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

Jonathan Adshead
jonathan.adshead@fau.de
Friedricht-Alexander-Universität Erlangen-Nürnberg

ABSTRACT

ToDo

1 IDEE

Die exponentielle Entwicklung im Bereich des Cloud-Computing und die zunehmende Verbreitung von Cloud-Diensten haben die Abhängigkeit von verteilten Computersystemen stark erhöht. Immer mehr Unternehmen und Organisationen verlagern Teile oder ihre gesamten Anwendungen in die Cloud, um von Skalierbarkeit, Flexibilität und Kosteneffizienz zu profitieren. Durch die Nutzung von sensiblen Daten und kritischen Anwendungen gewinnt die Sicherheit dieser Cloud Infrastrukturen eine wichtige Bedeutung, weil sie potentiell die sensiblen Informationen von Millionen von Nutzern und Organisationen verwalten.
Trusted Execution Environments (TEEs) sind dadurch zu einer wichtiger Technologie geworden, um die Sicherheit sensibler Daten und kritischer Anwendungen zu sichern. TEEs bieten eine sichere und geschützte Umgebung, in der sensible Daten vor potenziellen Bedrohungen geschützt sind. Die Nutzung von Cloud Diensten und die Verlagerung sensibler Daten in große Rechenzentren zeigt, dass TEEs als Schlüsselkomponente in der Datensicherheit zu sehen sind.
Doch trotz ihrer wichtigen Rolle sind TEEs nicht frei von Schwachstellen. Das Grundprinzip der TEEs erzwingt eine duale Weltansicht in den Hardwarekomponenten, welche in eine normale, möglicherweise kompromittierte und eine sichere und isolierte Welt aufgeteilt ist. Dabei können kompromittierte oder bösartige Systemsoftwareteile in der normalen, bzw. kompromittierten Welt keine Daten auf den sicheren Speicherbereich der Enklave lesen, welche einen isolierten Bereich im Adressraum bezeichnet, der aber näher in 2.1 beschrieben wird. Vor allem diese Schnittstelle zwischen den TEEs und zwischen TEE und unsicheren Welt stellt einen potenziellen Angriffsvektor dar. Diese Schwachstelle birgt das Risiko von Datenlecks und Datenkorruption, sowie Seitenkanalangriffen oder Rollbackangriffen. Ein erfolgreiches Ausnutzen von Angreifern könnte nicht nur sensible Informationen kompromittieren, sondern auch die Integrität der gesamten TEE gefährden.
Dieses Paper legt den Fokus auf die Sicherheitslücken zwischen den Kompartimenten, insbesondere zwischen sicheren Anwendungen innerhalb einer TEE und Anwendungen in der unsicheren Welt. Der Fokus liegt dabei auf Angriffen, die zu Datenlecks und Datenkorruption führen können. (? Zusätzlich wird diskutiert, ob Risiken gemindert und die Sicherheit von TEEs gestärkt werden können)

2 GRUNDLAGEN

Die Trusted Execution Environment (TEE) zeichnet sich durch drei wesentliche Eigenschaften aus, die ihre Funktionalität und Sicherheit beschreiben. Sie gewährleistet (1) die Authentizität des ausgeführten Codes, indem sie sicherstellt, dass dieser tatsächlich von der beabsichtigten Quelle stammt und nicht manipuliert wurde. Sie sichert (2) die Integrität des Codes, indem sie während der Ausführung sicherstellt, dass dieser nicht verändert wurde und sie schützt (3) die Vertraulichkeit von Code, Daten und Laufzeitvariablen, indem sie sicherstellt, dass diese nur von privilegierten Parteien eingesehen werden kann.
Aufgrund der Interaktion von Programmen innerhalb der TEE und Programmen außerhalb, kann die TEE auch als Kompartiment betrachtet werden. Diese Struktur ermöglicht eine klare Trennung zwischen den unterschiedlichen Kompartiments und ermöglicht so eine bessere Kontrolle über den Zugriff.

