MONTHLY SECURITY SUMMARY

)SCIP(

AUSGABE NOVEMBER 2021
CONTAINER SANDBOXING UND JWT LOGOUTS

SANDBOXING IN CONTAINERN NUTZEN

In unserem Artikel wird erörtert, wie dem Host, auf dem nicht vertrauenswürdiger Code in einem Container ausgeführt wird, zusätzliche Schutzschichten hinzugefügt werden können.

HERAUSFORDERUNGEN VON JWT MEISTERN

Entwicklern ist oft nicht bewusst, dass mit JWT zu Problemen beim Widerruf von Tokens und dem Logout von Benutzern kommen kann. Wir zeigen, wie damit umgegangen werden kann.

November 2021: Mythos Computerviren

Es gibt verschiedene Themengebiete, die sich aufgrund ihrer Mystifizierung und Legendenbildung geradezu für populäre Irrtümer anbieten. Neben *Nachrichtendiensten*, *politischen Verschwörungen* und der *traditionellen Kriegsführung* gehört der Bereich der *Computersicherheit* ohne Zweifel zum beliebtesten Spekulationsthema am Stammtisch.

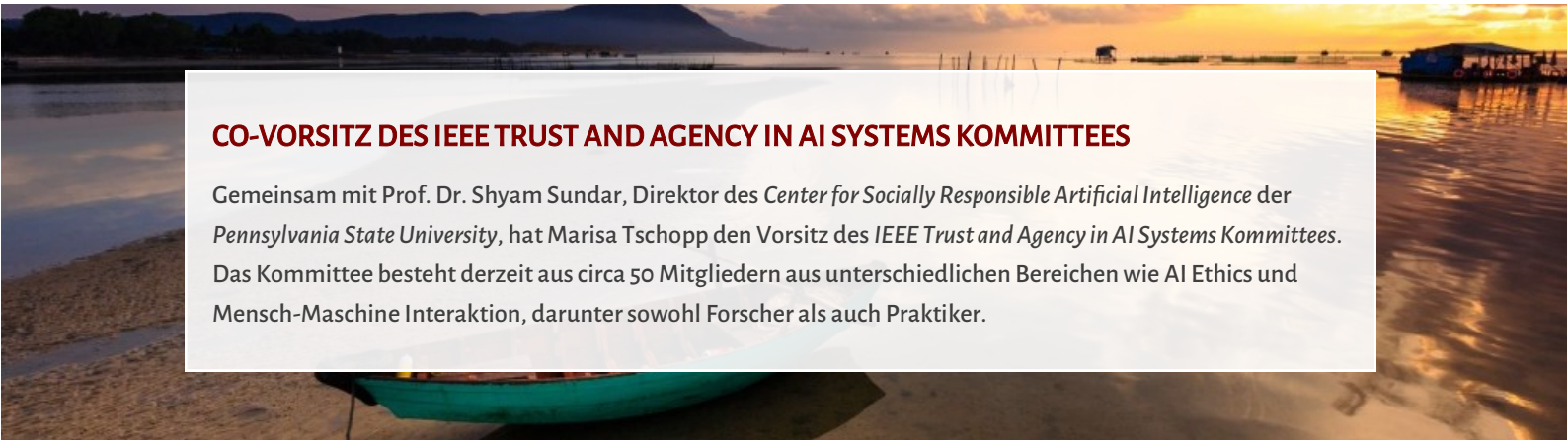
Wer kein Programm öffnet, kann keinen Virus haben? *Nein*, dies ist seit Jahren nicht mehr der Fall. Ursprünglich setzten Computerviren voraus, dass sich diese in *ausführbaren Dateien* (z.B. EXE und COM) einnisten konnten. Wird die Wirtsdatei ausgeführt, wird der Virus aktiv und geht seiner Arbeit nach. Deshalb hiess es, dass wer auf das Ausführen zwielichtiger Dateien verzichtet, sich keinen Virus einfangen kann.

Mittlerweile sind Computersysteme und Dokumentdateien so *komplex* geworden, dass sich selbst in diesen langwierige Dateioperationen einbringen lassen. Durch das Ausnutzen eines *Fehlers im Parsing* wird es auch möglich, Viren über Bilder und Musikdateien zu verbreiten (z.B. als Payload eines Pufferüberlauf). Jedes Bit, das von einem Computer interpretiert wird, kann also als Träger für einen korrupten Programmcode herhalten. Ein ausserordentlich gewiefter und effizienter Vertreter dieser Richtung war *W32.Blaster.Worm*. Glauben Sie also nicht immer alles, was erzählt wird!

