The Devil is in the Detail: Issues with Fine-Grained Trusted Execution Environments

Jonathan Adshead
jonathan.adshead@fau.de

ABSTRACT

ToDo

ACM Reference Format:
Jonathan Adshead. 2024. The Devil is in the Detail: Issues with Fine-Grained Trusted Execution Environments. In . ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 IDEE

Trusted Execution Environments (TEEs) haben sich als wichtige und notwendige Idee im Bereich der Computersicherheit etabliert, um robuste hardwaregestützte Sicherheitsmechanismen zu bieten. TEEs, wie etwa Intel SGX, ARM TrustZone oder Sancus, ermöglichen die Isolation und Signierung sicherer Anwendungsbereiche, die als Enklaven bezeichnet werden.

Das Prinzip der TEEs erzwingt eine duale Weltansicht, bei der selbst kompromittierte oder bösartige Systemsoftware in der normalen Welt keinen Zugriff auf den Speicherbereich der Enklave hat, die in einer isolierten sicheren Welt auf demselben Prozessor laufen. Dadurch kann die Trusted Computing Base (TBC) reduziert werden. Nur der Code, der in der Enklave läuft, muss überprüft, verschlüsselt und kontrolliert werden. Dennoch bieten TEEs nur eine grobgranulare Speicherisolation auf Hardware-Ebene und überlassen es den Enklaven- und Applikationsentwicklern, die Sicherheit auf Software-Ebene sicherzustellen.

Dadurch entstehen kontinuierlich neue Schwachstellen und Angriffsmöglichkeiten, wie Seitenkanalangriffe, bei denen bereits eine einzige erfolgreiche Attacke ausreicht, um die Isolation und Sicherheit der Enklave zu durchbrechen. In einem solchen Fall könnte die gesamte Enklave kompromittiert werden, was die Sicherheit der gesamten Anwendung gefährdet. Wegen dieser kritischen Schwachstelle kann ein zweiter Ansatz zur Stärkung der Sicherheit notwendig sein, die Kompartimentierung der Software. Gemeint ist dadurch, dass durch den Einsatz fein granularer TEEs die Sicherheit erheblich verbessert wird. Anstatt eine einzige TEE zu nutzen, werden mehrere TEEs implementiert. Dabei ist es auch möglich verschiedene Sicherheitsstufen der einzelnen Enklaven zu unterstützen. Diese Aufteilung bedeutet, dass ein potenzieller Angreifer mehrere, wenn nicht alle TEEs kompromittieren müsste, um das gesamte System zu übernehmen, was die Angriffskomplexität und das Risiko drastisch senkt.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Da das aber auch den größten Nachteil von Kompartimentierung übernehmen würde, die manuelle Aufteilung von Applikationen und die dadurch neue Angriffsfläche, die Compartment Interface Vulnerabilities (CIV), beschäftigt sich dieses Paper mit den Vorteilen, Problemen und Lösungsmöglichkeiten dieser Methode. Insbesondere wird untersucht, wie und ob die Sicherheit durch fein granularer TEEs gesteigert werden kann, ohne dabei die Komplexität und die Angriffsfläche der Anwendungsentwicklung unnötig zu erhöhen,

2 GRUNDLAGEN

2.1 Trusted Execution Environment

2.1.1 TEE Design. Der genaue Aufbau und das Verhalten von Trusted Execution Environments (TEEs) variieren, lassen sich aber im Allgemeinen in zwei Kategorien einteilen: (1) TEEs, die sich einen Adressraum mit dem unprivilegierten Host teilen, und (2) TEEs, bei denen die CPU logisch in eine normale und eine sichere Welt unterteilt ist.

In der ersten Kategorie erzwingt der Prozessor, dass nur die Enklave selbst auf ihren Speicherplatz zugreifen kann. Programme innerhalb der Enklave dürfen jedoch auch auf den Adressraum außerhalb der Enklave zugreifen, wobei Ein- und Ausgabedaten hauptsächlich aus Zeigern bestehen. Diese Struktur ermöglicht eine flexible Datenverarbeitung, birgt jedoch das Risiko unsicherer Speicherzugriffe, was potenziell zu Sicherheitslücken führen kann.