2.1 TEE

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 konzeptionell in eine normale und eine sichere Welt unterteilt ist.
In der ersten Kategorie hat nur die Enklave selbst vollen Zugriff auf ihren Speicherplatz und kann die Daten im Klartext lesen. Eine Enklave ist ein isolierter Speicherbereich innerhalb einer TEE, der dazu dient, Code und Daten vor unautorisiertem Zugriff und Manipulation zu schützen. Selbst privilegierte Benutzer und das Betriebssystem haben keinen Zugriff auf die innerhalb der Enklave gespeicherten Daten. Programme innerhalb der Enklave dürfen jedoch auch auf den Adressraum außerhalb der Enklave zugreifen.Diese Struktur ermöglicht einen besseren Datenaustausch, 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, jedoch auf Kosten der Datenübertragungsraten.
In beiden Kategorien gewährleistet sowohl der Prozessor, dass extern kein Zugriff auf die Daten möglich ist, als auch das TOS, dass der Code in der Enklave sicher ist. Ein weiterer Grundgedanke beim Designen einer TEE ist Angreifermodel. Man entwickelt die TEE unter der Annahme, dass sie in einem System ausgeführt wird, das als kompromittiert betrachtet wird und Angreifer die volle Kontrolle über sämtliche Hardware haben.
Das TOS übernimmt den Entry/Exit Prozess der TEE, in dem der Übergang zwischen der normalen und der sicheren Welt ausgeführt

wird. Dabei müssen die Daten, wie die Register flags oder der Call stack, aber auch die Eingabedaten, bereinigt werden. Im Exit prozess müssen wiederum die Daten bereinigt werden, damit sie nicht ausgelesen werden können.
Neben den Angriffen auf der Application Binary Interface (ABI), welche auf Schwachstellen beim erstellen und verlassen der TEE abzielen, gibt es noch die Apication Programming Interface, welche versuchen Sicherheitslücken während der Laufzeit zu nutzen.

2.2 Kompartimentierung

Die Kompartimentierung ist ein Konzept, das darauf abzielt, ein Computersystem oder einzelne Anwendungen in separate Bereiche aufzuteilen und diese möglichst isoliert voneinander operieren zu lassen. Diese Praxis beruht auf den Prinzipien der Sandbox und der Safebox, die eine sicherere und stabilere Systemumgebung ermöglichen.
Eine Sandbox ist eine isolierte Ausführungsumgebung, in dem ein Programm ausgeführt wird, welches keinen 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 Safebox 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 Kompartiments.
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 ein Zugang zu vertraulichen Daten erschwert wird.
Das Initialisieren eines Kompartments lässt sich in 3 Stufen unterteilen. Zunächst erfolgt die Identifizierung, wo das Programm logisch gut unterteilt werden kann und wie feingranular diese Unterteilung sein soll. Anschließend wird die Durchsetzung der Grenzen vorgenommen, indem die Programme logisch voneinander getrennt und ausschließlich über eine API zur Kommunikation zugelassen werden.
In der letzten Stufe werden diese Grenzen weiter verstärkt. Hierbei werden die API-Schnittstellen klarer definiert, der Datenfluss möglichst reduziert und Strategien implementiert, wie mit potenziell unsicheren Daten von der unsicheren Welt umgegangen wird.

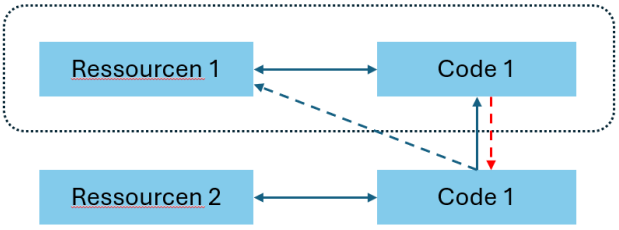

Figure 1: Beschreibung deiner Abbildung hier.

3 WARUM FEINGRANULARE TEES?

Ein wichtiger Begriff im Zusammenhang von TEEs ist die Trusted Computational Base (TCB). Die TBC beschreibt den Bereich eines

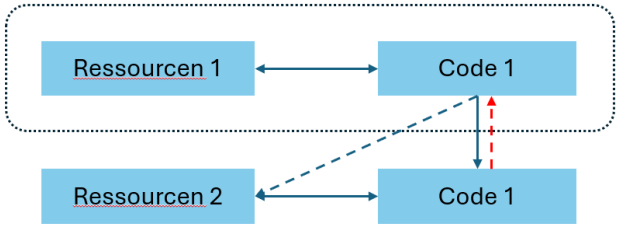

Figure 2: Beschreibung deiner Abbildung hier.

