are isolated, protecting against only specific types of attacks, and passive, responding only after a threat has occurred. Many systems rely on "detect-and-patch" approaches, which cannot prevent zero-day attacks or coordinated, multi-vector intrusions. Furthermore, these mechanisms lack collaboration across devices and layers—edge nodes rarely share threat intelligence or coordinate active defense strategies. Building adaptive, intelligent, and cooperative defense mechanisms that can anticipate and respond dynamically to evolving attacks is a critical challenge for the future.

4.7.6 Conclusion:

In summary, the current state of edge computing security is still immature. The absence of integrated design principles, the poor adaptability of existing frameworks, weak access control, and

passive defense strategies all contribute to an insecure environment. Future research must aim to create unified, scalable, and proactive solutions that embed security into the foundation of edge computing systems.

4.8 Future Research Directions:

4.8.1 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.

4.8.2 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.

4.8.3 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.

4.8.4 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.

4.8.5 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.