Marc Ruef
Head of Research

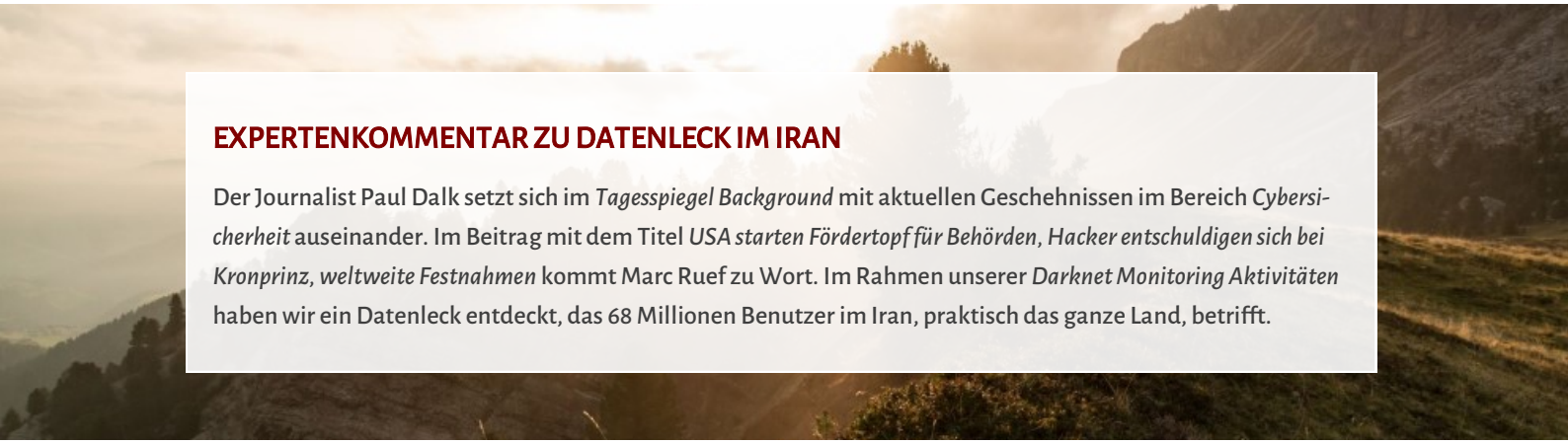
NEWS

WAS IST BEI UNS PASSIERT?



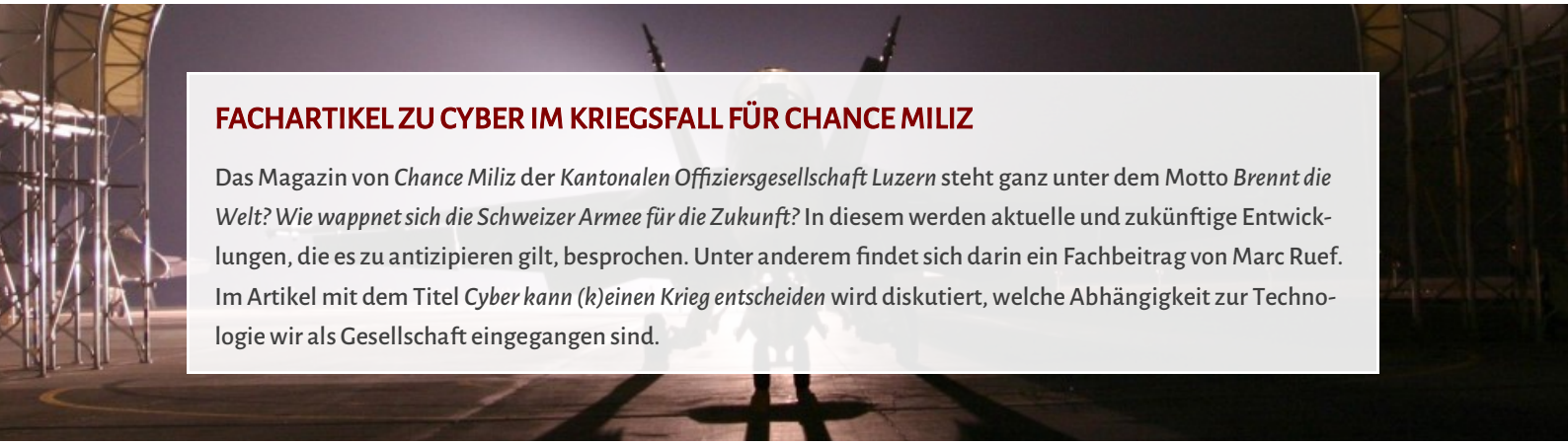
CO-VORSITZ DES IEEE TRUST AND AGENCY IN AI SYSTEMS KOMMITTEES

Gemeinsam mit Prof. Dr. Shyam Sundar, Direktor des *Center for Socially Responsible Artificial Intelligence* der *Pennsylvania State University*, hat Marisa Tschopp den Vorsitz des *IEEE Trust and Agency in AI Systems Kommittees*. Das Kommittee besteht derzeit aus circa 50 Mitgliedern aus unterschiedlichen Bereichen wie AI Ethics und Mensch-Maschine Interaktion, darunter sowohl Forscher als auch Praktiker.