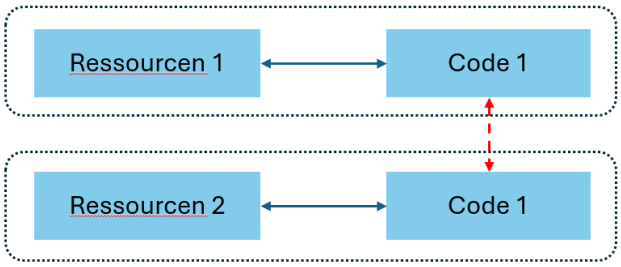

Figure 3: Beschreibung deiner Abbildung hier.

Programms, der besonders vor externen Angriffen geschützt werden muss, da er mit sensiblen Daten arbeitet oder andere kritischen Funktionen übernimmt.
Wird nun ausschließlich dieser Teil der Anwendung in einer TEE ausgeführt, wird von einer feingranularen TEE gesprochen. In einer feingranularen TEE läuft also nicht das gesamte Programm, sondern nur ein spezifischer Teil davon.
Außerdem können mehrere feingranulare TEEs genutzt werden, indem mehrere kritische Funktionen in separaten TEEs ausgeführt werden, was die Sicherheit weiter erhöhen kann. Unabhängig von der Anzahl der verwendeten TEES, bieten sie mehrere Vorteile.

3.1 Minimierung der Angriffsfläche

Der Hauptgrund für die Verwendung von TEEs ist die Minimierung der Angriffsfläche. Diese Minimierung wird erzielt, da der Prozess, der innerhalb einer sicheren Enklave läuft, keinen uneingeschränkten Zugriff auf externe Ressourcen hat.
Ebenso haben Programme in der unsicheren Umgebung keinen direkten Zugriff auf Daten, die innerhalb der sicheren Enklave verarbeitet werden. Da TEEs in diesem Kontext als Kompartiments behandelt werden können, gibt es für die Kommunikation zwischen sicherer und unsicherer Welt eine definierte Schnittstelle, auch als API bekannt.
Diese Schnittstelle ermöglicht es, genau zu kontrollieren, welche Daten und in welchem Format sie von außen akzeptiert werden. Obwohl diese Schnittstelle die Sicherheit erhöht, stellt die API-Schnittstelle einen potenziellen Angriffsvektor dar und führt zu einer neuen Klasse von Sicherheitsproblemen, den Compartment Interface Vulnerabilities (CIVs).

3.2 Isolierung von Schwachstellen

Eine weitere wichtige Eigenschaft feingranularer TEEs ist ihre Fähigkeit zur Isolierung von Schwachstellen. Jedes Kompartiment operiert unabhängig und sicher, ohne auf die Integrität anderer Kompartimente angewiesen zu sein.
Dadurch haben Sicherheitslücken in einem Kompartiment keine Auswirkungen auf die anderen. Je mehr Kompartimente eingesetzt werden, desto widerstandsfähiger wird das gesamte System gegen Angriffe. Allerdings steigt mit der Anzahl der feingranularen TEEs auch der Aufwand für die Erstellung, Überprüfung und Verwaltung der Daten sowie der gesamte Overhead. Dennoch kann ein erfolgreicher Angriff auf ein einzelnes Kompartiment möglicherweise nur einen Teil der Daten kompromittieren, während der Rest des Systems sicher bleibt.

3.3 Unterschiedliche Sicherheitsstandards

Ein Ziel neben der weiteren Minimierung der Angriffsfläche könnte die Einführung von verschiedenen Sicherheitsprofilen sein. Diese Strategie beinhalten nicht nur die Berücksichtigung der Interaktion zwischen den Kompartimenten bei ihrer Identifizierung, sondern auch die Festlegung welche Funktionen geschützt werden müssen und welche nicht.
Werden mit diesem Gedanken mehrere Kompartimente erstellt, können unterschiedliche Sicherheitsstandards eingeführt werden. So wird bei den Kompartimenten, in denen kritische Funktionen ausgeführt werden, besonders auf die Sicherheit geachtet. Durch diese Implementierung ist es möglich den Overhead von TEEs zu reduzieren, während ein nahezu gleiches Sicherheitsniveau beibehalten wird.
Erwähnenswert ist auch, dass bei der Identifizierung der Funktionen überprüft werden kann, ob bestimmte Funktionen einen ausschließlich lesenden oder schreibenden Zugriff erfordern. Funktionen mit einer dieser Eigenschaft können nicht zu Datenlecks oder Datenkorruption führen.

