Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Diseño Electrónico

Availability under threat - CXL Type 3 Vulnerability Assessment

---

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN DISEÑO ELECTRÓNICO

Presenta: **ERICK EDUARDO VITE LOPEZ**

Director: **MSC. RAMON EDUARDO GRADILLA NEGRETE**

Tlaquepaque, Jalisco. 12 de julio de 2024.

Acknowledgments

To my parents, Nabor Vite and Dalila Lopez, for being my inspiration and motivating me every day to become a better person, for showing me the values of perseverance and discipline, for helping me understand that to achieve success, one must travel a long path likely filled with difficulties and adversities, but above all, one must not give up and remain persistent. Thank you for all the unconditional support and love you have given me throughout my journey.

Also, to my friends Josue Martines, Alejandro Cazares, and Marisol Cervantes, who have walked alongside me on this long journey regardless of the adversities. They have shown their support in both good times and bad. There is no doubt that one is a reflection of the people around them, and the successes I share today are undoubtedly a reflection of their character.

And finally, but no less importantly, to my colleagues and teachers who have always been there to share their time and knowledge.

Summary

The growing demand for data processing, driven by technologies such as artificial intelligence, has led the industry to seek more efficient solutions. The Compute Express Link (CXL) protocol stands out as a promising option, offering high-speed, low-latency interconnections between the CPU and other devices, thereby improving resource management, and facilitating the scaling of data-intensive applications.

A case study of how the availability of a system could be compromised through a DoS attack targeting the CXL protocol is presented. The context of this work is based on a controlled, secure, and dedicated environment for this research. At no point is a real system/server compromised, and the study is based on the Next Generation Intel® Xeon Server CPU and a CXL type 3 test card.

Established in 2019 and publishing the first official specification of the CXL protocol, called CXL 1.0, in March 2020, The CXL Consortium is an open industry standards group established to create technical specifications that enable groundbreaking performance for new usage models, while also supporting an open ecosystem for data center accelerators and other high-speed enhancements.

Considering that the protocol is relatively young compared to other protocols, such as PCIe (Peripheral Component Interconnect Express), the "threat modeling" methodology will be followed to identify vulnerabilities that could potentially compromise the system's availability.

Contents

Introduction

Compute Express Link (CXL) is a high-speed, multi-protocol interconnect standard designed and specialized for data centers and high-performance computing environments. It provides efficient communication between high-performance processors, such as Central Processing Units (CPUs) and Graphics Processing Units (GPUs), accelerators like Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs), and memory devices. The protocols supported by CXL are CXL.cache, CXL.io, and CXL.mem (Chapter 1 will provide more technical details on the functioning and definition of the protocol).

A key point to highlight and understand is the rapid and widespread acceptance the industry has shown towards the protocol. Some of the key factors include:

- The growing demands of artificial intelligence and machine learning workloads, which require fast and efficient access to large volumes of data.
- Compatibility with the existing PCI Express (PCIe) standard, which facilitates its integration into current infrastructure and promotes its adoption by major hardware manufacturers and software developers.
- The scalability of the protocol, which allows it to support a wide range of system topologies, such as connecting multiple memory devices and accelerators to a single processor, and can adapt to systems with different amounts and types of computational resources.
- Different systems are allowed to share hardware resources efficiently through their ability to facilitate cache coherence and memory semantics between devices. This means that if one system is performing a light workload on the hardware it possesses, it can share these resources with another server experiencing more demanding workloads.

The hypothesis to be explored in this document is based on the assumption that the CXL protocol is relatively young compared to other protocols like PCIe, where security gaps and patches already have the maturity to protect the integrity, confidentiality, availability, and authenticity of system data. For the development and research of this document, the focus will be

solely on attempting to compromise system availability by injecting malicious code or junk code to disrupt legitimate users' access to system resources.

A concept or design protocol important to mention to highlight how critical it could be to compromise the availability of a system is the concept of "High Availability," which seeks to ensure a certain degree of absolute continuity of work and operation of a system. These metrics evaluate the reliability of services, especially in critical environments such as data centers, financial services, and life support systems, where high availability is crucial and downtime can translate into million-dollar losses and, even worse, human losses. Some examples are mentioned below:

- 99.9% = 43.8 minutes/month or 8.76 hours/year ("three nines")
- 99.99% = 4.38 minutes/month or 52.6 minutes/year ("four nines")
- 99.999% = 0.44 minutes/month or 5.26 minutes/year ("five nines")

1. Theorical framework

1.1. Fundamentals of Peripheral Component Interconnect Express

The PCI Express protocol represented a significant improvement and paradigm shift compared to its predecessors, PCI and PCI-X, which are parallel transmission buses. Restructuring the architecture, the current PCI Express transmission bus is a high-speed serial transport protocol using a bidirectional connection capable of sending and receiving information simultaneously. This type of connection is referred to as "dual-simplex" [Fig 1-1].

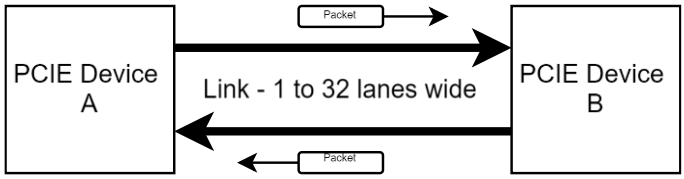

Fig. 1-1 Dual-Simplex Link

Utilizing the serialized structure means that data will be transmitted through one or several lanes in a sequence of bits individually, thereby reducing latency and increasing efficiency in data transmission.

The protocol is compatible and scalable to support various lane configurations, ranging from 1 (x1) lane to 32 (x32) lanes, allowing flexibility for different devices and applications to use the amount of bandwidth they need. The slot configurations can be found in different physical sizes corresponding to the number of lanes they support, such as x1, x4, x8, and x16.

In Table 1, we see the data transfer rates for the different generations of PCIe and their various configurations.

PCI Express Bandwidth (Dual-Simplex Link: GB/Second/Direction							
Slot Width	Transfer rate per lane	PCI 1.0 (2003)	PCI 2.0 (2007)	PCIe 3.0 (2010)	PCI 4.0 (2017)	PCI 5.0 (20019)	PCI 6.0 (20022)
x1	2.5 GT/s	0.250 GB/s	0.500 GB/s	0.985 GB/s	1.969 GB/s	3.938 GB/s	7.563 GB/s
x2	5.0 GT/s	0.500 GB/s	1.000 GB/s	1.969 GB/s	3.938 GB/s	7.877 GB/s	15.125 GB/s
x4	8.0 GT/s	1.000 GB/s	2.000 GB/s	3.938 GB/s	7.877 GB/s	15.754 GB/s	30.250 GB/s
x8	16.0 GT/s	2.000 GB/s	4.000 GB/s	7.877 GB/s	15.754 GB/s	31.508 GB/s	60.500 GB/s
x16	32.0 GT/s	4.000 GB/s	8.000 GB/s	15.754 GB/s	31.508 GB/s	63.015 GB/s	121.000 GB/s

Table 1 PCIe Bandwidth

1.2. History and Evolution of Attacks on PCIe

One of the first recorded attacks targeting the PCIe protocol was known as "Direct Memory Access (DMA)." This type of vulnerability or attack involves exploiting the capability of PCIe devices to read and write directly to memory without the need for this traffic or read/write transactions to pass through the CPU. In this way, it is possible to extract or manipulate system memory data, which can lead to obtaining sensitive or confidential information or even taking control of the affected system [1].

