230406.Papers.OP-TEE

Created: 2023-04-22 01:02:38

Questions

- 2017 vTZ
  - Paper: vTZ: Virtualizing ARM TrustZone `(2023-04-06)`
- 2019 TrustedVIM
  - Paper: Cloud & Edge Trusted Virtualized Infrastructure Manager (VIM) - Security and Trust in OpenStack `(2023-04-06)`
- Compare 2017 vTZ and 2019 TrustedVIM
  - What is difference between two papers? Please also summarize it in a table. `(2023-04-06)`
  - What is same between two papers? `(2023-04-06)`
  - What is difference of virtualization tech between two papers? `(2023-04-06)`
- 2020 HETEE
  - Title: Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment `(2023-04-06)`
- 2020 SoK TrustZone
  - Title: SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems `(2023-04-06)`
- 2020 RusTEE
  - Title: RusTEE: Developing Memory-Safe ARMTrustZone Applications `(2023-04-06)`
- 2020 Bao
  - Paper: Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems `(2023-04-06)`
  - What is static partitioning? `(2023-04-06)`
  - Is Bao a hypervisor? `(2023-04-07)`
- 2019 Research on TrustZone
  - Paper: Research on ARM TrustZone `(2023-04-06)`
  - The below context comes from “Other Platforms: IoT, Drone, Cloud” of the paper “Research on ARM TrustZone” `(2023-04-06)`
  - In the context, vTZ run in normal world. Why does not vTZ run in normal world need to trust REE hypervisor? `(2023-04-06)`
- 2019 TEEv
  - Paper: TEEv: Virtualizing Trusted Execution Environmentson Mobile Platforms `(2023-04-06)`
  - Please compare TEEv and vTZ and also summarize them in a table. `(2023-04-06)`
- 2020 Demystifying TEE Client API
  - Paper: Demystifying ARM TrustZone TEE Client API using OP-TEE `(2023/4/8)`
- 2022 WATZ
  - Paper: WATZ: A Trusted WebAssembly RuntimeEnvironment with Remote Attestation for TrustZone `(2023-04-17)`
  - What is the benefit of using WASM in TEE? `(2023-04-17)`
  - Is WATZ extended from OP-TEE? `(2023-04-17)`
  - What is Remote Attestation? `(2023-04-17)`
  - So the software is provided by the remote device and the party verify the signature of the software before run it. Right? `(2023-04-17)`
  - Is remote attestation only used to verify software? `(2023-04-17)`
  - How does a remote device declare that secure boot is enabled? `(2023-04-17)`
  - But how does the party(verifier) determine if the report is fake if the device is breached? `(2023-04-17)`
  - How does the party(verifier) verify the root of trust of the device? `(2023-04-17)`
  - Can remote attestation help the party know if the remote device is breached? `(2023-04-17)`
  - Can TEE guarantee the code? `(2023-04-17)`

Q & A

2017 vTZ

Question: Paper: vTZ: Virtualizing ARM TrustZone `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “vTZ: Virtualizing ARM TrustZone” addresses the challenge of virtualizing ARM TrustZone, which is not designed to be virtualizable, in order to securely share it among multiple virtual machines (VMs). ARM TrustZone is a security extension providing a secure world or trusted execution environment (TEE) for running security-sensitive code, primarily in mobile platforms. As ARM64 becomes more popular in server markets like cloud, TrustZone is expected to become a key component of cloud security.

The authors present vTZ, a solution that allows each guest VM to have a virtualized guest TEE using existing hardware. vTZ separates functionality from protection by maintaining a secure co-running VM to serve as a guest TEE, while utilizing hardware TrustZone for strong isolation among guest TEEs and the untrusted hypervisor. vTZ employs a tiny monitor within the physical TrustZone that securely interposes and virtualizes memory mapping and world switching. Additionally, it uses a Constrained Isolated Execution Environment (CIEE) with protected, self-contained code to ensure secure virtualization and isolation among multiple guest TEEs.

The authors implemented vTZ on Xen 4.8 on both ARMv7 and ARMv8 development boards and evaluated its performance using two common TEE-kernels, seL4 1 and OP-TEE. The results demonstrated that vTZ provides robust security with minimal performance overhead.

2019 TrustedVIM

Question: Paper: Cloud & Edge Trusted Virtualized Infrastructure Manager (VIM) - Security and Trust in OpenStack `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “Cloud & Edge Trusted Virtualized Infrastructure Manager (VIM) - Security and Trust in OpenStack” discusses the security challenges faced in 5G mobile networks as they adopt virtualization technologies, such as Network Functions Virtualization (NFV), and rely on edge computing for improved performance. The scattered nature of devices and their varying locations make man-in-the-middle, denial-of-service attacks, and tampering easier. Additionally, the dynamic nature of the cloud calls for real-time threat prevention.

