OBJECT CAPABILITY MODEL FOR TEE: A CHERI BASED COMPARTMENTALIZATION APPROACH

Bala Subramanyan

Verifoxx Ltd, London, UK

ABSTRACT

In this paper, we introduce a capability-driven approach to bolster security and granularity within Trusted Execution Environments (TEEs) [1]. By delivering precise privilege control and fine-grained compartmentalization, we aim to improve TEE security standards.

To address vulnerabilities within Trusted Execution Environments (TEEs) and enable selective privilege management and secure object sharing between secure and normal worlds, we introduce a TEE compartmentalization framework based on the CHERI object-capability model. Leveraging DSbD technologies, our framework provides an efficient prototyping environment for developing trusted applications while safeguarding against existing threats.

At Verifoxx Ltd, our architecture relies on TEEs to handle sensitive data, encompassing tasks such as extracting client secrets, managing commitments, sharding and executing cryptographic operations for zero-knowledge responses. The proposed approach holds promise where TEEs can enhance transaction security and enterprises seeking data protection.

Our approach introduces in-enclave compartments with controlled communication, facilitating domain transitions through sealed data capability delegations and hardware-assisted call/return mechanisms. This enables application layer compartmentalization by modularly separating concerns within the secure world, emphasising single responsibility, least privileges, and information hiding from unprivileged compartments. Furthermore, we ensure the integrity of lower-layer hardware and OS properties, effectively thwarting compromise attempts.

KEYWORDS

Trusted Execution Environments (TEE) [1], CHERI architecture [8][7], Object Capability Model, Compartmentalization, Memory Protection, DoS, Application Layer Security

1. INTRODUCTION

Trusted Execution Environments (TEEs) [1] have emerged as a critical component in enhancing the security and privacy of modern computing systems. However, like any technology, TEEs are not immune to vulnerabilities and threats. Addressing these challenges is paramount to ensuring the continued trustworthiness of TEEs, particularly in environments where sensitive data and critical operations are commonplace.

This paper presents a novel approach to fortifying TEEs by introducing a capability-based compartmentalization framework. The framework is built upon the principles of Capability Hardware Enhanced RISC Instructions (CHERI)[8][7]. It provides a robust and secure environment for developing trusted applications within TEEs, offering precise control over

privileges and the secure sharing of objects within as well as between the secure and normal worlds.

In today's digital landscape, where data breaches and cyberattacks continue to threaten individuals and organisations, the need for stronger TEE security is more evident than ever. The aspects of our work encompass the creation of in-enclave compartments with controlled communication, enabling secure domain transitions through sealed data capability delegations and hardware-assisted mechanisms. Furthermore, it aims to guarantee the integrity of lower-layer hardware and OS properties, effectively thwarting attempts at compromise.

By building on a capability architecture, our framework aims to minimise the attack surfaces by decomposing applications into isolated compartments with granularly selected access. Furthermore, we harness CHERI memory protection[6] to mitigate vulnerabilities associated with memory leaks, shared memory space, and address space isolations. By circumventing conventional kernel system calls for compartment delegations and empowering user threads with suitable capabilities, we significantly reduce the kernel attack surface, thereby preventing insecure threading and procedure calls. Our compartmentalized enclaves with assured pipelines are designed to tackle privilege separation-related attacks.

In the subsequent sections of this paper, we provide an exploration of our capability-based TEE compartmentalization concept and implementation, offering insights into its architecture, security features, and its potential impact on the broader technology landscape.

2. BACKGROUND

2.1. Trusted Execution Environment (TEE)

Trusted Execution Environments[1] are a pivotal component of modern computing systems, designed to create secure and isolated enclaves within a processor where sensitive code and data can be executed and stored. TEEs are engineered to provide a high degree of security and confidentiality, serving as a safeguard against various forms of attacks and data breaches. They have become indispensable in numerous applications, particularly those where security is paramount.

