MASTERARBEIT

AUTHENTIFIZIERUNG MIT FIDO2 UND DER WEBAUTHN SPEZIFIKATION

Moderne passwortfreie Authentifizierung im Web

ausgeführt am

Studiengang

Informationstechnologien und Wirtschaftsinformatik

Von: Florian Schmuck

Personenkennzeichen: 1810320013

Graz, am 25. Mai. 2019 .....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

KURZFASSUNG

Da immer mehr Geräte und Benutzer Zugang zum Internet bekommen, muss die Sicherheit von Web Applikationen immer mehr im Blick behalten werden. Besonders bei Authentifizierungsmechanismen steigen die Anforderungen enorm. Passwörter gelten schon länger als unsicher, um ein Benutzerkonto vor unbefugtem Zugriff zu schützen. Die in dieser Arbeit behandelte Spezifikation soll dabei Abhilfe schaffen. Bei dieser Spezifikation handelt es sich um die WebAuthn Spezifikation, welche von der FIDO Alliance entwickelt wurde. WebAuthn wird im Zuge dieser Arbeit auf Schwachstellen von derzeit gängigen Authentifizierungsmechanismen untersucht. Diese bedient sich bewährten kryptografischen Methoden, um Sicherheit für die Authentifizierung zu gewährleiten. Zur Verifizierung eines Authentifizierungsvorganges wird asymmetrische Kryptografie eingesetzt. WebAuthn benötigt Authentifikatoren, um diesen Vorgang durchführen zu können. Diese können FIDO2-zertifizierte Smartphones, Security Token oder biometrische Authentifikatoren sein. Bei der Registrierung einer Benutzerin oder eines Benutzers wird von Authentifikator und Server ein Schlüsselpaar, bestehend aus privatem und öffentlichem Schlüssel, generiert und die öffentlichen Schlüssel zwischen ihnen ausgetauscht. Nun ist es dem Client möglich, sich am Zielsystem mithilfe des registrierten Authentifikators anzumelden. Um Benutzerinnen und Benutzern sowie Entwicklerinnen und Entwicklern zu zeigen, welche Vor- und Nachteile WebAuthn gegenüber herkömmlicher Authentifizierung bietet, wurden bekannte Angriffe recherchiert und auf die WebAuthn Spezifikation ausgeführt. Attacken, welche darauf ausgeführt werden, beinhalten Angriffe, um an Zugangsdaten von Benutzerinnen oder Benutzern zu kommen, Angriffe, welche auf die Übernahme einer Sitzung des Clients abzielen, Angriffe welche nicht technischer Natur sind und Angriffe auf die Authentifikatoren.

ABSTRACT

As numbers of devices and users on the web are growing, security gains more and more weight in the development and usage of a web application. Authentication mechanisms need to be robust against a various amount of attacks but simple enough to enable usage to every user. Passwords are no longer safe to provide security against unauthorized access to a user account. The purpose of this thesis is to provide information about the new WebAuthn specification developed by the FIDO Alliance. This specification is designed to overcome the weaknesses of the common authentication mechanism used currently on the web. WebAuthn uses asymmetric cryptographic methods to provide security for the authentication process. To proof if WebAuthn is more secure than any other authentication mechanism, an experiment is run against a reference application. For this purpose, a collection of attacks from common authentication mechanism was evaluated and tested against the WebAuthn specification. Users and developers of web applications should gain awareness of improving security of accounts through the results of the experiment. Attacks on the specification include stealing user credentials, stealing client sessions, non-technical based attacks and attacks to bypass verification of an authenticator device.

INHALTSVERZEICHNIS

1 EINLEITUNG

Egal ob Forum, Online-Dienste oder soziales Netzwerk, fast jeder Dienst im World Wide Web muss nutzerspezifische Daten speichern, um den gewünschten Nutzen bieten zu können. Darum ist eine sichere Authentifizierung durch verschiedene Authentifizierungs-Mechanismen unerlässlich, um Nutzerdaten zu schützen. Es gibt verschiedenste Arten der Authentifizierung, welche jeweils Vor- und Nachteile haben. Beispielsweise müssen sich Benutzerinnen und Benutzer ihre Zugangsdaten, die oft komplizierten Regeln unterliegen und vielleicht sogar in bestimmten Zeiträumen geändert werden müssen, merken. Die neue sogenannte WebAuthn Spezifikation soll dem ein Ende setzen. Die Spezifikation soll die Authentifizierung für die Benutzerinnen und Benutzer erleichtern und gleichzeitig sicherer als andere Authentifizierungs-Mechanismen sein. Diese Arbeit prüft, ob die Web Authentication Spezifikation wirklich keine Anfälligkeit auf bekannte Schwächen anderer etablierter Authentifizierungs-Mechanismen zeigt.

1.1 Ausgangssituation

Derzeit ist die häufigste Form der Authentifizierung im Web jene, die via Namen und Passworteingabe erfolgt. Diese Art bringt jedoch einen Nachteil mit sich, da die Sicherheit der Authentifizierung von der Stärke des Passwortes abhängt. Folglich benötigt man für jede zusätzliche Webseite oder Webapplikation ein neues Passwort, welches eine angemessene Länge und idealerweise verschiedene Kombinationen aus Sonderzeichen, Zahlen und Klein- und Großbuchstaben aufweist, um die größtmögliche Sicherheit zu gewährleisten. So wird die Benutzerfreundlichkeit beeinträchtigt, da es für Benutzerinnen und Benutzer bei mehreren Diensten schwer wird, sich jedes Passwort zu merken. Das hat zur Folge, dass Nutzerinnen und Nutzer dasselbe Passwort für jeden ihrer Dienste verwenden. Sollte nun eine Hackerin oder ein Hacker das Passwort in die Finger bekommen, stehen ihr oder ihm alle Konten der betroffenen Person offen. Eine andere Möglichkeit ist, Abwandlungen eines Passworts zu verwenden was Angreifern wiederum erleichtert von einem Passwort auf ein anderes zu schließen. Da dieses Thema scheinbar noch immer nicht in das Bewusstsein von Benutzerinnen und Benutzer vorgedrungen ist, soll nun die Web Authentication Spezifikation durch passwortfreie Authentifizierung Abhilfe schaffen.

1.2 Zielsetzung der Arbeit

Ziel dieser Arbeit ist es, die Probleme der bestehenden Authentifizierungs-Mechanismen zu evaluieren und diese dann gegen die Web Authentication Spezifikation zu prüfen. Dazu wird eine Beispielapplikation mit der Web Authentication Spezifikation implementiert, gegen die geprüft werden kann.

1.3 Vorgehen

In diesem Abschnitt wird das Vorgehen für diese Arbeit erläutert.

1.3.1 Begriffserklärung und Recherche

In einer ersten Auseinandersetzung mit dem Begriff Web Authentication Spezifikation soll geklärt werden, was Authentifizierung im Web bedeutet. Des Weiteren sollen verschiedene Formen von Authentifizierung in Web Applikationen herausgearbeitet werden und jene für diese Arbeit relevanten Authentifizierungsmethoden inklusive deren Schwachstellen und anwendbare Attacken identifiziert werden. Diese Vorarbeit soll als Basis für die zu entwickelnde Referenzapplikation und das darauf auszuführende Experiment dienen.

1.3.2 Entwicklung einer Referenzapplikation

Für die Durchführung des Experiments dieser Arbeit soll eine Referenzapplikation der Web Authentication Spezifikation entwickelt werden. Gegen diese, werden die zuvor evaluierten Angriffe geprüft. Dabei soll auch eine technische Dokumentation mitgeführt werden.

1.3.3 Experiment

In diesem Schritt werden nun konkret Attacken auf die Referenzapplikation der Web Authentication Spezifikation angewandt und auf diese Weise geprüft, ob unerlaubter Zugriff möglich ist.

1.3.4 Ergebnisanalyse

Der letzte Schritt der Arbeit umfasst die Analyse der Ergebnisse des Experiments und soll die Forschungsfrage beantworten. Zusätzlich sollen noch Empfehlungen abgegeben werden, die besagen, worauf geachtet werden soll, um die Web Applikation, welche die Web Authentication Spezifikation implementiert, sicherer zu machen und vor unerlaubten Zugriff zu schützen.

2 ALLGEMEIN

In diesem Kapitel werden für die Arbeit relevanten Begriffe erklärt. Des Weiteren soll erläutert werden was Authentifizierung im Web bedeutet und welche Formen der Authentifizierung es im Web gibt.

2.1 Begriffsdefinition

Das Thema Sicherheit im Web beinhaltet viele Begriffe, die oft missverstanden oder falsch verwendet werden. Dieser Abschnitt dient dazu, Klarheit für die relevanten Begriffe dieser Arbeit zu schaffen.

2.1.1 Authentifizierung und Autorisierung

Die Begriffe Authentifizierung und Autorisierung werden oft falsch verwendet. Der folgende Absatz soll Klarheit schaffen.

2.1.1.1. Authentifizierung

Unter Authentifizierung versteht man den Prozess, der die Identität von Benutzerinnen und Benutzern darauf prüft, ob sie die oder derjenige sind, die sie vorgeben zu sein. (Boyd, 2012) Computersysteme, welche die Möglichkeit bieten, verschiedene Sitzungen für Benutzerinnen und Benutzer zu verwalten, müssen eine Art Authentifizierungs-Mechanismus bieten, um den Benutzern ein Login zu ermöglichen. Im gesamten Web, ob Client oder Webportal, spielt Authentifizierung eine wichtige Rolle in Bezug auf die Sicherheit der Applikationen und Systeme. Grundsätzlich soll der Zugriff von nicht autorisierten Personen unterbunden werden, gleichzeitig soll aber auch sichergestellt werden, dass befugten Personen der Zugriff gestattet wird. (Burnett, 2005)

2.1.1.2. Autorisierung

Im Gegenzug zur Authentifizierung bezeichnet man den Prozess der Rechteverifizierung von Benutzerinnen und Benutzern als Autorisierung. Dabei wird geprüft ob eine bestimmte Aktion ausgeführt werden darf, beispielsweise das Schreiben in ein Dokument oder das Anfordern von Daten. (Boyd, 2012)

2.1.2 Authentifizierungsfaktoren

Die vorherrschenden Name- und Passwort-Authentifizierungsmechanismen, welche die derzeit am Häufigsten verwendeten im Web sind, haben ihre Schwachstellen. Ein Passwort ist etwas was BenutzerInnen wissen – ein Faktor. Da Passwörter meist eine schwache Entropie aufweisen und noch schlimmer, mehrfach für verschiedene Seiten und Applikationen verwendet werden, sind sie dadurch leicht zu erraten oder zu stehlen. Eine Lösung dafür ist die Multifaktor-Authentifizierung. (Scambray, Liu, & Sima, 2010) Zur Authentifizierung einer Benutzerin / eines Benutzers gibt es grundsätzlich drei verschiedene Kategorien von Authentifizierungsfaktoren.

- Wissensfaktor (Was man weiß)
- Besitzfaktor (Was man hat)
- Zugehörigkeitsfaktor (Was man ist)

(Dawn M. Turner, 2016)

In der Tabelle 1 werden Authentifizierungsfaktoren und Beispiele zusätzlich angeführt.

Authentifizierungsfaktor	**Beispiele**
Was man weiß	Sicherheitsfrage, Passwort
Was man hat	Smart card, ID Token
Was man ist	Biometrische Merkmale

Tabelle 1 Authentifizierungsfaktoren

Einige Präventivmaßnahmen können getroffen werden, um Attacken auf die Authentifizierung zu erschweren. Es gibt mehrere Sicherheitsschichten, welche genutzt werden können, um Benutzerkonten zu schützen. Diese reichen von einfachen Sicherheitsrichtlinien, über starke Passwortanforderungen bis hin zu Multi-Faktor Authentifizierung. (Diogenes & Ozkaya, 2018)

2.1.3 Authentifizierungstypen

Die folgenden Authentifizierungstypen beschreiben die derzeit am Häufigsten genutzten Authentifizierungstypen im Web. (Dawn M. Turner, 2016)

2.1.3.1. Single-Faktor Authentifizierung

Von den oben erwähnten Faktoren wird hierbei nur einer verwendet. Da die Identität einer Person durch nur einen Faktor überprüft wird, kann erfahrungsgemäß keine ausreichende Sicherheit gewährleistet werden, um vor Missbrauch oder Datendiebstahl zu schützen. (Dawn M. Turner, 2016)

2.1.3.2. Zwei-Faktor Authentifizierung

Hierbei wir die Identität von Personen über zwei, voneinander unabhängigen Komponenten überprüft. Diese beiden Komponenten kommen auch aus zwei verschiedenen Kategorien der Authentifizierungsfaktoren. Als Beispiel könnten sich BenutzerInnen mit ihrer E-Mail-Adresse und ihrem Passwort bei ihrem Bankkonto anmelden, aber für die Durchführung von Transaktionen muss ein zusätzlicher Authentifikationsfaktor eingegeben werden. Da dieser aus einer anderen Kategorie stammen muss, müsste beispielsweise ein One-Time-Passwort (OTP) das entweder auf ein Mobiltelefon gesendet wird oder über ein Authentifizierungsgerät bezogen werden kann, eingegeben werden. OTPs sind generierte Passwörter, die nur einmal verwendbar sind. Allein dadurch wird die Sicherheit um ein Vielfaches gesteigert und bietet Schutz gegen eine Vielzahl von Angriffen. (Dawn M. Turner, 2016)

2.1.3.3. Multi-Faktor Authentifizierung

Multi-Faktor Authentifizierung ist der Zwei-Faktor Authentifizierung grundsätzlich sehr ähnlich. Anstatt nur zwei Authentifizierungsfaktoren zu kombinieren, werden hierfür einfach mehrere genommen, um die Sicherheit zu erhöhen. (Dawn M. Turner, 2016)

2.1.3.4. Starke Authentifizierung

Der Begriff starke Authentifizierung wird oftmals als Synonym für Zwei-Faktor oder auch Multi-Faktor Authentifizierung verwendet. Dabei werden bei starker Authentifizierung hauptsächlich nicht replizierbare Faktoren oder auch digitale Zertifikate eingesetzt, um die Identität einer Person festzustellen und die Sicherheit der Applikation zu erhöhen. Dabei sollte darauf geachtet werden, dass die Faktoren voneinander unabhängig sind, mindestens einer davon nicht replizierbar ist und über das Internet nicht gestohlen werden kann. Nicht-Replizierbarkeit wird oft über die Zeit oder über die Anzahl der Nutzung gesteuert. Das Beispiel dafür ist wieder das OTP, da es nur einmal für eine kurze Zeit genutzt werden kann. (Dawn M. Turner, 2016)

2.2 Authentifizierung im Web

Meistens wird Authentifizierung im Web mit dem Besuch eines großen Unternehmens verglichen. Man muss sich Einschreiben damit das Unternehmen weiß, dass man im Gebäude ist und der Zugang erteilt wird. Wenn man das Gebäude verlässt muss man sich wieder austragen. Das bedeutet, dass man das Recht im Gebäude zu sein nur dann hat, wenn man eingeschrieben ist und es verliert, wenn man sich wieder austrägt. Das Einloggen in einer Web Applikation oder Webseite scheint so ähnlich zu funktionieren. Man hat nur Zugriff auf

bestimmte Inhalte, wenn man sich authentifiziert. Nachdem man sich wieder ausgeloggt hat, verliert man den Zugriff auf die besagten Inhalte. Jedoch trifft dieser Vergleich nicht ganz auf die Authentifizierung im Web zu, da die Authentifizierung über das Hypertext Transfer Protocol (HTTP) vorgenommen wird, welches zustandslos ist. Das bedeutet, dass der Server jede Anfrage isoliert verarbeitet und keine Rücksicht auf vorherige Anfragen nimmt. Folglich können sich Benutzerinnen und Benutzer nicht einmalig einloggen und dann frei Anfragen an den Server stellen. Ein besserer Vergleich wäre eine Zugangskarte die immer Zugang gewährt, wenn man zurückkommt und die man beim Ausloggen wieder abgibt. Mit der Zugangskarte kann man jedoch immer nur das eine gerade Gewünschte erledigen und muss das Gebäude dann wieder verlassen, wenn man wieder etwas erledigen will, muss man erneut die Zugangskarte benutzen, um Zutritt zu erhalten. (Chapman, 2012) In Kapitel 2.3 wird erläutert wie mit dieser Zugangskarte umgegangen werden kann.