Another attack known as the "MMIO DoS Attack - Passthrough" is based on compromising system availability. This attack allows PCIe devices to be assigned directly to Virtual Machines (VMs). VMs executing DoS attacks on Passthrough devices can degrade the performance of I/O devices that share PCIe links with the DoS victim, which can affect concurrent VMs and the host [2].

To protect or attempt to mitigate these types of attacks, the industry and researchers have developed a series of protections, such as:

- Device Authentication – This involves verifying the identity of devices connected through the PCIe bus before allowing access to system resources. This can be achieved by implementing digital signatures, certificates, or directly by creating a whitelist of permitted devices.
- Secure Boot – This feature ensures that only trusted software is executed during the system's boot process by recognizing digital signatures. This prevents malicious devices from interfering with the boot process or injecting malicious code.
- Secure Virtualization – This refers to the creation of isolated and secure virtual environments where resources, such as PCIe devices, can be assigned to virtual machines without compromising the security of the host system or other virtual machines. This includes the use of technologies like IOMMU to protect against DMA attacks and the use of secure hypervisors that control access to hardware resources and manage the interaction between virtual machines and physical devices [3].

1.3. PCIe and CXL

Compute Express Link (CXL) is a high-speed, multi-protocol interconnect standard designed and specialized for data centers and high-performance computing environments. It is closely related to PCIe (Peripheral Component Interconnect Express) in several aspects, such as:

- Infrastructure – The physical and electrical layer specifications of CXL are based on the physical and electrical layer specifications of PCIe. This allows users to continue using the existing PCIe infrastructure without the need to add or modify existing hardware.
- Compatibility and Coexistence – The technical specifications of the CXL protocol are designed and developed to be compatible and coexist with PCIe. A great example of this is CXL.io, which will be detailed further in Chapter 2.
- Use of PCIe Lanes - Both protocols can utilize different lane configurations (x1, x4, x8, x16, etc.) to achieve the desired bandwidth, leveraging the flexibility and scalability of PCIe.
- Parallel Evolution - Both protocols are evolving in parallel, with PCIe continuing its development in terms of speed and efficiency, and CXL complementing these improvements with additional features for computing and memory.

1.4. Introduction to CXL protocol

Compute Express Link (CXL) is a high-speed, low-latency interconnect standard and multi-protocol that focuses on three key areas: CXL.io, CXL.mem, and CXL.cache, which will be defined and explained in more detail in Chapter 1. CXL has made significant advancements by providing a scalable interconnectivity solution for bandwidth and latency challenges in advanced computing centers. It plays a crucial role in leveraging the capabilities of next-generation devices and enhancing overall system performance.

1.5. Security in CXL

Some of the protection and security methods implemented in the protocol to ensure the integrity, confidentiality, and availability of data are as follows and will be discussed in more detail in Section 2

- CXL Integrity and Data Encryption (IDI)
- Device Authentication
- Secure Cache Coherence
- Protection Against Denial of Service (DoS) Attacks
- Use of IOMMU (Input-Output Memory Management Unit)
- Secure Firmware

1.6. Concepts of DoS (Denial of Service) y DDoS (Distributed Denial of Service)

A Denial of Service (DoS) attack aims to make a network resource, system, or service inaccessible to users by overwhelming it with junk traffic or requests to a specific system.

A Distributed Denial of Service (DDoS) attack has the same objective as a DoS attack. The difference lies in the scale and origin of the attack. A DoS attack seeks to compromise or disrupt system availability from a single point, whereas a DDoS attack uses a network of infected or hijacked devices to launch a massive, distributed, and synchronized attack.

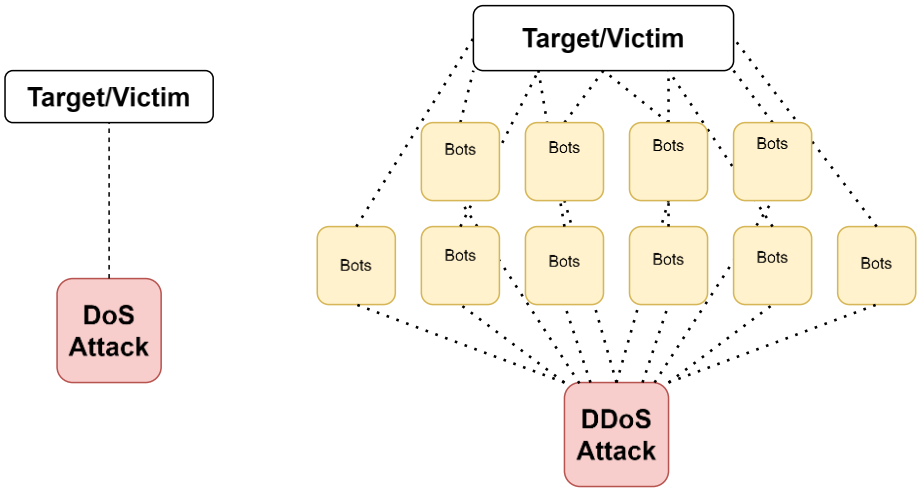

Fig 1.2 DoS vs DDoS

1.7. Impact of DoS/DDoS

Throughout history, this type of attack has had a significant impact on organizations, individuals, and the internet infrastructure in general. Some of the repercussions or objectives have been:

- Interruption of Services – Both in software and hardware, this implies the loss of availability, either partially by slowing down or making the system or resources intermittently accessible, or, more catastrophically, the total loss of the system.
- Economic Losses – Companies or systems affected by this type of attack can suffer direct economic losses due to the partial or total interruption of commerce. Additionally, the costs associated with mitigating these attacks and subsequent recovery can be significant.
- Reputational Damage – A successfully executed attack compromising system availability can damage the organization's reputation, diminish customer trust, and negatively affect the perception of the brand or organization.

Some important examples of DoS/DDoS attacks targeting hardware include:

Slowloris – This attack involves denial of service specifically to web servers by opening multiple HTTP connections to the target server. This is achieved by not completing the HTTP requests and sending partial headers or irregular intervals to keep the connections open.

Once multiple connections are simultaneously open, Slowloris exhausts the server's resources, such as processing threads and available connections, causing the server to be unable to generate new legitimate connections, resulting in a total denial of service rather than just a partial one.

To protect systems and servers from this type of attack, several measures have been implemented, such as adjusting the HTTP connection timeouts to close inactive connections more quickly and efficiently, limiting the number of multiple connections that can be generated from a single IP, and implementing reverse proxies and firewalls that can detect and block anomalous traffic patterns [4].

1.8. Mitigation and Prevention Strategies

Throughout history, some of the mitigation or prevention strategies against DoS or DDoS attacks have been the following:

- Implementation of Firewalls – Configuring and implementing firewalls to filter malicious traffic before it reaches hardware devices. These can block IP addresses categorized as suspicious and limit traffic originating from them.
- Intrusion Prevention and Detection Systems (IPS/IDS) – These are systems that can identify and block malicious traffic patterns.
- DoS/DDoS Mitigation Devices – Implementing and developing specialized and dedicated hardware for detecting and blocking high-magnitude attacks.
- Traffic Monitoring and Analysis – Implementing continuous monitoring to detect anomalous traffic patterns.