The authors propose a solution to address these challenges by incorporating trust into the virtualized edge infrastructure. They present their contributions in developing security services for platform authentication and integrity, which are hosted inside a trusted execution environment (TEE). The proposed solution leverages technologies such as OP-TEE, ARM TrustZone, Virtual Infrastructure Manager (VIM), and OpenStack to achieve the desired level of security.

The authors evaluate the performance overhead of their solution and suggest potential improvements for future work. Their approach aims to provide enhanced security and trust in virtualized cloud and edge computing environments, which are critical components of 5G mobile networks.

Compare 2017 vTZ and 2019 TrustedVIM

Question: What is difference between two papers? Please also summarize it in a table. `(2023-04-06)`

Answer:

The main difference between the two papers is their focus and proposed solutions. The first paper, “vTZ: Virtualizing ARM TrustZone,” focuses on virtualizing ARM TrustZone for secure sharing among multiple virtual machines (VMs). The second

paper, “Cloud & Edge Trusted Virtualized Infrastructure Manager (VIM) - Security and Trust in OpenStack,” aims to address security challenges in 5G mobile networks by incorporating trust into virtualized edge infrastructure.

Here is a summary table highlighting the key differences between the two papers:

Aspect	vTZ: Virtualizing ARM TrustZone	Cloud & Edge Trusted Virtualized Infrastructure Manager (VIM) - Security and Trust in OpenStack
Main Focus	Virtualizing ARM TrustZone	Enhancing security in virtualized edge infrastructure
Application Domain	Mobile platforms and cloud	5G mobile networks and edge computing
Proposed Solution	vTZ	Security services hosted inside a TEE
Key Technologies	ARM TrustZone, Xen 4.8	OP-TEE, ARM TrustZone, VIM, OpenStack
Implementation & Evaluation	ARMv7 and ARMv8 development boards	Performance overhead evaluation and future improvements

The first paper presents vTZ, a solution for virtualizing ARM TrustZone, while the second paper focuses on developing security services for platform authentication and integrity, hosted inside a TEE, for virtualized edge infrastructure in 5G networks.

Question: What is same between two papers? `(2023-04-06)`

Answer:

The similarities between the two papers lie in their overall goals, focus on security, and the use of Trusted Execution Environments (TEEs). Both papers:

1. Aim to enhance security: Both papers focus on improving security in their respective domains. The first paper addresses the challenge of securely sharing ARM TrustZone among multiple virtual machines, while the second paper aims to incorporate trust into virtualized edge infrastructure in 5G mobile networks.
2. Utilize Trusted Execution Environments (TEEs): Both papers leverage TEEs to provide secure environments for running security-sensitive code. The first paper uses ARM TrustZone as a TEE, while the second paper hosts security services inside a TEE.
3. Employ ARM TrustZone: ARM TrustZone is a key technology used in both papers. The first paper proposes a solution (vTZ) to virtualize ARM TrustZone, and the second paper uses ARM TrustZone as part of the trusted execution environment for hosting security services.

In summary, both papers share a common goal of enhancing security and use Trusted Execution Environments, specifically ARM TrustZone, to achieve their objectives.

Question: What is difference of virtualization tech between two papers? `(2023-04-06)`

Answer:

The main difference in virtualization technology between the two papers lies in their specific focus and approach to addressing security challenges in their respective domains.

1. “vTZ: Virtualizing ARM TrustZone”: This paper is primarily concerned with virtualizing ARM TrustZone, a security extension that provides a secure world or Trusted Execution Environment (TEE) to run security-sensitive code. The proposed solution, vTZ, allows each guest virtual machine (VM) to have a virtualized guest TEE using existing hardware. vTZ separates functionality from protection by maintaining a secure co-running VM to serve as a guest TEE while utilizing hardware TrustZone for strong isolation among guest TEEs and the untrusted hypervisor. vTZ employs a tiny monitor within the physical TrustZone that securely interposes and virtualizes memory mapping and world switching.
2. “Cloud & Edge Trusted Virtualized Infrastructure Manager (VIM) - Security and Trust in OpenStack”: This paper focuses on enhancing security in virtualized edge infrastructure in 5G mobile networks. The authors propose incorporating trust into the virtualized edge infrastructure by developing security services for platform authentication and integrity, hosted

inside a Trusted Execution Environment (TEE). The paper does not specifically focus on virtualizing a particular technology but rather aims to improve security in a virtualized infrastructure.

