TH Brandenburg
Studiengang IT Sicherheit
Fachbereich Informatik und Medien
Entwicklung Sicherer Softwaresysteme
Prof. Dr.-Ing. Martin Schafföner

Einsendeaufgabe: Threat Analysis

Sommersemester 2024
Abgabetermin 14. Mai 2024

Mara Schulke
Matrikel-Nr. 20215853

Zusammenfassung

In dieser Einsendeaufgabe wird die Online-Meeting-Plattform Google Meet der Firma Google hinsichtlich ihrer softwareseitigen Risiken und Gefahren analysiert. Es wird die Vorgehensweise der Gefahrenanalyse, die entdeckten Risiken und Gefahren zusammengefasst und eine Priorisierung mit Handlungsempfehlung daraus abgeleitet. Des Weiteren wird auf die Methodik der Gefahrenanalyse eingegangen und von alternativen Vorgehensweisen abgegrenzt.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Projektauswahl: Google Meet

Die Auswahl des zu analysierenden Projektes fiel auf die Online-Meeting-Plattform Google Meet, da diese mehrere Punkte abdeckt:

1. Ihre Funktionsweise und ihr Umfang ist im Rahmen einer Kurzanalyse greifbar
2. Die Anwendungsarchitektur lässt sich aus vergleichbaren Echtzeit-Meeting-Anwendungen ableiten und vereinfachen
3. Sie hat einen weitreichenden Bekanntheitsgrad

Vereinfacht lässt sich Google Meet in einer Client-Server-Architektur darstellen, auch wenn an dieser Stelle angemerkt werden muss, dass durch die Menge an Nutzern, die Vielzahl an Integrationen und die Stabilität der Software davon auszugehen ist, dass es sich nicht um eine naive Architektur der Server bzw. Server-Infrastruktur handelt.

Im Rahmen dieser Einsendeaufgabe wird die Annahme getroffen, dass Google Meet der folgenden Architektur entlang aufgebaut ist:

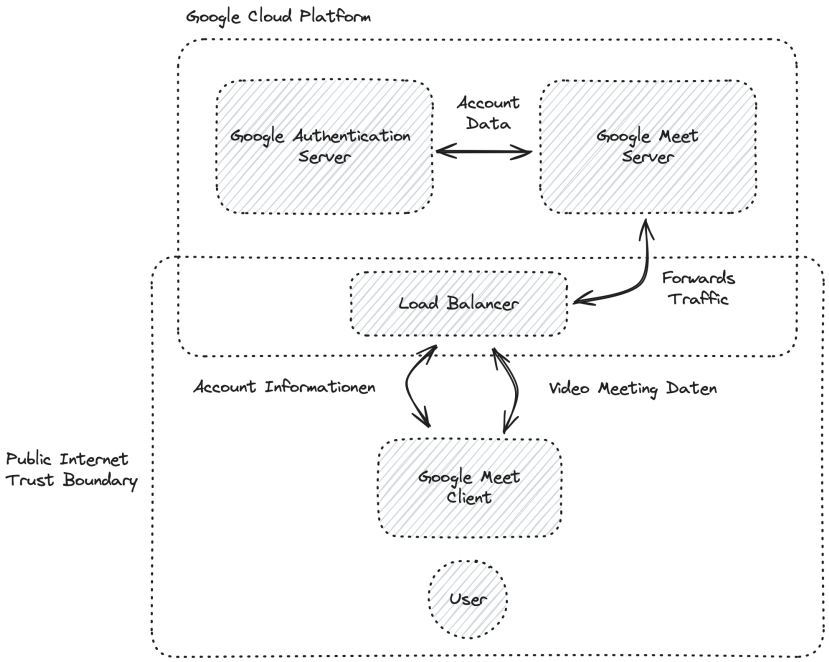

Abbildung 1: Anwendungsarchitektur Google Meet (stark vereinfacht)

Im Rahmen dieser Aufgabe wird von einer monolithischen Anwendungsarchitektur von Google Meet ausgegangen, um in einem zeitlich angemessenen Rahmen in der Lage zu sein, eine hypothetische Gefahrenanalyse durchführen zu können. Der Google Meet Server in der beistehenden Grafik übernimmt die Verwaltung der Sitzungen, die Account-Authentifizierung und die Aushandlung der WebRTC-Sitzungen zwischen verschiedenen Nutzern.

2 Auswahl der Gefahrenanalyse Software

Um eine automatisierte Gefahrenanalyse durchzuführen, stehen mittlerweile (Stand Mai 2024) eine Vielzahl an verschiedenen Tools zur Verfügung. Nachfolgend sind einige beliebte bzw. bekannte, kostenlose *Threat-Modelling*-Programme aufgelistet:

Kostenlose Software

Da diese Einsendeaufgabe von geringem Umfang ist und keine weiteren Anwendungsfälle für eine ausgereifte Threat-Modelling Software absehbar sind, beschränke ich den Vergleich verschiedener Software-Lösungen auf kostenlose bzw. Open-Source-Programme.