While TEEs are essential for enhancing the security of various applications, as illustrated in Figure 1 below, TEEs have faced significant challenges and vulnerabilities including:

- **Insecure Threading and Procedure Calls**: Traditional TEEs often rely on the underlying operating system for managing threads and procedure calls. This dependency has led to vulnerabilities such as race conditions and code injection attacks, where malicious threads can compromise the integrity of TEE-enabled applications.
- **Memory Vulnerabilities**: Buffer overflows and memory-related vulnerabilities have been exploited to compromise the confidentiality and integrity of TEEs. Attackers can manipulate memory to gain unauthorised access to secure data or execute arbitrary code.
- **Return-Oriented Programming (ROP) Attacks**: TEEs have been susceptible to ROP attacks, where attackers construct malicious code sequences using existing functions in the TEE's memory. This technique allows them to evade security checks and execute unauthorised operations.
- **Inadequate Address Space Isolations**: Insufficient separation of address spaces between the secure and normal worlds within TEEs has exposed vulnerabilities.

Attackers can exploit these weaknesses to gain access to sensitive data or escalate privileges.

- **Secret Leaks Due to Insecure Shared Memory Buffers**: TEEs often use shared memory buffers for communication between secure and normal worlds. However, if these buffers are not adequately protected, they can become targets for data leakage attacks.
- **Insufficient Privilege Separation**: TEEs typically enforce a broad privilege model, where all enclaves within the TEE share the same set of rules and instructions. This lack of fine-grained privilege separation increases the attack surface and makes it challenging to prevent lateral movement by attackers.
- **Horizontal Privilege Escalation Attacks**: Attackers have exploited TEE vulnerabilities to escalate their privileges horizontally, gaining unauthorised access to other compartments or critical parts of the system.
- **Host Kernel Memory Modifications (BOOMERANG Attacks)**: Certain attacks, like BOOMERANG attacks, aim to modify the host kernel's memory from within the TEE. This represents a severe security breach, as it can compromise the entire system's integrity.

Summary of analyzing TEE-enabled applications and known attacks.

	Category	Num	Thread	Memory	IPC/RPC	Priv
TEE-enabled Applications	Reference monitor & Auditing	8	◐	●	●	●
	Web apps	7	●	●	●	●
	Data analytics	5	●	◐	●	●
	Key management	4		◐	◐	◐
	Attestation	2		●	◐	◐
	Databases	4	●	◐	●	●
	SSL/TLS	5	◐	●	●	●
	Blockchain	6	◐	●	●	●
	HPE [76]	95			●	●
	BOOMERANG [51]			●	●	◐
	COIN Attacks [41]	10	●	●	●	◐
TCB Vulnerabilities	CVE-2019-1010298		◐	●	●	◐
	CVE-2018-11950		◐	◐	●	●
	CVE-2017-8252		◐	◐	◐	●
	CVE-2017-8276		◐	◐	●	●
	CVE-2016-10297		●	◐	●	◐
	CVE-2016-5349		◐	●	◐	●
	CVE-2016-2431			●	●	●
	CVE-2015-4422			●	◐	●

● Object/feature is involved, ◐ partially involved
Priv: Inadequate Privilege separation

Figure 1: TEE Vulnearability Summary

2.2. Capability Hardware Enhanced RISC Instructions (CHERI)

Capability Hardware Enhanced RISC Instructions (CHERI)[8] improves software security by extending traditional Instruction-Set Architectures (ISAs)[7] with innovative capability-based primitives. This architectural paradigm shift is rooted in the fundamental security principles of "least privilege" and "intentional use."

Key aspects of CHERI include:

- **Fine-Grained Memory Protection**: CHERI introduces fine-grained code protection through the use of in-address-space memory capabilities, which replace conventional integer-based virtual address representations of code and data pointers. This fine-grained protection operates at the instruction level, limiting the potential damage that can result from software bugs. CHERI capabilities safeguard the integrity and origin of pointers themselves, ensuring the secure handling of in-memory data and code to which the pointers refer. This approach provides robust protection against a wide array of memory- and pointer-based vulnerabilities and exploits, including buffer overflows, format-string attacks, and control-flow manipulation.

- **Software Compartmentalization**: Secure encapsulation is another key aspect of CHERI's security model, allowing for the isolation of larger software components. This is achieved through the efficient implementation of in-address-space software compartmentalization, often referred to as object capabilities. Object capabilities enable the explicit definition of isolation boundaries and communication channels within software. While this approach necessitates clear software structure and communication descriptions, it significantly reduces the risk of application-level vulnerabilities, such as logical errors or malicious code introduced via software supply chains.