2. Compute Express Link

2.1. Introduction to the CXL (Compute Express Link) protocol

Compute Express Link (CXL) is a high-speed, multi-protocol interconnect standard designed and specialized for data centers and high-performance computing environments. It provides efficient communication between high-performance processors, such as Central Processing Units (CPUs) and Graphics Processing Units (GPUs), accelerators like Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs), and memory devices. Announced in March 2019 by a consortium led by Intel, its objective is to create an open interconnect standard aimed at addressing the limitations of existing protocols like PCIe in terms of coherence and cache.

Since its announcement, CXL has rapidly gained industry support, with leading companies such as AMD, ARM, Google, Microsoft, and NVIDIA joining the consortium. The first official specification, CXL 1.0, was released in 2019, followed by CXL 2.0 in 2020, which introduced significant improvements such as the ability to share memory between multiple hosts and support for persistent memory devices. The CXL 3.0 specification, released in 2022, introduced new features such as mesh topology to enhance scalability and efficiency in data centers. The future of CXL focuses on integration with emerging technologies and expanding its use in artificial intelligence, machine learning, and high-performance computing applications.

Evolution and Future of CXL – The evolution of the protocol has advanced by leaps and bounds, always maintaining a focus on interoperability and performance. The key points that have enabled this growth are:

- Performance Improvement
- Latency Reduction
- Scalability
- Flexibility and Efficiency
- Compatibility with PCIe

2.2. CXL System architecture

CXL is a high-performance input/output bus architecture designed to connect peripheral devices. These devices can include traditional non-coherent I/O devices, memory devices, or accelerators with additional capabilities. The types of devices that can be connected via CXL and the general system architecture are detailed below:

2.2.1. CXL Type 1

CXL Type 1 devices have unique requirements that benefit from having a fully coherent cache on the device. For instance, a device that needs to perform complex atomic operations, which are not included in the standard set of atomic operations available in PCIe, would find this particularly valuable. Basic cache coherence allows an accelerator to implement any ordering model it prefers and enables the execution of an unlimited number of atomic operations. CXL Type 1 devices are acceleration devices that utilize the capabilities of CXL.io and CXL.cache.

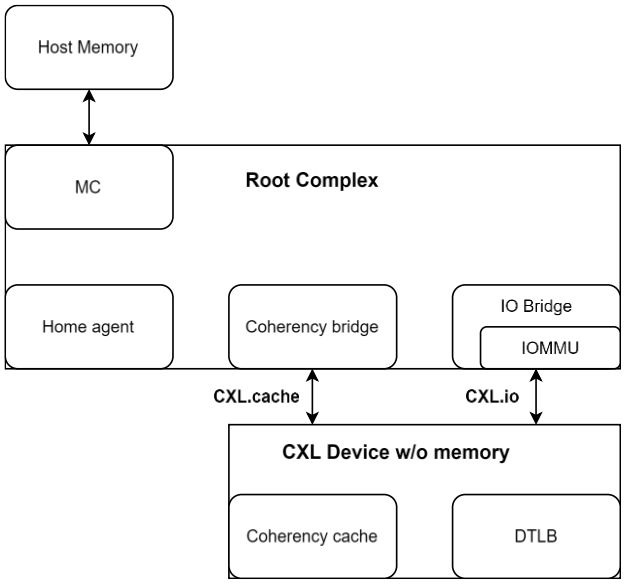

Fig 2.1 Type 1 CXL Device

2.2.2. CXL Type 2

CXL Type 2 devices support CXL.cache, CXL.mem, and CXL.io. In addition to having a fully coherent cache, these devices also have memory (e.g., DDR, High Bandwidth Memory (HBM), etc.) attached to the device. Their performance comes from the enormous bandwidth between the accelerator and the memory attached to the device. The primary goal of CXL is to enable the host to send operands to the memory attached to the device and retrieve results from that memory without adding software and hardware costs that negate the benefits of the accelerator.

An example of such a device is a GPGPU with attached GDDR. These devices treat the memory attached to them as private, meaning that this memory is not accessible to the host and does not maintain coherence with the rest of the system. It is entirely managed by the hardware and the device controller, primarily serving as a buffer for the device when handling large data sets. The clear disadvantage of this model is that it necessitates high-bandwidth copies between the host memory and the device-attached memory, as operands are transferred in and results are written out.

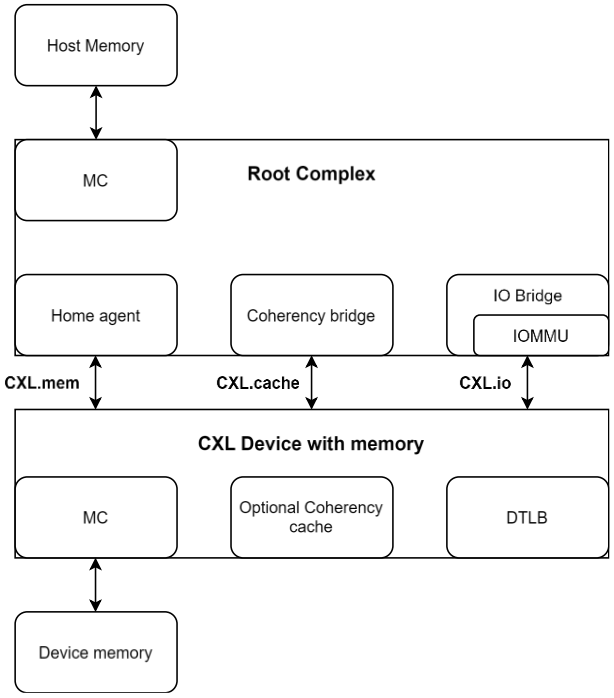

Fig 2.2 Type 2 CXL Device

2.2.3. CXL Type 3

A CXL Type 3 device supports the CXL.io and CXL.mem protocols. An example is an HDM-H memory expander for the host. A passive memory expansion device uses the HDM-H memory region and typically does not directly manipulate the memory contents while it is exposed to the host.

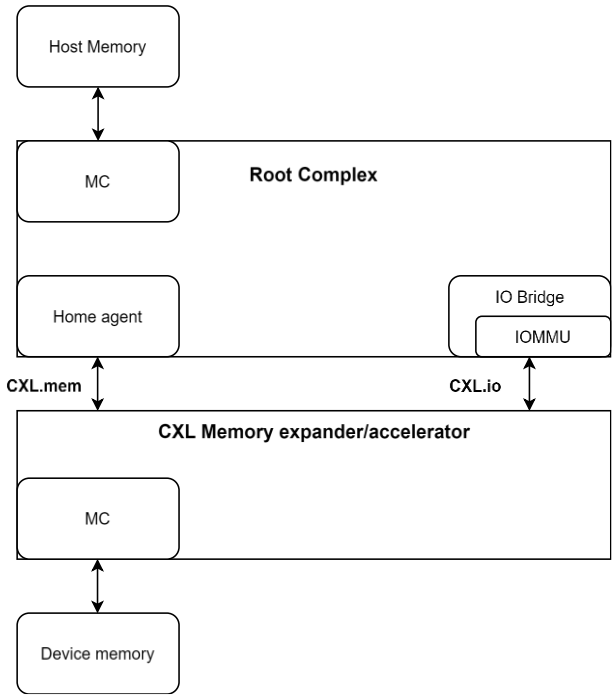

Fig 2.3 Type 3 CXL Device

2.3. CXL Transaction Layer

The architecture of CXL is based on three main components:

2.3.1. CXL.cache

This protocol focuses on cache coherence between the CPU and connected devices, such as accelerators and memory. For example, in a system using CXL.cache, an accelerator like a GPU

can access and cache data from the system memory coherently with the CPU cache. This means that both the CPU and the accelerator can work with the same copy of the data without conflicts or inconsistencies, thereby improving system performance and efficiency.