In summary, the first paper targets virtualizing ARM TrustZone to enable secure sharing among multiple VMs, while the second paper addresses security challenges in virtualized edge infrastructure in 5G networks by incorporating trust and providing security services hosted inside a TEE.

2020 HETEE

Question: Title: Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “Enabling Rack-scale Confidential Computing using Heterogeneous Trusted Execution Environment” addresses the limitations of today’s Trusted Execution Environments (TEEs) in supporting large-scale confidential computing involving high-throughput accelerators like GPUs, FPGAs, and TPUs. Existing solutions that extend CPU-like enclaves to GPUs require chip-level changes, may introduce new security risks due to side-channel leaks, and are constrained by CPU TEE resources.

The authors present HETEE, a novel Heterogeneous TEE design that supports large-scale compute or data-intensive (CDI) computing without requiring chip-level changes. HETEE is a device for centralized management of all computing units (e.g., GPUs and other accelerators) in a server rack. It works with today’s data centers and clouds, leveraging resource pooling technologies to dynamically compartmentalize computing tasks, enforce strong isolation, and reduce Trusted Computing Base (TCB) through hardware support.

HETEE uses PCIe ExpressFabric to allocate accelerators to server nodes for non-sensitive CDI tasks and moves them back into a secure enclave when confidential computing is required. A thin TCB stack for security management runs on a security controller (SC), while other software components run on integrated microservers that operate enclaves. Enclaves are physically isolated through hardware and verified by the SC at their inception, with microservers and computing units restored to a secure state upon termination.

The authors implemented HETEE on a real hardware system and evaluated its performance using popular neural network inference and training tasks. The evaluation results show that HETEE can easily support real-world scale CDI tasks, with a maximum throughput overhead of 2.17% for inference and 0.95% for training on ResNet152.

2020 SoK TrustZone

Question: Title: SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems” investigates the security vulnerabilities and limitations that affect existing TrustZone-assisted Trusted Execution Environment (TEE) systems, the challenges in building them correctly, and potential contributions from the research community to address these issues. TEEs, which are used to protect security-critical applications and OS components, are often assumed to be highly secure. However, various successful attacks on TEEs have been reported in recent years, revealing security flaws in TEE systems.

The authors conduct a security analysis of popular TrustZone-assisted TEE systems, targeting Cortex-A processors, developed by Qualcomm, Trustonic, Huawei, Nvidia, and Linaro. They study publicly documented exploits and vulnerabilities and reverse-engineer the TEE firmware to identify critical vulnerabilities across existing systems. The findings raise legitimate concerns about the security of commercial TEE implementations.

This paper provides valuable insights into the types of vulnerabilities that affect TrustZone-assisted TEE systems and highlights the need for robust security measures to protect TEEs from potential attacks. The research contributes to a better

understanding of TEE security and helps inform future improvements in TEE design and implementation.

2020 RusTEE

Question: Title: RusTEE: Developing Memory-Safe ARMTrustZone Applications `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “RusTEE: Developing Memory-Safe ARM TrustZone Applications” addresses the issue of vulnerabilities in trusted applications running in ARM TrustZone’s Trusted Execution Environment (TEE). While TrustZone provides isolation to protect trusted applications from malicious software in the untrusted rich execution environment (REE), it cannot prevent vulnerabilities within trusted applications themselves. Many trusted applications are written in memory-unsafe languages like C, leading to reported memory corruption vulnerabilities.

The authors propose RusTEE, a trusted application mechanism that uses Rust, a memory-safe programming language, to enhance the security of trusted applications. They address several challenges in adopting Rust for mobile TEEs, as Rust does not currently support TrustZone-assisted TEE systems. The paper presents the following contributions:

1. Extending the Rust compiler: The authors extend the existing Rust compiler to provide support for TrustZone-assisted TEE systems.
2. Implementing comprehensive security mechanisms: They resolve two security issues of trusted applications, namely, securely invoking high-privileged system services and securely communicating with untrusted REE.
3. Developing a prototype: The authors implement a prototype of RusTEE as an SDK for trusted applications, supporting both emulators and real hardware devices.

Experimental results show that RusTEE can compile applications with close-to-C performance on the evaluated platforms, demonstrating the potential of using Rust to enhance the security of trusted applications in ARM TrustZone TEE systems.

2020 Bao

Question: Paper: Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems” addresses the need for a virtualization solution in increasingly complex and mixed-criticality modern embedded systems. Current hypervisors like KVM and Xen were not designed for embedded constraints and requirements. The static partitioning architecture used by Jailhouse seems to address embedded concerns, but it still relies on Linux to boot and manage its virtual machines (VMs).