CHERI has been primarily associated with research and development efforts aimed at improving the security and robustness of computing systems and is designed to facilitate incremental adoption within existing security-critical software systems, including operating system kernels, critical libraries, language runtimes, and applications. Its innovative memory protection mechanisms[6] have the potential to significantly enhance the security posture of Trusted Execution Environments, making it a valuable technology for securing TEE-enabled applications.
By incorporating the principles of CHERI, our proposed capability-based compartmentalization framework aims to leverage these advancements in memory protection and access control to create a more secure and resilient TEE environment, mitigating the limitations and vulnerabilities associated with conventional TEEs. Morello [3] is an Arm experimental platform for evaluation of CHERI in the Arm architecture context, to explore its potential for mass-market adoption.

3. CAPABILITY-BASED TEE COMPARTMENTALIZATION FRAMEWORK

In our construction, we aim to develop a Trusted Execution Environment (TEE)-aware compartmentalization framework that enhances security by providing strong isolation within the secure world of a simple "Secure Key Management and Storage application", named "morello-Tower". This framework focuses on the application layer, separating concerns and decomposing capabilities within, achieving modular abstraction, separation of responsibilities, and adhering to the principle of least privilege. Our solution leverages the CHERI hardware software architecture and CheriBSD [2] kernel, minimising the need for additional hardware resources and reducing attack surfaces.

3.1. Implementation Details

As illustrated in Figure 2 below, we have compartmentalized our Key management and storage system using vertical and work-bounded compartmentalization patterns. This employs a series of

compartments to perform specific processing of data while limiting the access and exposure of the given compartment within the pipeline.

Host Compartment (Normal World)

- Purpose: This compartment serves as the interface for user interactions and initialises the key generation process.
- Function: It forwards an incoming User ID as a sealed data capability to the Enclave Entry Compartment, thereby requesting the generation of a User key.

Enclave Compartment (Secure World)

This component emulates the secure world and consists of a well-structured pipeline comprising multiple isolated compartments. Each compartment is endowed with specific access privileges and distinct responsibilities. These compartments include:

- ***Entry (Seed) Compartment***: Responsible for generating a seed, based on delegated User ID capability and a source of randomness. The subsequent compartments do not have access to the User Id data capability.
- ***Crypto Compartment***: Executes operations like generating a key pair from the delegated seed.
- ***Exit (Storage) Compartment***: With dispatched key pairs object, the given compartment handles data storage. This compartment has dedicated access to the storage location with write privileges.

Our framework facilitates controlled communication between these compartments, ensuring mutual distrust is enforced. Domain transitions between host and enclave compartments, as well as within the enclave, occur through sealed data capability delegations with hardware-assisted call/return mechanisms.