2.3.2. CXL.io

This handles basic input/output and device initialization. An example of CXL.io would be its use in device enumeration and the initial configuration of an accelerator connected via CXL. When a system starts up, CXL.io allows the CPU to discover and configure the accelerator, assigning resources such as memory space and establishing how they will communicate. This is similar to how PCIe works for device initialization and configuration, but CXL.io is optimized to work more integrally with other CXL protocols, such as CXL.cache and CXL.memory, for superior performance in heterogeneous systems.

2.3.3. CXL.mem

It handles memory semantics and memory access operations between the CPU and connected devices. An example of CXL.mem would be its use in a system where an accelerator needs to access system memory directly and efficiently. With CXL.mem, the accelerator can read and write to system memory without going through the CPU, reducing latency and increasing overall system performance. This is especially useful in data analysis and machine learning applications, where quick access to large datasets is critical.

Physical Infrastructure – CXL is not only compatible with PCIe infrastructure but also uses it as a foundation. This allows for its integration with existing systems without the need for significant hardware changes, while still benefiting from the virtues and advantages of CXL as a high-speed interconnect protocol used to connect peripheral devices to the motherboard.

2.4. CXL Security

Some of the protection and security methods implemented in the protocol to ensure the integrity, confidentiality, and availability of data are as follows:

- CXL Integrity and Data Encryption (IDE) defines mechanisms to ensure confidentiality, integrity, and replay protection for data traversing the CXL link. The cryptographic schemes employed are aligned with industry best practices. CXL IDE supports various usage models and guarantees broad interoperability. It can be used to secure traffic within a Trusted Execution Environment (TEE) composed of multiple components[11]
- Device Authentication – By implementing authentication mechanisms, the identity of connected devices is verified before allowing them access to system resources. This prevents unauthorized or malicious devices from connecting and ensures that only trusted devices can interact with and access system resources.
- Secure Cache Coherence – The cache coherence mechanisms in CXL not only ensure that data is consistent between the CPU and devices but also include security measures to protect against unauthorized access and tampering. This helps maintain data integrity and prevents corruption of shared memory.
- Protection Against Denial of Service (DoS) Attacks – The CXL protocol includes measures and mechanisms to mitigate DoS attacks, such as request rate limiting and implementing timeouts for inactive connections. This helps prevent a malicious device from overloading the system and compromising or affecting its availability.
- Use of IOMMU (Input-Output Memory Management Unit) – CXL benefits from the use of IOMMU to manage and isolate memory access. The IOMMU maps virtual memory addresses to physical addresses, providing an additional layer of security that prevents devices from accessing unauthorized memory areas. This also protects against Direct Memory Access (DMA) attacks and ensures that devices can only access assigned memory areas.
- Secure Firmware – The use of secure and up-to-date firmware on connected devices is promoted. This includes verifying digital signatures and applying security patches to fix known vulnerabilities. This reduces the risk of firmware-based attacks and ensures that devices operate with reliable and secure software.

2.5. Impact on Industry Performance and Future of CXL

Compute Express Link (CXL) has significantly improved performance in high-performance computing systems and modern data centers. By offering high-speed, low-latency interconnection between the CPU and memory and acceleration devices, CXL increases the efficiency and responsiveness of critical applications. The cache coherence mechanisms and shared memory access enable faster and more efficient communication, reducing latency and improving overall system performance. This is especially beneficial for applications that require intensive data processing, such as artificial intelligence, machine learning, and big data analytics.

In the future, CXL is expected to continue evolving and expanding. Future versions of the protocol, such as CXL 3.0, will introduce new features and improvements that will increase scalability and efficiency. These improvements include a mesh topology for better interconnection between multiple devices and integration with emerging technologies such as persistent memory and specialized hardware accelerators. The growing adoption of CXL by the technology industry and its compatibility with the existing PCIe infrastructure ensure that CXL will play a crucial role in the next generation of data centers and high-performance computing systems [5].

3. Threat Modeling

Threat Modeling is a structured and defined methodology used for identifying, evaluating, and mitigating potential security threats to a system, application, or infrastructure. It can be used to shape the design and meet security and reliability objectives. This process involves creating a model that analyzes and describes potential threats and the necessary countermeasures to mitigate them.

The primary objective is to provide a clear and understandable view of potential security risks, enabling the development and security team to implement proactive measures to protect the system.

Why Follow This Methodology?

Threat Modeling helps improve the security of systems and applications by identifying risks and implementing appropriate mitigation measures [6]. Some of the key points offered by this methodology are:

Holistic Security View: Provides a comprehensive view of security risks, considering both internal and external threats.

Cost Reduction: By proactively identifying and mitigating threats, the costs associated with security breaches and service interruptions can be avoided.

Facilitation of Decision-Making: Provides valuable information for making informed decisions about the implementation of security measures and risk management.

Adaptability and Scalability: Threat modeling can be adapted to different types of systems and scales, from small applications to large, complex infrastructures.

Early Detection of Vulnerabilities: Enables the early detection of vulnerabilities, facilitating their correction before they can be exploited by attackers.

The points that comprise this methodology will be discussed below.

3.1. Identification of Assets

Some of the susceptible assets can be systems focused on Artificial Intelligence, Machine Learning, or big data analysis. The use of the CXL protocol is essential to increase and improve the efficiency of these tasks by efficiently handling the processing and analysis of large volumes of data. Some examples of the volumes that these types of systems can present include:

Systems Focused on Social Media Data Analysis – Some platforms like Facebook, Instagram, or Twitter generate and process enormous amounts of data daily, such as posts, comments, likes, and other user-generated data. This is done with the intention of better understanding user behavior, detecting trends, and offering targeted content to users. The transactions resulting from this processing and analysis can include:

- Facebook – Generates around 4 petabytes of data per day, handling over 1 million interactions (likes, shares, comments) per minute [7].
- Twitter – Processes approximately 500 million tweets per day, around 6,000 tweets per second. Each tweet has an average size of 200 bytes, which translates to approximately 1.2 megabytes per second just in tweets [8].