The authors present Bao, a minimal, standalone, and clean-slate implementation of the static partitioning architecture for Armv8 and RISC-V platforms. Bao is a lightweight hypervisor designed specifically for modern multi-core embedded systems, offering strong spatial and temporal isolation. The main contributions of the paper include:

1. A minimal, standalone implementation: Bao is a clean-slate implementation of the static partitioning architecture, removing the dependency on Linux for booting and managing VMs.
2. Support for Armv8 and RISC-V platforms: Bao is designed to work with both Armv8 and RISC-V platforms, making it suitable for a wide range of embedded systems.
3. Low virtualization overhead: Preliminary results regarding size, boot, performance, and interrupt latency demonstrate that Bao incurs minimal virtualization overhead.

The authors plan to make Bao publicly available, aiming to engage both industry and academia in improving Bao’s safety, security, and real-time guarantees. This paper presents a novel approach to virtualization in embedded systems, addressing the unique constraints and requirements of these systems while minimizing overhead.

Question: What is static partitioning? `(2023-04-06)`

Answer:

Static partitioning is a method of resource allocation in which system resources, such as memory, CPU time, and I/O devices, are divided into fixed, non-overlapping partitions. Each partition is assigned to a specific application, task, or virtual machine. The allocation of resources is determined at design time or during system initialization, and it remains fixed during the runtime of the system.

Static partitioning is often used in real-time and safety-critical systems, where strong spatial and temporal isolation between tasks is required to ensure predictable system behavior and to prevent interference between tasks. This isolation helps in maintaining the performance, safety, and security of the system.

In the context of hypervisors, static partitioning refers to creating fixed, isolated partitions for virtual machines (VMs) or operating system instances. Each partition is allocated a fixed set of resources, such as CPU cores, memory, and I/O devices. This approach contrasts with dynamic partitioning, where resource allocation can be adjusted during runtime based on the changing needs of the system. Static partitioning in hypervisors can help achieve better isolation, predictability, and reduced overhead compared to dynamic partitioning.

Question: Is Bao a hypervisor? `(2023-04-07)`

Answer:

Yes, Bao is a hypervisor. Specifically, it is a lightweight static partitioning hypervisor designed for modern multi-core embedded systems. The Bao hypervisor provides a minimal, standalone, and clean-slate implementation of the static partitioning architecture for Armv8 and RISC-V platforms. Its main goal is to achieve strong spatial and temporal isolation with minimal virtualization overhead in embedded systems.

2019 Research on TrustZone

Question: Paper: Research on ARM TrustZone `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “Research on ARM TrustZone” discusses ARM TrustZone, a hardware-based security feature providing a high-privilege and isolated execution environment known as the Trusted Execution Environment (TEE). TrustZone technology was proposed in 2002, but it gained widespread adoption in 2009 when Apple used it to protect Touch ID in the iPhone 5s. TrustZone ensures that user fingerprint data remains secure even if the iOS is compromised.

In 2017, Google mandated TEE for all Android devices with a fingerprint scanner. Today, almost all mobile phones and tablets have TEE deployed. ARM has integrated TrustZone into ARM64 and ARMv8-M to support a broader range of platforms, including servers and IoT devices.

The paper provides valuable insights into the ARM TrustZone technology, discussing its importance in providing a secure and isolated execution environment. It highlights the adoption of TrustZone by major tech companies like Apple and Google and emphasizes its relevance in the growing field of servers and IoT devices.

Question: The below context comes from “Other Platforms: IoT, Drone, Cloud” of the paper “Research on ARM TrustZone” `(2023-04-06)`

Answer:

In this section, the authors discuss how ARM TrustZone technology is applied to different types of platforms, such as IoT devices, drones, and cloud computing. They provide examples of various projects and research that utilize TrustZone to improve the security of these platforms:

1. Ditio: This project aims to enhance the security of IoT devices by recording sensor activity logs using TrustZone and virtualization support. Ditio captures access to sensor registers without modifying the Rich Execution Environment (REE) operating system. This generates a complete log of sensor activities that can be audited and checked for compliance with

specific policies. Ditio also offers a tool for easier log analysis.

2. PROTC: This project utilizes TrustZone on drone platforms to protect peripherals. PROTC deploys a monitor in the secure world to enforce a secure access control policy for some peripherals on the drone, ensuring that only authorized applications can access specific peripherals.
3. vTZ: This research addresses the limitation of TrustZone, which can only provide one secure world, meaning multiple virtual machines (VMs) on a server must share the single secure world. vTZ virtualizes TrustZone to offer each VM an isolated secure world. With vTZ, software in a VM can use the Secure Monitor Call (SMC) instruction to switch between the REE and a virtual TEE. This preserves all TrustZone features after virtualization. The implementation requires an REE hypervisor but does not need to trust it.

