2023-04-12

Created: 2023-05-01 22:18:02

Chats

- Papers.OP-TEE.0406.md
  - `[10:36]` What are Security Building Blocks? `(Papers.OP-TEE.0406.md)`
  - `[10:37]` What compare Security Building Block and TCB and also summarize them in a table. `(Papers.OP-TEE.0406.md)`
- FF-A.0409.md
  - `[10:59]` Is TF-A persistent in memory at run-time? `(FF-A.0409.md)`
- OP-TEE.0327.md
  - `[14:46]` What is size of tee.bin? `(OP-TEE.0327.md)`
  - `[14:47]` What is about size of tee.bin? `(OP-TEE.0327.md)`
  - `[15:37]` Is SMAF of OP-TEE? `(OP-TEE.0327.md)`
  - `[15:39]` How is SMAF used in DRM? `(OP-TEE.0327.md)`
  - `[15:41]` What is Clear Key? `(OP-TEE.0327.md)`
- Arm.0410.md
  - `[16:48]` Can I get Armv8-A with only 32-bit for cost down? `(Arm.0410.md)`

Q & A

Papers.OP-TEE.0406.md

Question: `[10:36]` What are Security Building Blocks? `(Papers.OP-TEE.0406.md)`

Answer:

Security Building Blocks refer to the fundamental components or technologies used to design and implement a secure system. These building blocks form the foundation for creating secure software, hardware, and network architectures. They are essential for safeguarding sensitive information and ensuring that systems are resistant to unauthorized access, manipulation, or other forms of attack. Some common security building blocks include:

1. Encryption: The process of encoding data to protect it from unauthorized access. Encryption algorithms and protocols, such as AES, RSA, or TLS, are widely used to secure data during transmission or storage.
2. Authentication: The process of verifying the identity of a user, device, or system to ensure that only authorized entities can access the system or its resources. Common authentication methods include passwords, biometric data, or digital certificates.
3. Authorization: The process of granting or denying access to specific resources or actions based on an authenticated entity's privileges. Role-based access control (RBAC) and attribute-based access control (ABAC) are examples of authorization models.
4. Integrity: Ensuring that data and systems remain unaltered and free from tampering or corruption. Cryptographic hash functions, digital signatures, and checksums are examples of techniques used to maintain data integrity.
5. Confidentiality: The protection of sensitive information from unauthorized disclosure. Encryption, access control, and secure storage solutions are used to maintain data confidentiality.
6. Auditing and Logging: The process of recording system events, such as user actions or security incidents, to detect potential threats and ensure compliance with security policies or regulations.
7. Secure Communication: Ensuring that data transmitted between systems or devices remains confidential, authentic, and

unaltered. Secure communication protocols like TLS/SSL and VPNs are used to protect data during transmission.

8. Secure Boot: The process of verifying the integrity of a system's firmware and software during startup, ensuring that only authorized code is executed.
9. Hardware Security Modules (HSMs): Physical devices that provide secure storage and processing for cryptographic keys and other sensitive information.

These building blocks, when combined and implemented correctly, create a robust and secure foundation for any system, ensuring the protection of data and the prevention of unauthorized access or attacks.

Question: `[10:37]` What compare Security Building Block and TCB and also summarize them in a table. `(Papers.OP-TEE.0406.md)`

Answer:

Security Building Blocks and Trusted Computing Base (TCB) are two related concepts in the context of secure system design. Here is a comparison between the two:

Aspect	Security Building Blocks	Trusted Computing Base (TCB)
Definition	Fundamental components or technologies used to design and implement a secure system.	The smallest set of software and hardware components that are critical to the security of a system.
Purpose	To create secure software, hardware, and network architectures by providing a foundation for safeguarding sensitive information and ensuring system resistance to unauthorized access or manipulation.	To ensure that the system operates securely and enforces security policies by isolating critical components from the rest of the system.
Components/Elements	Encryption, Authentication, Authorization, Integrity, Confidentiality, Auditing and Logging, Secure Communication, Secure Boot, and Hardware Security Modules (HSMs).	TCB includes the operating system's kernel, security mechanisms, hardware components (e.g., CPU, memory), and other essential components that enforce security policies.
Trust	Security Building Blocks are used to create a trusted system by combining various security mechanisms and techniques.	The TCB is the core of a trusted system, and it is assumed to be secure and trustworthy. Its correct implementation and operation are crucial to the overall security of the system.
Scope	Security Building Blocks cover a wide range of security aspects in a system, such as data protection, access control, and secure communication.	TCB is a more focused concept, mainly concerned with the critical components that enforce security policies and protect the system's security properties.

In summary, Security Building Blocks are the fundamental components and technologies used to create secure systems, while the Trusted Computing Base (TCB) represents the minimal set of critical components essential for ensuring the security of a system. Security Building Blocks can be seen as the elements that are combined to create a secure system, whereas the TCB is the core of the system, ensuring its overall security.

FF-A.0409.md

Question: `[10:59]` Is TF-A persistent in memory at run-time? `(FF-A.0409.md)`

Answer:

Yes, the Trusted Firmware-A (TF-A) is typically persistent in memory at run-time. It resides in the secure memory region and provides various services, such as Secure Monitor Call (SMC) handling, Power State Coordination Interface (PSCI) implementation, and other platform-specific services. Since it acts as a secure monitor, it must remain present in memory

during the system's operation to handle any secure world requests or transitions between secure and non-secure worlds.

OP-TEE.0327.md

Question: `[14:46]` What is size of tee.bin? `(OP-TEE.0327.md)`

Answer:

The size of `tee.bin` (the OP-TEE OS binary) can vary depending on the configuration, target platform, and enabled features. It is not possible to provide an exact size, as it can change with each build based on the selected options.