Systems Focused on Training AI and ML Models – Training models focused on Artificial Intelligence (AI) and Machine Learning (ML) requires processing large volumes of data to adjust and mature the model parameters, performing analysis and processing of images, text, and audio. Some examples of the volumes of data processed include:

- ImageNet: Contains over 14 million labeled images, occupying several terabytes of storage. During training, models process thousands of images per second. Each image can have an average size of 200 kilobytes, which translates to several gigabytes per minute during training [9].
- GPT-3: During training, the model processes millions of text tokens per second. It is trained with 570 gigabytes of text. Each text token has an average size of 2 bytes, which translates to several gigabytes per minute during training [10]

3.2. Definition of test enviroment

As mentioned earlier, the test environment is a fully controlled and secure setting that will be used solely for research purposes. At no point is an unauthorized or restricted system compromised. The components to be used are as follows:

Processor: 5th Gen Intel Xeon processors

CXL Test Card: Astera Labs Leo CXL Memory Expansion Card with DIMMs (Fig 3.1)

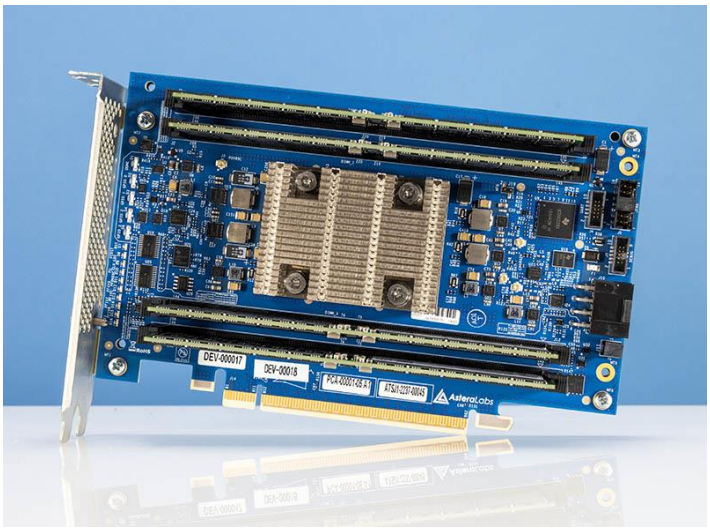

Fig 3.1 CXL test card - Astera Labs Leo

3.3. Identification of Threats and Potential Attackers

For the case being presented, we will focus on addressing the threat of Denial of Service (DoS) attacks. Historically, potential attackers are not limited to a particular group or entity, as over time, individual hackers, hacktivist groups, cybercriminal groups, and even nation-states have perpetrated this type of attack on various platforms/technologies for different purposes.

3.4. Identification of Attack Vectors and Selection of DoS Attack

The attack vectors that will be targeted or exploited are as follows:

Flooding the CXL.mem link with malicious/junk traffic

- The objective for this attack vector is to flood the CXL link with a large amount of malicious or junk traffic to deny or degrade the communication capacity between the CPU and the CXL devices connected to the system. CXL is designed to provide high speed and low latency, but it has a capacity limit. Flooding the link with malicious traffic can exceed this capacity, causing congestion. This results in significant delays (increased latency) and potentially disrupts communication between devices, denying service to legitimate users.

Exploitation of Vulnerabilities in Shared Memory Management

- Using the same concept mentioned in the previous point, the aim is to exploit the partial or total denial of service to legitimate users, but in this case by exploiting vulnerabilities in the management of shared memory provided by CXL. Shared memory in systems allows 1 to N virtual machines/instances under the same system (under the same hardware). This enables the CPU and other devices to access the same memory, improving efficiency and performance. However, if not managed properly, it can become a point of vulnerability.
- Similarly, by injecting junk or malicious traffic, the goal will be to degrade or deny service. The difference, when hosting multiple virtual instances under the same server, is that if a user manages to exploit this vulnerability, all other legitimate users will experience high latency in their transactions and, in the most catastrophic case, a total denial of their services.

3.5. Attack Desing

The attack can be divided into three main sections:

3.5.1 Search and Identification of the Potential Device to Attack

For this section, it is important to first mention how the "System Address Map" is constituted or functions. This is a very important concept and component when discussing computer system architecture. Its relevance lies in the fact that it is a structure or table that defines how memory addresses and I/O resources will be assigned coherently and efficiently, allowing correct and rapid communication between the CPU and connected devices. In Fig. 3.2, we can see a representation of what this table might look like. It is also important to mention that this can vary depending on the architecture and manufacturer.

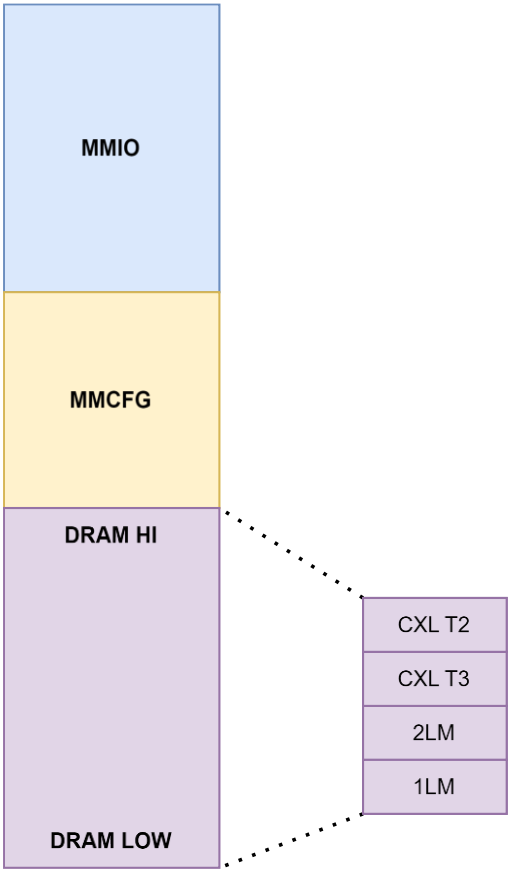

Fig 3.2 Address Map

We have several ways to obtain a list of the elements connected to our system, for example, using lspci in Unix/Linux environments. In the case of this experiment, we can obtain the following information about our device as shown in Table 3.1:

CXL DEVICES:

d#	BDF	DID	RID	TC_Name	ImgRev	SPEC typ	Status	Base	Size	Intlv
1	27:00.0	01e0	01	ASTRA Leo	0.75	2.0	3 ?	1080000000	28 GB	256B

Total CXL memory = 128 GB

3.5.2 Initiate Data Injection

Once we have identified the potential CXL Type 3 devices that could be a target, it is necessary to understand how memory transactions occur. The HDM-H address region in the CXL Type 3 device being used in this case will be used as a memory expander. This allows transaction flows to HDM-H to be simplified into just two classes: reads and writes originating from CXL.mem or a specific device, as shown in Fig. 3.3.

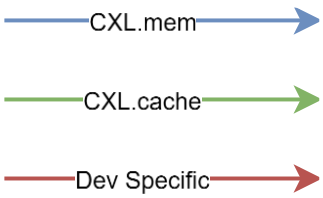

Fig 3.3 legend for the diagrams that follow

In Fig. 3.4, the optimized read flow for the HDM-H address region is shown. In this flow, only a data message is returned.