EXPERTENKOMMENTAR ZU DATENLECK IM IRAN

Der Journalist Paul Dalk setzt sich im *Tagesspiegel Background* mit aktuellen Geschehnissen im Bereich *Cybersicherheit* auseinander. Im Beitrag mit dem Titel *USA starten Fördertopf für Behörden, Hacker entschuldigen sich bei Kronprinz, weltweite Festnahmen* kommt Marc Ruef zu Wort. Im Rahmen unserer *Darknet Monitoring Aktivitäten* haben wir ein Datenleck entdeckt, das 68 Millionen Benutzer im Iran, praktisch das ganze Land, betrifft.



FACHARTIKEL ZU CYBER IM KRIEGSFALL FÜR CHANCE MILIZ

Das Magazin von *Chance Miliz* der *Kantonalen Offiziersgesellschaft Luzern* steht ganz unter dem Motto *Brennt die Welt? Wie wappnet sich die Schweizer Armee für die Zukunft?* In diesem werden aktuelle und zukünftige Entwicklungen, die es zu antizipieren gilt, besprochen. Unter anderem findet sich darin ein Fachbeitrag von Marc Ruef. Im Artikel mit dem Titel *Cyber kann (k)einen Krieg entscheiden* wird diskutiert, welche Abhängigkeit zur Technologie wir als Gesellschaft eingegangen sind.

Weitere News zu unserem Unternehmen finden Sie auf unserer Webseite.

SCIP BUCHREIHE

VERÖFFENTLICHUNG DES NEUEN JAHRBUCHS

Erneut veröffentlichen wir die aktuelle Ausgabe unseres Jahrbuchs. Bereits zum 11ten Mal fassen wir in diesem die Fachbeiträge von einem Jahr Forschung im Bereich Cybersecurity zusammen.

Das Buch ist wiederum sowohl in deutscher (ISBN 978-3-907109-26-7) als auch in englischer Sprache (ISBN 978-3-907109-27-4) verfügbar. Das Vorwort wurde von Marko Rogge, seines Zeichens IT-Sicherheitsbeauftragter für Mobile Security bei der Deutschen Bahn AG, verfasst.

In unserem Katalog finden sich ebenfalls themenspezifische Bücher zu Künstlicher Intelligenz (ISBN 978-3-907109-14-4) und Sicherheit in der Hotellerie (978-3-907109-16-8).

Weitere Informationen und Bestellungen auf unserer Webseite.

ISBN 978-3-907109-26-7 [de]
ISBN 978-3-907109-27-4 [en]

EINFACHHEIT IST OFT DIE LÖSUNG

Bildquelle: https://pixabay.com/images/id-1205611/

ROCCO GAGLIARDI

SO FUNKTIONIERT SANDBOXING VON CONTAINERN

Über *Container* zu sprechen, kann sehr kompliziert sein. Es gibt eine Menge Software, Schnittstellen, Standards und Spezifikationen, die alle sehr komplex sein können. *Kubernetes* (und Derivate wie OpenShift) wurden zum Standard für die Ausführung von containerisierten Anwendungen, aber das Ziel von Kubernetes ist es, die Container zu orchestrieren und nicht, sie auszuführen. Selbst Kubernetes oder OpenShift benötigen eine komplexe zugrunde liegende Technologieschicht. Die Bestandteile dieser Schicht können nach und nach geändert oder abgestimmt werden. In diesem Beitrag werden wir erörtern, wie Sie dem Host, auf dem nicht vertrauenswürdiger Code in einem Container ausgeführt wird, zusätzliche Schutzschichten hinzufügen können.

Der Begriff Container-Laufzeit kann sehr verwirrend sein. Im Allgemeinen bezieht sich der Begriff auf die Software, die für die Ausführung des Containers verantwortlich ist, aber einige Laufzeiten (sogenannte High-Level-Laufzeiten) sind eine Reihe von Tools für die Ausführung und Verwaltung von Containern. Ian Lewis erklärt den Begriff Container Runtimes im Detail.