2.2.1 Cookies

Um einen Zustand zwischen Client und Server zur transportieren werden häufig sogenannte Cookies im Web verwendet. Diese Daten werden als Message Header der eigentlichen Nachricht mitgeschickt und lassen den, ansonsten zustandslosen Server, den übertragenen Status annehmen. Benutzerinnen und Benutzer können im Web Browser Cookies verbieten, jedoch gibt es mittlerweile so viele Web Applikationen und Webseiten, welche nahezu vollständig von Cookies abhängig sind. All diese würden unbenutzbar sein, wenn man Cookies verbietet. Die Message Header für Cookies sind Set-Cookie und Cookie. Der Server beginnt immer mit dem Austausch eines Cookies und der Client sendet das Cookie im Normalfall mit jeder Folgeanfrage unverändert zurück bis das Cookie abläuft. (Chapman, 2012) Nachfolgender Code zeigt einen Set-Cookie Header

```
Set-Cookie: exp_last_activity=123456; expires=Wed, 01-Feb-2014 19:00:00 GMT;
path=/
```

Hier wird ein Cookie mit dem Namen exp_last_activity erstellt. Jeder Teil des Headers besteht aus einem Schlüssel, dem ein Wert zugewiesen wird. Es gibt einige Schlüsselwörter, die im Cookie-Standard definiert sind und die spezielle Bedeutungen haben. Diese besonderen Schlüsselwörter werden Attribute genannt. Der erste Schlüssel eines Cookies wird als Name für das Cookie verwendet und wird zusammen mit dem Wert im Browser gespeichert. Die übrigen Schlüssel setzen die Werte der Attribute des Cookies. Beispielsweise wird mit dem expires-Attribut der Zeitpunkt festgelegt, wann das Cookie seine Gültigkeit verliert. Falls dieses Attribut ausgelassen wird, verliert das Cookie seine Gültigkeit am Ende der jeweiligen Browsersitzung. Natürlich kann auch die Benutzerin / der Benutzer manuell ein Cookie löschen. Daher kann nie sichergestellt werden, dass ein Cookie wirklich bis zu dem Ende der Gültigkeit besteht. Das letzte Attribut im Beispiel besagt, dass das Cookie für jede Anfrage im spezifizierten Pfad zurückgesendet wird. In diesem Beispiel wird das Cookie für alle Anfragen des Pfades zurückgeschickt, da es den Wurzelknoten des Pfads angibt. (Chapman, 2012)

2.2.2 Token

Es gibt verschiedene Formate von Token bei Authentifizierungs-Mechanismen im Web. Davon sind die namhaftesten SAML und JavaScript Object Notation (JSON) Web Token. Nach einer erfolgreichen Authentifizierungsanfrage erhält man einen solchen Token zurück und kann ihn einfach durch Identity Provider Koordinaten verifizieren. Zusätzlich haben Token Bestandteile, die es ermöglichen, mehr als nur die Identitätsprüfung von Benutzerinnen und Benutzern durchzuführen. (Bertocci, 2016) Ein Autorisierungsserver erstellt das Token und legt es in einer Datenbank ab, auf der Applikationen, welche den Zugriff auf bestimmte Ressourcen validieren müssen, den Abgleich des Tokens durchführen können. Da es nicht immer praktikabel ist, eine Datenbank für verschiedene Ressourcen und Applikationen zu warten können wir zwei andere Wege unter untersuchen: strukturierte Token und Token Introspektion. (Boyd, 2012)

2.2.2.1. Strukturierte Token: JSON Web Token (JWT)

Ein JSON Web Token ist kann eine digitale Signatur oder eine Verschlüsselung aufweisen. Damit wird die Identität der Benutzerin oder des Benutzers vom Identity Provider bei einer Anfrage validiert. (Boyd, 2012) Diese Art von Authentifizierungstoken macht keinen Abgleich in einer Datenbank, sondern beinhaltet bereits alle notwendigen Informationen im Textkörper. Dadurch kann der Autorisierungsserver indirekt mit der geschützten Ressource kommunizieren, ohne dabei auf eine Anfrage über das Netzwerk abhängig zu sein. In das Token werden Informationen wie Zeitstempel des Ablaufs und Benutzerin oder Benutzer, welche sich autorisieren oder authentifizieren wollen eingetragen. Das JWT besteht grundsätzlich aus einem JSON Objekt, das in ein Übertragungsformat gebracht wird. Unsignierte Token sind dabei die einfachste Ausprägung. Ein Beispiel dafür wäre hier:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9
```

Es handelt sich hierbei um ein Format, das auf den ersten Blick wie eine zufällige Zeichenkette aussieht. Unter der Haube spielt sich jedoch mehr ab als man vermuten würde. Man kann sehen, dass sich das JWT in zwei Bereiche teilt. Die Punkte separieren diese. Kein Wert innerhalb dieser Bereiche ist zufällig, sondern in ein Base64-kodiertes JSON-Objekt. Würden wir diese Zeichenkette dekodieren und parsen, erhielten wir folgendes Objekt:

```
{
"typ": "JWT",
"alg": "none"
}
```

Bei diesem Header handelt es sich immer um ein JSON-Objekt. Die bereitgestellten Informationen geben Auskunft über die restlichen Teile des Tokens. Der **typ** Header stellt der Applikation Informationen über den zweiten Bereich des Tokens zur Verfügung, was in diesem Fall aussagt, dass es sich um ein JWT handelt. Unsignierte Token werden mit dem **alg** Header und dem Wert **none** bereitgestellt. Die eigentlichen Nutzdaten stecken im zweiten Bereich des Objekts. Nutzdaten könnten wie folgt aussehen:

```
{
"sub": "1234567890",
"name": "John Doe",
"admin": true
}
```

Zusätzlich zu den Nutzdaten kann ein JWT noch sogenannte Claims beinhalten. Dabei handelt es sich um Zusatzinformationen, welche über verschiedene Applikationen genutzt werden können. Alle Claims sind optional, aber Applikationen und Services können so konfiguriert sein, dass sie einige dieser Informationen benötigen. Für eine eigene Applikation können zusätzliche Felder definiert werden, um besondere Anforderungen abbilden zu können. Die folgende Tabelle zeigt die Standard Claims von JWT. (Richer & Sanso, 2017)

Name	Beschreibung
iss	Ersteller des Tokens. Meistens der Autorisierungsserver
sub	Das Subjekt des Tokens. Dabei handelt es sich meist um einen eindeutigen Schlüssel zum Identifizieren der Eigentümerin oder des Eigentümers der Ressource.
aud	Der Wert dieses Schlüssels beinhaltet die Information über den Empfänger des Tokens – die Audience. Damit kann spezifiziert werden, wer das Token annehmen soll.
exp	Zeitstempel, wie lange das Token gültig ist
nbf	Eine Integerzahl welche beschreibt wann das Token gültig ist. In manchen Systemen kann es vorkommen das man das Token schon vor dem Start der Gültigkeit erhält. Die Zahl beschreibt die Sekunden seit der Unix Epoche - 1. Jänner 1970 GMT.
iat	Dieses Feld ist ebenfalls eine Integerzahl und gibt wieder die Sekunden seit der Unix Epoche an. Es beschreibt, wann das Token ausgestellt wurde.
jti	Zur eindeutigen Identifizierung des Tokens wird in diesem Feld ein Wert vom Ersteller eingesetzt. Meist handelt es sich dabei um einen kryptografischen Zufallswert, um Kollisionen zu verhindern. Dieses Feld ist wichtig, um das Erraten des Tokens und Replay Attacken zu verhindern.

Tabelle 2 JWT Claims (Richer & Sanso, 2017)

2.2.3 Fast Identity Online (FIDO)

FIDO ist das Konsortium, welches frei zugängliche, passwortfreie und Phishing-sichere Authentifizierungsstandards entwickelt. Von FIDO gibt es bereits eine ganze Protokollfamilie, die von der FIDO Alliance entwickelt wurde. Gegründet wurde die FIDO Alliance 2013 und umfasst mittlerweile schon mehr als 250 Mitglieder über die ganze Welt verteilt. Es wurden bereits die folgenden drei Protokolle von FIDO entwickelt:

- Universal Authentication Framework (UAF)
- Universal Second Factor (U2F)

- FIDO2

(Ackermann Yuriy, 2019)

2.2.4 FIDO2

FIDO2 dient als Projektbezeichnung für ein neues, moderneres, einfaches, Phishing-sicheres und passwortfreies Authentifizierungsprotokoll. Dieses umfasst derzeit zwei Spezifikationen:

- WebAuthn
- Client to Authenticator Protocols (CTAP)

Dabei dient WebAuthn als Client Applikationsprogrammierschnittstelle (API) und CTAP als API zwischen Computer und Authentifikatoren. (Ackermann Yuriy, 2019)

2.2.5 Client to Authenticator Protocols

CTAP sind eine Sammlung von Protokollen auf niedriger Ebene. Das heißt sie kommunizieren mit Authentifikatoren, beispielsweiße via Bluetooth, Nahfeldkommunikation (NFC) oder auch Universal Serial Bus (USB).

2.2.6 WebAuthn

Als WebAuthn wird die JavaScript API für den Browser bezeichnet. Diese beschreibt die Schnittstelle, um öffentliche Schlüssel (Public Keys) zu erstellen und zu verwalten. Streng genommen ist WebAuthn eine Erweiterung der Credentials Management API, welche kurz gesagt, eine JavaScript Autovervollständigung für Anmeldeinformationen ist. (Ackermann Yuriy, 2019)

2.2.7 Authentifikator

Authentifikatoren können verschiedene Formen annehmen. Einerseits können es Hardwareschlüssel sein, die an ein Endgerät angeschlossen werden und durch Tastendruck die Authentifizierung vornehmen. Andererseits kann es ein mobiles Endgerät oder eine Applikation sein, an die ein One-Time-Passwort gesendet wird, dieses muss schließlich noch beim Authentifizierungsprozess eingegeben werden.

2.2.8 Digitale Signaturen und Public Key Kryptografie

Dieser Abschnitt beschreibt, welche Rolle Digitale Signaturen und Public Key Kryptographie in der Authentifizierung spielen.

2.2.8.1. Digitale Signatur

Digitale Signaturen sind ein fundamentaler Teil der Authentifizierung. Durch eine digitale Signatur wird eine Identität an eine Information gebunden, um sie dadurch unverwechselbar zu machen. Dadurch kann beispielsweise eine Nachricht die digital signiert ist, einer Person eindeutig zugewiesen werden. (Menezes, van Oorschot, & Vanstone, 2001)

Was macht eine Signatur, ob handschriftlich oder digital, aus? Die folgenden Punkte sollen das erläutern:

- Signaturen sind authentisch und zeigen dem Empfänger oder der Empfängerin, dass eine Nachricht oder Ähnliches von der unterzeichnenden Person stammt.
- Signaturen können nicht einfach nachgemacht werden und zeigen so an, dass eine Signatur sicher von der unterzeichnenden Person kommt.
- Da Signaturen Teil eines Dokumentes oder einer Nachricht sind, können sie nicht einfach an eine andere Stelle verschoben und dort genutzt werden.
- Nachdem ein Dokument, eine Nachricht oder Ähnliches unterzeichnet wurde, kann nichts mehr verändert werden. Um eine Änderung einzufügen, muss alles neu erzeugt werden.
- Die signierende Person kann nicht abstreiten, dass sie nicht unterzeichnet hat, da die Signatur nur von dieser Person kommen kann.

Erfahrungen in der realen Welt haben jedoch gezeigt, dass diese Punkte nicht komplett wahr sind. Signaturen wurden schon nachgemacht, verschoben oder Dokumente nach dem Signieren verändert. Da es aber nicht einfach ist, dies durchzuführen, leben wir mit den Problemen. Auf Computern wären handschriftliche Signaturen noch einfacher zu fälschen, da Kopieren, Einfügen und Verschieben von Dateien und Bildausschnitten sehr einfach ist. Daher gibt es einige Ansätze, um das Signieren auf Computern zu ermöglichen. Eine davon ist das Signieren mit Public Key Kryptografie. (Schneier, 2015)

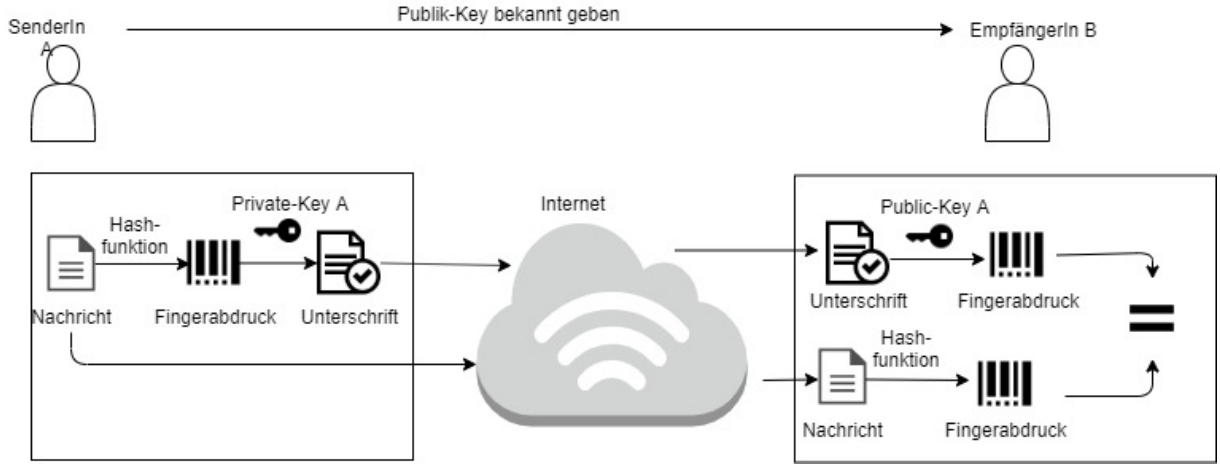

Abbildung 1 Digitale Signatur (Alexander Lauert, 1999)

2.2.8.2. Public Key Kryptografie (Kryptografie mit öffentlichem Schlüssel)

Bei der Public Key Kryptografie oder auch asymmetrische Kryptografie wird ein Schlüsselpaar erzeugt, welches verwendet wird, um eine Nachricht für den sicheren Transport zu verschlüsseln oder entschlüsseln zu können. Beim Start einer Verbindung zwischen den Teilnehmern einer Übertragung erhält jeder der Teilnehmer und Teilnehmerinnen einen öffentlichen und einen privaten Schlüssel von einer Zertifizierungsstelle. Wenn nun ein Teilnehmer oder eine Teilnehmerin eine verschlüsselte Nachricht an einen anderen Teilnehmer oder eine Teilnehmerin senden möchte, kann er oder sie sich den öffentlichen Schlüssel des Gegenübers aus einem öffentlichen Verzeichnis holen und mit diesem Schlüssel die Nachricht verschlüsseln. Hat das Gegenüber nun die verschlüsselte Nachricht erhalten, kann diese mithilfe des privaten Schlüssels entschlüsselt werden. Da niemand anderes auf diesen privaten Schlüssel Zugriff haben sollte, kann die Nachricht nur von dem vorgesehenen Empfänger oder Empfängerin entschlüsselt werden. Diese Vorgangsweise wird durch eine sogenannte Einbahn oder auch Falltür-Funktion ermöglicht. Dabei handelt es sich um eine Funktion, die in eine Richtung einfach ausgeführt werden kann, aber von der anderen Seite nahezu unmöglich berechnet werden kann. Ein Beispiel dafür wäre das Ergebnis von zwei Zahlen zu bestimmen. Ist das Ergebnis und einer der Faktoren bekannt, kann einfach auf den zweiten Faktor geschlossen werden. Ist nun nur das Ergebnis bekannt, ist es, um einiges schwieriger die richtigen Faktoren zu finden. Wenn also der private Schlüssel der Berechnung bekannt ist, wird die schwierige Richtung der Berechnung um einiges einfacher. (Menezes et al., 2001)

Diesen Algorithmus kann man nun zum Verschlüsseln und Entschlüsseln von Nachrichten verwenden. Dabei lässt sich eine Nachricht vom Sender oder der Senderin digital signieren. Das heißt, wenn die Nachricht mit dem privaten Schlüssel des Senders oder der Senderin verschlüsselt wird, lässt sich diese nur mit dem passenden öffentlichen Schlüssel entschlüsseln. Daher kann damit der Sender oder die Senderin authentifiziert werden. (Margaret Rouse, o. J.)

2.3 Authentifizierungs-Mechanismen im Web

In diesem Abschnitt werden verschiedene Authentifizierungs-Mechanismen und ihre Unterschiede beschrieben.

2.3.1 Sitzungs-basierte Authentifizierung

Viele Web Applikationen haben das Problem, dass der Zustand, ob eine Benutzerin oder ein Benutzer am System angemeldet ist, gemerkt werden muss. Da http aber ein zustandsloses Protokoll ist, werden Anfragen zu einem Server isoliert voneinander behandelt. Das bedeutet nach dem Senden der Antwort gehen alle Informationen der Anfrage verloren und eine andere Anfrage derselben Benutzerin oder desselben Benutzers kann der vorherigen Anfrage nicht

zugeordnet werden. Das führt zu Herausforderungen für einige Applikationen. Als Beispiel in einem Webshop kann eine Benutzerin oder ein Benutzer, während sie oder er auf der Seite ist, den Warenkorb mit Artikel befüllen bis es zum Abschluss des Kaufs kommt und alle Artikel auf einmal bezahlt werden. Zwischen dem Hinzufügen des ersten Artikels und dem Kaufabschluss werden viele Anfragen und Antworten zwischen Server und Client ausgetauscht. Daher müssen die Artikel im Warenkorb zwischen den Anfragen gespeichert werden. Über das zustandslose http Protokoll ist das nicht möglich. Somit muss die Sitzung (Session) der Benutzerin oder des Benutzers gespeichert werden. Eine Art Darstellung des Zustands der Interaktion der Benutzerin oder des Benutzers muss zwischen Server und Client bei jedem Request ausgetauscht werden. In einer http-Anfrage gibt es drei Stellen, welche diese Informationen beinhalten können:

- Nachrichtenkopf (Message Header)
- Anfragedaten (Request Data)
- Antwortkörper (Response Body)

Die am Häufigsten verwendete Art der Darstellung, ist die Nutzung von Message Headers, um den Zustand einer Applikation zu transportieren. Dafür werden meistens Cookies eingesetzt. (Chapman, 2012)

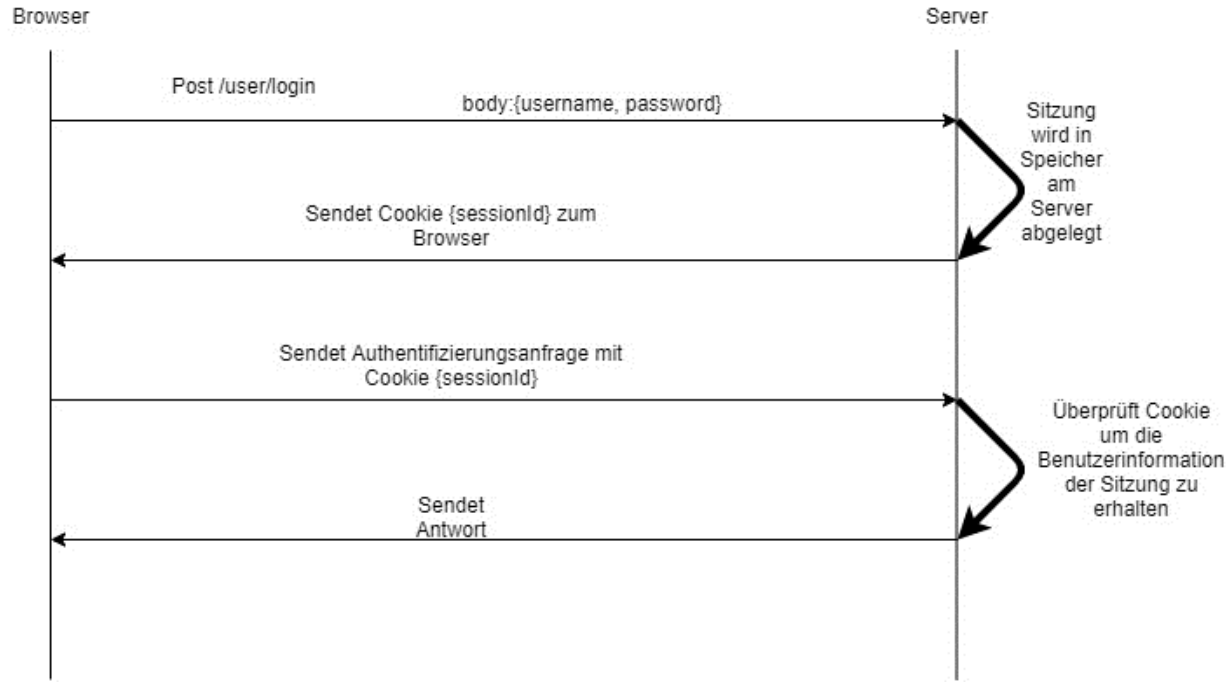

Abbildung 2 Ablauf Sitzungs-basierte Authentifizierung (Sherry Hsu, 2018)

2.3.2 Token-basierte Authentifizierung

Im Gegensatz zur sitzungsbasierten Authentifizierung können Web Applikationen auf einen Token setzen, die bei jedem Request mitgesendet werden, um die Identität der Benutzerin oder des Benutzers zu verifizieren. Meistens werden dafür JSON Web Token eingesetzt. Dieses

Token wird vom Server mit einem Geheimnis (Secret) erstellt und an den Client gesendet. Der Client legt das Token im Speicher ab, meistens ist das der local storage, um es bei jeder Anfrage im Header anzuhängen. Das Token wird dann bei eingelangten Anfragen vom Server evaluiert und sendet bei positiver Evaluierung die Antwort. Meistens wird diese Form der Authentifizierung für moderne Web Applikationen aus Gründen der Skalierbarkeit eingesetzt. Da Token auf dem jeweiligen Endgerät gespeichert werden, wird der Speicher des Servers nicht belastet und kann so besser skalieren. Der große Unterschied zwischen sitzungsbasierter Authentifizierung und tokenbasierter Authentifizierung ist, dass das Zustandsmanagement vom Server zum Client in das Token wechselt. (Sherry Hsu, 2018)

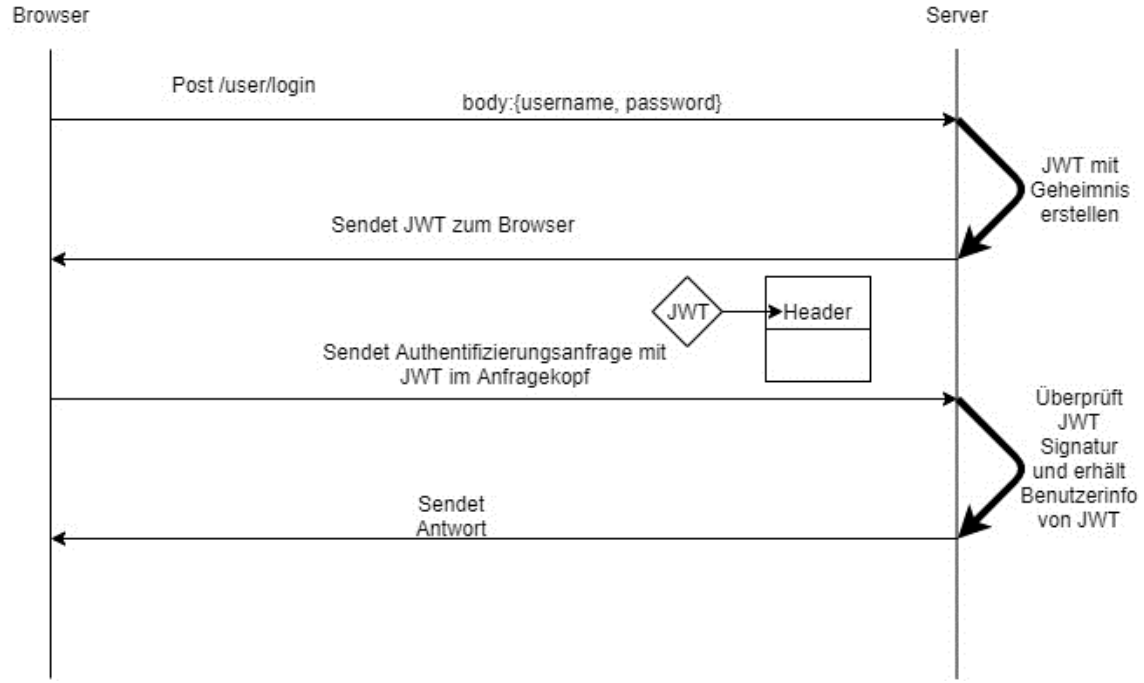

Abbildung 3 Token-basierte Authentifizierung (Sherry Hsu, 2018)

2.3.3 FIDO2-basierte Authentifizierung

Die Authentifizierung mit FIDO2 besteht aus drei Teilen:

- WebAuthn Web Applikation oder Website
- Client
- FIDO2 Authentifikator

Die WebAuthn Web Applikation oder Website ist die Vertrauenspartei (Relying Party) gegen die sich die Benutzerin oder der Benutzer authentifiziert. Der Client – das kann in diesem Fall auch ein Browser sein – übernimmt in diesem Szenario die Rolle des Mittelmanns um den Authentifikator mit dem WebAuthn Server verbindet. Der Authentifikator kann ein USB Token, ein Smartphone oder ähnliches Gerät sein, welches die Spezifikation unterstützt. Grundsätzlich gibt es für FIDO2 zwei verschiedene Abläufe. Dabei handelt es sich um die Registrierung eines

neuen Schlüssels zum Konto der Benutzerin oder des Benutzers und der Authentifizierung gegen den WebAuthn Server. (Jad Karaki, 2019)

2.3.3.1. Registrierung

Bei der Registrierung werden vom Authentifikator neue Anmeldeinformationen erstellt, die zur Signierung einer Forderung (Challenge) der Relying Party genutzt werden kann. Die Anmeldeinformationen bestehen aus einem privaten und einem öffentlichen Teil. Der Public Key, kann zusammen mit der signierten Challenge an die Relying Party gesendet und von dieser abgespeichert werden. Damit kann die Relying Party später die Anmeldeinformationen verifizieren.

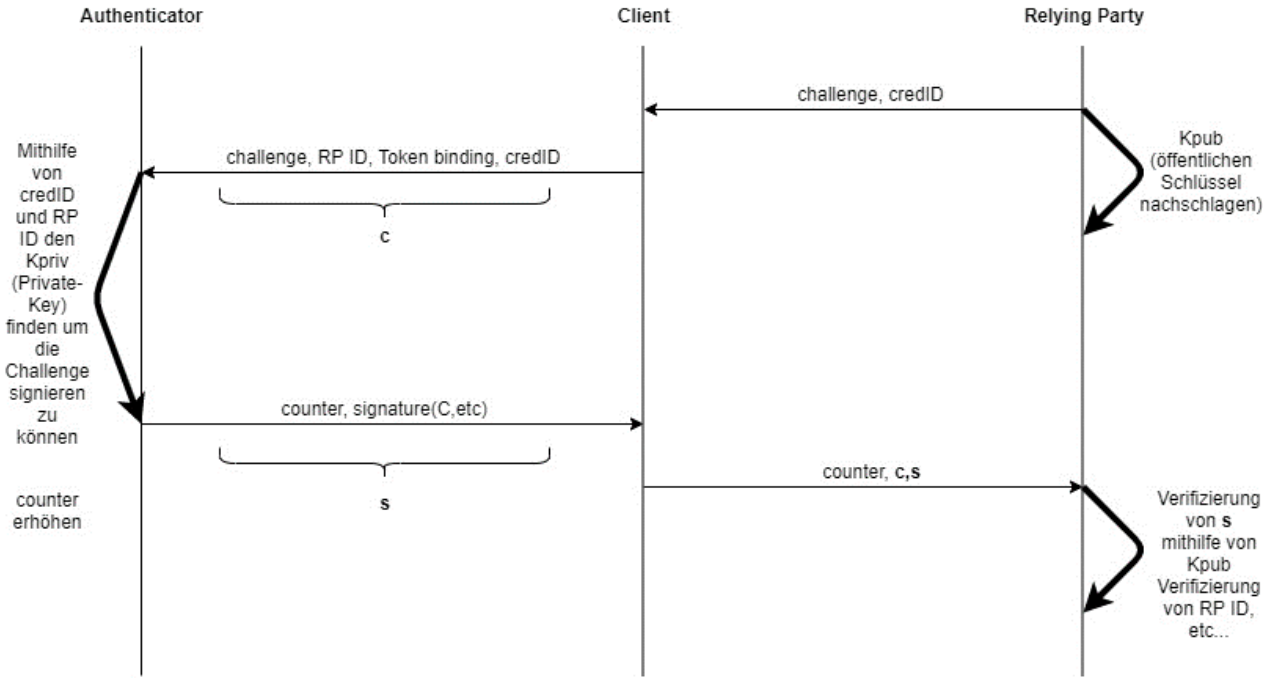

Abbildung 4 FIDO2 Authentifizierung (Ackermann Yuriy, 2019)

2.3.3.2. Authentifizierung

Bei der Passwortfreien Authentifizierung gibt es drei Schritte, die für eine erfolgreiche Anmeldung durchgeführt werden müssen. Zum Start des Authentifizierungsvorganges, gibt eine Benutzerin oder ein Benutzer den Benutzername in der Anmeldemaske ein. Daraufhin wird der Relying Party der Benutzername übermittelt und diese generiert eine Challenge, eine Zufallszahl, welche dann an den Client, der angemeldet werden soll, gesendet wird. Nun wird diese Challenge an den Authentifikator zur Verifizierung weitergegeben. In diesem Schritt muss die Benutzerin oder der Benutzer dem Authentifikator beweisen, dass sie oder er die richtige Person ist. Bei manchen Authentifikatoren reicht ein Tastendruck, bei anderen kann ein biometrisches Merkmal überprüft werden. Nun kann der Authentifikator die Challenge signieren

und über einen sicheren Kanal wieder an die Relying Party zur Vollendung des Authentifizierungsvorgangs übertragen. Die Relying Party muss nun nur mehr mit dem Public-Key des Authentifikators die signierte Challenge überprüfen und kann bei positivem Ergebnis den Client am Zielsystem anmelden. (Ackermann Yuriy, 2019)

3 ATTACKEN AUF AUTHENTIFIZIERUNGSMECHANISMEN UND ZUGANGSKONTROLLE

In diesem Kapitel wird erklärt, was die derzeit größten Schwachstellen von Authentifizierungsmechanismen sind und welche Attacken dadurch ermöglicht werden. Diese Attacken werden im Kapitel „Experiment gegen die WebAuthn Spezifikation“ geprüft, um zu testen, ob diese neuartige Authentifizierungsmethode den Angriffen standhält.

3.1 Problem mit Passwörtern

Als die am Häufigsten verwendeten Zugangsdaten für Authentifizierungs- und Authorisierungsverfahren im Web gelten typischerweise die Kombination aus Benutzername und Passwort. Diese Zugangsdaten kommen auf den meisten Login-Seiten sowie bei HTTP-Basic und HTTP-Digest Authentifizierungsverfahren zum Einsatz. Es kann jedoch problematisch sein, eine Benutzerin oder einen Benutzer nach dem Passwort zu fragen. Die folgenden Punkte sollen die Probleme erläutern (Boyd, 2012):

3.1.1 Vertrauen in die Applikation

Es könnte sein, dass Benutzerinnen oder Benutzer der Applikation nicht vertrauen und deshalb ihr Passwort nicht preisgeben wollen. Leider ist es meistens jedoch umgekehrt. Wenn Benutzerinnen oder Benutzer erst einmal Vertrauen in verschiedene Web Applikationen gewonnen haben und ihre Passwörter für diese preisgeben, hat das auf lange Sicht einen Nachteil. Die Effektivität von Passwort-Diebstahl durch Phishing-Angriffe steigt, da oftmals Benutzerinnen oder Benutzer einfach ihre Zugangsdaten in die Eingabemaske eingeben, ohne den Link zur Login Seite oder die Seite selbst auf Echtheit zu überprüfen. (Boyd, 2012)

3.1.2 Risiko des erweiterten Zugriffs

Hat eine Benutzerin oder ein Benutzer ihr oder sein Passwort eines Benutzerkontos, welches für mehrere verschiedene Applikationen verwendet wird, der Applikation zur Verfügung gestellt, bekommt die Applikation Zugriff zu allen Daten des Benutzerkontos, nicht nur die Daten, welche von der Applikation benötigt werden. Dadurch fällt es nun der Applikation zu, die Passwörter der Benutzer sicher zu verwahren und gegen Attacken gerüstet zu sein. (Boyd, 2012)

3.1.3 Begrenzte Zuverlässigkeit

Bei Benutzerkonten, die über mehrere Applikationen hinweg verwendet werden, verliert die Applikation den Zugriff auf das Benutzerkonto, in dem Moment, in welchem die Benutzerin oder der Benutzer ihr oder sein Passwort ändert. Erst wenn er oder sie sich erneut in der Applikation anmeldet, wird das neue Passwort in die Applikation übernommen. (Boyd, 2012)

3.1.4 Rücknahmeanforderungen

Möchte eine Benutzerin oder ein Benutzer nun den Zugriff einer Applikation auf das Benutzerkonto widerrufen, bleibt ihr oder ihm nichts anderes übrig als das Passwort zu ändern. Dadurch verlieren aber auch alle anderen verbundenen Applikationen ihren Zugriff und er oder sie müssen sich erneut bei den gewünschten Applikationen anmelden. (Boyd, 2012)

3.1.5 Passwortanforderungen

Das wohl größte Problem mit Passwörtern, die im Web Anwendung finden ist, dass jede Benutzerin oder Benutzer viele verschiedene Applikationen nutzt, welche unterschiedliche Nutzerprofile erfordern. Das heißt für jedes Nutzerprofil muss ein eigenes Passwort angelegt werden und dieses dann auch gemerkt werden. Problem dabei ist jedoch, dass man schnell den Überblick verliert. Deshalb tendieren Benutzerinnen und Benutzer dazu, einfach Passwörter zu verwenden und diese über mehrere Applikationen wiederzuverwenden. Hat ein Angreifer durch eine erfolgreiche Attacke Zugriff zu einem Passwort eines Nutzerkontos erlangt, hat er mit großer Wahrscheinlichkeit auch Zugriff zu anderen Nutzerkonten dieser Benutzerin oder dieses Benutzers. Laut Microsoft sollte ein Passwort folgende Kriterien erfüllen, um den Sicherheitsanforderungen gerecht zu werden:

- Änderung des Passworts mindestens alle 60 Tage
- Passwörter von zumindest acht Zeichen Länge
- Einsatz einer Mischung aus Groß- und Kleinbuchstaben für das Passwort
- Kombination aus Alphanumerischen Zeichen und Symbolen
- Einzigartiges Passwort pro Applikation
- Achten auf Speicherung in umkehrbarer Verschlüsselung

Es ist also verständlich, dass Benutzerinnen und Benutzer einfache Passwörter wählen und diese auch wiederverwenden, da es schwierig ist sich viele - idealerweise komplett unterschiedliche - Passwörter zu merken. Allein diese Anforderungen würden bedeuten, dass es eine immens große Anzahl an verschiedenen Passwort-Kombinationen pro Nutzer gäben müsste, welche jeweils nach 60 Tagen erneut geändert werden müsste. (Peter Ray Allison, 2016)

3.2 Attacken

In diesem Kapitel werden die häufigsten Attacken auf Authentifizierung und Zugriffskontrolle erklärt. Das Open Web Application Security Project (OWASP) gibt eine Top-10-Liste die Sicherheitsrisiken für Applikationen heraus. Diese Liste beinhaltet folgende Attacken, vor denen gewarnt wird:

- Broken Authentication
- Broken Access Control
- Sensitive Data Exposure
- Security Misconfiguration

3.2.1 Broken Authentication

Durch unterschiedliches Design und jeweils spezifische Implementierung von Authentifizierungs- und Zugriffskontrollen ist die am weittesten verbreitete Methode die Umgehung der Benutzerauthentifizierung. Das Ziel von Attacken auf zustandsabhängigen Applikationen ist oft das Sitzungsmanagement, da die Benutzerauthentifizierung und Zugangskontrolle stark vom Sitzungsmanagement abhängig sind. Automatische Werkzeuge und manuelles Ausnutzen von Schwachstellen erlauben es Angreiferinnen und Angreifern die Benutzerauthentifizierung und Zugangskontrolle mit beispielsweise Wörterbuchattacken oder Passwortlisten anzugreifen. Wenn Angreifer auf einige Benutzerkonten oder ein Konto einer Administratorin oder Administrators zugreifen und dadurch das System kompromittieren, kann es beispielsweise für Geldwäsche, Sozialversicherungsbetrug oder Identitätsdiebstahl missbraucht werden. Die häufigsten Attacken zur Umgehung der Benutzerauthentifizierung und Zugangskontrolle sind folgende:

- Credential Stuffing durch Listen von validen Passwörtern und Benutzernamen
- Automatische Attacken durch Brute-Force oder andere automatische Angriffe
- Ausnutzen von einfachen Passwörtern, die durch das System zugelassen werden (admin, Passwort1)
- Schwache oder ineffektive Passwortrücksetzung wie beispielsweise Sicherheitsfragen
- Passwörter werden als Reintext oder mit schwachem Verschlüsselungs- oder Hashingverfahren gespeichert
- Fehlende oder ineffektive Multi-Faktor Authentifizierung
- Sitzungs-ID wird in der URL preisgegeben
- Sitzungs-ID wird nach erfolgreichem Login nicht gewechselt

- Sitzungen werden beim Logout Vorgang nicht richtig invalidiert, sodass die Sitzung wieder aufgegriffen werden kann (OWASP, 2017)

3.2.2 Broken Access Control

Die Zugriffskontrolle beschränkt die Benutzerin oder den Benutzer, sodass er oder sie nicht außerhalb ihrer oder seiner Erlaubnis fungiert. Tritt dabei ein Fehler auf, führt das meistens zu unerlaubtem Zugriff auf Daten. Bestenfallswerden Daten „nur“ modifiziert, schlimmstenfalls sogar die komplette Sammlung an Daten vernichtet. Meist jedoch werden dabei Geschäftstätigkeiten durchgeführt, die sonst nicht möglich wären. Folgende Liste soll Schwachstellen von Zugriffskontrollen aufzeigen:

- Durch Modifikation des Uniform Resource Locators (URL), des Zustandes der Applikation oder HTML-Seite oder durch Nutzung eines Application Programming Interface Werkzeugs zum Angriff kann die Zugangskontrolle umgangen werden.
- Der Primärschlüssel zur Identifizierung wird auf den einer anderen Benutzerin oder eines anderen Benutzers verändert, um ihr oder sein Benutzerkonto zu modifizieren oder persönliche Informationen anzusehen.
- Angreiferinnen und Angreifer können durch Attacken ihre Rechte erhöhen, um dann als Benutzerin oder Benutzer zu fungieren, ohne authentifiziert zu sein oder die Administrator Rolle einnehmen, obwohl sie nur ein normales Konto besitzen.
- Über Manipulation von Metadaten wie das Wiederholen oder Fälschen eines JWT Zugriffstoken, Cookies oder verstecktem Feld, können die Rechte der Angreiferin oder des Angreifers auch erhöht werden.
- Falsche Konfiguration von Cross-Origin Resource Sharing (CORS) erlaubt Zugriff auf API, welche nicht erreichbar sein sollte.
- Seiten oder Dienste nutzen, welche Authentifizierung erfordern würden, aber dennoch erreichbar sind.

(Abhay Bhargav, 2019)

Schwachstellen in der Zugriffkontrolle treten oft auf, da es selten eine automatische Erkennung dafür gibt. Applikationsentwickler müssen auch effektive funktionale Tests auf die Zugriffskontrolle machen, um solche Schwachstellen zu finden. Das automatische Testen der Zugriffskontrolle ist nicht ganz einfach. Der beste Weg, um Schwachstellen der Zugriffskontrolle zu finden ist der manuelle Test. Dabei sollten HTTP-Methoden, Controller und Objektreferenzen der Applikation mitgetestet werden. (OWASP, 2017) Um die Zugriffskontrolle sicher zu gestalten, muss sie gezwungener Maßen auf der Serverseite der Applikation oder einer Serverlosen Schnittstelle durchgeführt werden. Dadurch kann eine Angreiferin oder ein Angreifer die Überprüfung der Zugangskontrolle oder die Metadaten nicht modifizieren. Einige Regeln für das sichere Gestalten der Zugangskontrolle sind:

- Von Vorneherein den Zugriff auf Funktionalität nicht automatisch zulassen

- Listen zur Zugangskontrolle führen und auf Rollen basierte Authentifizierungsmechanismen etablieren
- Funktionalität nicht verstecken, sondern nicht zugänglich machen (Abhay Bhargav, 2019)

3.2.3 Sensitive Data Exposure

Als Offenlegung von sensiblen Daten oder Sensitive Data Exposure bezeichnet man eine Schwachstelle die Angreiferinnen und Angreifer ausnutzen können, um Daten von Applikationen, einem Browser oder Ähnlichem abgreifen können. Meistens wird diese Schwachstelle durch unzureichende Absicherung der Informationen verursacht. So kommen Kriminelle in den Besitz von beispielsweise Kreditkarteninformationen, Gesundheitsdaten oder Passwörtern. Mit diesen Daten in ihrem Besitz ist es möglich, unerlaubte Käufe durchzuführen oder sich als eine andere Person auszugeben. Sensitive Data Exposure kann das Ende für eine Webseite oder der Organisation dahinter bedeuten. Es treten dabei Schäden in Form von Aufwänden für die Ursachensuche des Datenlecks auf und zusätzlich muss man mit einem Imageschaden rechnen. (Joyce Tammany, 2018)

Szenarios für Attacken sind beispielsweise dann vorhanden, wenn eine Web-Applikation oder Webseite keine Transport Layer Security (TLS) verwendet. Eine Angreiferin oder ein Angreifer hören den Datenverkehr im Netzwerk ab und Stufen das Übertragungsprotokoll von Hypertext Transfer Protocol Secure (HTTPS) auf HTTP herunter. Damit ist es ihr oder ihm möglich, Anfragen anzufangen und das Session Cookie des Opfers zu stehlen. Mit diesem Cookie kann die Hackerin oder der Hacker nun erneut am Server anfragen und die authentifizierte Sitzung des Opfers stehlen. Damit steht es ihr oder ihm nun offen, die Daten des Opfers einzusehen, zu verändern oder zu löschen. Natürlich könnte die Angreiferin oder der Angreifer auch einfach die Datenpakete, welche über das Netzwerk geschickt werden, einsehen und beispielsweise Empfänger einer Geldtransaktion verändern. (OWASP, 2017)

Für den Schutz gegen Sensitive Data Exposure, muss sich eine Betreiberin oder ein Betreiber einer Web-Applikation erst einmal bewusst machen, welche Daten als sensibel gelten. Wenn nun ein E-Commerce System betroffen ist, wird man sich um Zahlungsinformationen kümmern müssen. In Falle eines Forums sollte wiederum besondere Vorsicht bei Zugangsdaten gelten. Grundsätzlich sollten persönliche Informationen wie Name, E-Mailadresse, Adresse oder Telefonnummer immer als sensible Information angesehen werden. Um die Sicherheit zu stärken, sollte ein Secure Sockets Layer (SSL)-Zertifikat installiert werden. Mithilfe dieses wird eine sichere Verbindung zwischen Server und Client aufgebaut. Fast alle modernen Browser und Suchmaschinen markieren eine Web-Applikation oder Webseite als unsicher, wenn kein SSL-Zertifikat verwendet wird. HTTPS trägt auch zur Sicherheit der Verbindung bei. Wie im Kapitel Ineffektive Speicherung und schwache Verschlüsselung oder Hashingverfahren erwähnt, sollten Passwörter oder Daten nie als Klartext übertragen werden. Daten sollten immer durch einen starken Verschlüsselungsalgorithmus verschlüsselt werden. In Bezug auf das Thema Backup sollten Betreiberinnen und Betreiber bedenken, dass sie das Backup der Daten