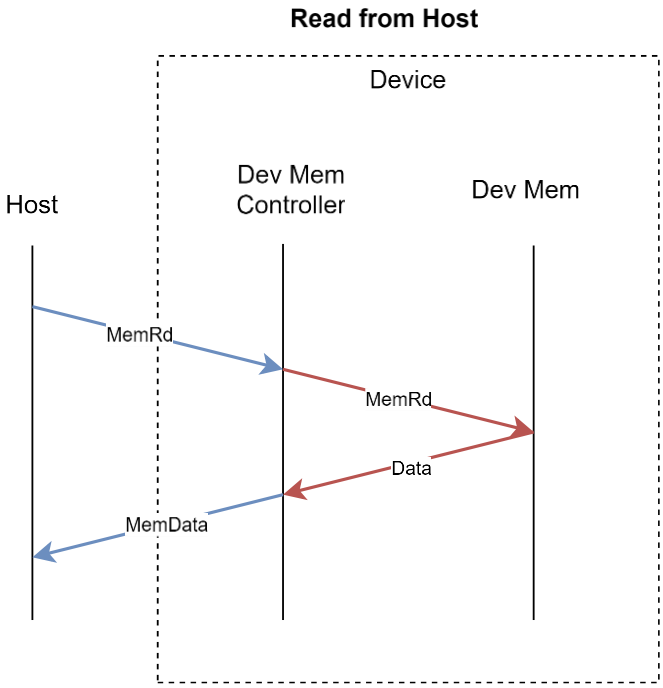

Fig 3.4 Read from Host to HDM-H

Writes to the HDM-H region use the same flow as the HDM-D/HDM-DB region and are always completed with an S2M NDR Cmp message. This write flow is shown in Figure 3.4.

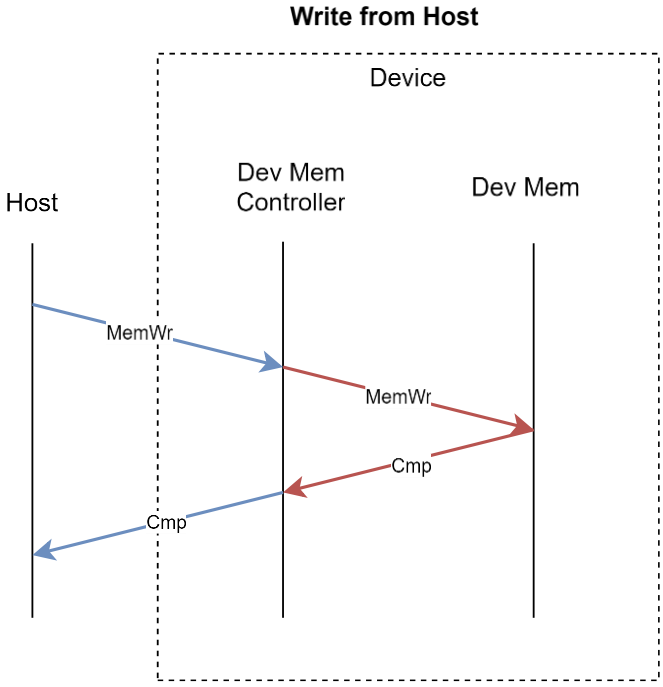

Fig 3.4 Write from Host to HDM-H

3.5.3 And finally, system monitoring

For monitoring how and how many resources our attack is demanding, a simple bash script will be implemented. This script will collect the percentage of Type 3 memory usage in the system, which will later be analyzed and processed in the Results and Analysis of the Attack section.

En la figura 3.5 podemos observar de manera resumida los flujos anteriormente mencionados y como es que están interactuando entre ellos, un punto que no se menciona anteriormente es que de no encontrarse un dispositivo CXL en el sistema el ataque no se llevara a cabo y simplemente se finalizara el programa

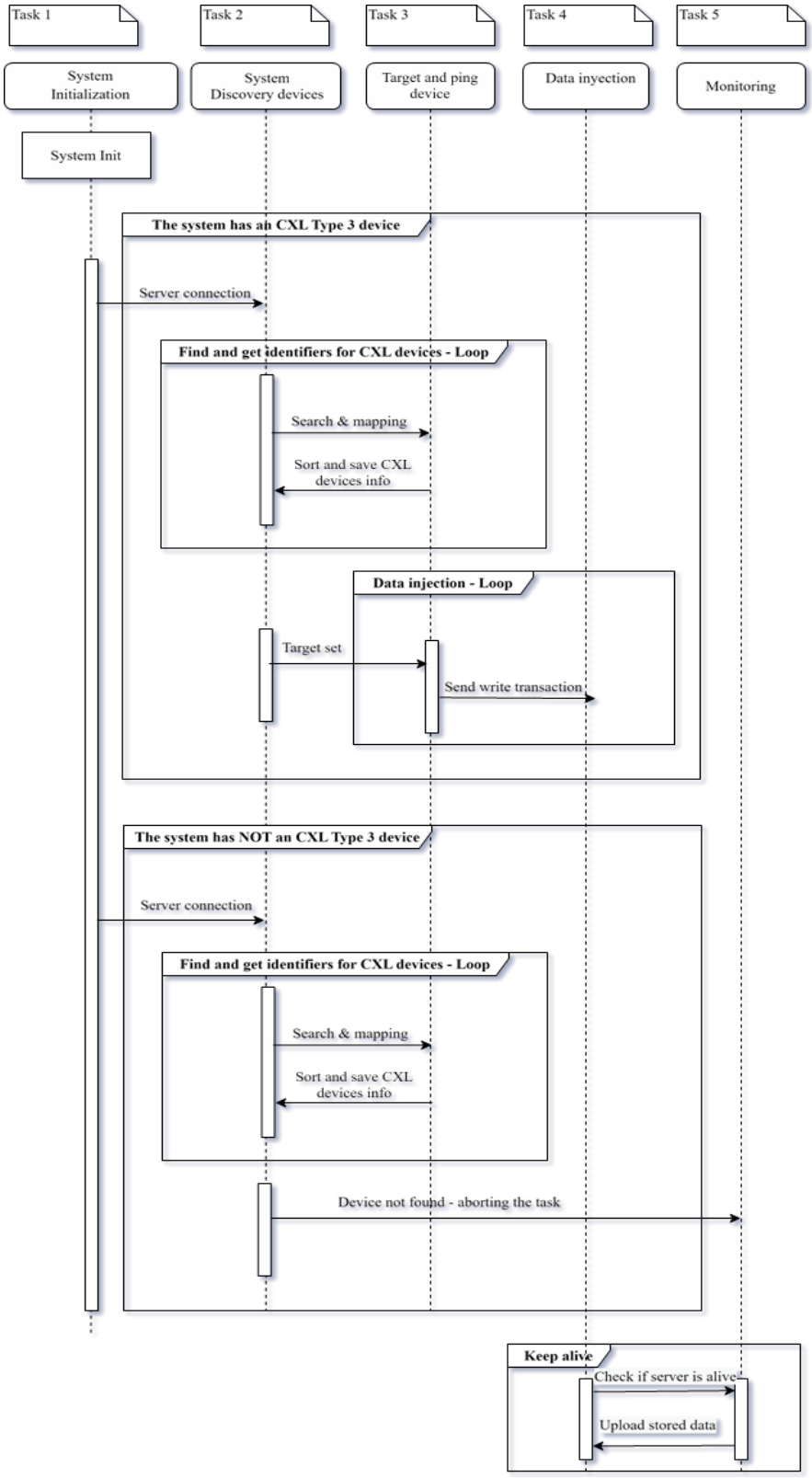

Fig 3.5 General attack flow

3.6 Execution and analysis of the Attack

Before starting the injection of transactions, in Fig. 3.6 and Fig. 3.7, we can see the percentage of resources being used with the system in an idle state.

