SECW2

1244 words

Summary

1. **Linux Security Modules[1]:**

This paper introduces the Linux Security Modules (LSM) framework. It aims to address the inadequacy of existing access control mechanisms in operating systems due to a lack of consensus on optimal security solutions. Then, the design and implementation of LSM is presented.

LSM can support various security models while minimizing changes to the existing Linux kernel. It uses a modular approach, enabling security policies to be loaded as separate modules to accommodate diverse needs. The default "capabilities" module is loaded if no module is used. LSM provides a registration mechanism for security modules to register and unregister themselves with the kernel for dynamically switching between different models; it allows multiple modules to operate concurrently, enabling more complex policies. It also allows the POSIX.1e functions to be separated from the base, introducing minimal overhead.

This work is a significant advancement in making Linux more secure and adaptable.

2. **seL4[2]:**

The paper presents the first formal, machine-checked verification of the seL4 microkernel. It confirms that the seL4 implementation is mathematically proven to be consistent with its specification, ensuring no programming errors and functional correctness.

The paper details the approach that blends formal methods with conventional OS development techniques, optimizing the kernel for verification without sacrificing performance. The verification process covers the entire kernel, from its high-level abstract specifications to the C implementation. This approach demonstrates that formal verification, often considered an academic exercise with limited practical applications, can be applied to real-world systems like OS kernels.

It stresses "design for verifiability," where the system is designed with verification in mind, facilitating formal analysis and proof. It also sets a new benchmark in OS development and encourages the adoption of formal methods in developing software for secure and reliable computing infrastructures.

3. **Cure[3]:**

The paper introduces CURE, a Trusted Execution Environment (TEE) security architecture. Existing TEE solutions uses one-size-fits-all approach which is insufficient to meet the evolving security demand or applications and are prone to cache side channel attacks. CURE addresses the limitations by introducing customizable enclave types—sub-space, user-space, and self-contained enclaves—each designed to meet different security requirements. CURE also focus on minimizing performance overhead and hardware modifications. CURE achieves this through hardware changes, demonstrating a geometric mean performance overhead of just 15.33% on standard benchmarks. It also protects against cache side-channel attacks by implementing a fine-grained mapping between cache resources and individual enclaves.

Overall, CURE's contributions to hardware-assisted security architectures mark a significant step forward in developing secure and efficient TEE solutions.

Key Themes

Similarities:

Modularity and Flexibility: LSM and CURE both emphasis on modularity and flexibility. LSM introduces a framework that allows for integrating various security models into the Linux kernel as loadable modules, enabling a customizable approach to access control. Similarly, CURE's architecture offers multiple types of enclaves, providing the adaptability needed to meet the diverse security requirements. This modular approach allows LSM and CURE to cater to a wide range of security needs without significantly altering the existing system structure.

Minimal Impact: Both the seL4 microkernel and CURE prioritize minimizing their impact on system performance and complexity. seL4 achieves this through formally verified microkernel design, which ensures correctness while maintaining a lean and efficient architecture. Conversely, CURE focuses on minimal hardware changes to implement its security features, striving to balance security with performance overhead.

Security as a Core Consideration: seL4's comprehensive formal verification process underscores a commitment to building a fundamentally secure and reliable operating system kernel. LSM and CURE, while employing different methodologies, also centre their designs around strengthening system security—LSM through its general-purpose access control framework and CURE through its resilient enclaves and protection against side-channel attacks.

Contrasts:

Scope and focus:

- LSM targets within the Linux kernel, aiming to enhance its security by introducing a modular framework for access control, focusing on the integration of various security models without altering the kernel's core structure.
- seL4 focuses on creating a formally verified kernel that serves as a reliable base for building secure systems and eliminating vulnerabilities. Its scope extends to provide verification approach for all kernels.
- CURE addresses security challenges within Trusted Execution Environments (TEEs). It focuses on creating secure, isolated enclaves that can be customized to protect sensitive applications, extending its relevance to a wide range of computing platforms, from embedded devices to cloud servers.

Methodological Approaches:

- LSM adopts a modular approach that allows for the dynamic addition and configuration of security modules, emphasizing flexibility and minimal intrusion into the existing kernel architecture.
- seL4 employs a formal verification process to ensure the correctness of microkernel through mathematical proofs.
- CURE combines architectural innovations with minimal hardware changes to achieve its objectives, balancing enhanced security features with the practical considerations of performance overhead and implementation feasibility.These contrasts highlight the diverse perspectives and approaches each paper takes in addressing the multifaceted challenges of system security. While LSM and seL4 focus on different layers of operating system security, CURE extends the discussion to the specialized context of TEEs, underscoring the varied strategies and solutions employed in the quest for more secure computing environments.