4 SICHERHEITSPROBLEME VON FEINGRANULAREN TEES

Wird die Sicherheit der Daten betrachtet, offenbart sich ein signifikanter Unterschied zwischen Trusted Execution Environments (TEE) mit einem gemeinsamen Adressraum und solchen mit einem separaten. Bei streng separierten Enklaven gestaltet es sich äußerst schwierig für unsichere Prozesse, die TEE zur Freigabe von Daten aus ihrem Speicher freizugeben. Die Konfiguration des Trusted Operating Systems (TOS) kann entweder eine Spiegelung eines Teils des Adressraums oder den Zugriff auf externe Daten für den Datenaustausch in und aus der Enklave ermöglichen. In beiden Fällen ist jedoch ein direkter Zugriff des Programms auf den Speicher der Anwendung nicht mehr gegeben.
Hugo Lefeuvre et al. haben zwei große Kategrorien von Sicherheitslücken identifiziert: Datenlecks und Datenkorruption. Diese Kategorisierung erweist sich als äußerst nützlich, da nahezu alle bekannten Angriffe auf TEEs in eine dieser beiden Kategorien passen. Auch die Angriffsvektoren, die von Jo Van Bulck et al. gefunden und untersucht wurde, lassen sich in diese Kategorien einteilen, was die Gültigkeit dieser Klassifizierung weiter unterstützt. Die Einteiung in Datenlecks und Datenkorruption ermöglicht eine Identifizierung der Angriffsziele und eine Einschätzung der potenziellen Auswirkungen auf die Sicherheit und Integrität der Daten. [3] [2]

4.1 Datenlecks

Die Kategorie der Datenlecks umfasst Angriffe, die darauf abzielen, in irgendeiner Form Daten aus der Enklave zu extrahieren.
Ein besonders anfälliges Angriffsziel ist der Zustand der TEE beim Verlassen oder Beenden. Unabhängig davon, ob die Beendigung geplant oder durch ein Interrupt erzwungen wird, muss sichergestellt werden, dass keine Daten in einem lesbaren Zustand im Speicher verbleiben. Da das Bereinigen des Speichers eine explizite Softwareaufgabe darstellt, liegt es in der Verantwortung der Entwickler, dies korrekt zu implementieren. Selbst wenn das Bereinigen der Registerzustände in normalen Prozesswechseln üblich ist und automatisch erfolgt, ist diese Schwachstelle nicht zu unterschätzen. Wenn dieser Prozess nicht ordnungsgemäß durchgeführt wird, besteht für Angreifer die Möglichkeit, Registereinträge oder den Stack auszulesen. Dadurch können sie Rückschlüsse auf die Funktionsweise der Anwendung und die letzten Aktionen innerhalb der TEE ziehen.
Die Bedeutung dieser Sicherheitsmaßnahmen wird durch die potenziellen Konsequenzen unzureichender oder falscher Speicherbereinigung unterstützt. Angreifer könnten sensitive Informationen wie kryptografische Schlüssel, Passwörter oder andere vertrauliche Daten extrahieren und somit die Sicherheit der gesamten Anwendung kompromittieren. Daher ist es von entscheidender Bedeutung, dass Entwickler sorgfältig darauf achten, dass alle sicherheitskritischen Daten beim Verlassen der TEE vollständig und sicher gelöscht werden.
Neben dem Verlassen ist die TEE auch während der Laufzeit angreifbar. Konkret ist das über die Parameter möglich, die die TEE von unsicheren Prozessen von außen bekommt. Es ist essenziell, dass diesen Parametern nicht von anfang an vertraut wird. Diese Daten müssen zuerst überpüft und in manchen fällen bereinigt werden. So muss z.B in jedem Fall nicht nur überprüft werden, dass der Inputpointer außerhalb der Enklave liegt, sondern auch, dass der gesammte Speicherbereich des Pointers außerhalb liegt, was je nach Datentyp oder Strukt nicht nur aus einer einfach Überprüfung besteht. Da die größe des Pointers Teil der Eingabe ist und nach dem Angreifermodell davon ausgegangen wird, dass das komplette system möglicherweise komromittiert ist, kann diesem Wert nicht vertraut werden
So ist es z.B möglich, Situationen zu erzeugen in dem die Size wegen Überläufen nicht mehr richtig berechnet wird und so nicht erkannt wird, dass der Datenbereich eigendlich den der Enklave schneidet. Aus diesem Grund ist es dringen notwendig, beim Berechnen von Adressen und beim grundsätzlichen Verwenden von unsicheren Daten, sichere Arithmetik zu verwenden um fehlerhafe Berechnungen zu vermeiden.
Dieser Angriff kann aber Mithilfe der Manipulation von Strings noch weiter geführt werden. Bei der Verwendung einer Low-Level-Sprache wie C verhält sich ein String auch als ein Pointer, welcher auf ein Array zeigt, in dem die Länge mit angegeben wird. Schafft es nun ein Angreifer, z.B. mithilfe der oben beschriebenen Möglichkeit, diesen Datenbereich mit dem der Enklave überschneiden zu lassen, ist es ihm nun Möglich, nahezu den kompletten Speicher auszulesen.