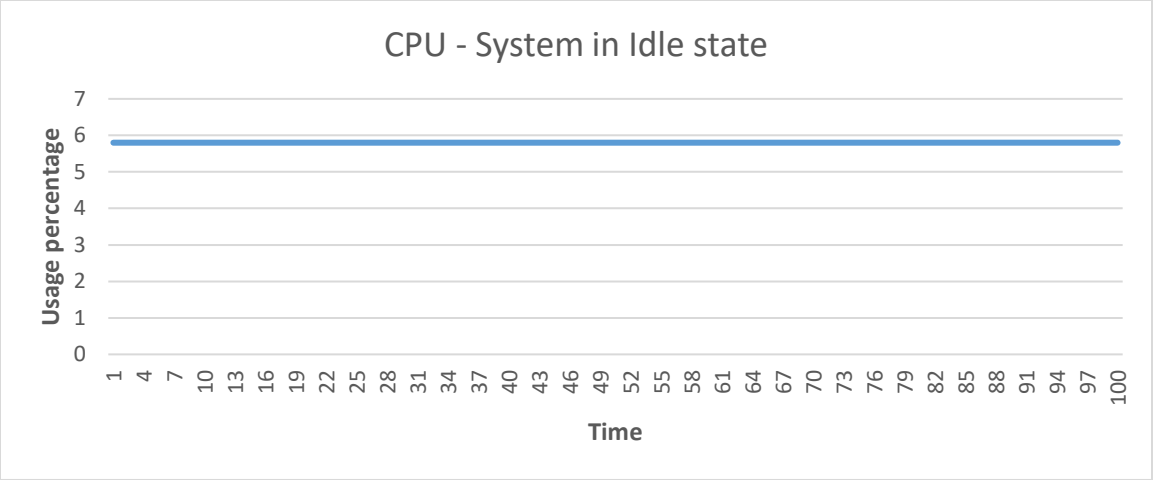

Fig 3.6 CPU utilization percentage – Idle State

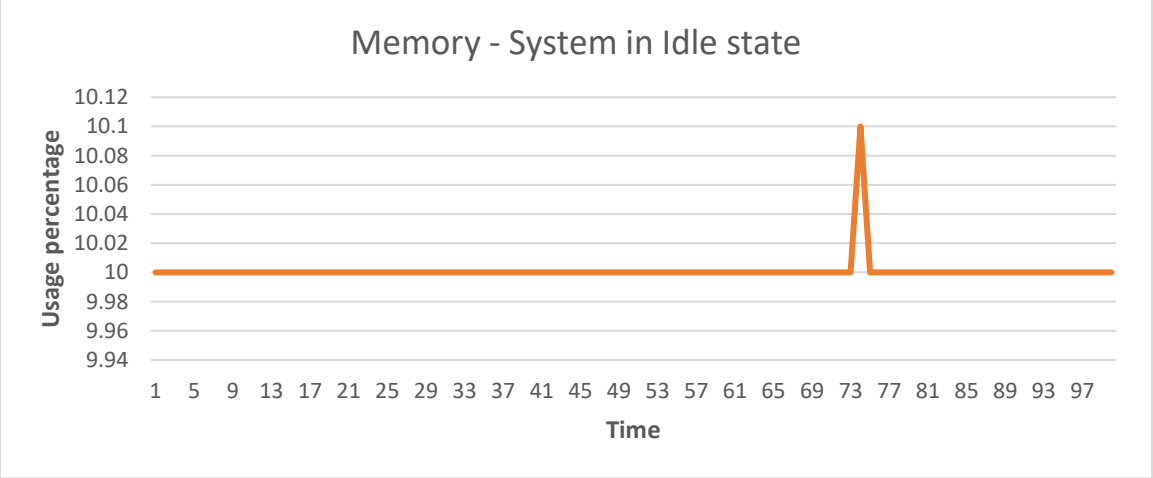

Fig 3.7 Memory utilization percentage – Idle state

It can be observed that when the platform is in an idle state, only 6% of the CPU and 10% of the total available memory resources are being used. This utilization percentage corresponds to essential kernel and operating system tasks. Once transactions begin, we can see in Fig. 3.8 and Fig. 3.9 how the demand for system resources increases, both for the CPU and the memory attached to the CXL device.

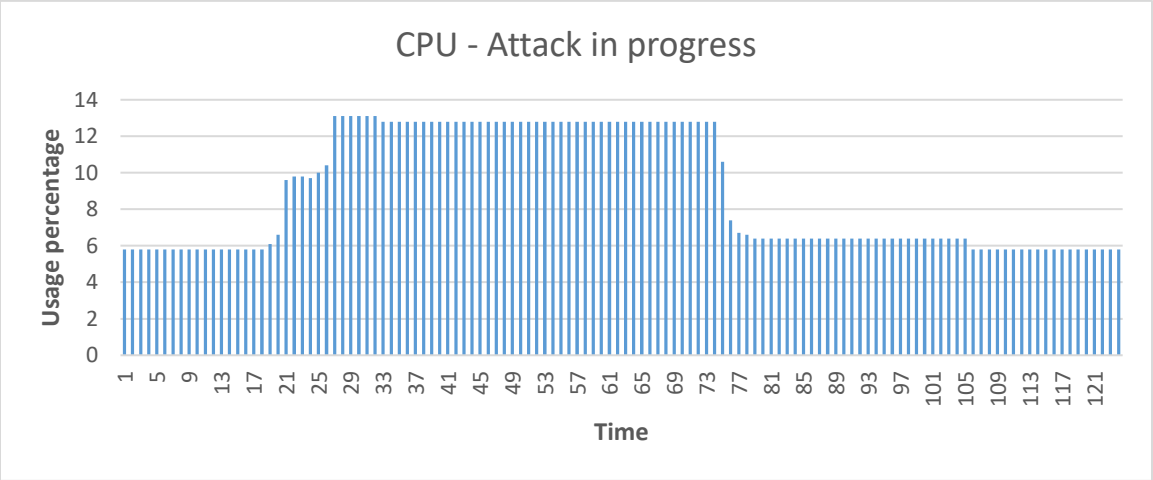

Fig 3.8 CPU utilization percentage – Attack State

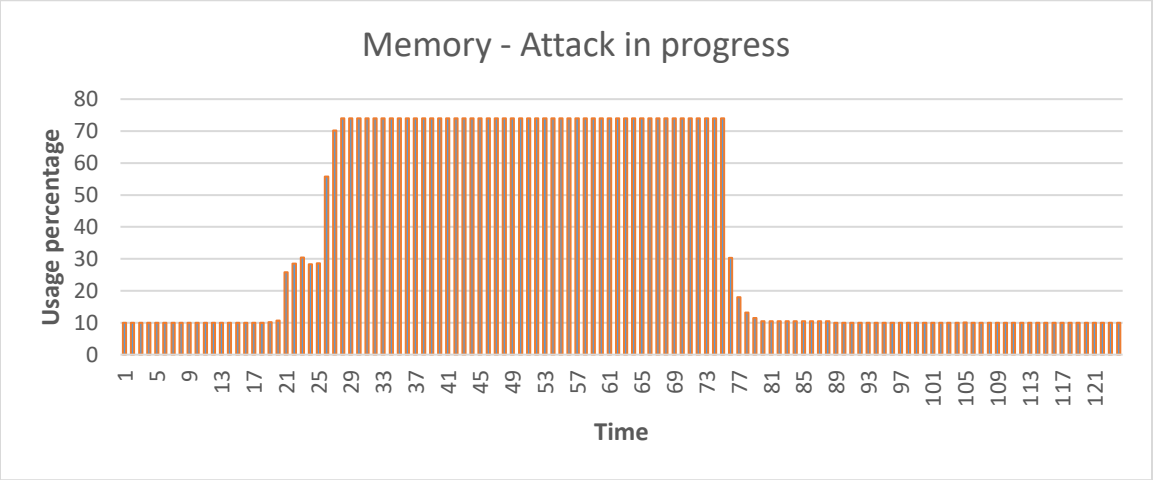

Fig 3.9 Memory utilization percentage – Attack State

While the attack or junk transaction injection is being implemented, it can be observed that the CPU demand increases from 6% to approximately 13%. This is due to the transactions being generated towards the memory. In the case of memory, we can see a more significant demand for resources, increasing from approximately 10% to a maximum of 74% during the transaction injection.