In vielen Artikeln im Internet wird die Geschichte der Container-Runtimes erzählt und warum Kubernetes Docker als Standard-Runtime abgelöst hat. Docker begann als eine Reihe verschiedener einfacher Tools (gemäss der Unix-Philosophie), die jeweils eine ganz bestimmte Aufgabe hatten und im Allgemeinen auf die Kommunikation mit Peripheriegeräten ausgerichtet waren, z.B. den Umgang mit Netzwerken oder Speicher. Im Laufe der Zeit entwickelten Docker, CoreOS, Google und andere angesichts der verschiedenen Probleme, die dieser Ansatz bei gross angelegten Umsetzungen verursachte, die *Open Container Initiative* (OCI), die offene Industriestandards für Containerformate und Laufzeiten definiert. Die OCI enthält derzeit zwei Spezifikationen: Die *Runtime Specification* (runtime-spec), die definiert, wie ein Image ausgeführt wird, und die *Image Specification* (image-spec), die definiert, wie Informationen in ein lauffähiges Image gepackt werden (Manifest, Dateisystem, Konfiguration).

Docker hat das Containerformat und die Laufzeit *runC* entwickelt und der OCI zur Verfügung gestellt, um als Referenzimplementierung zu dienen. Um mit den verschiedenen Laufzeiten zu kommunizieren, wurde ein weiterer Standard entwickelt: *Container*

Runtime Interface (CRI), der es den Administratoren ermöglicht, verschiedene Laufzeiten zu wählen, solange sie OCI-kompatibel sind.

Nachdem die Standards für die verschiedenen Ebenen definiert wurden, wurden für jede Ebene verschiedene Tools entwickelt, so dass wir für jeden Bedarf die beste Kombination wählen können.

CONTAINER-SECURITY

Container wurden lange Zeit mit einer Art *Sandkasten* verwechselt, aber Container sind keine Sandkästen, sie schützen das Hostsystem nicht standardmässig, und es sind Konfiguration und Tuning erforderlich, um ihr Potenzial unter Kontrolle zu halten.

Der typische Konfigurationsprozess umfasst:

- **Minimieren der Benutzerrechte**: Beschränkung des Zugriffs auf Komponenten, Netzwerk, Dateisystem, Signals, Syscalls
- **Funktionen abschalten**: Deaktivieren Sie nicht mehr benötigter Funktionen
- **C-Gruppen-Tuning**: Ressourcen begrenzen, Kerne zuweisen, Gerätezugriff kontrollieren
- **Namespace einrichten**: Isolieren der Container

Beispielsweise kann es gefährlich sein, Zugriff auf alle Syscalls (348) zu gewähren, die der Kernel anbietet, wenn Sie keine vollständige Kontrolle über den Code haben, der innerhalb des Containers läuft. Und selbst dann wäre es besser, die Verwendung aller unnötigen Ressourcen zu verhindern.

Docker war in seinen Anfängen sehr grosszügig mit der Rechteverwaltung, gepaart mit der Verwirrung der Rollen (Docker kann alles: Images ausführen, Images, Volumes, Netzwerk und so weiter verwalten). Dies sind einige der Gründe, die zur Entwicklung anderer Laufzeiten geführt haben, die darauf ausgelegt sind, den Funktionsumfang zu minimieren und über klar definierte Rollen zu verfügen.

NATIVE ODER VIRTUALISIERTE LAUFZEITUMGEBUNG

Die OCI-Familie von Laufzeiten kann in zwei Gruppen unterteilt werden:

- **nativ**: Führt den containerisierten Prozess auf demselben Host-Kernel aus; Low-Level-Laufzeiten wie *runC* oder *crun* (RedHat-Implementierung der OCI-Spezifikationen).

- **virtualisiert**:

    - Sandbox-Laufzeiten mit einem eigenen Kernel, die eine weitere Isolierung des Hosts vom containerisierten Prozess ermöglichen; Beispiele sind gVisor oder nabla.

    - Leichtgewichtige virtuelle Maschinen (VMs), die sich wie Container anfühlen und funktionieren, aber die Workload-Isolierung und die Sicherheitsvorteile von VMs bieten, z.B. Kata.

Die Verwendung virtualisierter Laufzeiten könnte die Lösung sein, um die Sicherheit der Virtualisierung (VMs) und die Geschwindigkeit des Containers zu nutzen, aber es gibt einige Nachteile: Im Gegensatz zu nativen Laufzeiten beeinträchtigen Laufzeiten in Sandkästen die Leistung während der gesamten Lebensdauer eines containerisierten Prozesses.

In Container-Sandboxen gibt es eine zusätzliche Abstraktionsebene: Der Prozess läuft auf dem Unikernel, der Anweisungen an den Host-Kernel weitergibt. Der Zugriff auf das Dateisystem, der über einen Proxy erfolgt, oder auf Geräte wird ebenfalls kontrolliert und von der Leistung beeinflusst.

In der virtualisierten Familie gibt es eine zusätzliche Unterscheidung: VMs und Syscall-Filter.