Durch die absichtlich falsche Größenangabe und das Weglassen der Nullterminierung, die das Ende eines Strings definiert, kann der Angreifer über das Programm in der TEE auf die Daten innerhalb der Enklave zugreifen. Nun kann der Angreifer über diesen Seitenkanal die Daten auslesen. Ein Seitenkanalangriff meint dabei, dass Rückschlüsse über die Daten anhand von der Ausführzeit oder den Cache Zugriffen gewonnen werden können. [3] Wird nun der String ausgelesen kann der Angreifer anhand der Dauer feststellen, wann das nächste Nullbyte im Speicher liegt. Wird dies nun an beliebigen oder auch allen Speicheradressen gemacht, können alle Nullbytes in der Enklave identifiziert werden.
Je nach Implementation der TEE ist dies anders oder überhaupt Möglich. So ist es z.B. bei Intel SGX-SDK sogar möglich für Angreifer, ein volles Speicherabbild zu generieren. Konkret möglich ist das durch das EDL Attribut, welches das Ende eines Strings definiert. Da dies Standardmäßig das Nullbyte ist, können normalerweise auch nur die Nullbytes ausgelesen werden. Wird dieses EDL Attribut aber mit einer eins überschrieben ist es möglich alle einser in der Enklave zu lokalisieren. Wird dieser Vorgang bis 255 wiederholt, ist es für einen Anwender möglich ein komplettes Speicherabbild zu generieren, was die Sicherheit der TEE komplett untergraben würde. [1] [3]
Bei diesem Angriff ist aber Anzumerken, dass TEEs nach dem single-World-Prinzip deutlich anfälliger sind, da die Enklave in den speicher des Angreifers eingebettet ist. Bei TEEs nach dem Two-Worlds-Prinzip werden die Daten nur über das TOS geteilt. Da aber in keinem System eine absolute Sicherheit möglich ist, gibt es auch hier Schwachstellen, die von Angreifern ausgenutz werden können.

4.2 Datenkorruption

5 FAZIT

ToDo: Fazit zu meinem Paper

5.1 A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes

Meinung zum Paper

5.2 Assessing the Impact of Interface Vulnerabilities in Compartmentalized Software

Meinung zum Paper

REFERENCES

[1] Victor Costan and Srinivas Devadas. Intel SGX explained. Cryptology ePrint Archive, Paper 2016/086, 2016. https://eprint.iacr.org/2016/086.

[2] Hugo Lefeuvre, Vlad-Andrei Bădoiu, Yi Chen, Felipe Huici, Nathan Dautenhahn, and Pierre Olivier. Assessing the impact of interface vulnerabilities in compartmentalized software. In *Proceedings 2023 Network and Distributed System Security Symposium*, NDSS 2023. Internet Society, 2023.

[3] Jo Van Bulck, David Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D. Garcia, and Frank Piessens. A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1741–1758, New York, NY, USA, 2019. Association for Computing Machinery.