In der zweiten Kategorie ist der Adressraum strikt in zwei separate Bereiche unterteilt, wobei keiner der beiden Bereiche auf den jeweils anderen zugreifen kann. Der Datenaustausch zwischen diesen beiden Welten erfolgt über das Trusted Operating System (TOS). Da das TOS privilegierten Zugriff hat, kann es einen geteilten Adressraum einrichten, in dem Daten sicher ausgetauscht werden können. Diese Methode bietet eine stärkere Isolation und somit ein höheres Maß an Sicherheit, da direkte Speicherzugriffe zwischen der normalen und der sicheren Welt verhindert werden.

In beiden Fällen gewährleistet der Prozessor, dass extern kein Zugriff auf die Daten möglich ist und das TOS, dass die Enclave nicht kompromittiert ist. Die Daten werden ausschließlich im Prozessor entschlüsselt, und durch Signaturen ist es möglich festzustellen, ob die Integrität der TEE noch gewährleistet ist oder ob von außen unautorisierte Zugriffe oder Veränderungen vorgenommen wurden.

2.1.2 TEE erstellen und verlassen. Der Enklave Entry/Exit-Prozess spielt eine zentrale Rolle in Trusted Execution Environments (TEEs) und umfasst die Mechanismen ecall (Entry) und eexit (Exit), die die Kontrolle über den Übergang zwischen der normalen und der sicheren Welt, der Enklave, ermöglichen.

Beim Betreten der Enklave erfolgt dies typischerweise über einen ecall. Hierbei wird von der normalen Welt aus eine spezielle Anweisung an das TOS gesendet, um den Eintritt in die Enklave zu initiieren. Bevor die Ausführung der Enklaven Anwendung gestartet

wird, bereinigt das TOS den Prozessorzustand, um sicherzustellen, dass keine Parameter, Flags oder andere Daten die Funktionsweise der TEE beinflussen können. Die Parameter werden dabei als untrusted parameters an die CPU weitergegeben, damit diese erst überprüft werden, bevor sie als sicher gelten.

Der Enklave-Exit-Prozess wird durch eexit realisiert. Dies geschieht, wenn die Enklave ihre Ausführung beendet, eine externe Anfrage erhält, die den Übergang in die normale Welt erfordert oder ein Interrupt ausgelöst wird. Durch den Aufruf von eexit wird der Zustand der Enklave gespeichert, und die Kontrolle wird an das TOS zurückgegeben. Das TOS bereinigt den Prozessorzustand, um sicherzustellen, dass keine sensiblen Informationen zurückbleiben, bevor der Prozessor aus dem Enklave-Modus in den normalen Betriebsmodus wechselt. Je nach Design werden die Daten für das unsichere OS direkt im Speicher abgelegt oder dafür an das TOS weitergegeben.

Die ecall- und eexit-Mechanismen bieten eine standardisierte Möglichkeit, den Eintritt und Austritt aus Enklaven zu verwalten und gewährleisten eine sichere Ausführung von Anwendungen innerhalb einer Enklave. Diese Mechanismen ermöglichen einen sicheren Übergang zwischen der normalen und der sicheren Welt unter strikter Kontrolle des Trusted Operating Systems.

2.2 Kompartimentierung

Die Kompartimentierung ist ein Konzept, das darauf abzielt, ein Computersystem in separate Bereiche aufzuteilen und diese möglichst isoliert voneinander operieren zu lassen. Diese Praxis beruht auf den Prinzipien der Sandbox und der Savebox, die eine sicherere und stabilere Systemumgebung ermöglichen.

Eine Sandbox ist eine isolierte Ausführungsumgebung, in dem ein Programm ausgeführt wird, welches kein Zugriff auf Daten anderer Programme hat. Dies gewährleistet nicht nur die Sicherheit sensibler Informationen, sondern verhindert auch die Übernahme von Daten oder die Beeinflussung des Programmes durch potenziell kompromittierte Softwareteile.

Im Gegensatz dazu dient die Savebox dem Schutz der eignen Daten. Dies wird ermöglicht, indem nur privilegierten Prozessen Zugriff auf die Daten gestattet wird. Diese Zugangskontrolle verhindert den Zugriff auf Daten eines anderen Kompartimenten.

Die Kombination aus Sandboxing und Safeboxing ermöglicht eine Trennung zwischen den einzelnen Teilen eines Programmes und sorgt so dafür, dass eine Übernahme des Systems oder Zugang zu vertraulichen Daten erschwert wird.

3 ???

3.1 Sicherheitslücken von TEEs

3.2 Sicherheitslücken von Kompartimentierung