Kata Container implementiert die VM-Virtualisierung. Die wichtigsten Funktionen lassen sich wie folgt zusammenfassen:

- Vollständiger Kernel auf einer leichtgewichtigen VM

- Da Syscalls durchlaufen werden, gibt es eine vollständige Syscall-Unterstützung

- Leistungseinbussen aufgrund der VM-Schicht

- Kann jede Anwendung ausführen

- Kann in verschachtelten virtualisierten Umgebungen ausgeführt werden, wenn der Hypervisor und die Hardware dies unterstützen

Solange die Leistung der sicherheitsorientierten Laufzeiten nicht an die der nativen Laufzeiten heranreicht, werden wir eine Mischung aus all diesen Lösungen sehen.

GVISOR

Im Jahr 2018 hat Google gVisor veröffentlicht, einen User-Space-Kernel für Container. gVisor präsentiert sich den Anwendungen als normaler Kernel, stellt aber eine geringere Anzahl von Syscalls zur Verfügung, verwendet eine andere Sprache (Go) und reduziert so die Möglichkeit, dass in beiden Kerneln dieselben Fehler auftreten. gVisor filtert das Dateisystem und den Netzwerkzugriff über einen speziellen und separaten Mechanismus.

Die wichtigsten Merkmale lassen sich wie folgt zusammenfassen:

- Kernel (teilweise) im Userspace. Von 348 Syscalls haben 260 Syscalls eine vollständige oder teilweise Implementierung.
- Abfangen und Filtern von Syscalls durch ein *Sentry Modul* (Interpretation und Filterung).
- Leistungseinbussen zur Laufzeit aufgrund der Filterung von Syscalls bei syscall-intensiven Anwendungen.
- Leistungseinbussen zur Laufzeit bei dateisystemintensiven Anwendungen aufgrund des Proxys durch Gofer der Dateisystemzugriffe.
- Es können nur Anwendungen ausgeführt werden, die implementierte Syscalls verwenden.

Wie bereits erwähnt, gibt es Nachteile bei der Verwendung von gVisor, vor allem hinsichtlich der Leistung. Für weitere Analysen lesen Sie bitte *Die wahren Kosten von Containing: Eine gVisor-Fallstudie*.

Anwendung

Die Installation ist ziemlich einfach. Nach der Installation kann eine neue Laufzeitumgebung zur Docker -Konfiguration hinzugefügt werden:

```
{
   "runtimes": {
       "runsc": {
            "path": "/usr/local/bin/runsc",
            "runtimeArgs": [
                "—overlay",
                "—network=sandbox",
                "—debug-log=/tmp/runsc/",
                "—debug",
                "—strace"
            ]
       }
   }
}
```

Angabe eines zusätzlichen Schutzes:

- **Overlay**: Umhüllen Sie Dateisystem-Einhängungen mit einem beschreibbaren Overlay. Alle Änderungen werden im Speicher innerhalb der Sandbox gespeichert.
- **Netzwerk**: Gibt an, welches Netzwerk verwendet werden soll: Sandbox (Standard), Host, keins. Die Verwendung eines Netzwerks innerhalb der Sandbox ist sicherer, da es vom Host-Netzwerk isoliert ist.
- **Debug-Optionen**

Verwenden Sie *runsc flags*, um die Dokumentation der Laufzeitoptionen anzuzeigen.

Die Ausführung desselben Containers mit runc und runsc zeigt den Unterschied zwischen nativ (Weitergabe von Aufrufen an den Host-Kernel) und sandboxed (präsentiert sich als anderer Kernel):

```
[root@localhost cnt]# docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED
SIZE
alpine       latest    14119a10abf4   2 months
ago   5.6MB
[root@localhost cnt]# docker run —rm —
runtime=runc alpine uname -r
4.18.0-348.el8.x86_64
```

```
[root@localhost cnt]# docker run —rm —
runtime=runsc alpine uname -r
4.4.0
```

runsc gibt sich gegenüber der Anwendung als Kernel 4.4.0 aus.

Gleichzeitig funktionieren einfache Angriffe wie Root-Mount nicht.

ANWENDUNGSFÄLLE

Im Allgemeinen ist der perfekte Anwendungsfall für gVisor, wenn ein Container nicht vertrauenswürdigen Code ausführen muss. In diesem Fall kann gVisor mit einem einfachen Schalter in der Docker- oder Kubernetes-Konfiguration zusätzliche Sicherheitsebenen hinzufügen.