MTMT – Microsoft Threat Modeling Tool

MTMT ist das Threat Modeling Tool der Firma Microsoft. Es ist ein weit verbreitetes Werkzeug zur Modellierung von Software und der automatisierten Analyse dieser Modelle. Da der Anbieter Microsoft ist, besteht leider die Anforderung, MTMT auf einem Windows-System auszuführen.

Das MTMT ist weiterhin ein sehr umfangreiches Werkzeug, das ebenfalls im professionellen Kontext Einsatz findet – dies resultiert in einer komplexen Oberfläche mit vielen Möglichkeiten und ausgereiften Konzepten.

Leider musste ich durch die Gegebenheit, kein Windows-System zu betreiben, die Analyse des MTMT hier beenden, da es mir leider nicht möglich war, eine ausführbare Version für Unix-Systeme zu finden, ohne Virtualisierung zu verwenden.

OWASP ThreatDragon

OWASP ThreatDragon ist eine Software der Organisation OWASP (kurz für *Open Worldwide Application Security Project*). Die Software ist kostenlos und browserbasiert. Sie enthält Funktionen für die Diagramm-Zeichnung und anschließende Erstellung eines Sicherheits-Reports. Die Arbeitsstände müssen allerdings immer heruntergeladen und in lokalen Dateien gespeichert werden – dies ist eine relativ umständliche Arbeitsweise, da anders als bei nativer Software nicht eine Datei durchgehend geöffnet und bearbeitet werden kann. Außerdem bietet ThreatDragon nur einen begrenzten Umfang an erkannten Sicherheitslücken bei einer offensichtlich fehlerhaften Softwarearchitektur.

AWS ThreatComposer

AWS ThreatComposer ist eine Open-Source-Software der Firma Amazon. Sie ist ebenfalls browserbasiert, allerdings etwas ausgereifter und nutzerfreundlicher als die OWASP- Alternative. Dennoch gibt es auch hier einige Unannehmlichkeiten für Nutzer: Die Benutzeroberfläche ist unklar (z.B. Diagramme sind nur zur Dokumentation hinzuzufügen, nicht zur automatisierten Auswertung). Es gibt keine klare Datenverwaltung bzw. keinen klaren Datenexport und der allgemeine Funktionsumfang wirkt etwas eingeengt.

Threagile

Threagile ist kostenlose Open-Source-Software zur Gefahrenanalyse ohne eine Benutzeroberfläche. Dies ist eine Eigenheit im Vergleich zu den anderen vorgestellten Programmen, die hauptsächlich über das Desktop- oder Web-Interface zu bedienen sind. Threagile lässt sich über eine .yaml-Datei bedienen, die einem vorgegebenen Schema entsprechen muss. So lassen sich im Quelltext Komponenten (und ihre Beziehungen), Datenbanken, Verschlüsselungen, Datenflüsse, verwendete Technologien und vieles mehr dokumentieren.

Der klare Vorteil von Threagile liegt in der Möglichkeit, diese YAML-Datei bzw. die analysierte Architektur ohne Weiteres in git oder anderen VCS (kurz für *Version Control Systems*) zu versionieren. Bei anderer proprietärer Software ist dies zwar grundsätzlich möglich, aber da VCS in der Regel auf Klartext-Dateien ausgelegt sind, sind die Möglichkeiten begrenzt, parallel im Team an einer Datei zu arbeiten.

Des Weiteren ist es einfach, mittels Threagile eine Gefahrenanalyse auf der YAML-Datei auszuführen und der generierte Report hat eine hohe Qualität (optisch, strukturell und inhaltlich).

2.1 Begründung der Auswahl

Im Rahmen dieser Einsendeaufgabe habe ich mich unter den oben dargestellten Abgrenzungen der verschiedenen Softwarelösungen für die Nutzung von Threagile entschieden. Threagile ist einfach zu bedienen, reproduzierbar (mittels Docker), und eine nützliche Ressource für zukünftige berufliche Situationen.

3 Ergebnis der automatisierten Gefahrenanalyse

Der automatisierte Report von Threagile hat basierend auf der oben dargestellten, stark vereinfachten Anwendungsarchitektur insgesamt 21 mögliche Risiken und 12 verschiedene Risikoklassen identifizieren können.