These examples showcase the versatility of TrustZone technology and its potential for improving security on various platforms beyond mobile phones.

Question: In the context, vTZ run in normal world. Why does not vTZ run in normal world need to trust REE hypervisor? `(2023-04-06)`

Answer:

vTZ is designed to virtualize ARM TrustZone to offer each virtual machine (VM) an isolated secure world. Although vTZ runs in the normal world, it does not need to trust the REE hypervisor because of the hardware-level isolation provided by ARM TrustZone.

TrustZone ensures a strong isolation between the secure world (trusted execution environment) and the normal world (rich execution environment). When vTZ is implemented, it leverages the security features provided by TrustZone to maintain a separation between the virtual TEEs and the REE hypervisor.

As a result, even though vTZ operates in the normal world and interacts with the REE hypervisor, the secure world components (virtual TEEs) remain protected from potential attacks or vulnerabilities in the REE hypervisor. This isolation allows vTZ to run securely without having to trust the REE hypervisor.

2019 TEEv

Question: Paper: TEEv: Virtualizing Trusted Execution Environmentson Mobile Platforms `(2023-04-06)`

Answer:

Paper Summary:

The paper titled “TEEv: Virtualizing Trusted Execution Environments on Mobile Platforms” discusses the challenges of transitioning from vendor-controlled, single-purpose Trusted Execution Environments (TEEs) to open TEEs that host Trusted Applications (TAs) from multiple sources with independent tasks. This transition can lead to an enlarged attack surface and a lack of trust or isolation among TAs and the TEE.

To address these challenges, the authors propose TEEv, a TEE virtualization architecture that supports multiple isolated, restricted TEE instances (vTEEs) running concurrently. TEEv relies on a tiny hypervisor called TEE-visor, allowing TEE instances from different vendors to run in isolation on the same smartphone and host their own TAs. This ensures that a compromised vTEE cannot affect its peers or the Rich Execution Environment (REE), and TAs no longer have to run in untrusted or unsuitable TEEs.

The paper presents an implementation of TEEv on a development board and a real smartphone, running multiple commercial TEE instances from different vendors with minimal porting effort. The evaluation results show that TEEv can isolate vTEEs and defend against all known attacks on TEE with only mild performance overhead.

Question: Please compare TEEv and vTZ and also summarize them in a table. `(2023-04-06)`

Answer:

TEEv and vTZ are two different approaches to virtualize Trusted Execution Environments (TEEs) on mobile platforms. Here is

a comparison and summary of both in the form of a table:

Aspect	TEEv	vTZ
Objective	Virtualize TEEs to support multiple isolated, restricted TEE instances	Virtualize TrustZone to offer each VM an isolated secure world
Architecture	Relying on a tiny hypervisor (TEE-visor)	Requires an REE hypervisor but does not need to trust it
Implementation	Runs multiple commercial TEE instances from different vendors	Uses SMC instruction to switch between REE and a virtual TEE
Security	Isolates vTEEs and defends against known attacks on TEE	Preserves all TrustZone features after virtualization
Performance	Mild performance overhead	Not explicitly mentioned

TEEv is a TEE virtualization architecture that supports multiple isolated, restricted TEE instances (vTEEs) running concurrently. It relies on a tiny hypervisor called the TEE-visor, allowing TEE instances from different vendors to run in isolation on the same smartphone and host their own Trusted Applications (TAs). TEEv can isolate vTEEs and defend against all known attacks on TEEs with only mild performance overhead.

vTZ aims to virtualize TrustZone to provide each VM an isolated secure world. This approach requires an REE hypervisor but does not need to trust it, as the hardware-level isolation provided by TrustZone is leveraged to maintain separation between the virtual TEEs and the REE hypervisor. With vTZ, software in a VM can use the SMC instruction to switch between the REE and a virtual TEE, just as it does in a non-virtualized environment, preserving all the features of TrustZone after virtualization.

2020 Demystifying TEE Client API

Question: Paper: Demystifying ARM TrustZone TEE Client API using OP-TEE `(2023/4/8)`

Answer:

Title: Demystifying ARM TrustZone TEE Client API using OP-TEE

Abstract: With the increased use of mobile devices for various purposes, such as payment systems, the need for secure storage of critical data has become more important. Trusted Execution Environment (TEE) technology has emerged as a solution to provide a secure execution environment for sensitive data. ARM TrustZone technology, primarily used in mobile devices, divides a physical processor into the Normal World and Secure World, ensuring a safer execution environment. While various manufacturers have adopted TrustZone technology, existing commercial TEEs have limitations in conducting security research using TrustZone.