Für eine typische Kundenumgebung, in der kein nicht vertrauenswürdiger Code gehostet werden muss, könnte eine Mischung aus gVisor für offene Anwendungen und Standard-Laufzeiten für Anwendungen mit überarbeitetem oder vertrauenswürdigem Code in Frage kommen. Im Falle eines Kubernetes-Clusters ist es möglich, eine Untergruppe von Knoten für die Ausführung von sensiblem Code zu erstellen und gVisor auf diesen Knoten zu installieren. Der Rest der Knoten kann Container mit standardmässigen, leistungsfähigeren Laufzeiten ausführen.

ZUSAMMENFASSUNG

Container sind sehr beliebt, da sie leistungsstark und flexibel sind, aber die komplexe Technologie, die die Umgebung steuert, stellt viele Herausforderungen an die Sicherheit. Wenn Sie nicht vertrauenswürdigen Code auf Ihren Containern ausführen müssen, wäre es sinnvoll, eine zweite Schutzschicht zu implementieren. In diesem Fall ist gVisor eine gute Wahl für den Anfang.

Es erfordert zwar eine ausführliche Prüfung der Kompatibilität mit der Anwendung und eine sorgfältige Abwägung der Auswirkungen auf die Leistung, aber durch die Verwendung einer anderen Sprache (go) und die strenge Kontrolle der Syscalls bietet gVisor eine solide zweite Verteidigungslinie für eine Umgebung, die dem Code nicht vertraut.

In Anbetracht der sicheren containerisierten Umgebungen mit aktualisierter Bedrohungsmatrix für Kubernetes kann gVisor dazu beitragen, einige Techniken zu entschärfen, die in den Taktiken der *Ausführung*, *Persistenz* und *Rechteerweiterung* verwendet werden. Wir werden solche Implementierungen in Zukunft noch häufiger sehen.

Rocco Gagliardi

next gen vulnerability intelligence

VulDB

Warten Sie nicht, bis es brennt.

Angreifer finden täglich neue Schwachstellen und suchen lohnende Ziele, um sie anzuwenden. Durch die tägliche Dokumentation neuer Schwachstellen, detaillierte Analyse der technischen Hintergründe, exklusive Details zu Exploiting und Gegenmassnahmen erhalten Sie mit vuldb.com ein durchschlagskräftiges Werkzeug in die Hand, um sich proaktiv dagegen wehren zu können. Setzen Sie auf uns, noch heute!

)SCIP(
official data provider

> https://vuldb.com

ANDREA HAUSER

DAS SIND DIE SCHWIERIGKEITEN VON JWT

In letzter Zeit wurden während unseren Tests einige Webapplikationen angetroffen, welche *JSON Web Token* (JWT) als klassische *Session Tokens* verwendeten. Dabei wurde festgestellt, dass vielen Entwicklern nicht bewusst war, dass es mit *JWTs* zu Problemen beim Widerruf solcher Tokens und somit auch dem Logout von Benutzern kommen kann. In diesem Artikel soll auf diese und weitere Problematiken von JWTs eingegangen und aufgezeigt werden, in welchen Situationen der Einsatz von JWTs sinnvoll ist.

DEFINITION JWT UND KLASSISCHES SESSION TOKEN

RFC 7519 definiert JWTs als ein Mittel zur *Repräsentation von Forderungen* beziehungsweise *Claims*, die zwischen zwei Parteien übertragen werden. Wenn JWTs dazu genutzt werden, die Sessions von Benutzern zu verwalten, werden dementsprechend die *Berechtigungen eines Benutzers* im JWT festgehalten. Ein solcher Claim, der in einem JWT transportiert wird, könnte dementsprechend wie folgt aussehen:

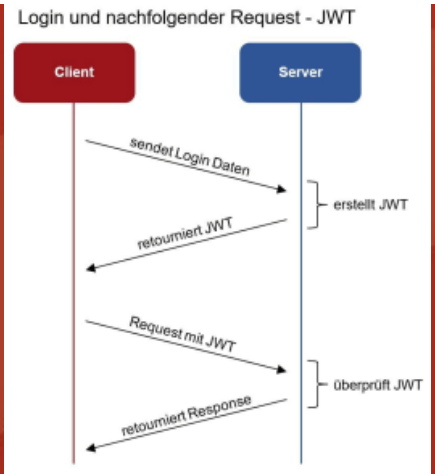
```
{"user":"test", "id":"123", "gui":"full_access",
"admin":"true", "exp":"1634205600"}
```