nicht auf dem gleichen Server wie die Originaldaten speichern. Im Falle einer Attacke, kann so das Backup einfach wieder eingespielt werden. Wäre das Backup und die Daten am gleichen Server, müsste man davon ausgehen, dass möglicherweise beides kompromittiert wurde. Zum weiteren Schutz der Daten können Web-Application-Firewalls (WAF) eingesetzt werden, um die Applikation vor automatisierten Attacken zu schützen. Zuletzt sollte sichergestellt werden, dass der Browser keine sensiblen Daten speichern darf. Dazu sollte man den Header untersuchen und prüfen, ob dieser Zugangsdaten oder andere sensible Information enthält. (Joyce Tammany, 2018)

3.2.4 Security Misconfiguration

Der Name Security Misconfiguration verrät schon, dass es sich hierbei um falsche Konfiguration von Sicherheitskritischen Elementen in einer Web-Applikation handelt. Oft erhalten Systeme, Geräte oder Bibliotheken keine Sicherheitsupdates mehr und werden auch nicht durch ein neueres Exemplar, welches diese Sicherheitslücken schließen würde, ersetzt. Diese Schwachstellen werden gern von Angreiferinnen und Angreifern ausgenutzt, um sich unerlaubten Zugriff auf das Zielsystem zu verschaffen. In jeder Ebene einer Applikation und sogar darüber hinaus kann eine Fehlkonfiguration stattfinden. Um diese zu finden, können automatische Werkzeuge eingesetzt werden wie beispielsweise ein Standardkonto, welches nur zu Testzwecken gedacht ist, des Weiteren Dienste, welche nicht benötigt werden oder Optionen, die nicht mehr verwendet werden. Solche Sicherheitslecks geben Hackerinnen und Hackern oft nicht nur unerlaubten Zugriff auf das Zielsystem, sondern können im schlimmsten Fall sogar dazu führen, dass das komplette System übernommen wird. (OWASP, 2017)

Wie bereits angesprochen, kann die Fehlkonfiguration der Sicherheit für eine Web Applikation in verschiedenen Schichten auftreten. Daher kann es verschiedene Fehlerquellen geben, die für eine Schwachstellen sorgen. Bedient sich eine Entwicklerin oder ein Entwickler sicherer Codepraxis, hängt es immer noch von der Integration in der Praxis ab, ob dieser Code auch sicher in den Produktionsbetrieb geht. Verantwortliche wie Systembetreiberinnen und Systembetreiber sowie Geschäftsführung sollten Regeln definieren um Systeme, Applikationen und Geräte auf dem neuesten Sicherheitsstandard zu halten. Die folgenden Punkte zeigen die häufigsten Sicherheitsfehlkonfigurationen:

- Systeme, Applikationen und Geräte, die nicht auf dem neuesten Stand sind
- Standardzugangsdaten wie test/test oder admin/admin
- Ungeschützter Zugriff auf Ordner, Daten oder das komplette Dateisystem
- Links zu Seiten, die unbenutzt sind
- Fehlkonfiguration von Netzwerkgeräten

Fehlkonfigurationen können durch Mitarbeiterinnen und Mitarbeiter auftreten, die nicht dafür ausgebildet sind oder auch durch Systemadministratorinnen und Systemadministratoren,

welche sich nicht über Updates informieren, da sie nie darauf geschult wurden. Unternehmen können ein großes Risiko eingehen, wenn sie nicht in Sicherheit ihrer Systeme investiert. Wichtig ist es außerdem, nicht blind auf Updates zu vertrauen, sondern diese vorerst auf einem Testsystem zu prüfen. Wurde das Update zum Beispiel von einer nicht vertrauenswürdigen Quelle heruntergeladen, würde der Schaden zumindest nur das Testsystem betreffen und dem Unternehmen viel Geld sparen. Schlecht definierte Sicherheitsregeln und ungeschulte Systemadministratorinnen und Systemadministratoren schaffen eine Arbeitsumgebung in der Standardkonten verwendet werden können, welche Hackerinnen und Hacker gerne als Angriffsziel verwenden. (Tyra Appleby, 2018)

3.2.5 Credential Stuffing

Definition

Unter Credential Stuffing versteht man einen automatisierten Angriff auf die Benutzerauthentifizierung und Zugangskontrolle einer Web Applikation oder Webseite. Mithilfe eines automatisierten Skripts werden Zugangsdaten an die Authentifizierungsschnittstelle geschickt. Dieser Angriff funktioniert aufgrund des Wiederverwendens von Zugangsdaten von Benutzerinnen und Benutzern. Unglücklicherweise sind meistens gleich mehrere Organisationen betroffen, wenn ein Datenleck der Zugangsdaten auftritt. (McDonough, 2019)

Ausführung

Um die Attacke ausführen zu können, beschafft sich eine Angreiferin oder ein Angreifer Zugangsdaten, die durch ein Datenleck öffentlich gemacht wurden – solche sind günstig aus dem Internet oder einer Seite im dem Dark Web zu erwerben. Danach können Credential Stuffing-Werkzeuge mit den Zugangsdaten gefüttert werden, um den Angriff automatisiert ausführen zu können. Sind nun Logins erfolgreich, hat die Angreiferin oder der Angreifer Zugang zu beispielsweise Kreditkarteninformationen oder persönliche Informationen erlangt. (McDonough, 2019)

Gegenmaßnahmen

Für Benutzerinnen und Benutzer können Passwortregeln und Anleitungen eingeführt werden, die helfen sollen, dass sie sich starke Passwörter aussuchen, welche nur für eine Applikation in Verwendung sind. Es können auch Regeln eingeführt werden, wie die periodische Änderung des Passworts vornehmen zu müssen, um eine Weiterverwendung zu vermeiden oder die Einführung von sogenannten Completely Automated Public Turing to tell Computer and Humans Apart (CAPTCHA). Dieser erfordert bei jeder Authentifizierung eine Analyse eines Puzzles, welches visuell oder akustisch auftreten kann. Wurde dieses Ergebnis dieser Analyse korrekt eingegeben, wird die Anmeldung weitergegeben. Ein Monitoring System, um solche Angriffe zu erkennen, kann eingeführt werden. Mit diesen Informationen können bestimmte Benutzerinnen oder Benutzer in bestimmten Sitzungen, mit bestimmten IP-Adressen oder einem bestimmten Standort blockiert werden. Als Betreiberin oder Betreiber einer Web

Applikation können auch die zur Verfügung stehenden Datenlecks im Internet nach Benutzerinnen und Benutzern auf dem System durchsucht werden. Betroffene Benutzerinnen und Benutzer können so aufgefordert werden, ihr Passwort so schnell wie möglich nach den vorherrschenden Sicherheitsregeln zu ändern. Zuletzt könnten auch Authentifizierungsfehlversuche mitdokumentiert werden. Hat nun eine Sitzung, IP-Adresse, Standort oder Gerät auffällig viele Authentifizierungsfehlversuche, können diese wiederum blockiert werden oder im Vorhinein nur eine bestimmte Anzahl an Authentifizierungsfehlversuchen zugelassen werden. (Colin Watson & Tin Zaw, 2018)

3.2.6 Brute-Force

Definition

Wie der Name bereits sagt, wird diese Attacke auf die Benutzerauthentifizierung durch rohe Gewalt ausgeführt. Um an das Passwort einer Benutzerin oder eines Benutzers zu kommen, werden verschiedene Kombinationen ausprobiert – solange bis ein erfolgreicher Login-Vorgang erzielt wird. Die Gefahr bei diesem Angriff geht von den vielen verschiedenen Kombinationen aus: Hat eine Benutzerin oder ein Benutzer ein einfaches, unsicheres Passwort, besteht die Chance, dass tatsächlich die gesuchte Kombination in einer vernünftigen Zeit gefunden wird. Darum wird nahegelegt, sich an IT-Sicherheitsmaßnahmen und Vorgaben zu halten und auf die eigenen Daten aufzupassen. (Shema, 2012)

Ausführung

Durch Automatisierung mithilfe von Programmen oder einfacher Eingabe in Login-Masken werden verschiedene Kombinationen ausprobiert bis ein erfolgreicher Login-Vorgang erzielt wird. (Scambray et al., 2010)

Ein durchschnittlicher Computer kann in einer Sekunde ungefähr acht Millionen Passwörter ausprobieren. Im Gegensatz dazu konnte ein High-end Computer mit 25 Graphical Processing Units (GPUs) 350 Milliarden Versuche ein Passwort zu finden in der Sekunde durchführen. Das bedeutet, dass ein einfaches Passwort in sehr kurzer Zeit gefunden werden kann. (Peter Ray Allison, 2016)

Gegenmaßnahmen

Es gibt 2 Ebenen, auf denen man Maßnahmen gegen eine Brute-Force Attacke setzen kann. Eine davon ist die Ebene der Benutzerinnen und Benutzer. Er oder sie müssen selbst dafür sorgen, dass ihre Passwörter sicher genug sind und nicht durch eine Brute-Force Attacke geknackt werden können. Man sollte seine Passwörter an den derzeitigen Standard anpassen. Sichere Passwörter sollten aus Kombinationen von Sonderzeichen, Groß- und Kleinschreibung sowie Zahlen bestehen, um optimalen Schutz zu gewährleisten. Die zweite Ebene, auf der Maßnahmen gesetzt werden können, ist die der Betreiber, welche Server und Dienste im Internet betreiben. Technische Maßnahmen können beispielsweise gesetzt werden, um nach einer bestimmten Anzahl an erfolglosen Anmeldeversuchen keine weiteren mehr zuzulassen.

Größtenteils können so automatisierte Angriffe verhindert werden. Die Betreiber können auch die Benutzerinnen und Benutzer auffordern, keine Passwörter aus Passwortlisten zu verwenden. Damit sinkt die Chance einer erfolgreichen Brute-Force Attacke drastisch. Die meisten Skripte und automatisierten Programme dürften damit vor einer kaum zu lösenden Herausforderung stehen. In Zeiten wo Rechenleistung immer günstiger wird sind solche Maßnahmen von höchster Wichtigkeit, um einer Brute-Force Attacke vorzubeugen. (Scambray et al., 2010)

3.2.7 Passwortrücksetzung

Definition

Bei dieser Attacke wird die Passwort-Wiederherstellungsfunktion ausgenutzt, um an ein Benutzerkonto zu kommen. Das Institute of Electrical and Electronic Engineers (IEEE) publizierte einen Artikel über diese Art von man-in-the-middle (MitM) Angriff. Der Artikel beschreibt wie es Forschern des College of Management Academic Studies in Israel herausgefunden haben, dass Kriminelle die Ähnlichkeiten zwischen Registrierungsprozess und Passwortrücksetzung ausnutzen können, um an ein Benutzerkonto zu kommen. (Larry Loeb, 2017) Ein weiteres Problem mit statischen Sicherheitsabfragen ist, dass die Antworten dazu sehr leicht in Profilen auf diversen Social-Media Plattformen des Opfers zu finden sind. Die meisten dieser Sicherheitsabfragen sind wiederholend und auch schwer zu merken. Beispiele dafür sind: „Was war der Name ihres ersten Haustieres?“ oder „Was war ihr Lieblingsbuch als Kind?“. Die Auswirkung von Social Engineering wird in diesem Fall sehr stark unterschätzt, wenn nicht sogar komplett ignoriert. Für Angreiferinnen und Angreifer ist es in diesem Szenario einfacher diese Barrieren auf dem korrekten Weg zu überwinden als sie mit einer Hintertür zu durchbrechen. (LeBlanc & Messerschmidt, 2016)

Ausführung

Zur Durchführung einer solchen Attacke muss die Angreiferin oder der Angreifer eine Web-Applikation oder eine Webseite unter seine Kontrolle bringen. Registriert sich jetzt eine Benutzerin oder ein Benutzer auf dieser Applikation oder Webseite, benutzt die Angreiferin oder der Angreifer die angegebenen Informationen, um eine Passwortrücksetzung auf einer anderen Web-Applikation oder Webseite durchzuführen. Dafür wird das originale Passwort nicht benötigt. Das Zielsystem fordert das Opfer nun auf eine Sicherheitsfrage zu beantworten, um der Angreiferin oder dem Angreifer die Antwort zu liefern. Oftmals geht die Passwortrücksetzung mit einer Zwei-Faktor-Authentifizierung einher, um eine Komplexitätsebene hinzuzufügen. Ist diese Zwei-Faktor-Authentifizierung nun schwach implementiert, kann sie durch einen weiteren Trick der Angreiferin oder des Angreifers umgangen werden. Mehr dazu im Kapitel Attacken auf schwache Multi-Faktor Authentifizierung. (Larry Loeb, 2017)

Gegenmaßnahmen

Zum Schutz gegen einen Angriff auf den Passwortrücksetzungsprozess empfehlen Forscher keine statischen Sicherheitsfragen zu verwenden und Codes zur Passwortrücksetzung nach kurzer Zeit zu invalidieren. Bei Online-Diensten sollte zusätzlich eine Benachrichtigung per E-Mail ausgesendet werden, welche die Benutzerin oder den Benutzer darüber informiert, dass ihr oder sein Passwort zurückgesetzt wird. Des Weiteren sollte nicht nur ein Code eingegeben werden müssen, um ein neues Passwort zu aktivieren, sondern auch ein Link geklickt werden. (Larry Loeb, 2017)

3.2.8 Ineffektive Speicherung und schwache Verschlüsselung oder Hashingverfahren

Definition

Hat man einen kleinen Kreis aus Benutzerinnen und Benutzern, welche im gleichen Netzwerk arbeiten, ist die Versuchung groß, Passwörter unverschlüsselt in der Datenbank abzulegen. Dadurch wäre es möglich bei Verlust oder Vergessen eines Passworts einfach in der Datenbank nachzusehen und der Benutzerin oder dem Benutzer das Passwort mitzuteilen. Diese Praktik sollte jedoch auf keinen Fall implementiert werden. Ein Passwort sollte etwas Persönliches für jede Benutzerin und Benutzer sein und niemand anderes sollte darüber Bescheid wissen. Sieht ein Administrator beispielsweise das Passwort eines Mitarbeiters, könnte er ein Muster erkennen und so auf Zugangsdaten von anderen Benutzerkonten dieser Benutzerin oder dieses Benutzers schließen. (Paul Ducklin, 2013) Passwörter sollten deshalb im schlimmsten Fall zumindest als kryptografischer Hash abgelegt werden, was jedoch Angreiferinnen und Angreifer gut ausnutzen können. Zu diesem Hash kann noch ein sogenannter Salt hinzugefügt werden, um die Entropie zu erhöhen. (Shema, 2012)

Ausführung

Bei Passwörtern, die in Klartext abgelegt worden sind, reicht einfaches Durchprobieren wie bei der Brute-Force Methode oder man wendet eine Wörterbuchattacke an. Im Fall von abgelegten Hashes sind die Passwörter zwar unlesbar, jedoch können sich Hackerinnen und Hacker eine sogenannte Rainbow-Table zunutze machen. Eine Rainbow-Table ist ein vorbereitetes Wörterbuch aus Wörtern mit ihrem dazugehörigen Hashwert und ist eine Technik, um einen Kompromiss zwischen Zeit und Speicherverbrauch zu schlagen. Statt wie bei der Brute-Force Methode durch das Wörterbuch zu iterieren, generiert die Hackerin oder der Hacker Hashwerte für alle Einträge im Wörterbuch. Danach kann der Zielhashwert einfach mit der Liste der vorgefertigten Hashwerte abgeglichen werden. Es ist, um einiges schneller eine große Datenmenge abzugleichen als in jedem Schritt den Hashwert zuerst zu generieren. Der Kompromiss ist hier jener, dass man zuerst Zeit aufwenden, um die Rainbow-Table zu generieren und die Möglichkeit haben diese große Menge an Daten abzulegen. Rainbow-Tables können Monate brauchen, um aufgebaut zu werden und einige Terabytes benötigen, um gespeichert werden zu können, jedoch benötigt man danach nur noch einige Minuten, um einen Eintrag darin finden zu können. Wird dem Passwort nun noch ein Salt hinzugefügt, muss dieser

natürlich auch vorbereitet werden. Ist ein Salt bekannt, reduziert sich die Stärke des Passworts auf die Länge im Brute-Force Fall oder auf Glück im Fall der Rainbow-Table. Mit unbekanntem Salt erhöht sich die Sicherheit um die Länge des Passworts plus der Länge des Salts. (Shema, 2012)

Gegenmaßnahmen

Ein Passwort sollte so wenig Zeit wie möglich im Klartextformat verbringen, was bedeutet, dass es im Authentifizierungsprozess so früh wie möglich verschlüsselt werden sollte. Nach diesem Verschlüsselungsverfahren soll die Klartextform des Passworts nie mehr auf irgendeine Weise aufscheinen. Technisch gesehen werden Passwörter nicht verschlüsselt, sondern mit einer kryptografischen Hashfunktion gehasht. Verschlüsselung würde bedeuten, dass es einen Weg zurück zum Klartext des Passworts gibt. Ein Verhalten, dass man im Falle von Passwörtern nicht haben möchte. Kryptografische Hashverfahren erzeugen immer einen Hash mit fixer Länge, egal welche Länge, der zu hashende Inhalt hatte. Beispiele für solche Hashverfahren sind SHA-1 oder SHA-256. Durch den speziell dafür entworfenen Kompressionsmechanismus, würde beispielsweise aus einem Passwort, welches aus 15 Zeichen besteht, ein 160-Bit langer SHA-1 Hash generiert werden. Analog dazu, ein 256-Bit langer Hash über das SHA-256 Hashverfahren. Sicherheit von Hashverfahren wird durch ihre Resistenz gegen Kollisionen gegeben und dass es nahezu unmöglich ist, den unbekannten Klartext über den bekannten Chiffrenwert zu berechnen. Zusätzlich unwahrscheinlich ist, dass zwei verschiedene Eingaben den gleichen Hashwert erzeugen. Im Fall von Passwörtern wird meist noch ein Salt hinzugefügt, um alternierende Hashes zu erzeugen. Der Salt verhindert, dass Hackerinnen und Hacker sich vorgefertigte Wörterbücher von Hashes aufbauen können. Um weitere Verbesserungen durchzuführen, kann noch Password-Based Key Derivation Function 2 (PBKDF2) eingesetzt werden. Diese wird verwendet, um den Hashwert des Passworts zu generieren. Bei dem PBKDF2 Algorithmus handelt es sich um ein Verfahren, welches mehrere Iterationen von Hashdurchläufen auf den Klartext anwendet. Der dafür verwendete Hashalgorithmus kann frei gewählt werden. Primärziel dieses Verfahrens ist es, die Brute-Force Methode noch schwieriger zu machen. Ein Beispiel für den Einsatz dieses Verfahrens ist die WPA2 Authentifizierung, welche in 802.11x Netzwerken eingesetzt wird. Dabei werden 4096 Runden mit dem PBKDF2 Verfahren durchgeführt, um das Passwort zu hashen. (Shema, 2012)

3.2.9 Attacken auf schwache Multi-Faktor Authentifizierung

Definition

Multi-Faktor Authentifizierung dient dazu, dem Authentifizierungsverfahren mehr Komplexität beizugeben. Ein Passwort ist ein Authentifizierungsfaktor, welchen man kennt. Fügt man nun eine Smart-Card oder ein kryptografisches Gerät hinzu, wird aus dem Authentifizierungsprozess eine Zwei-Faktor Authentifizierung. Nun können beispielsweise noch Fingerabdrücke oder Retina Scans eingebaut werden, um letztendlich eine Multi-Faktor Authentifizierung zu machen.

Es ist aber leider so, dass nicht jeder Faktor gleich sicher ist. Man kann auch SMS, E-Mail Nachrichten, Gesichtserkennung oder Stimmenerkennung als zusätzlichen Faktor miteinbeziehen. Wie schon im Kapitel Passwortrücksetzung erwähnt, können manche Authentifizierungsfaktoren von Angreiferinnen und Angreifern mithilfe von Passwortrücksetzungsmechanismen ausgehebelt werden. (Shostack, 2014)

Ausführung

Ausgangspunkt für den Angriff ist wieder ein Authentifizierungsserver, der unter der Kontrolle einer Angreiferin oder eines Angreifers steht. Als zweiten Faktor zur Authentifizierung wird eine SMS mit einem Code an die Benutzerin oder den Benutzer gesendet, der dann in der Eingabemaske eingegeben werden muss. Will sich nun eine Benutzerin oder ein Benutzer am Zielsystem anmelden, wird ihm/ihr eine falsche Anmeldemaske gezeigt, in der er oder sie die Telefonnummer für die SMS, in der der zweite Faktor enthalten ist, angibt. Danach wird eine normale Loginmaske vorgegaukelt, in der sich die Benutzerin oder der Benutzer anmeldet. Die Zugangsdaten werden nun von der Angreiferin oder dem Angreifer zwischengespeichert und an das echte Zielsystem gesendet, um den Loginvorgang zu vollziehen. Das Opfer erhält nun eine SMS mit dem Code für das Zielsystem und gibt diesen in die falsche Eingabemaske ein. Der Angreifer nimmt diesen und meldet sich legitim am Zielsystem an. Die Angreiferin oder der Angreifer leiten den Datenverkehr nun zwischen Zielsystem und Opfer hin und her, kann aber jede Übertragung mitlesen und sogar als Opfer fungieren. (Shostack, 2014) Dasselbe kann über den Passwortrücksetzungsprozess gespielt werden. Dabei wird nur der SMS-Code vom Opfer abgegriffen, um die Passwortrücksetzung am Zielsystem zu vollziehen. Meistens wird die Quelle der Authentifizierungsfaktoren nicht hinterfragt, auch wenn es von einer unbekannten Nummer im Falle der SMS kommt. (Larry Loeb, 2017)

Für biometrische Sensoren besteht die Gefahr, dass sich eine Angreiferin oder ein Angreifer eine Abbildung des benötigten Körperteils macht und diese als Eingabe verwendet, um das Zielsystem zu täuschen. Um Eingaben solcher biometrischen Sensoren validieren zu können, greifen Applikationen auf bestimmte Schablonen zurück. Gibt man dieser Schablone nun vor dieses Körperteil zu sein oder analysiert die Angriffspunkte dieser Schablone, um vorzugeben man hat den richtigen biometrischen Faktor, ist es möglich sich unerlaubt Zugang zu verschaffen. Um an beispielsweise Abbildungen von Fingerabdrücken oder Gesichtern zu kommen reicht es diverse Soziale Netzwerke zu durchsuchen, um einige brauchbare Fotos zu erhalten. Einen echten Fingerabdruck kann man auf Gläsern, Türen oder Ähnlichem finden. Für Authentifizierungsfaktoren, die man besitzt, besteht die Gefahr darin, dass diese verloren gehen oder auf irgendeine Weise unbrauchbar gemacht oder zerstört werden. Die größte Gefahr jedoch ist der Diebstahl beispielsweise einer Zutrittskarte oder eines Hardware Tokens. (Shostack, 2014)

Wie schon in vorherigen Kapiteln erwähnt stellen Authentifizierungsfaktoren, die man kennt, die größte Gefahr dar. Das beste Beispiel dafür ist das Passwort, welches über verschiedenste Wege und Attacken herausgefunden werden kann. (Shostack, 2014)

Gegenmaßnahmen

Als Betreiber einer Web Applikation sollte man sich vor Augen führen, welche Konsequenzen welche Art von Authentifizierungsfaktor haben kann und welche Gefahr davon ausgeht. Im Laufe von 2019 haben bereits einige österreichische Banken ihren Authentifizierungsmechanismus von SMS-Codes auf One-Time-Passwörter, welche aus einer Applikation generiert werden, eingeführt. Schwachstelle dabei kann natürlich sein, dass das Gerät, welche die Applikation installiert hat, in die Hände von Angreifern fällt und kompromittiert wird. Es ist jedoch ein wichtiger Schritt, um Sicherheit im Web voranzutreiben.

Auf der Seite der Benutzerinnen und Benutzer sollten Authentifizierungsfaktoren, welche vom Betreiber gesendet werden, immer kritisch hinterfragt werden. Mit Faktoren, die man besitzt, sollte man umgehen wie mit Kreditkartendaten oder dem Schlüsselbund. Zuletzt kann man sich noch für Passwörter und dergleichen beispielsweise Passwortmanagern, zunutze machen. Damit können starke Passwörter generiert werden, die völlig unterschiedlich sind. Der Vorteil dabei ist, dass sich eine Benutzerin oder ein Benutzer keines dieser Passwörter merken muss, da die meisten Passwortmanager die Zugangsdaten automatisch ausfüllen können. Des Weiteren würde diese automatische Ausfüllfunktion auch auf falschen Links nicht funktionieren, da der Passwortmanager diese nicht erkennt. Zur Nutzung muss sich die Benutzerin oder der Benutzer nur mehr ein Masterpasswort, das in Kombination mit einer Schlüsseldatei oder andere Faktoren verwendet werden kann, um den Passwortmanager zu öffnen.

3.2.10 Login Spoofing

Definition

Spoofing beschreibt den Vorgang, bei dem eine Angreiferin oder ein Angreifer seine wahre Identität verschleiert. Die Fälschung der Identität im Netzwerk wird dabei über eine falsche Quellenadresse des Datenpakets erzielt. Die Spoofing Attacke wird dazu verwendet, um die wahre Quelle der Attacke zu verschleiern oder Zugriffskontrolllisten zu umgehen, welche Zugriffe anhand von Quelladressen regulieren. (Meier, 2003) Diese Technik kann nun auch dazu verwendet werden, um dem Opfer eine gefälschte Eingabemaske auf der Loginseite der Web Applikation zu zeigen. Das Opfer gibt nun seine Zugangsdaten in diese Maske ein und die Zugangsdaten landen direkt bei der Hackerin oder dem Hacker. (Shostack, 2014)

Ausführung

Das Opfer der Attacke besucht eine gefälschte Loginseite eine Web-Applikation oder Webseite. Diese Seite beinhaltet Eingabefelder, welche die Zugangsdaten direkt in die Hände der Angreiferin oder dem Angreifer spielen. Die gefälschte Loginseite sieht so weit wie möglich aus wie die Originalseite, um keinen Verdacht beim Opfer zu erwecken. Dieser Angriff geht meist Hand in Hand mit einer Phishing Attacke. (Shin, 2017)

Gegenmaßnahmen

Bei solchen Attacken kommen besonders Authentifizierungsfaktoren ins Spiel, welche man besitzt. Hat nun eine Hackerin oder ein Hacker zwar die Zugangsdaten aus der gefälschten Loginseite erhalten, braucht er oder sie noch immer den Code des zweiten

Authentifizierungsfaktors. Wenn das Gerät, dass diesen Code erzeugt, nicht gerade gestohlen wurde, muss die Angreiferin oder der Angreifer irgendwie an den erzeugten Wert des Gerätes kommen. (Shema, 2012) Einem Authentifizierungsverfahren, welches erlaubt nach dem Code in einer Eingabemaske einzugeben, könnte mit dieser Attacke wieder umgangen werden. Hier kommt WebAuthn ins Spiel, welches keine Eingabe im herkömmlichen Sinn erlaubt.

3.2.11 Wörterbuchattacke

Definition

Die Wörterbuchattacke oder auch Dictionary Attack genannt, ist eine Angriffsmethode aus der Familie der Angriffe, bei denen Passwörter erraten werden. Wichtig dabei ist die richtige Vorbereitung, um den Angriff effektiv und effizient durchführen zu können. Zum Beispiel wird eine Liste der potentiellen Opfer des Zielsystems angelegt, um keine Zeit damit zu verschwenden Passwörter für Benutzerinnen oder Benutzer zu raten, welche am Zielsystem nicht existieren. Um solche Listen zu befüllen kann zum Beispiel auf Fehlermeldungen des Systems zurückgegriffen werden. Beispiel wären dafür wären:

- Der eingegebene Benutzername existiert nicht
- Das eingegebene Passwort ist falsch
- Die eingegebene Kombination aus Benutzername und Passwort ist falsch

Mithilfe dieser Information lassen sich schon einige Einträge für das Wörterbuch anlegen, um die Attacke leichter durchzuführen. (Scambray et al., 2010)

Ausführung