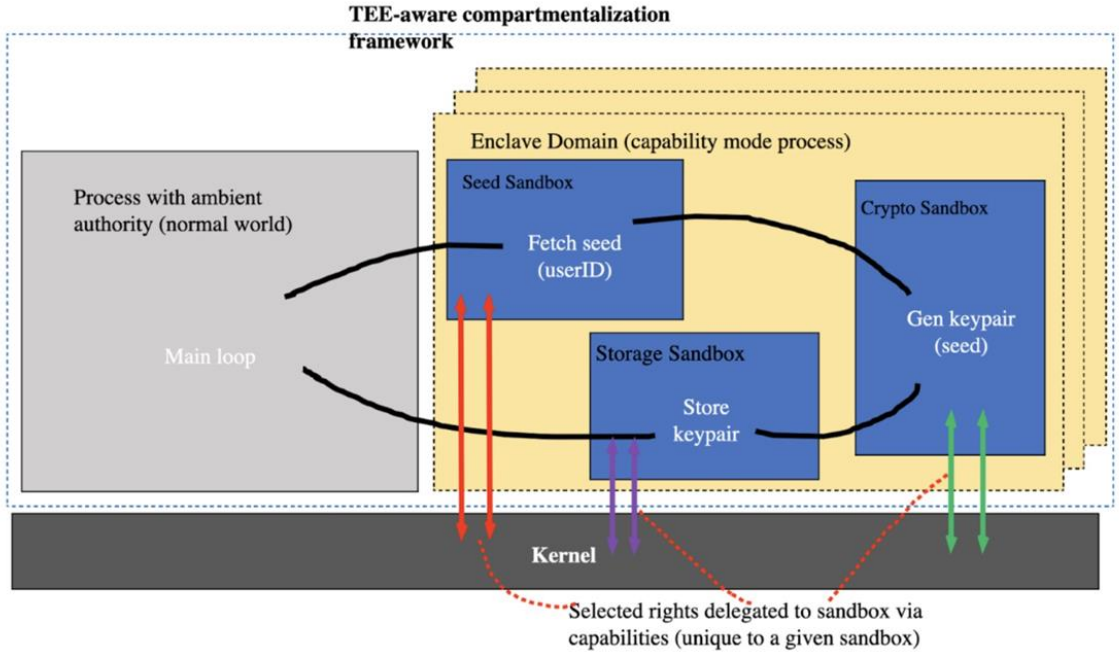

Figure 2 : TEE- aware compartmentalization framework

3.2. Key Aspects

Object-Capability Model

- Objective: To enable selective privileges and secure sharing of objects between the secure and normal worlds.
- Approach: The foundation of morelloTower is the object-capability model. It provides the means to securely delegate and control access rights to specific objects or resources within the TEE, preventing unauthorised access.

Modular Compartmentalization

- Objective: To compartmentalize TEE-aware functions with single responsibility principles, enforcing the least privilege.
- Approach: morelloTower enforces a modular separation of concerns with each compartment having a well-defined responsibility and access rights. This approach ensures that compartments only have access to the resources required for their specific tasks, minimising the attack surface. Also if any unsafe operation within a compartment can be gracefully handled by the caller of the compartment, which can then terminate or reset the connection.

Information Hiding

- Objective: To protect sensitive data from unauthorised access or leakage.
- Approach: The framework incorporates mechanisms for information hiding within compartments. Sensitive data is shielded from unprivileged compartments, ensuring confidentiality and integrity.

Hardware-Assisted Security

- Objective: To enhance the security of TEE operations.
- Approach: Hardware-assisted call and return mechanisms are utilised to ensure secure transitions between compartments. These mechanisms prevent tampering and unauthorised code execution, contributing to the overall security of the TEE.

Kernel Attack Surface Reduction

- Objective: To minimise exposure to potential kernel-level attacks.
- Approach: morelloTower eliminates the need for traditional kernel system calls for compartment delegations. This reduction in kernel interactions lowers the attack surface, making it resistant to rootkit-based attacks and ensuring the TEE kernel's integrity.

Protection Against Denial of Service (DoS)

- Objective: To maintain TEE functionality even in the presence of disruptive actions.
- Approach: By limiting the amount of work each compartment performs per invocation, denial-of-service attacks can be mitigated. Computations from one flow group will not effect processing of other pipelines, when multiple instances of same processing are performed on different data instances.

Memory Protection

- Objective: To mitigate vulnerabilities associated with memory leaks and shared memory spaces.
- Approach: CHERI memory protection is employed to safeguard against memory vulnerabilities. It isolates memory spaces, reducing the risk of leaks and enhancing overall security.

Secure Communication

- Objective: To prevent command injection attacks and unauthorised communication.
- Approach: morelloTower implements sealed data capability delegations and controlled communication interfaces. These measures protect against command injection attacks and ensure that communication channels remain secure.

In conclusion, the construction of morelloTower centred around the object-capability model, modular compartmentalization, information hiding, hardware-assisted security, kernel attack

surface reduction, DoS protection, memory protection, and secure communication. These elements collectively contribute to the robust security and compartmentalization of the TEE aware framework.

4. CONCLUSION AND FUTURE WORK

In this paper, we have presented morelloTower, a framework designed to achieve TEE-aware secure and granular compartmentalization. By leveraging the CHERI object-capability model and advanced hardware features, morelloTower provides an innovative solution to address critical security concerns and vulnerabilities associated with conventional TEEs.

The framework aims to offer modular compartmentalization, enforcing the principle of least privilege and single responsibility for each compartment. This design minimises the attack surface, reduces the risk of unauthorised access, and enhances the overall security posture of the TEE-aware compartments. The integration of information hiding mechanisms ensures the confidentiality and integrity of sensitive data within these compartments.

Furthermore, morelloTower addresses the issue of denial of service (DoS) attacks by implementing isolation & redundant resilience mechanisms that allow TEE compartments to continue functioning even in adverse conditions. Secure communication interfaces and sealed data capability delegations prevent command injection attacks and unauthorised communication, further fortifying the security of the TEE. By emphasising these key features and technologies, morelloTower demonstrates a step forward in the domain of secure computations.

In summary, morelloTower, with its CHERI enabled architecture and emphasis on granular computation compartmentalization, offers a promising avenue for securing critical applications and data in an ever-evolving digital landscape.