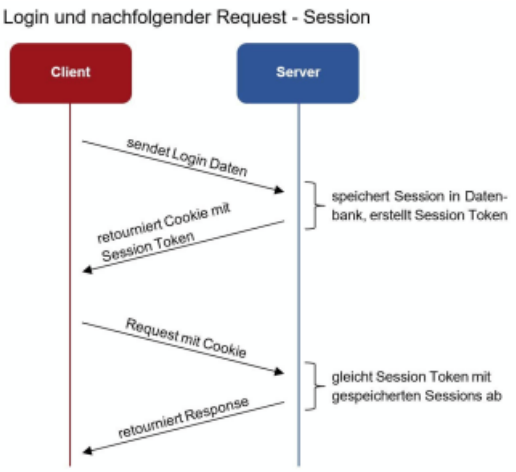
In diesem Claim befindet sich alles, was für die Validierung der Berechtigungen eines Benutzers serverseitig notwendig ist. Dementsprechend kann durch die Validierung der Signatur des JWTs die Validität des Claims festgestellt werden. Es besteht also keine Session per se, sondern es wird mit jedem Request die Validität des Claims überprüft. Der im Beispiel oben abgebildete Claim ist bis am 14. Oktober 2021 um 12:00:00 Uhr gültig. Danach wird das JWT aufgrund des *exp-Parameters ablaufen*.

Ein *klassisches Session Token*, das meist mittels Cookies im Browser gespeichert wird, beinhaltet lediglich eine *ID*, mit der *serverseitig* die Informationen zum Benutzer *abgerufen* werden können. Mit dem Session Token wird eine spezifische Session eines Benutzers verbunden. In der Regel wird eine Session nach einer gewissen Zeit aufgrund von *Inaktivität* oder bei einem *Logout beendet* und das Session Token kann nicht mehr verwendet werden.

PROBLEMATIK BENUTZUNG VON JWTS ALS SESSION TOKEN

Bereits im Artikel *Signierte JSON Web Token* wurde auf einige Schwachstellen wie der Algorithm:None-





Angriff sowie die Problematik von gewissen verwendeten Signaturalgorithmen eingegangen. Auf diese Probleme wird hier nicht im Detail eingegangen.

Wie bereits aus der Definition der JWTs oben ersichtlich ist, gibt es für JWTs, sobald sie einmal ausgestellt sind, keinen Weg diese zu invalidieren oder zu widerrufen. Dies bedeutet auch, dass ohne Eigenentwicklungen kein funktionierendes Logout umgesetzt werden kann. Des Weiteren ist es *nicht möglich*, die in einem Claim enthaltenen *Berechtigungen anzupassen*, solange ein JWT gültig ist. Wenn der Claim

```
{"user":"test", "id":"123", "gui":"full_access",
"admin":"true", "exp":"1634205600"}
```

als Beispiel genommen wird, bedeutet das also, dass der Benutzer *test* bis zum Ablauf des JWTs administrative Berechtigungen hat, auch wenn ihm diese serverseitig schon entzogen wurden. Da die Überprüfung der Berechtigungen nur über das JWT durchgeführt wird, kann die serverseitige Anpassung von Berechtigungen nicht erkannt werden. Diese Einschränkungen müssen Entwicklern bewusst sein, bevor der Einsatz von JWTs umgesetzt wird. Wie auch OWASP in ihrem *JWT Cheat Sheet* empfiehlt, sollten JWTs nur eingesetzt werden, wenn die Webapplikation komplett stateless ist. Ansonsten sollten klassische Systeme mit Sessions in Betracht gezogen werden.

PROBLEMATIK ABSPEICHERN VON JWT

Neben der Problematik mit dem Invalidieren von JWTs gibt es noch die Problematik mit dem Abspeichern eines JWTs. Im Normalfall werden JWTs über einen *Header* versandt. Dementsprechend muss mit *JavaScript auf das JWT zugegriffen* werden können, um diesen Header erstellen zu können. Dies eröffnet allerdings die Problematik, dass das Token entweder im localStorage oder in einem Cookie ohne httpOnly-Schutz abgespeichert werden muss und dementsprechend bei einer *XSS-Schwachstelle* ausgelesen werden kann.

Um dieses Problem zu lösen, werden von OWASP zwei unterschiedliche Lösungsansätze vorgestellt. Einerseits kann das JWT in einem *gehärteten Cookie* gespeichert und direkt aus dem Cookie verarbeitet werden. Dabei muss das gehärtete Cookie mindestens die folgenden Attribute beinhalten:

```
Secure; HttpOnly;
```

Zusätzlich muss beachtet werden, dass mit dem Verwenden von Cookies nun Gegenmassnahmen gegen Cross Site Request Forgery Angriffe umgesetzt werden müssen. Für moderne Browser sollte das Setzten des Attributs *SameSite=Strict* beim Cookie reichen. Eine weitgehende Beschreibung eines CSRF-Angriffs sowie Gegenmassnahmen werden in unserem Artikel *Cross Site Request Forgery – Ist CSRF tot?* behandelt.