This paper introduces OP-TEE, an open-source project for implementing ARM TrustZone technology, and the TEE Client API that enables communication with Trusted Applications within the TrustZone Secure World. To demystify the TEE Client API, the authors also implemented a simple trusted application for communication between the Normal World and Secure World in the OP-TEE OS using the QEMU emulator. This research aims to make the TEE Client API more accessible and easier to understand for researchers and developers working with ARM TrustZone technology.

2022 WATZ

Question: Paper: WATZ: A Trusted WebAssembly RuntimeEnvironment with Remote Attestation for TrustZone `(2023-04-17)`

Answer:

The paper “WATZ: A Trusted WebAssembly Runtime Environment with Remote Attestation for TrustZone” presents WATZ, a runtime environment designed for the efficient and secure execution of WebAssembly (Wasm) code within Arm’s TrustZone trusted execution environment (TEE). The runtime also incorporates a lightweight remote attestation system tailored to Wasm applications running in TrustZone, as TrustZone lacks built-in mechanisms for attestation.

The primary research questions that WATZ aims to address include: 1. System challenges in embedding Wasm into Arm

TrustZone. 2. The trustworthiness of remote execution of Wasm applications. 3. Performance comparison of Wasm applications when hosted in the trusted environment.

WATZ extends OP-TEE, a popular open-source trusted OS, and the prototype is validated with Arm hardware. The system successfully hosts Wasm applications in TrustZone, leveraging the WASI standard, a POSIX-like layer for Wasm, to interact with the TEE facilities. The paper also proposes a remote attestation protocol to attest Wasm code embedded in the trusted environment, ensuring that executed Wasm binaries are trustworthy, similarly to Intel SGX.

In terms of performance, WATZ is on par with Wasm applications executed outside of the TEE, and up to 2.12 times slower compared to native execution. This difference is considered negligible given the security benefits provided by WATZ.

Question: What is the benefit of using WASM in TEE? `(2023-04-17)`

Answer:

Using WebAssembly (WASM) in a Trusted Execution Environment (TEE) provides several benefits, making it a valuable choice for enhancing security and interoperability in various applications:

1. Portability: WASM is platform-agnostic, allowing developers to write code once and run it on various platforms, including TEEs. This increases the reusability and flexibility of applications running in TEEs.
2. Isolation and security: WASM provides a sandboxed execution environment, which ensures that applications are isolated from each other and from the host system. This isolation helps mitigate potential security risks, such as privilege escalation or lateral attacks.
3. Interoperability: WASM allows developers to build software components using modern programming languages and leverage existing code, making it easier to deploy applications in TEEs without the need to rewrite or recompile them.
4. Efficiency: WASM is designed for near-native performance, which means that applications running within TEEs can benefit from the efficiency and speed of WASM execution, even in resource-constrained environments like IoT devices.
5. Support for non-standard system interfaces: WASM can interact with non-standard system interfaces, making it suitable for TEEs that might have unique APIs or restrictions.

Overall, integrating WASM in TEEs allows for enhanced security, performance, and interoperability in various use cases, including smartphones, automotive applications, IoT devices, and edge computing.

Question: Is WATZ extended from OP-TEE? `(2023-04-17)`

Answer:

Yes, WATZ is extended from OP-TEE. OP-TEE (Open Portable Trusted Execution Environment) is a popular open-source trusted operating system for Arm TrustZone. WATZ extends OP-TEE by adding support for running WebAssembly (WASM) code inside Arm's TrustZone TEE and providing a lightweight remote attestation system optimized for WASM applications running in TrustZone. This extension allows WATZ to offer a secure and efficient runtime environment for WASM applications within the TrustZone TEE.

Question: What is Remote Attestation? `(2023-04-17)`

Answer:

Remote attestation is a security mechanism used to verify the authenticity and integrity of software running on a remote device, typically within a trusted execution environment (TEE) or a secure enclave, like Intel SGX or Arm TrustZone. It helps establish trust between a remote device and a verifying party by providing evidence that the software running on the remote device is genuine, unaltered, and operating in a secure environment.

In a remote attestation process, the remote device generates a cryptographic signature or proof, which includes information about the software's state and the secure environment. This proof is then sent to the verifying party, who can validate the signature using a public key or a pre-shared key. If the signature is valid and the software's state meets the expected criteria, the verifying party can trust the remote device and its software.