In Fig. 3.10, the difference in resource demand can be clearly observed when comparing the idle state to when the attack transactions are being injected.

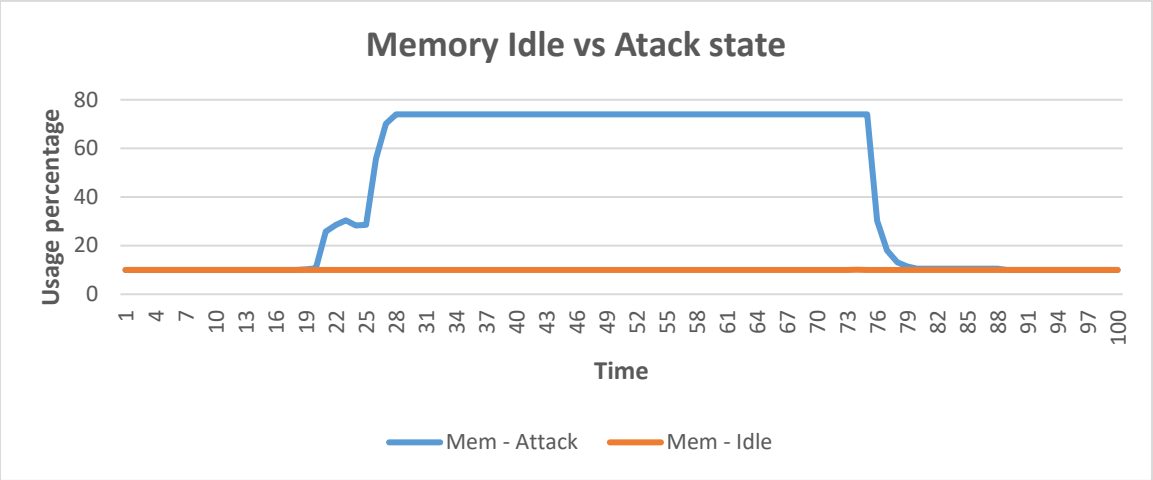

Fig 3.10 Memory utilization percentage – Idle vs Attack State

As shown in Figs. 3.6 to 3.10, although it was not possible to completely compromise the availability of the CXL link, it was possible to degrade availability by approximately 74%. Once the potential legitimate user begins to generate their traffic or transactions, they may experience high latency and possibly a loss of some services.

Now, another couple of important points to mention for future improvements or implementations of this project:

The first point is that the development and implementation of this project were carried out in a controlled laboratory with hardware and software fully dedicated to the research of this project. Therefore, at no point was a system compromised that was not intended for the purposes. With that clarified, all transaction injections were performed locally as the root user. However, is it possible to execute these transactions on a larger scale?

To answer this question, there are a couple of points to analyze. Although the entire program responsible for searching and selecting potential CXL devices and injecting CXL transactions is developed in C, which would allow us to generate a binary for execution of just a few kilobytes, implementing this attack on real systems would require access to execute this binary. This already represents a potential impediment to executing this attack on a larger scale, as services like Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and others manage permissions and access as a crucial part of system security and administration. They govern access to these permissions under the principle of least privilege, which means that users only have the essential permissions to carry out their tasks.

Possible or Hypothetical Situations Where This Attack Could Be Carried Out on a Larger Scale

The first hurdle an attacker would face is achieving privilege escalation to become a root user on the platform. If this privilege escalation were somehow achieved, it would indeed be feasible to carry out this attack on a larger scale.

Conclusions and future recommendations.

Throughout this document, the significant threat that a DoS attack on the CXL protocol, specifically CXL Type 3, can represent has been explored. With the growing demand in modern industry for applications that require high data transfer, processing, and analysis capacity for applications such as artificial intelligence, machine learning, data analysis, or even the growing trend of serverless computing (such as AWS, Azure, or Google Cloud), the impacts could translate into losses of millions of dollars.

Although the implementation achieved in this case did not completely compromise the system's availability, it did result in a degradation of availability, rendering approximately 74% of the system's resources (in terms of memory) unusable. Similarly, even though this type of attack does not compromise 100% availability for a legitimate user, it still represents a more than considerable degradation. This leads to consequences such as partial interruptions in communication between the CPU and connected devices, affecting critical applications and essential services, delays in shared memory management, which ultimately also translates into economic losses for organizations, both in terms of direct revenue and costs associated with recovery and mitigation. Additionally, this can affect user trust in the affected services, damaging the reputation of the organizations.

Bibliography

[1] Piegdon, David Rasmus (2006-02-21). Hacking in Physically Addressable Memory - A Proof of Concept (PDF). Seminar of Advanced Exploitation Techniques, WS 2006/2007.

[2] Richter, A., Herber, C., Wild, T., & Herkersdorf, A. (2015). Denial-of-Service attacks on PCI passthrough devices: Demonstrating the impact on network- and storage-I/O performance. Journal of Systems Architecture, 61(10), 592-599. https://doi.org/10.1016/j.sysarc.2015.07.003

[3] Popek, G. J., & Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures. Communications of the ACM, 17(7), 412-421. https://doi.org/10.1145/361011.361073

[4] Yampolskiy, M., Horvath, P., & Koutsoukos, X. (2018). Security of Cyber-Physical Systems in the Internet of Things Era. IEEE Transactions on Cybernetics, 48(12), 3309-3322. https://doi.org/10.1109/TCYB.2018.2854590

[5] Yao, Y., & Li, J. (2020). Enhancing Data Center Performance with Compute Express Link (CXL). Proceedings of the IEEE International Conference on High Performance Computing. https://doi.org/10.1109/HPCC.2020.00023

[6] Shostack, A. (2014). Threat Modeling: Designing for Security. Wiley.Swiderski, F., & Snyder, W. (2004). Threat Modeling. Microsoft Press.

[7] Facebook Research. (n.d.). Retrieved from https://research.fb.com/

[8] Twitter Developer. (n.d.). Twitter API Documentation. Retrieved from https://developer.twitter.com/en/docs/twitter-api

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 248-255). IEEE. Retrieved from http://www.image-net.org/

[10] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165. Retrieved from https://openai.com/research/gpt-3

[11] Compute Express Link Consortium. (2023). CXL Specification (rev. 3.1, ver. 1.0, Release Candidate). Public Clean.