Grundsätzlich sollten die Claims aus einem JWT clientseitig nicht benötigt werden. Falls allerdings dennoch clientseitig auf Informationen aus dem JWT zugegriffen werden soll, kann der zweite Lösungsansatz von OWASP umgesetzt werden. Die Details und Implikationen einer solchen Implementation werden in den Abschnitten *Tokenspeicherung auf der Client-Seite* und *Token Sidejacking* im OWASP JWT Cheat Sheet genauer beschrieben.

FÜR WAS SIND JWTS SINNVOLL?

Wie bereits erwähnt, sind JWTs vor allem dann sinnvoll, wenn eine Applikation vollständig stateless ist. Ansonsten können JWTs als Token-Mechanismus in OAuth 2.0 Flows verwendet werden. Ebenfalls ein sehr guter Einsatzort für JWTs sind Anwendungsfälle, in denen sie als einmaliges Autorisierungs-Token verwendet werden können. Als Beispiel für einen solchen Anwendungsfall kann eine Applikation genommen werden, die es dem Benutzer unter anderem erlaubt Dateien herunterzuladen. Dabei handelt es sich beim Server, der die Dateien zur Verfügung stellt, um einen separaten, stateless Server oder Service. Dann kann von der Hauptapplikation ein Download-Token im JWT-Format ausgestellt werden, das vom Benutzer genutzt werden kann, um beim Download-Server seine Datei herunterzuladen. In diesem Fall haben die JWTs die folgenden Eigenschaften:

- **Token sehr kurz gültig**: Es muss lediglich für wenige Minuten gültig sein, damit der Benutzer den Download starten kann.
- **Token nur ein Mal verwendet**: Für jeden neuen Download wird ein neues Token erstellt. Dementsprechend handelt es sich beim Download-Server um eine Applikation, die tatsächlich komplett stateless ist.
- **Lediglich Download-Server verwendet JWT für Autorisierung**: Die Applikation selbst ist nicht

stateless und verwendet Sessions. Eine Kombination von Session Tokens und JWTs kann also je nach Anwendungsfall sehr sinnvoll sein.

FAZIT

JWTs sind nicht grundsätzlich etwas Schlechtes. Sie müssen einfach in den für sie angedachten Situationen eingesetzt werden, um sinnvoll zu sein. In einer Webapplikation, in der die Abmeldung von Benutzern oder das Anpassen von Berechtigungen umgehend geschehen muss, sind JWTs ohne zusätzliche eigene Logik aufzubauen nicht die richtige Lösung. JWTs sollten nur in Bereichen eingesetzt werden, die tatsächlich vollständig stateless sind. Ansonsten sollten bewährte klassische Session Tokens zum Einsatz kommen. Wenn JWTs für die Webapplikation verwendet werden, sollte darauf geachtet werden, wie diese abgespeichert werden, damit es aufgrund der gewählten Speicherungsmethode nicht zu Problemen bei XSS- oder CSRF-Angriffen kommt.

Andrea Hauser

VOR ALLEM AUS ERFAHRUNGEN
KANN MAN VIEL LERNEN

Bildquelle: https://pixabay.com/images/id-5360340/

SCIP MONTHLY SECURITY SUMMARY

IMPRESSUM

ÜBER DEN SMSS

Das *scip Monthly Security Summary* erscheint monatlich und ist kostenlos.

Anmeldung: smss-subscribe@scip.ch
Abmeldung: smss-unsubscribe@scip.ch

Informationen zum Datenschutz.

Verantwortlich für diese Ausgabe:
Marc Ruef

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion des Herausgebers, den Redaktoren und Autoren nicht übernommen werden. Die geltenden gesetzlichen und postalischen Bestimmungen bei Erwerb, Errichtung und Inbetriebnahme von elektronischen Geräten sowie Sende- und Empfangseinrichtungen sind zu beachten.

ÜBER SCIP AG

Wir überzeugen durch unsere Leistungen. Die scip AG wurde im Jahr 2002 gegründet. Innovation, Nachhaltigkeit, Transparenz und Freude am Themengebiet sind unsere treibenden Faktoren. Dank der vollständigen Eigenfinanzierung sehen wir uns in der sehr komfortablen Lage, vollumfänglich herstellerunabhängig und neutral agieren zu können und setzen dies auch gewissenhaft um. Durch die Fokussierung auf den Bereich Information Security und die stetige Weiterbildung vermögen unsere Mitarbeiter mit hochspezialisiertem Expertenwissen aufzuwarten.

Weder Unternehmen noch Redaktion erwähnen Namen von Personen und Firmen sowie Marken von fremden Produkten zu Werbezwecken. Werbung wird explizit als solche gekennzeichnet.

scip AG
Badenerstrasse 623
8048 Zürich
Schweiz

+41 44 404 13 13
www.scip.ch