As part of our ongoing research, we intend to explore life cycle management, asynchronous object capability invocations, and protocol design for multi-compartment computations. Additionally, we are actively assessing the performance, semantics, vulnerability mitigation, and comparative advantages of CHERI-enabled morelloTower in contrast to existing TEE environments.

Furthermore, at VerifoxxLtd[4], we are extending our research endeavours to enhance the Digital Security by Design (DSbD) [9] software ecosystem. To this end we are developing a CHERI-aware WebAssembly Micro Runtime (WAMR) [10] for software modules that can be seamlessly embedded into morelloTower. This strategic integration aims to reinforce the security posture of the double sandboxed strategy within morelloTower. We are dedicated to sharing our ongoing research findings, results, and developmental efforts through forthcoming publications.

AVAILABILITY

MorelloTowerGithub - https://github.com/Verifoxx-LTD/morello_tower.git

ACKNOWLEDGEMENTS

We would like to acknowledge University of Cambridge, SRI International and all the members of the wider CHERI and Morello teams, for their work. This work was funded by the Innovate UK project under the Digital Security by Design (DSbD) Ecosystem and Validation project, 106897, to validate and enrich the DSbDtech CHERI ecosystem.

The views, opinions, and/or findings contained in this paper are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] V. Costan and S. Devadas. Intel sgx explained, 2016. https://eprint.iacr.org/2016/086

[2] CheriBSDDSbD. Cheribsd - DSbD. https:// www.cheribsd.org/.

[3] Arm Limited. Architecture reference manual sup- plement: Morello for a-profile architecture, 2020.

[4] VerifoxxLtd. https://verifoxx.com/

[5] MorelloTower. Morello Tower github repo. https://github.com/Verifoxx-LTD/ morellotower.git.

[6] Alexander Richardson Peter G. Neumann John Baldwin-Simon W. Moore David Chisnall Jessica Clarke Nathaniel Wesley FilardoKhilanGudka Alexandre Joannou Ben Laurie A. Theodore Markettos J. Edward Maste Alfredo Mazzinghi Edward Tomasz Napierala Robert M. Norton Michael Roe Peter Sewell Stacey Son Jonathan Woodruf Robert N. M. Watson, Brooks Davis. Cheriabi: Enforcing valid pointer provenance and minimising pointer privilege in the posix c run-time environment, 2015.

[7] Jonathan Woodruff Michael Roe HeshamAlmatary Jonathan Anderson John Baldwin Graeme Barnes David Chisnall Jessica Clarke Brooks Davis Lee Eisen Nathaniel Wesley Filardo Richard Grisenthwaite Alexandre Joan-nou Ben Laurie A. Theodore Markettos Simon W. Moore Steven J. Murdoch KyndylanNien-huis Robert Norton Alexander Richardson Peter Rugg Peter Sewell Stacey Son Hongyan Xia Robert N. M. Watson, Peter G. Neumann. Capability hardware enhanced risc instructions: Cheri instruction-set architecture (version 8), 2015.

[8] Peter G. Neumann Simon W. Moore Jonathan Anderson David ChisnallNirav Dave Brooks Davis KhilanGudka Ben Laurie Steven J. Murdoch Robert Norton Michael Roe-Stacey Son MunrajVadera Robert N. M. Watson, Jonathan Woodruff. A hybrid capability system architecture for scalable software compartmentalization, 2015. https://www.cl. cam.ac.uk/research/security/ctsrd/pdfs/201505-oakland2015-cheri-compartmentalization.pdf.

[9] Digital Security by Design. DSbD - https://www.dsbd.tech/about/

[10] WebAssembly Micro Runtime. WAMR - https://github.com/bytecodealliance/wasm-micro-runtime

AUTHOR

BalaSubramanyan, CTO & Co-Founder: 14+ years of expertise in Architecting and Implementing Confidential Computing, Verifiable Computations and Machine Learning Solutions across Financial Research and Healthcare domains.

Bala is the Co-Founder and CTO of Verifoxx, a cutting edge identity verification solution that prioritises privacy through zero-knowledge proof-based verification of entity's identities and financial profiles. Before co-founding Verifoxx in 2019, Bala established his career across organisations such as JP Morgan, S&P Global & Nationwide where he specialised in implementing confidential computing, verifiable computations and machine learning solutions.