Legacy

1. **Linux Security Modules:**

The work of Crispin Cowan[4] and James P. Anderson[5] are cited by this paper. Cowan studied SubDomain and emphasis on a least privilege mechanism tailored for programs, that aligns with LSM's modular approach to security. Anderson's study focused on identifying urgent security needs for USAF, highlights the necessity for continuous research and development in security technologies. It directly informs the motivations behind the LSM framework, reinforcing the importance of built-in security mechanisms. LSM's design reflects an integration of these insights, embodying a flexible, modular architecture that addresses the security needs for Linux systems.

The works of Aleksandar Milenkoski[6] and Sari Sultan[7] are both influenced by LSM. The former, with its focus on evaluating intrusion detection systems, benefits from LSM's modular framework. For the latter, LSM's framework for enhancing kernel security through modular access control mechanisms is particularly relevant to the software-based solutions for container security. By leveraging LSMs, container security architectures can implement fine-grained access control policies that are critical for the first three use cases of container security.

2. **seL4:**

Victor Costan[8] introduces Sanctum, a software isolation architecture that protects against memory access pattern attacks. Sanctum's minimal hardware changes and trusted software implementation resemble seL4's security philosophy.

Stuart Russell[9] focuses on AI, but it also shows that formal verification of seL4 ensures system integrity is essential for hosting critical AI applications.

Andrew Baumann[10] introduces shielded execution to protect applications and data from cloud infrastructure compromise. Using hardware protections like Intel SGX to run unmodified legacy applications complements seL4's approach. Andrew builds on a formally verified kernel like seL4 to show how hardware and software security mechanisms can work together to protect the stack.

Dayeol Lee[11], builds customizable TEEs(Keystone) to address security threat models and resource constraints across devices. Keystone's use of simple hardware abstractions for memory isolation and a secure layer beneath untrusted components follows seL4's minimalism and modularity in secure system design.

3. **Cure[3]:**

Adil Ahmad[12] uses Intel SGX to overcome the limitations of existing program obfuscation schemes by addressing commodity hardware side-channel vulnerabilities. The CURE paper emphasizes customizable and resilient enclave designs, so obfuscation and memory access pattern protection could be integrated into CURE's architecture to secure enclaves from sophisticated side-channel attacks.

Adil[13] also addresses of protecting file system operations in SGX enclaves from side-channel attacks in another paper. OBLIVIATE's strategies and optimizations helped the CURE architecture secure data access within enclaves and mitigate side-channel risks.

Reference list

[1] Wright, Chris, et al. "Linux security modules: General security support for the linux kernel." *11th USENIX Security Symposium (USENIX Security 02)*. 2002.

[2] Klein, Gerwin, et al. "seL4: Formal verification of an OS kernel." *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. 2009.

[3] Bahmani, Raad, et al. "CURE: A security architecture with CUstomizable and resilient enclaves." *30th USENIX Security Symposium (USENIX Security 21)*. 2021.

[4] Cowan, Crispin, et al. "SubDomain: Parsimonious Server Security." *14th Systems Administration Conference (LISA 2000)*. 2000.

[5] Anderson, James P. *Computer security technology planning study*. Vol. 1. ESD-TR-73-51, 1972.

[6] Milenkoski, Aleksandar, et al. "Evaluating computer intrusion detection systems: A survey of common practices." *ACM Computing Surveys (CSUR)* 48.1 (2015): 1-41.

[7] Sultan, Sari, Imtiaz Ahmad, and Tassos Dimitriou. "Container security: Issues, challenges, and the road ahead." *IEEE access* 7 (2019): 52976-52996.

[8] Costan, Victor, Ilia Lebedev, and Srinivas Devadas. "Sanctum: Minimal hardware extensions for strong software isolation." *25th USENIX Security Symposium (USENIX Security 16)*. 2016.

[9] Russell, Stuart, Daniel Dewey, and Max Tegmark. "Research priorities for robust and beneficial artificial intelligence." *AI magazine* 36.4 (2015): 105-114.

[10] Hunt, Andrew Baumann Marcus Peinado Galen. "Shielding applications from an untrusted cloud with Haven."

[11] Lee, Dayeol, et al. "Keystone: An open framework for architecting trusted execution environments." *Proceedings of the Fifteenth European Conference on Computer Systems*. 2020.

[12] Ahmad, Adil, et al. "Obfuscuro: A commodity obfuscation engine on intel sgx." *Network and Distributed System Security Symposium*. 2019.

[13] Ahmad, Adil, et al. "OBLIVIATE: A Data Oblivious Filesystem for Intel SGX." *NDSS*. 2018.