Meistens werden Dictionary Attacks offline gegen ein Passwort ausgeführt. Natürlich kann die Attacke auch auf ein Onlineziel gestartet werden, wenn sie richtig genutzt wird. Für die Durchführung wird eine Liste von Wörtern aufgebaut, welche dann durchprobiert wird, bis ein valides Passwort gefunden werden konnte oder die Liste zu Ende ist. Oft werden dafür ein richtiges Wörterbuch oder auch Onlinequellen wie https://sourceforge.net/projects/wordlist/ verwendet. Zur Ausführung auf ein Zielsystem sind automatische Werkzeuge verfügbar, welche die Attacke gegen verschiedenste Zielsysteme beherrschen. Diese Werkzeuge haben meist auch eine Funktion die Wörterliste, um einfache Varianten der darin enthaltenen Wörter zu erweitern. (Burnett, 2005)

Gegenmaßnahmen

Um die Web-Applikation sicherer gegen Angriffe solcher Art zu machen, sollten die Passwörter durch Hashingverfahren laufen und mit hinzugefügtem Salt in der Datenbank abgelegt werden. Am besten wäre es den PBKDF2-Algorithmus dafür zu verwenden, um die größtmögliche Sicherheit zu bieten. Damit muss die Dictionary Attacke viel mehr Varianten der Wörter bilden, um auf ein positives Ergebnis zu erzielen. Grundsätzlich ist es nicht unmöglich einen Treffer zu landen, jedoch ist es sehr unwahrscheinlich. Des Weiteren sollten Benutzerinnen und Benutzer wiederum keine Passwörter für zwei oder mehr verschiedene Systeme verwenden. Taucht so

ein Passwort in einer dieser Wörterlisten auf, ist es leicht möglich gleich mehrere Benutzerkonten damit zu knacken. Außerdem sollten keine zu einfachen Passwörter wie 12345678, verwendet werden oder Namen von Tieren oder Personen. Solche Passwörter erleichtern die Attacke um einiges. (Shema, 2012)

3.2.12 Sniffer-Attacke

Definition

Gestiegene Popularität von kabellosem Internetzugang und damit der Trend immer und überall online sein zu müssen, setzen die Vertraulichkeit des Webs enorm unter Druck. Als Sniffing bezeichnet man das Abhören und Überprüfen von Datenverkehr in einem Netzwerk. Der Datenverkehr von Web-Applikationen und Webseiten kann leicht abgehört werden, wenn die Verbindung zwischen Server und Client nicht über einen sicheren Kanal aufgebaut wird. Somit können Angreiferinnen und Angreifer einfach den Datenverkehr in einem öffentlichen Wireless Local Area Network (WLAN) abhören und Informationen wie Kreditkartendaten, E-Mail-Adressen und Passwörter oder andere persönliche Informationen mitlesen, welche unverschlüsselt über das Netz gehen und von der Benutzerin oder dem Benutzer als privat angesehen werden. Besonders WLAN Netzwerke sind anfällig auf solche Angriffe. Da sich die Angreiferin oder der Angreifer nicht mit einem Kabel an einem Hardwaregerät anschließen muss, um in das Netzwerk zu kommen, können solche Attacken gut in Flughäfen, öffentlichen Stadtnetzwerken oder Schulen durchgeführt werden. (Shema, 2012)

Ausführung

Mithilfe eines einfachen Werkzeugs zum Abhören der Netzwerkpakete, kann eine Angreiferin oder ein Angreifer ganz simpel den Datenverkehr in Klartext mitlesen. Sind Netzwerkpakete nur durch ein einfaches Hashingverfahren oder eine schwache Verschlüsselung geschützt, kann eine Hackerin oder ein Hacker versuchen, diese Verschlüsselung zu knacken, welche eigentlich als sicher betrachtet wurde. Um erfolgreich den kompletten Datenverkehr abhören zu können, muss sich die Angreiferin oder der Angreifer in das Netzwerk zwischen den Server und Client Verbindungen setzen. (Meier, 2003)

Gegenmaßnahmen

Zum Schutz vor einer Sniffing-Attacke sollte die Verbindung zum Server über einen sicheren Kanal hergestellt werden. Dafür bieten sich Verschlüsselungen wie SSL oder HTTPS an. (Scambray et al., 2010)

3.2.13 Session Hijacking

Definition

Unter Session Hijacking wird eine Angriffstechnik bezeichnet, bei der die Angreiferin oder der Angreifer das Session Cookie oder den Authentifizierungstoken des Opfers stiehlt, um damit eine authentifizierte Sitzung aufzubauen. Web Applikationen benutzen meist irgendeine Art von

Authentifizierungstoken, um Benutzerinnen und Benutzern nicht jedes Mal um eine Passworteingabe auffordern müssen, wenn sie sich innerhalb von verschiedenen Bereichen, welche Authentifizierung verlangen, bewegen. (Scambray et al., 2010) Meistens verbergen sich Authentifizierungstokens in Cookies oder einfach in URLs und sind ein leichtes Ziel für eine Angreiferin oder einen Angreifer. Hat er oder sie das Authentifizierungstoken abgefangen, kann eine valide Anfrage an die Applikation mit der authentifizierten Sitzung des Opfers getätigt werden. Dabei nimmt der Angreifer oder die Angreiferin für die Web Applikation die Identität des Opfers ein. Folgende Schwachstellen machen eine Web Applikation für diese Attacke anfällig:

- Sitzungsidentifikation wird ungeschützt in der URL übertragen
- Personalisierte Cookies werden mit Authentifizierungsrelevanten Cookies vermischt
- Authentifizierungstoken werden unverschlüsselt übertragen

(Meier, 2003)

Das OAuth2 Authorization Code Verfahren sendet mit bei der Authentifizierung einer Benutzerin oder eines Benutzers einen eindeutigen Code mit dem Authentifizierungstoken. Nehmen wir an, eine Benutzerin oder ein Benutzer besucht eine Web Applikation auf einem öffentlichen Computer. Er oder sie meldet sich am System an, erledigen ihre Arbeit, melden sich wieder ab und verlassen den Computer. Nun hat eine Angreiferin oder ein Angreifer die Chance, die Sitzung über den eindeutigen Code wieder aufzunehmen. Dazu meldet sie oder er sich bei der Web Applikation mit ihren oder seinen eigenen Zugangsdaten an, nehmen aber aus dem Browserverlauf den eindeutigen Authentifizierungscode des Opfers mit und fälschen ihre Authentifizierung damit. Damit ist die Angreiferin oder der Angreifer mit ihrem oder seinem Konto angemeldet, besitzt aber den Ressourcenzugriff des Opfers. (Richer & Sanso, 2017)

Ausführung

Zur Ausführung einer solchen Attacke gibt es verschiedene Möglichkeiten:

- Attacke auf die Sitzungsidentifikation (Session ID)

In älteren Web Applikationen kommt es oft vor, dass Session-IDs zu einfach bestimmt werden. Es kann sein, dass eine neue Session-ID einfach hinauf gezählt wird oder das schwache Algorithmen für Pseudozufallszahlengeneratoren eingesetzt werden und dadurch die Session-ID vorhersagbar beziehungsweise berechenbar machen. Natürlich kann auch eine Brute-Force Attacke gegen die Session-ID gestartet werden bis eine valide ID gefunden wird. Dafür muss eine Vielzahl an Anfragen gestellt werden und je nach Stärke des Algorithmus zur Erstellung von Session-IDs kann dies ein machbares Unterfangen sein. Eine Variante dieser Attacke wird Session Fixation genannt. Dabei wird eine Session-ID vor der Authentifizierung der Benutzerin oder des Benutzers von einer Angreiferin oder einem Angreifer festgelegt. Da die Angreiferin oder der Angreifer im Voraus weiß, welche Session-ID vergeben wird, ist das Opfer sofort der Attacke ausgesetzt. (Scambray et al., 2010)

- Token Replay Attacke

Bei einer zustandslosen Web Applikation kann kein Zusammenhang zweier Anfragen über den Zustand ermittelt werden. Daher muss bei jeder Anfrage ein Authentifizierungstoken mitgeschickt werden, um verifizieren zu können, ob die Benutzerin oder der Benutzer diese Anfrage auch stellen darf. Jedoch gibt es bei verschiedenen Web-Applikationen die Anforderung, die Schritte der Benutzerin oder des Benutzers durch die Applikation mitzuführen. Beispiel dafür wäre ein Webshop. Man meldet sich am Shopsystem an, legt Waren in den Warenkorb, entfernt wieder einige Waren und sendet die Bestellung dann ab. Da es keinen Zustand für den Warenkorb gibt, und dieser zwischen den Anfragen an den Server zum hineinlegen und herausnehmen nicht verloren geht, müssen die Informationen immer wieder zwischen Server und Client übertragen werden. Um die Übertragung einer eindeutigen Benutzersitzung zuordnen zu können, werden meist Sessiontokens verwendet. Diese können sich in Cookies, Tokens, versteckten Feldern in einem HTML-Formular oder Teil des Uniform Resource Identifiers (URI) sein. (Shema, 2012) Eine Angreiferin oder ein Angreifer beschafft sich nun das Authentifizierungstoken des Opfers mithilfe einer Sniffing-Software oder anderen Methoden und benutzt dasselbe Authentifizerungstoken, um selbst die Session des Opfers aufzubauen. (Meier, 2003)

Gegenmaßnahmen

Abhören des Datenverkehrs im Netzwerk ist die einfachste Möglichkeit an Authentifizierungstokens zu kommen. Als Gegenmaßnahme sollte die Verbindung über einen verschlüsselten Kanal laufen. Abhilfe schafft hier SSL oder HTTPS, um den sicheren Kanal aufzubauen. Damit sollte es Angreiferinnen und Angreifern um einiges schwerer fallen, an das Authentifizierungstoken zu kommen und dieses für eine Replay-Attacke zu nutzen. Hat man nun die Anforderung, selbst ein Authentifizierungstoken designen zu müssen, sollte man darauf achten, dass das Token in keiner Weise vorhersagbar oder gar berechenbar ist. Dafür muss ein Pseudozufallsgenerator mit einem starken Algorithmus eingesetzt werden, der dies verhindert und zusätzlich auch die Ausführung einer Brute-Force Attacke verhindert. (Scambray et al., 2010) Des Weiteren sollten personalisierte Cookies und Authentifizierungscookies voneinander getrennt werden. Anfrageparameter sollten niemals die Session-ID beinhalten. Eine Möglichkeit die Replay-Attacke zu verhindern ist, jedes Mal aufs Neue eine Authentifizierungsaufforderung an die Benutzerin oder den Benutzer zu stellen, wenn eine sicherheitskritische Operation ausgeführt werden soll. (Meier, 2003) Das OAuth 2 Verfahren bietet mit dem zusätzlichen Authentifizierungscode auch Abhilfe, um einer solchen Attacke vorzubeugen. Wie zuvor erwähnt, wäre es möglich, über den Browserverlauf an den Benutzerspezifischen Authentifizierungscode zu kommen, auch wenn sich die Benutzerin oder der Benutzer vom System abgemeldet hat. Damit man diesen Authentifizierungscode aus dem Browserverlauf nicht für eine Attacke nutzen kann, ist er immer nur für eine Authentifizierung gültig. (Richer & Sanso, 2017)

3.2.14 Social Engineering Attacken

Definition

Manchmal werden Attacken von Hackerinnen und Hackern nicht auf technischer Ebene durchgeführt. Sie fragen einfach nach dem Passwort. Es ist wohl ein uralter Trick, jedoch kann dieser sehr wirksam sein. (Burnett, 2005) Als Social Engineering Attacken wird eine Sammlung von Attacken bezeichnet, welche sich nicht auf technische Hacks stützen, sondern das Opfer durch soziale Manipulation dazu bringen, seine Informationen preiszugeben. Unter den vielen verschiedenen Arten von Social Engineering Attacken, geht die größte Gefahr von Phishing aus. (Shin, 2017)

Ausführung

Als Angreiferin oder Angreifer gibt vor, der Helpdesk oder Support einer Web-Applikation zu sein. Möglicherweise erhält man als Opfer eine E-Mail oder wird direkt angerufen, um das Passwort für das Zielsystem preiszugeben (Burnett, 2005). Meistens fällt so ein Angriff auch in die Kategorie Phishing, Vishing oder Whaling. Mehr dazu in den genannten Kapiteln. Hackerinnen und Hacker müssen das Vertrauen des Opfers gewinnen, um sie dazu zu bringen, ihre Informationen preiszugeben. Es ist, um einiges leichter, an die Zugangsdaten von Opfern zu kommen, als in ein stark gesichertes System mit verschiedenen Attacken technologischer Art einzudringen. Die Angreiferin oder der Angreifer können sich für einen solchen Angriff gut vorbereiten, anstatt einfach zu versuchen mit Lügen zum Opfer durchzukommen. Das Internet bietet eine Vielzahl an Web-Applikationen und Webseiten, auf denen persönliche Informationen von potenziellen Opfern zu finden sind. Mit diesen Informationen im Gepäck kann der Angriff gestartet werden. (Shin, 2017) Besonders in der heutigen Zeit, wo Information in großem Ausmaß aus dem Web gezogen werden können, können Social Engineering Attacken größtenteils automatisiert durchgeführt werden. (Shostack, 2014)

Gegenmaßnahmen

Das beste Vorgehen, um diese Attacke zu verhindern ist, niemals ein Passwort preiszugeben. Sicherheitsexperten empfehlen als Gegenmaßnahme Mitarbeiterinnen und Mitarbeiter gründlich über die Wichtigkeit und den Wert von Informationen zu schulen. Zusätzlich soll das Bewusstsein über solche Attacken immer wieder ins Gedächtnis gerufen werden. Social Engineering Attacken sind die größte Gefahr für jedes Sicherheitssystem. (Margaret Rouse, 2006)

3.2.15 Shoulder Sniffing

Definition

Shoulder Sniffing oder auch Shoulder Surfing ist eine simple Attacke. Man sieht einem Opfer über die Schulter, um an seine Zugangsdaten zu kommen. Dabei gibt es verschiedene Arten wie dieser Vorgang vonstattengehen kann. (Jared Kee, 2008)

Ausführung

Die Ausführung dieser Attacke kann auf verschiedene Weisen geschehen:

- Angreiferin oder Angreifer ist abwesend und eine Kamera oder Software kümmert sich um die Beschaffung der Zugangsdaten
- Angreiferin oder Angreifer steht in der Nähe des Opfers und sieht ihr oder ihm beim Eintippen der Zugangsdaten zu. Dies kann durch einfaches Zusehen oder mit einem Fernglas passieren.

Es kommt nicht oft vor, jedoch kann es sein, dass man sich auf einem öffentlichen Computer in einem System anmelden muss. Natürlich gilt dasselbe, wenn man mit seinem eigenen Laptop oder Smartphone in der Öffentlichkeit ist. Dabei sollte man sehr auf seine Umgebung achten damit keiner die Zugangsdaten mitlesen kann. Manche Leute kleben ihre Zugangsdaten sogar auf ihr Gerät, was ein großes Sicherheitsrisiko birgt. Wird das Gerät gestohlen, muss sich die kriminelle Person nicht mehr darum kümmern, an die Zugangsdaten zu kommen. Überwachungskameras in der Öffentlichkeit sollten zwar ein Gefühl von Sicherheit vermitteln, man weiß aber nie, wer sich Zugriff auf diese verschaffen kann. (Jared Kee, 2008)

Gegenmaßnahmen

Die beste Gegenmaßnahme für diese Attacke ist es, immer auf seine Umgebung zu achten. Während dem Eingeben von Zugangsdaten immer die Sicht versperren und wirklich niemals Zugangsdaten niederschreiben oder gar auf ein Gerät zu kleben. Shoulder Sniffing hört sich nach einer simplen Attacke an, jedoch müssen viele Faktoren bedacht werden, um sich bewusst zu machen, wie man überwacht werden könnte. (Jared Kee, 2008)

3.2.16 Phishing, Vishing, Spear Phishing

Definition

Phishing Attacken bedienen sich der Methode des Social Engineering sowie technischen Attacken, um Zugangsdaten und Identitäten von Benutzerinnen und Benutzern zu stehlen. Opfer erhalten Täuschungsemails, in denen vorgegeben wird, dass diese von einer echten Quelle stammen. Ziel dabei ist es, die Opfer auf eine gefälschte Webseite zu locken, um dort deren Zugangsdaten für beispielsweise ihr Bankkonto einzugeben, um damit Transaktionen durchführen zu können. Ein technischer Ansatz für Phishing ist es, Schadprogramme und kompromittierten Code auf dem Computer der Opfer einzuschleusen. Diese Überwachen den Computer dann von innen und stehlen so direkt die Zugangsdaten oder lassen das Opfer eine falsche Anmeldeseite besuchen. (APWG, 2019) Es gibt verschiedene Ausprägungen von Phishing:

- Phishing

Dient als Überbegriff für solche Arten von Attacken. Es handelt sich dabei um eine Technik, mit der man auf betrügerische Weise sensible Informationen über eine Firma oder Person

bekommt. Sie gilt als eine der ältesten Tricks von Hackerinnen und Hackern und wird meist per E-Mail ausgeführt. (Diogenes & Ozkaya, 2018)

- Vishing

Als Vishing oder auch Phone Phishing wird eine Spezialform der Phishing Attacke bezeichnet. Dabei werden Telefonanrufe anstatt E-Mails genutzt, um an die Informationen des Opfers zu kommen. Für die Ausführung kommen dabei interaktive Sprachausgaben zum Einsatz, mit denen vorgeben wird, eine Bankangestellte oder Bankangestellter oder Ähnliches zu sein. (Diogenes & Ozkaya, 2018)

- Spear Phishing

Als Spear Phishing bezeichnet man die Phishing Attacke, die auf ein bestimmtes Opfer abzielt. Dafür muss eine Angreiferin oder ein Angreifer gute Vorbereitung leisten, um das richtige Opfer für ihren Angriff zu identifizieren. Ist die Vorbereitung abgeschlossen, erstellt die Angreiferin oder der Angreifer eine E-Mail mit einem Inhalt, welcher für das Opfer von großem Interesse ist. Im Gegensatz zu normalen Phishing Attacken, die eine Erfolgsquote von 3% aufweisen, hat Spear Phishing eine Erfolgsquote von 70%. (Diogenes & Ozkaya, 2018)

Ausführung

Zur Ausführung von Phishing Attacken können Tricks aus dem Social Engineering-Bereich verwendet werden. Zum Beispiel werden E-Mails an potenzielle Opfer ausgesandt. Der Inhalt dieser E-Mail fordert die Opfer dazu auf, bei Gericht zu erscheinen. Zusätzlich enthält die E-Mail noch einen Link, um sich die Einzelheiten der Vorladung anzusehen. Klickt das Opfer nun diesen Link, wird Schadsoftware auf ihrem oder seinem Rechner installiert. Der Inhalt der E-Mail kann auch so vorgetäuscht sein, dass es so aussieht, als käme sie von einer echten Quelle. In der E-Mail wird dann ein Link angegeben, über den sich Kunden bei einer Web Applikation anmelden können. Dieser Link verweist aber auf eine nachgebaute Seite, welche aussieht wie das Original. Meldet sich eine Benutzerin oder ein Benutzer auf dieser gefälschten Seite an, erfasst die Angreiferin oder der Angreifer die eingegeben Daten und hat so die Zugangsdaten erbeutet. Für Vishing Attacken werden oft gratis Telefonnummern bereitgestellt. Ruft eine Benutzerin oder ein Benutzer diese Nummern an, wird er oder sie an eine Sprachausgabe verwiesen, welche die echte bearbeitende Person vortäuscht. Danach wird das Opfer aufgefordert Informationen preiszugeben, um beispielsweise Geld von einem Bankkonto zu stehlen. Für die Spear Phishing-Methode gilt dasselbe, wie für Phishing, nur muss sich die Angreiferin oder der Angreifer speziell auf die Zielperson vorbereiten. Natürlich können diese Attacken auch über Social Media Benachrichtigungen ausgeführt werden. Diese Benachrichtigungen enthalten dann einen Link, um die Opfer zur Preisgabe ihrer Information zu bringen oder Schadsoftware zu installieren. (Diogenes & Ozkaya, 2018)

Gegenmaßnahmen

Da Phishing eine große Gefahr darstellt, gibt es diverse Dienste, die Benutzerinnen und Benutzer helfen sollen, potenzielle Phishing Betrügereien zu identifizieren. Große

Suchmaschinenanbieter zum Beispiel markieren Seiten, die eine Gefahr darstellen. Manche Browser bieten sogar eine eingebaute Funktion dafür an. Für Benutzerinnen und Benutzer kann es helfen, E-Mails in Klartextformat zu lesen, um zu erkennen, dass die Quelle dieser Nachricht vorgetäuscht wurde. (Scambray et al., 2010) Weitere Gegenmaßnahmen wird die WebAuthn Spezifikation aufzeigen.

3.2.17 Authentifikator Attacken

Definition

Gegenstände und Geräte, welche die Multi-Faktor-Authentifizierung ermöglichen, können auch Ziel eines Angriffs werden. Jedoch ist es bei diesen viel schwieriger, ein erfolgreiches Ergebnis zu bekommen, als bei einem Passwort. Immerhin ist die Wahrscheinlichkeit, ein positives Ergebnis zu bekommen, abhängig von der Größe des Schlüssels des Authentifikators. Dieser Schlüssel wird wiederum von einem Algorithmus berechnet. Ist dieser nun schwach, hat man eine bessere Chance den Authentifikator zu knacken. Wenn die Attacke darauf Online ausgeführt wird, kann es sein, dass die Web Applikation Alarm schlägt, wenn es zu viele Fehlversuche gibt. Bei einem Offline-Versuch muss der Basiswert für den Geheimwert, der im Authentifikator gespeichert ist, herausgefunden werden. Für biometrische Authentifikatoren ist der Ansatz dafür etwas schwieriger. (Woodward, Orlans, & Higgins, 2003)

Ausführung

Offline Attacken können von einem System nicht überprüft werden, was eine höhere Wahrscheinlichkeit birgt, dass sie erfolgreich sind. Gestartet wird die Attacke durch das Sammeln von mehreren Authentifikatorenwerten und - falls verfügbar - die dazugehörigen Challenges, um den Basisgeheimwert berechnen zu können. Dazu werden alle möglichen Werte für den Basisgeheimwert gegen die gesammelten Authentifikatorenwerte geprüft. Ergibt nun die Berechnung mit einem Basisgeheimwert einen Authentifikatorwert als Ergebnis, war die Attacke erfolgreich. Diese Attacke macht praktisch nur für Algorithmen Sinn, welche kurze Schlüssellängen haben. Für biometrische Authentifikatoren ist diese Attacke bei weitem schwieriger. In diesem Fall können keine Werte errechnet werden, sondern dem Authentifikator muss vorgegeben werden, das biometrische Muster des Opfers zu besitzen.

Bei manchen biometrischen Sensoren könnte eine sogenannte Team-Attacke möglich sein. Eine Gruppe von Angreiferinnen oder Angreifern, versuchen beispielsweise abwechselnd mit jedem Finger einen Fingerabdrucksensor zu täuschen. Für Sensoren mit Stimmerkennung versucht sich jede und jeder daran, genauso wie das Opfer zu klingen. Bei Gesichtserkennung muss jede Angreiferin und jeder Angreifer versuchen den Authentifikator mit ihrem oder seinem Aussehen zu täuschen. Theoretisch könnte einer Gruppe ein erfolgreicher Versuch gelingen, wenn diese groß genug ist. In der Praxis sollte das Authentifikatorsystem jedoch Alarm schlagen, wenn es zu viele Fehlversuche gibt. (Woodward et al., 2003)
Es gibt auch bekannte Attacken auf biometrische Sensoren, bei denen der Fingerabdruck mithilfe eines Gummibonbons oder nachgebauten Gummifingern kopiert wird und damit die

Benutzerverifizierung durchgeführt wurde. (Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada, & Satoshi Hoshino, 2010)

Gegenmaßnahmen

Da es sich bei dieser Art von Attacken um einfaches Ausprobieren handelt, sollten sich Systeme und Authentifikatoren dagegen wappnen. Für Authentifikatoren ist die false acceptance rate (FAR) wichtig um zu bestimmen ob durch einfaches Ausprobieren der Basisgeheimwert gefunden oder ein biometrischer Sensor getäuscht werden kann. Diese sagt den Prozentsatz der Zeit aus in dem eine große Anzahl von Authentifizierungsversuchen durchgeführt wird und erfolgreich endet. Natürlich können Authentifikatoren gestohlen werden und deshalb sollte man immer besonders auf diese Geräte und Gegenstände aufpassen. Es ist zwar nicht unmöglich biometrische Merkmale zu stehlen oder vorzutäuschen, jedoch kommt es immer auf das Authentifikatorsystem an ob es die Merkmale annimmt. Zuletzt könnte noch ein Versuch gestartet werden den Wert, der über das Netzwerk geht, abzuhören und sich damit anzumelden. Am Ende wird jede Authentifizierungsmethode in Bits über zur Web Applikation übertragen. Erwartet sich die Web Applikation nun einen bestimmten Wert des Authentifikators, ist es einer Angreiferin oder einem Angreifer möglich eine Replay Attacke auszuführen. Siehe Session Hijacking.

Für dieses digitale Spoofing wurde die Challenge-Response Prozedur entworfen. Dabei wird zwischen Server und Client nicht der Basisgeheimwert genutzt, sondern ein neuer, einmaliger Geheimwert aus diesem berechnet und ausgetauscht. Da biometrische Sensoren geschätzte Übereinstimmungen prüfen, wird dafür ein anderer Ansatz gewählt. Um Spoofing des Sensors zu verhindern, werden kryptografische Techniken eingesetzt, um zu verifizieren, dass die biometrische Messung wirklich von einem Authentifikator stammt. Um die Replizierung von Authentifikatoren zu verhindern, gibt es einige Möglichkeiten. Bei nicht-biometrischen Authentifikatoren ist es wichtig zu verhindern, dass der Geheimwert dieses Authentifikators nicht extrahiert werden kann. Token die man käuflich erwerben kann sowie Smart-Cards bieten bereits Möglichkeiten diese Attacke vorzubeugen. Gegenmaßnahmen für die Replizierung von biometrischen Sensoren, sind meist eine Art von Lebendigkeit in die Information zu bringen. Zum Beispiel prüfen manche Sensoren nicht nur die statischen Daten eines Fingerabdrucks oder einer Gesichtserkennung. Sie nehmen auch Daten wie die Temperatur des Fingers, den Puls oder Bewegung des Gesichts oder der Augen in die Messung mit auf, um nicht von einem Bild oder Abdruck getäuscht werden zu können. Bei Stimmerkennung könnte jedes Mal eine andere Phrase abgefragt werden, um sicherzugehen, dass die Stimme nicht einfach aufgenommen werden kann und daraus eine Phrase zur Authentifizierung vorbereitet werden kann. (Woodward et al., 2003)

3.2.18 Cross-Site Request Forgery

Definition

Als Cross-Site Request Forgery (CSRF) wird ein Angriff bezeichnet, bei dem eine Hackerin oder ein Hacker den Browser des Opfers dazu bringt, eine Anfrage zu ihren Gunsten abzusetzen.

(Shostack, 2014) Dabei muss das Opfer nur auf der Web Applikation oder Webseite des Zielsystems angemeldet sein und in einem weiteren Tab desselben Browsers eine andere Seite aufrufen welche mit Hypertext Markup Language (HTML) Elementen einer Angreiferin oder eines Angreifers angereichert ist. Es ist keine Phishing Attacke, kann aber Teil einer solchen sein. (Shema, 2012)

Ausführung

Eine Benutzerin oder Benutzer surfen im Web auf verschiedenen Web Applikation und Webseiten. Er oder sie wird nach Zugangsdaten gefragt und Cookies werden je nach Applikation oder Seite gesetzt. Eine Web Applikation oder Webseite erkennt die Benutzerin oder den Benutzer nur anhand der Internet Protocol (IP) Adresse, mitgesendete Header, Cookies oder Anfragen. Dadurch dass es für die Web Applikation oder Webseite nur über diese Parameter definiert ist, kann eine Angreiferin oder Angreifer eine CSRF Attacke über den Browser des Opfers durchführen und dabei im Sicherheitskontext des Zielsystems bleiben. Dadurch das der Browser des Opfers und die veränderte Seite der Angreiferin oder des Angreifers die Arbeit des Angriffs übernehmen, bekommt das Zielsystem nie Datenverkehr der Angreiferin oder des Angreifers zu Gesicht. (Shema, 2012) Als Beispiel dafür besucht eine Benutzerin oder ein Benutzer eine Web Applikation einer Bank, um eine Transaktion zu tätigen. Die Web Applikation erlaubt es über eine HTTP GET Anfrage Transaktion zwischen Konten zu tätigen. Angenommen die Anfrage sieht folgendermaßen aus:

```
http://xsrf.verwundbarebank.at/transaktion.jsp?
konto=12345&betrag=1000.00&waehrung=EUR
```

Eine Angreiferin oder ein Angreifer erstellen nun eine HTML Seite die Schadcode beinhaltet. Diese fügen zeigen sie dann in einer Seite, welche unter ihrer Kontrolle ist, an. Beispiel für solchen Schadcode wäre:

```
<script type="text/javascript">
var img = document.createElement("image");
img.src = "http://xsrf.verwundbarebank.at/transaktion.jsp?
konto=CSRF_KONTO&betrag=1000.00&waehrung=EUR";
</script>
```

Damit wird die Schwachstelle mit einem einfachen Bildelement, dass nicht einmal sichtbar sein muss, bei Besuch der Seite ausgelöst. Durch das Bildelement wird das Zielkonto auf das der Angreiferin oder des Angreifers gesetzt. Der Client Browser muss nur dazu gebracht werden die HTML Seite mit dem Schadcode zu besuchen, während man auf der Bankenseite angemeldet ist, um die HTTP GET Anfrage ausführen zu können. Für die Bankenapplikation sieht es so aus als habe das Opfer das Ziel ihrer Transaktion willentlich gewählt. (Scambray et al., 2010)

Gegenmaßnahmen

Um sich vor einer CSRF Attacke zu schützen sollte man sich zuerst bewusst machen, dass der Browser bei Anfragen an eine Web Applikation oder Webseite immer alle Cookies, welche mit der Zieldomäne, Port und Pfad in Verbindung stehen, mitsendet. (Scambray et al., 2010) Da es sich bei Anfragen, die durch eine CSRF Attacke durchgeführt werden, um valide Anfragen

handeln, welche sich nur in ihrem Inhalt unterscheiden, ist es schwierig die Attacke über den Datenverkehr zu verhindern. Jedoch kann die Web Applikation gegen solche Anfragen gewappnet werden, indem der Ablauf angepasst wird, um eine CSRF Attacke zu verhindern. Zum Beispiel kann als erste Ebene die Eingabe validiert werden. (Shema, 2012) Die folgenden drei Präventionstechniken kommen häufig zum Einsatz:

- **Double-posted Cookie**

Im Formular, in dem sensible Anfragen durchgeführt werden, wird ein verstecktes Feld eingebettet, welches einen generierten Zufallswert aus dem Clientcookie oder den Wert der Session ID beinhaltet. Wenn das Formular nun an den Server gesendet wird, wird der Wert aus dem HTTP Anfragekopf, mit dem aus dem Cookie verglichen. Passen die Werte nicht zusammen, wir die Anfrage abgewiesen und ein Log Eintrag der Anfrage geschrieben.

- **Unique Form Nonce**

Bei dieser Strategie zur Vorbeugung von CSRF Attacken, erhält jedes Formular ein eigenes verstecktes Feld, das bei jeder Anfrage eine sicheren generierten Zufallswert beinhaltet. Zur Generierung eines solchen Nonce muss ein kryptografisch sicherer Pseudozufallsgenerator eingesetzt werden, um die Sicherheit zu gewährleisten. Erhält der Server nun eine Anfrage wird die Nonce mit dem Wert im Speicher verglichen. Der Server weist Anfragen ab sollte der Wert nicht zusammenpassen oder akzeptiert sie, wenn der Wert ident ist.

- **Require authentication credentials**

In dieser Präventionsstrategie wird die Benutzerin oder der Benutzer jedes Mal zur Eingabe ihres oder seines Passworts aufgefordert, wenn eine sensible Anfrage getätigt werden soll. Die Aufforderung zu Passworteingabe wird auch bei bereits authentifizierter Sitzung gestellt. Meist wird diese Strategie für Web Applikationen und Webseiten eingesetzt, die nur wenige Stellen besitzen, an denen sicherheitskritische Anfragen abgesendet werden. Zum Beispiel das Ändern der Benutzerinformationen. Hierbei sollte jedoch bedacht werden einen Audit- und Aussperrmechanismus zu integrieren, um eine CSRF Brute-Force Attacke vorzubeugen.

Die Unique form nonce Strategie ist, die wohl am häufigsten eingesetzte Strategie, um CSRF Attacken vorzubeugen. (Scambray et al., 2010)

3.3 Conclusio

Angriffe auf Zugangsdaten haben ein sehr ähnliches Schema, jedoch kommt es darauf an wie viel Zeit und Ressourcen in das herausfinden dieser eingesetzt werden soll. Natürlich hängt es auch sehr von der verwendeten Form von Authentifizierungsmechanismus ab, ob ein bestimmter Angriff überhaupt so leicht ausführbar ist. Die OWASP Top 10 sind ein guter Anhaltspunkt, um etwaige Schwachstellen herauszufinden beziehungsweise diese zu überprüfen ob sie in einer vorhandenen Applikation ausgenutzt werden können. Bevor man mit der Implementierung einer Benutzerverwaltung einer Applikation beginnt sollte man die Sicherheitsrisiken anhand der OWASP Top 10 Liste überprüfen und Gegenmaßnahmen planen.

Am Beispiel der Ineffektiven Speicherung von Zugangsdaten kann man sehen was man nur im Fall der Speicherung bedenken muss, um ein robustes System zu erstellen. Sogar bei mehrfachen Authentifizierungskriterien können sich Schwachstellen einschleichen. Die beiden größten Probleme bleiben jedoch noch immer Phishingattacken und Social Engineering Angriffe. Einer Person kann mit einer gut gestalteten Phishingattacke leicht vorgetäuscht werden, dass es sich um die richtige Anmeldeseite handelt oder mit der richtigen Person gesprochen wird, um an Zugangsdaten oder andere sensible Informationen zu kommen. Im Fall von Social Engineering Angriffen gibt es ein breites Spektrum an Methoden. Diese reichen vom einfachen über die Schulter sehen bis hin zum Diebstahl von Authentifikatoren. Meistens gehen Phishing und Social Engineering Angriffe zusammen einher, da sie in Kombination effektiver sind. Im Kapitel 4 Experiment wird nun geprüft ob die WebAuthn Spezifikation Schwachstellen gegen die beschriebenen Angriffe aufweist.

4 EXPERIMENT

In diesem Kapitel wird das Experiment dieser Arbeit beschrieben. Dabei sollen die, aus Kapitel 3 Attacken auf Authentifizierungsmechanismen und Zugangskontrolle, beschriebenen Attacken gegen eine Referenzimplementierung der WebAuthn Spezifikation getestet werden. Die Basis dieser Referenzimplementierung wurde vom GitHub User https://github.com/herrjemand Ackermann Yuriy freundlicherweise zur Verfügung gestellt. Er ist Teil der FIDO Alliance und hat an der WebAuthn Spezifikation mitgewirkt. Das Projekt kann auf GitHub unter dem Link https://github.com/fido-alliance/webauthn-demo gefunden werden. Dieser Abschnitt wird folgendermaßen aufgebaut:

- Erklären des Aufbaus und funktionelle Elemente des Referenzprojekts
- Ausführen der Attacken auf das Referenzprojekt und Beschreibung der Ergebnisse
- Empfehlungen für Gegenmaßnahmen
- Conclusio

Um die Attacken auf das Referenzprojekt ausführen zu können, wird auch eine Infrastruktur benötigt, die im Laufe des Experiments beschrieben wird. Die Beschreibung der Ergebnisse umfasst warum ein Angriff erfolgreich war beziehungsweise warum dieser bei der WebAuthn Spezifikation nicht möglich ist.

4.1 Aufsetzen des Experiments

Die Experimentapplikation, auf das die Attacken ausgeführt werden sollen, ist eine Referenzimplementierung in JavaScript. Im folgenden Abschnitt sollen die wichtigsten Teile dieser Applikation erklärt werden.

4.1.1 Aufbau der Referenzimplementierung

Das Projekt wurde in der Programmiersprache JavaScript entwickelt. Sie umfasst sowohl den Clientteil als auch den Serverteil, um die Authentifizierung mit WebAuthn durchzuführen. Für den Ablauf dieser Applikation gibt es mehrere Wege:

- Registrierung in der Web Applikation
- Anmeldung in der Web Applikation

4.1.1.1. Registrierung in der Web Applikation

Für die Registrierung müssen zuerst Zugangsdaten generiert werden. Dafür wird eine MakeCredentials Abfrage abgesetzt:

```
navigator.credentials.create({ publicKey })
    .then((newCredentialInfo) => {
        /* Public key credential */
    }).catch((error) => {
        /* Error */
    })
```

Für diese Anfrage muss zuerst die Variable **publicKey** definiert werden. Dafür gibt es mehrere Parameter, welche die Erstellung beeinflussen:

```
let publicKey = {
        challenge: challenge,
        rp: rp,
        user: user,
        pubKeyCredParams: pubKeyCredParams,
    }
```

Das **challenge**-Attribute des **publicKey**-Objekts definiert einen 32-Byte langen Buffer, um zufällige Challenges zu erzeugen. Damit sollen Man-in-the-Middle Attacken verhindert werden.

```
let challenge = new Uint8Array(32);
window.crypto.getRandomValues(challenge);
```

Um die Informationen der Relying Party festzulegen, können die Attribute im **rp**-Objekt ausgefüllt werden. Neben dem Namen der Relying Party, kann noch ein Link zum Icon der Relying Party oder deren **id** angeben werden. Der **name** ist das einzige Attribut, das ausgefüllt werden muss.

```
let rp = {
        'name': 'Experiment Relying Party'
    },
```

Das **user**-Objekt enthält alle relevanten Informationen über die Benutzerin oder den Benutzer dieser Zugangsdaten. Die Felder **id**, **name** und **displayName** sind zwingend auszufüllen. Auch das user-Objekt besitzt ein Feld für das Icon, dieses ist jedoch optional. Die user.id ist ein vom Server generierte Kennzeichnung der Benutzerin oder der Benutzer und darf keine Benutzerinformationen enthalten. Bestenfalls sollte diese ID von einem Zufallsgenerator erzeugt werden und hat den Zweck die Zugangsdaten in der Datenbank der Relying Party zu finden. Das **name**-Attribut kann verschiedene Werte entgegennehmen, je nachdem was die Relying Party zur Ablage und primären Kennzeichnung für die Benutzerin oder den Benutzer nutzen

möchte. Zuletzt enthält noch der **displayName** den wirklichen Namen der Benutzerin oder des Benutzers. Zum Beispiel „Florian Schmuck“.

```
let userId = '12345678'

var id = Uint8Array.from(window.atob(userId), c=>c.charCodeAt(0))

    let user = {

        'id': id,

        'name': 'fschmuck@test.at',

        'displayName': 'Florian Schmuck'

    }
```

Das letzte Attribut, welches noch nötig ist, um Zugangsdaten zu erzeugen ist **pubKeyCredParams**. Dieses enthält Signierungsalgorithmen, welche vom Server unterstützt werden. FIDO2 Server müssen zumindest RS1, RS256 und ES256 als Signierungsalgorithmen unterstützen. Der Wert -7 für das Feld **alg** bedeutet das der ES256 Algorithmus eingesetzt wird.