Remote attestation is particularly useful in scenarios where devices are deployed in untrusted or potentially compromised environments, such as IoT devices or cloud servers. It enables the establishment of trust between distributed devices and helps ensure the confidentiality and integrity of sensitive data and processes.

Question: So the software is provided by the remote device and the party verify the signature of the software before run it. Right? `(2023-04-17)`

Answer:

Yes, that's correct. In remote attestation, the software running on the remote device generates a cryptographic signature or proof, which includes information about the software's state and the secure environment. This proof is then sent to the verifying party.

The verifying party checks the signature using a public key or a pre-shared key, and if the signature is valid and the software's state meets the expected criteria, the verifying party can trust that the software running on the remote device is genuine and unaltered.

In other words, remote attestation allows the verifying party to confirm the authenticity and integrity of the software running on the remote device before deciding to trust it and establish secure communication or share sensitive data.

Question: Is remote attestation only used to verify software? `(2023-04-17)`

Answer:

Remote attestation is primarily used to verify the authenticity and integrity of software running in a secure environment, such as a trusted execution environment (TEE) or secure enclave. However, remote attestation can also be extended to cover aspects of the underlying hardware or firmware.

In some cases, remote attestation can be used to verify the configuration and state of the hardware or firmware to ensure it meets specific security requirements or complies with certain policies. For example, it can be used to verify that secure boot has been enabled, or that a specific hardware component is genuine and not tampered with.

So, while remote attestation is primarily focused on software, it can also be applied to hardware and firmware components in certain scenarios to provide a higher level of assurance about the security and trustworthiness of the entire system.

Question: How does a remote device declare that secure boot is enabled? `(2023-04-17)`

Answer:

A remote device can declare that secure boot is enabled by providing evidence in the form of a signed measurement or attestation report. This process involves the following steps:

1. Secure boot process: When the device boots up, the secure boot mechanism ensures that each component in the boot chain (bootloader, firmware, and operating system) is verified using digital signatures before being executed. This process creates a series of measurements (hashes) of the loaded components.
2. Platform Configuration Registers (PCRs): The measurements obtained during the secure boot process are stored in the device's Trusted Platform Module (TPM) or another secure element, in special registers called Platform Configuration Registers (PCRs).
3. Attestation request: A remote party (verifier) can request an attestation report from the device to verify that secure boot is enabled. The device generates a report that includes the PCR values, a nonce provided by the verifier, and other relevant information about the device's state.
4. Signing the report: The device signs the attestation report using a private key that corresponds to a public key known to the verifier. This key can be an endorsement key, an attestation identity key, or another key tied to the device's secure element.
5. Verifying the report: The remote party receives the signed report and verifies the signature using the corresponding public key. If the signature is valid, the remote party can examine the PCR values and other information in the report to determine if secure boot is enabled and if the device's configuration meets the required security policies.

This process allows the remote party to verify that secure boot is enabled on the device and that the device's boot chain components are genuine and have not been tampered with.

Question: But how does the party(verifier) determine if the report is fake if the device is breached? `(2023-04-17)`

Answer:

The remote attestation process relies on the hardware-backed security provided by secure elements like TPM (Trusted Platform Module) or TEE (Trusted Execution Environment) to ensure the integrity and authenticity of the attestation report. However, if the device is breached and the attacker can compromise these secure elements, the remote attestation process could potentially be fooled.

To minimize the risk of accepting fake reports, the verifier should follow these best practices:

1. Verify the root of trust: Ensure that the device's secure element (e.g., TPM or TEE) has a legitimate and trusted root of trust, such as a hardware-based root of trust or a known and trusted manufacturer's certificate.
2. Use a strong attestation protocol: Employ a secure and well-established attestation protocol, which has been formally verified and widely reviewed by the security community.
3. Verify the attestation report's signature: Ensure that the report is signed with a valid and trusted key tied to the device's secure element. The verifier must have access to the corresponding public key and validate the signature.
4. Check the nonce: The verifier should provide a unique nonce (number used once) during the attestation request, which the device must include in the attestation report. This ensures that the report is fresh and generated for the specific request, mitigating replay attacks.
5. Evaluate the report's content: The verifier should carefully analyze the attestation report's contents, including PCR values and other device state information, to ensure that it meets the required security policies and that the device's configuration is as expected.

It is important to note that no security measure is foolproof, and sophisticated attackers might still find ways to breach a system. However, following best practices and employing a defense-in-depth approach can significantly increase the difficulty for an attacker to compromise the remote attestation process.