To get the size of your specific `tee.bin` file, you can use the following command in your terminal (assuming you have already built the OP-TEE OS binary):

```
ls -l path/to/your/tee.bin
```

Replace `path/to/your/tee.bin` with the actual path to the binary file. This command will display the file size, among other details.

If you want to get only the file size in bytes, you can use the following command:

```
stat --printf="%s" path/to/your/tee.bin
```

Again, replace `path/to/your/tee.bin` with the actual path to the binary file. This command will output the file size in bytes.

Question: `[14:47]` What is about size of tee.bin? `(OP-TEE.0327.md)`

Answer:

The size of the `tee.bin` file, which is the OP-TEE OS binary, can vary depending on various factors such as the target platform, configuration options, and enabled features. It's difficult to give an exact size for a specific `tee.bin` file without knowing the specific build configuration.

However, generally speaking, the size of the `tee.bin` file is relatively small, usually ranging from a few hundred kilobytes to a couple of megabytes. This small footprint is one of the reasons why OP-TEE is suitable for use in constrained environments, such as embedded systems and IoT devices.

Keep in mind that the actual size of the `tee.bin` file in your specific case can be different depending on the factors mentioned earlier.

Question: `[15:37]` Is SMAF of OP-TEE? `(OP-TEE.0327.md)`

Answer:

SMAF (Secure Memory Allocation Framework) is not directly related to OP-TEE. SMAF is a generic term used to describe a framework for securely allocating memory in a computer system. It is designed to prevent unauthorized access to memory regions and protect sensitive information.

OP-TEE (Open Portable Trusted Execution Environment) is a separate project that provides a TEE (Trusted Execution Environment) implementation for ARM TrustZone-enabled platforms. OP-TEE is designed to secure critical code and data by isolating it from the normal world operating system.

While OP-TEE itself doesn't include a specific component called SMAF, it does provide secure memory allocation and management within the TEE. OP-TEE manages memory allocations securely to ensure the integrity and confidentiality of data within the TEE.

Question: `[15:39]` How is SMAF used in DRM? `(OP-TEE.0327.md)`

Answer:

SMAF (Secure Memory Allocation Framework) can be used in Digital Rights Management (DRM) systems to provide an additional layer of security for protecting sensitive content, such as media files or streaming data. In a DRM context, SMAF can help ensure that only authorized applications or devices can access the protected content.

Here's a high-level overview of how SMAF can be utilized in a DRM system:

1. Secure content storage: SMAF can be used to allocate and manage secure memory regions for storing encrypted content, keys, and DRM-related metadata. This can help protect content from unauthorized access or tampering.
2. Content decryption and playback: When authorized applications or devices request access to protected content, the DRM system can use SMAF to allocate secure memory for the decryption process. This secure memory can be used to store decrypted content or keys temporarily, ensuring that they are not exposed to potentially untrusted parts of the system.
3. Secure communication: SMAF can be employed to allocate secure memory for encrypted communication channels between DRM components. This can help protect sensitive information, such as decryption keys or usage rights, from eavesdropping or tampering.
4. Trusted execution environment: In some DRM systems, SMAF can be used in conjunction with a Trusted Execution Environment (TEE) like OP-TEE, which provides an isolated environment for running security-sensitive code. The TEE can leverage SMAF to allocate secure memory for executing DRM-related tasks, further enhancing the overall security of the system.

By using SMAF in a DRM system, content providers can enhance the security of their content and ensure that it is only accessed by authorized users, devices, and applications.

Question: `[15:41]` What is Clear Key? `(OP-TEE.0327.md)`

Answer:

Clear Key is a simple key management system used in the context of the Encrypted Media Extensions (EME) API, which is part of the HTML5 standard for providing protected media playback in web browsers. The EME API enables web-based applications to use encrypted content and interact with DRM (Digital Rights Management) systems to handle the encryption and decryption of the content.

Clear Key is a basic, non-proprietary key management system provided as part of the EME specification. It is intended primarily for testing and development purposes, rather than for robust content protection in a production environment. Clear Key is not a full-fledged DRM solution; instead, it is a simple way to demonstrate the functionality of the EME API.

In the Clear Key system, the content keys (used for encryption and decryption) are transmitted in the clear, without any obfuscation or additional protection. This makes it straightforward to use but provides minimal security against unauthorized access to the content. The simplicity of Clear Key allows developers to quickly implement and test EME features without having to integrate with a more complex DRM system.

It is important to note that Clear Key is not suitable for commercial use or protecting high-value content, as the keys are exposed and can be easily intercepted by malicious actors. For secure content protection, commercial DRM solutions like Google Widevine, Apple FairPlay, or Microsoft PlayReady should be used instead. These systems provide stronger protection mechanisms to ensure that content remains secure and can only be accessed by authorized users or devices.

Arm.0410.md

Question: `[16:48]` Can I get Armv8-A with only 32-bit for cost down? `(Arm.0410.md)`

Answer:

While the Armv8-A architecture includes both 64-bit (AArch64) and 32-bit (AArch32) execution states, there is no specific variant of Armv8-A that exclusively supports 32-bit operation for cost reduction. The idea behind Armv8-A's dual execution states is to provide compatibility with existing 32-bit software while taking advantage of 64-bit enhancements.

If you are looking for a lower-cost, 32-bit ARM processor option, you may want to consider Cortex-M or Cortex-R series

processors. These processors are designed for cost-sensitive, low-power, and real-time applications, respectively.

Cortex-M series processors, in particular, are widely used in microcontroller and Internet of Things (IoT) applications, where power efficiency and lower costs are essential. Cortex-R series processors cater to real-time systems that require fast, deterministic response times and high reliability. While both of these processor families are primarily 32-bit, the Armv8-M architecture for Cortex-M series also introduced a 64-bit variant (Armv8-M Mainline).