Folgende Risikoklassen wurden mit der Stufe *Elevated – Wahrscheinlicher Angriff mit hohem Schaden* markiert:

1. Cross-Site-Scripting
2. Missing Cloud Hardening
3. Missing Hardening
4. Server-Side-Request-Forgery

Alle anderen entdeckten Risikoklassen wurden mit der Stufe *Medium – Unwahrscheinlicher Angriff mit hohem Schaden* markiert:

1. Container Base Image Backdooring
2. Cross-Site-Request-Forgery
3. DoS-Risky Access Across Trust Boundary
4. Missing Identity Propagation
5. Missing Identity Provider Isolation
6. Missing Two-Factor Authentication
7. Missing Vault
8. Mixed Targets on Shared-Runtime

Zusammenfassend lässt sich über die entdeckten Risiken sagen, dass diese ein konkretes, greifbares Risiko darstellen und Threagile wenige "False Positives" entdeckt hat (bspw. Fehler, die lediglich aufgrund einer Ungenauigkeit der Diagramm-Funktion markiert werden). Threagile lässt eine sehr detaillierte Dokumentation von Verschlüsselungsmethoden, übertragenen Datensätzen und verwendeten Datenformaten etc. zu. Die Liste der einzelnen Risiken finden Sie im Threagile-Report (Anhang 1).

4 Priorisierung der Gefahren

Die logische Schlussfolgerung der oben eingestuften Risikoklassen ist die Priorisierung der Gefahren nach absolutem Risiko:

$$Risiko = Schaden \times Wahrscheinlichkeit$$

Demzufolge bezieht sich diese Priorisierung lediglich auf die Gefahren aus der Klasse *Elevated*:

1. Cross-Site Scripting (XSS) risk at Google Meet Server (*Likely with High impact*)
2. Missing Cloud Hardening risk at Google Cloud (*Unlikely with Very High impact*)

3. Missing Cloud Hardening risk at Public Internet (*Unlikely with Very High impact*)
4. Cross-Site Scripting (XSS) risk at Auth Server (*Likely with Medium impact*)
5. Missing Hardening risk at Google Meet Server (*Likely with Medium impact*)
6. Server-Side Request Forgery (SSRF) risk at Auth Server server-side web-requesting the target Google Meet Server via Google Meet Server (*Likely with Medium impact*)
7. Server-Side Request Forgery (SSRF) risk at Google Meet Server server-side web-requesting the target Auth Server via User Authentication Access: (*Likely with Medium impact*)
8. Server-Side Request Forgery (SSRF) risk at Google Meet Server server-side web-requesting the target Google Meet Client via Web Interface (*Likely with Medium impact*)

5 Detaillierte Betrachtung

5.1 Cross-Site-Scripting (XSS)

Da Google Meet textbasierte Nutzereingaben als eine der Kernfunktionen (Chat) zulässt, stellt Cross-Site-Scripting ein erhebliches Risiko dar. Allerdings muss an dieser Stelle angemerkt werden, dass die Empfänger eines bösartigen Angriffs lediglich die Nutzer einer Sitzung sind (meist ein- bis zweistellige Nutzerzahlen) – wohingegen auf Plattformen wie Twitter und YouTube etc. Nutzer öffentlich Beiträge posten können, was die Anzahl der betroffenen Nutzer drastisch erhöht.

Davon unberührt ist allerdings die Ausführung von Code bzw. Befehlen auf dem Google Meet Server (auch bekannt als *RCE – Remote Code Execution*) mittels XSS.

5.2 Missing Cloud Hardening

Die fehlende Absicherung der Cloud gegen unbefugten Zugriff (bspw. via SSH, Netzwerkschwachstellen etc.) stellt neben XSS die größte Gefahr dar, da diese mit einer möglichen "Escalation of Privilege" einhergehen könnte und somit ein Angreifer nach der Infiltration einer der cloud-internen Komponenten womöglich weitere Komponenten infizieren könnte. Mögliche Lösungsstrategien zur Absicherung der Cloud bestehen darin, verwaltete Services (für Netzwerkverbindungen, Datenbanken, Load-Balancing, Server- Hosting) der GCP in Anspruch zu nehmen, anstatt mittels VM eine manuelle Infrastruktur einzurichten. Bei Verwendung der verwalteten Lösungen (was bei Googles eigener Software sehr wahrscheinlich ist) besteht keine Notwendigkeit, dass sich das Google Meet Team um Updates der Netzwerkssoftware, Firewall etc. kümmert.

Threagile schlägt zur Unterbindung dieser Gefahr vor, bekannte Absicherungs-Frameworks (unter anderem von OWASP) zu verwenden, um die cloud-interne Architektur zu überprüfen.

6 Zusammenfassung

Insgesamt lassen sich aus dem generierten Threagile-Report mit angemessenem Aufwand wichtige Hinweise zu grundlegenden Problemen der verwendeten Software-Architektur ableiten. Anhand der von Threagile vorgegebenen Risikoklassen und der Risikoformel lässt sich eine grobe Priorisierung für die Schwachstellen ableiten. Für eine detaillierte Priorisierung wäre jedoch tiefgreifendes Wissen über das Geschäftsmodell, die Nutzer, die Umsatzrelevanz einzelner Komponenten, Reparaturaufwand einer Schwachstelle etc. notwendig.