Question: How does the party(verifier) verify the root of trust of the device? `(2023-04-17)`

Answer:

To verify the root of trust (RoT) of a device, the verifier (the party receiving the attestation report) must follow these steps:

1. Obtain the device's attestation report: The attestation report, generated by the device's secure element (e.g., TPM or TEE), includes information about the device's hardware, firmware, and software configuration, as well as a digital signature created using the device's private attestation key.
2. Verify the attestation key: The device's attestation key must be tied to a trusted certificate issued by the device manufacturer or another trusted authority. The verifier should validate the attestation key's certificate chain up to a trusted root certificate that is pre-installed or obtained from a trusted source. This process ensures the attestation key is genuine and belongs to a legitimate device.
3. Verify the attestation report's signature: The verifier must use the device's public attestation key (corresponding to the private attestation key used to sign the report) to verify the digital signature on the attestation report. If the signature is valid, it means the report was generated by the device with the genuine attestation key.
4. Check the nonce: To ensure the freshness of the attestation report and prevent replay attacks, the verifier should provide a unique nonce (number used once) during the attestation request. This nonce must be included in the attestation report, and the verifier should confirm its presence and correctness.
5. Examine the report's content: The verifier should carefully analyze the attestation report's contents to ensure that the device's hardware, firmware, and software configuration meet the required security policies. This includes checking measurements of the secure boot process, firmware versions, and other relevant security properties.

By following these steps, the verifier can establish trust in the device's root of trust, making it difficult for an attacker to

generate a fake attestation report. However, it is essential to maintain up-to-date trust anchors (e.g., root certificates) and follow best practices to ensure the security of the remote attestation process.

Question: Can remote attestation help the party know if the remote device is breached? `(2023-04-17)`

Answer:

Remote attestation can help the party (verifier) determine if the remote device might be breached, but it does not provide a 100% guarantee. Remote attestation provides information about the device's hardware, firmware, and software configuration, as well as measurements of the boot process and the integrity of the executed code.

If the device is compromised, there is a chance that the integrity measurements included in the attestation report will not match the expected values. The verifier can then detect the discrepancy and decide not to trust the device. However, if the breach is sophisticated enough and can tamper with the attestation process itself, it might be challenging to detect the breach using remote attestation alone.

To further enhance the security of remote attestation and reduce the chances of not detecting a breach, it is essential to follow best practices:

1. Use hardware-based roots of trust (e.g., TPM or TEE) to increase the difficulty for attackers to tamper with the attestation process.
2. Keep trust anchors (e.g., root certificates) up-to-date and manage them securely.
3. Regularly update the device's firmware, software, and security configurations to minimize vulnerabilities.
4. Use secure communication channels to transmit attestation reports and other sensitive information.
5. Combine remote attestation with other security mechanisms (e.g., intrusion detection systems, secure logging, and monitoring) to increase the chances of detecting breaches.

By employing a comprehensive security approach and using remote attestation as one of the security mechanisms, the verifier can have a higher level of confidence in detecting if a remote device is breached. However, it is essential to remember that no security measure can provide absolute certainty.

Question: Can TEE guarantee the code? `(2023-04-17)`

Answer:

A Trusted Execution Environment (TEE) can provide a high level of assurance regarding the integrity and confidentiality of the code executed within it, but it cannot provide an absolute guarantee. A TEE isolates sensitive code and data from the rest of the system, protecting them from unauthorized access or tampering by malicious software or external attackers.

TEEs, such as ARM TrustZone or Intel SGX, leverage hardware-based security features to create a secure enclave where sensitive operations can be performed. These enclaves provide a range of security properties, including:

1. Code isolation: The code running in the TEE is isolated from the rest of the system, ensuring that other applications or the OS cannot access or modify it.
2. Data protection: Data stored or processed within the TEE is encrypted and protected from unauthorized access or tampering.
3. Secure boot: TEEs often support secure boot, which ensures that the code loaded into the TEE has not been tampered with and comes from a trusted source.
4. Remote attestation: TEEs can provide remote attestation capabilities, allowing a remote party to verify the integrity and authenticity of the code running in the TEE.

Despite these security properties, TEEs are not infallible, and vulnerabilities have been discovered in the past. For example, side-channel attacks have been demonstrated against Intel SGX, potentially allowing an attacker to extract sensitive information from the enclave.

While TEEs can provide a high level of assurance for the code they execute, it is essential to remember that no security measure can offer absolute certainty. Regularly updating the software, firmware, and security configurations of the TEE, as well as employing a comprehensive security approach that includes other security mechanisms, can help reduce the risk of compromise.