```
let pubKeyCredParams = [

        { 'type': 'public-key', 'alg': -7  }

    ]
```

Diese Basiskonfiguration reicht aus, um Zugangsdaten für die Registrierung mit WebAuthn zu erstellen. Es können noch weitere optionale Attribute angegeben werden, um die Erstellung der Zugangsdaten zu verfeinern:

- Timeout

Damit steuert man wie lange die Registrierung offen bleibt bis eine Zeitüberschreitung einen Fehler wirft.

- excludeCredentials

Dient dazu doppelte Registrierung mit einem Authentifikator zu verhindern. Hier kann eine Liste mit Zugangsdaten angegeben werden, die bereits von einer Benutzerin oder einem Benutzer registriert wurden. Wird eine davon vom Authentifikator erkannt, wird ein CREDENTIALS_EXISTS Fehler geworfen.

- authenticatorSelection

Mit dieser Eigenschaft lassen sich Authentifikatoren filtern, welche für die Registrierung und Anmeldung verwendet werden können. Sie enthält mehrere Untereigenschaften zur feineren Einstellung:

    - authenticatorAttachment

AuthenticatorAttachement kann zwei Werte annehmen. Wird der Wert „platform“ gesetzt, können nur Authentifikatoren, welche im verwendeten Gerät verbaut sind, verwendet werden. Der Wert „cross-platform“ hingegen, ermöglicht es verschiedene Arten von Authentifikatoren wie Smartphones, Security Keys oder ähnliches zu verwenden.

- requireResidentKey

Diese Eigenschaft kann nur Werte zwischen true oder false annehmen mit denen eingestellt werden kann ob der Private-Key im Authentifikator, dem Client oder einem Clientgerät gespeichert werden muss oder nicht. Wird die Eigenschaft auf true gesetzt, können die sogenannten resident credentials auf dem Authentifikator über die ID der Relying Party gefunden werden. Das heißt, dass die Authentifizierung selbst alle Resident-Keys findet und für jeden davon die Prüfung durchführt. Dem Client werden dann alle verfügbaren Zugangsdaten angezeigt und nach Auswahl von der Benutzerin oder dem Benutzer an die Relying Party gesendet.

- userVerification

Dies spezifiziert wie eine Benutzerin oder ein Benutzer die Erstellung der Zugangsdaten verifizieren muss. Standardwert dafür ist preferred, was bedeutet, dass falls kein Client PIN, biometrische Verifizierung oder ähnliches verfügbar ist, auf Test of User Presence (TUP) zurückfällt. Dabei muss zumindest ein Schalter gedrückt werden. Der Wert discouraged führt den TUP Zustand direkt herbei.

- attestation

Attestation hat den Zweck die Daten, welche von einem Authentifikator kommen, zu bestätigen. Es gibt drei Werte, welche die Antwort steuern können. Mit **direct** fordert die Relying Party die Bescheinigung, die vom Authentifikator generiert wurde. Der Wert **indirect** lockert diese Forderung dahingehend, dass diese Bescheinigung vom Client durch eine anonymisierte ausgetauscht werden kann, um die Privatsphäre der Benutzerin oder des Benutzers zu schützen. Wird aber **none** als Wert gesetzt, wird keine Bescheinigung von der Relying Party gefordert.

- extensions

Eine optionale Eigenschaft, die es erlaubt zusätzliche Parameter anzufordern, welche dann vom Client und dem Authentifikator verarbeitet werden können. Nun können die Zugangsdaten für eine Benutzerin oder Benutzer angelegt werden. (Ackermann Yuriy, 2019)

4.1.1.2. Anmeldung in der Web Applikation

Nachdem ein Authentifikator zum Profil einer Benutzerin oder eines Benutzers hinzugefügt wurde, kann er oder sie sich mit FIDO2 an der Web Applikation authentifizieren. Dafür muss die challenge der Web Applikation überprüft werden. Diese Prüfung wird über die get-Funktion des navigator.credentials-Objekts durchgeführt. Diese kann wieder mittels Parameter konfiguriert werden.

```
let publicKey = {
    challenge: challenge,
    timeout: timeout,
    rpId: rpId,
```

```
    allowCredentials: [
        { type: "public-key", id: credentialId }
    ]
    userVerification: userVerification,
    extensions: extensions
}
navigator.credentials.get({ 'publicKey': publicKey })
```

- Challenge

Die Challenge des publicKey-Objekts, stellt wieder, wie bei der Registrierung, den Buffer für die kryptografische Challenge bereit. Dieser Wert wird dann von einem Authentifikator signiert und zurück an den Server gesendet.

- Timeout

Wie bei der Registrierung kann damit gesteuert werden, wie lange auf die Antwort der Prüfung gewartet wird. Wert wird in Millisekunden angegeben.

- rpId

Die ID der Relying Party. Wenn diese nicht angegeben wird, wird der Wert der Ursprungsdomäne verwendet. Ein Beispiel dafür wäre login.fschmuck.at.

- allowCredentials

Damit werden die vorhandenen Zugangsdaten gefiltert, welche für die Authentifizierung zugelassen werden. Diese Eigenschaft enthält einige Untereigenschaften zur Verfeinerung.

    - Type

Typ der verwendet werden darf. Zurzeit dieser Arbeit kann nur der Wert public-key verwendet werden.

    - Id

Ein Buffer zum Auffinden einer public-key Zugangsdaten ID. Die ID wird bei der Erstellung der PublicKeyCredential Instanz erstellt und wird grundsätzlich vom Server bereitgestellt.

    - Transports

In diesem Feld können mögliche Authentifikatoren für die Authentifizierung angegeben werden. Mögliche Werte sind:

- „usb“; zum Beispiel ein Security Token
- „nfc“; zum Beispiel mittels Smartphone

- „ble“; zum Beispiel mittels Smartphone zu Gerät verbinden
- „internal“: zum Beispiel ein Laptop mit biometrischem Sensor

Als Beispiel hierfür wird nur ein USB Sicherheitstoken als akzeptables Authentifizierungsgerät konfiguriert. Kann kein Authentifikator dieser Konfiguration gerecht werden, wird vom Client ein „NotAllowedError“ produziert.

```
allowCredentials: [
    {
      transports: "usb",
      type: "public-key",
      id: new Uint8Array(26) // actually provided by the server
    },
```

- userVerification

Dieselbe Eigenschaft wie bei der Registrierung. Konfiguriert welche Rolle die Verifizierung bei der Authentifizierung spielt.

- Extensions

Gleich wie bei der Registrierung können zusätzliche Erweiterungen angegeben werden. Zum Beispiel Rückwärtskompatibilität zur FIDO JavaScript API.

Die einzige Eigenschaft, die für die Authentifizierung erforderlich ist, ist die Challenge. Damit kann nun auch eine Authentifizierung einer Benutzerin oder eines Benutzers stattfinden.

Die get-Methode des navigator.credentials-Objekts, liefert ein asynchrones Ergebnis. Ein sogenanntes Promise. Dieses Promise liefert nun das Ergebnis für einen Authentifikator der auf die angegeben Parameter passt. Falls nichts gefunden wird, ist der Wert null. Bei der Durchführung werden zuerst die Kriterien aus den bereitgestellten Optionen gesammelt und mit diesen, die besten Zugangsdaten herausgesucht. Die Optionen können auch erfordern, dass eine Benutzerin oder ein Benutzer eine Wahl treffen muss, welche Zugangsdaten er verwenden möchte. Eine typische Anmeldung mit der WebAuthn-Spezifikation sieht so aus:

4.1.2 Infrastrukturaufbau

Die Infrastruktur des Experimentes wird folgendermaßen aufgebaut:

- Ein Computer der als Server für die WebAuthn Applikation fungiert
- Ein Computer, von dem aus, sich der Client bei der WebAuthn Applikation registriert und authentifiziert
- Ein Computer, von dem aus, die Attacken durchgeführt werden
- Ein biometrischer Authentifikator und ein USB Security Key Authentifikator um sich bei der WebAuthn Applikation registrieren und authentifizieren zu können

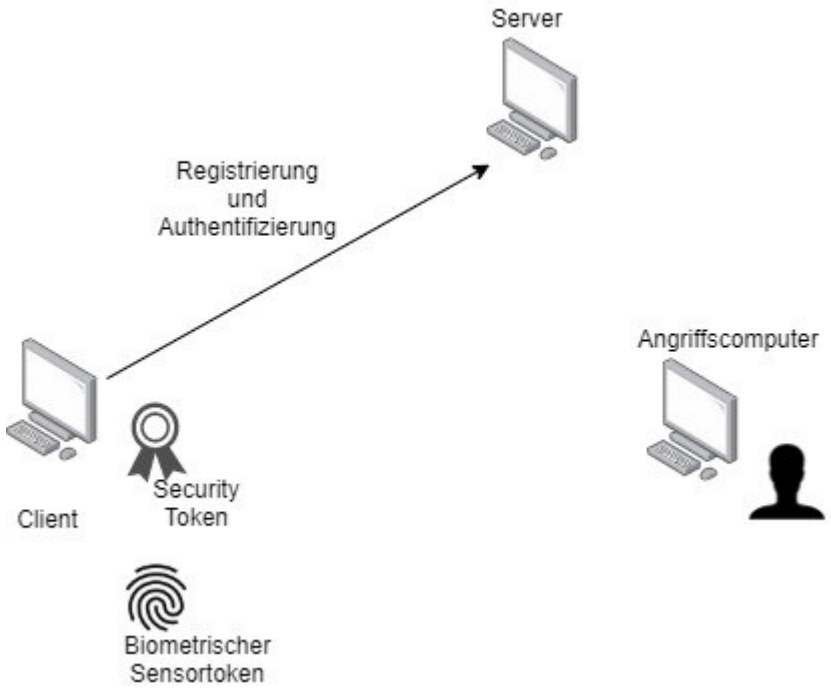

Abbildung 5 Infrastruktur des Experiments (Quelle: eigene Darstellung)

In der WebAuthn Applikation wird ein Testuser registriert, mit dem die Authentifizierung für die Attacken durchgeführt wird.

Abbildung 6 Registrierung Testbenutzer (Quelle: eigene Darstellung)

4.2 Ausführung der Attacken

In diesem Kapitel werden die beschriebenen Attacken aus dem Kapitel Attacken auf Authentifizierungsmechanismen und Zugangskontrolle gegen die Registrierung und Authentifizierung der WebAuthn-Spezifikation getestet.

4.2.1 Attacken auf Passwörter

Da WebAuthn keine Passwörter besitzt, müssen diese Attacken etwas abgewandelt werden, um durchgeführt werden zu können.

4.2.1.1. Credential Stuffing auf vorhandene Zugangsinformationen

Die Credential Stuffing Attacke wird gegen den Testbenutzer ausgeführt. Dabei wird einmal ein Benutzername ausprobiert, der nicht vorhanden ist und als zweiten Schritt der vorhandene Testbenutzer geprüft.

Versuch 1 wird mit Benutzername **Admin** durchgeführt:

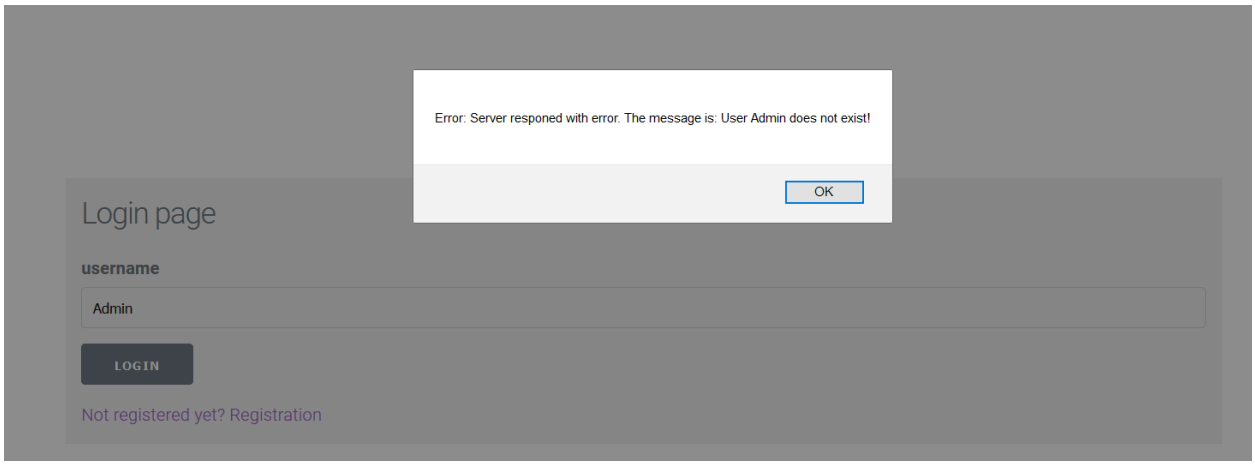

Abbildung 7 Credential Stuffing mit Admin (Quelle: eigene Darstellung)

Die Web Applikation liefert einen Fehler mit dem Hinweis das der Benutzer Admin nicht existiert. Der nächste Versuch wird nun mit dem vorhandenen Benutzernamen Test durchgeführt. Jedoch ist die Angreiferin oder der Angreifer nicht im Besitz eines registrierten Authentifikators für dieses Konto.

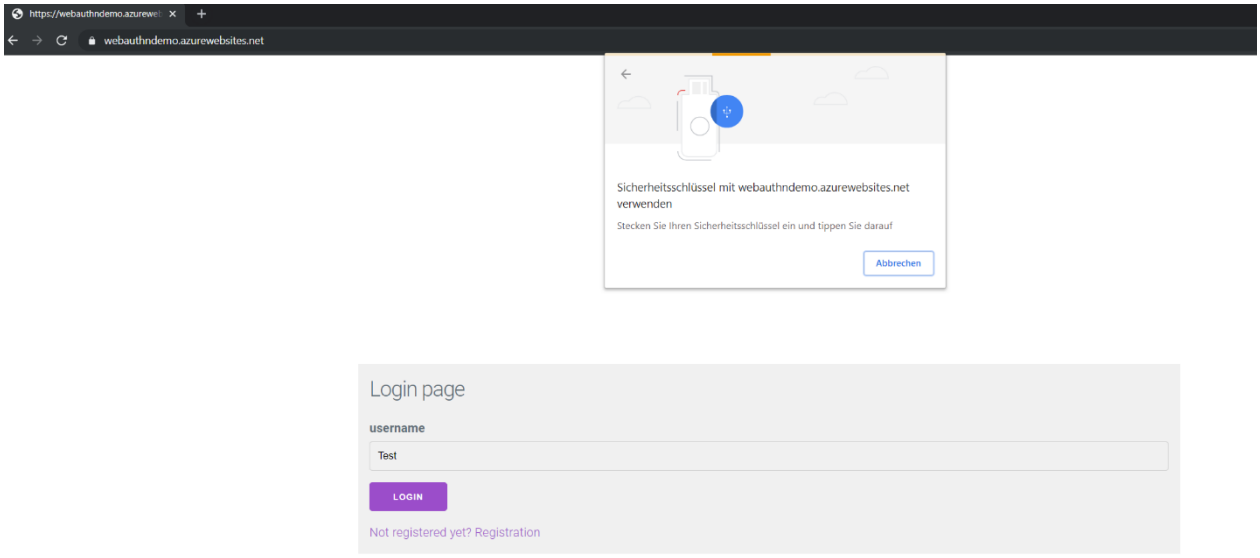

Abbildung 8 Credential Stuffing mit Benutzer Test (Quelle: eigene Darstellung)

An diesem Punkt müsste der Authentifikator die Challenge des WebAuthn Servers signieren, um die Authentifizierung erfolgreich abzuschließen. Dafür müsste der Private-Key geraten

werden und bei diesem Versuch richtig sein. Ansonsten schlägt der Authentifizierungsversuch fehl und bei einem neuen Versuch ändert sich die Challenge.

Die Credentials Stuffing Attacke bietet keine Möglichkeit an das Benutzerkonto eines Opfers zu gelangen. Gefundene Benutzernamen könnten jedoch für Angriffe auf andere Web Applikationen ohne der WebAuthn Spezifikation gesammelt werden.

4.2.1.2. Brute-Force Attacke

Die Brute-Force Attacke sieht der Credential Stuffing Attacke im Fall von WebAuthn sehr ähnlich. Man muss jeden möglichen Benutzernamen ausprobieren bis man auf einen erfolgreichen Versuch stoßt. Repräsentativ für diese Versuche wird wieder ein negativer Versuch mit dem Brute-Force Benutzernamen **A** und dem **Test** Benutzer ausprobiert. Der Benutzername A wurde deshalb ausgewählt, weil Benutzernamen in den häufigsten Fällen nur aus Buchstaben bestehen oder zumindest mit Buchstaben beginnen und es der erste Buchstabe im Alphabet ist der von der Brute-Force Attacke ausprobiert werden kann.

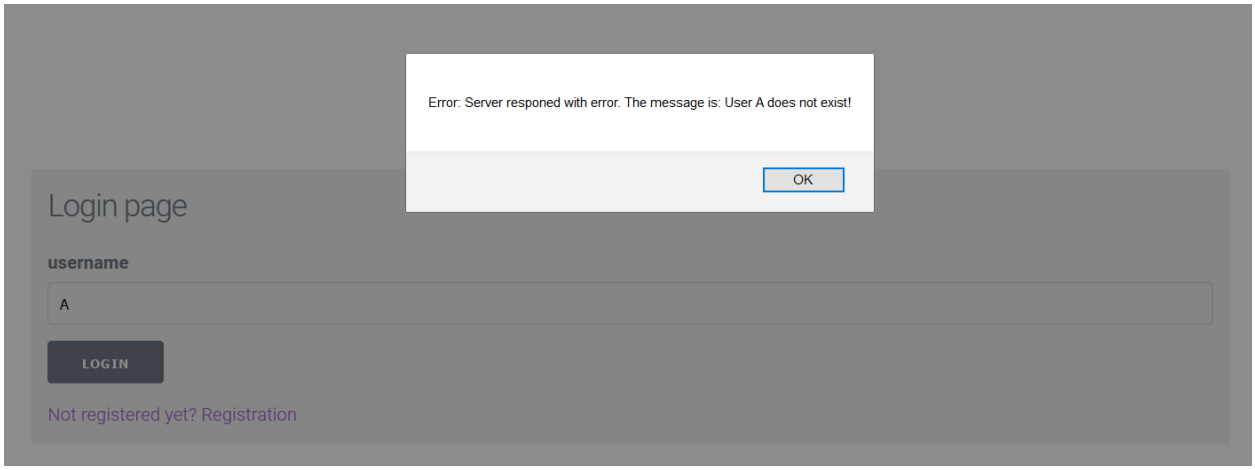

Abbildung 9 Brute-Force mit Benutzername A (Quelle: eigene Darstellung)

Der Benutzername A konnte nicht gefunden werden. Dafür wird ein Fehler der Applikation ausgegeben, welche die Meldung enthält, dass keine Benutzerin oder Benutzer mit diesem Namen existiert. Die Brute-Force Attacke würde jetzt so lang Variationen von Benutzernamen ausprobieren bis es einen positiven Treffer gibt. Im Fall dieses Experiments, wäre das der Benutzername Test.

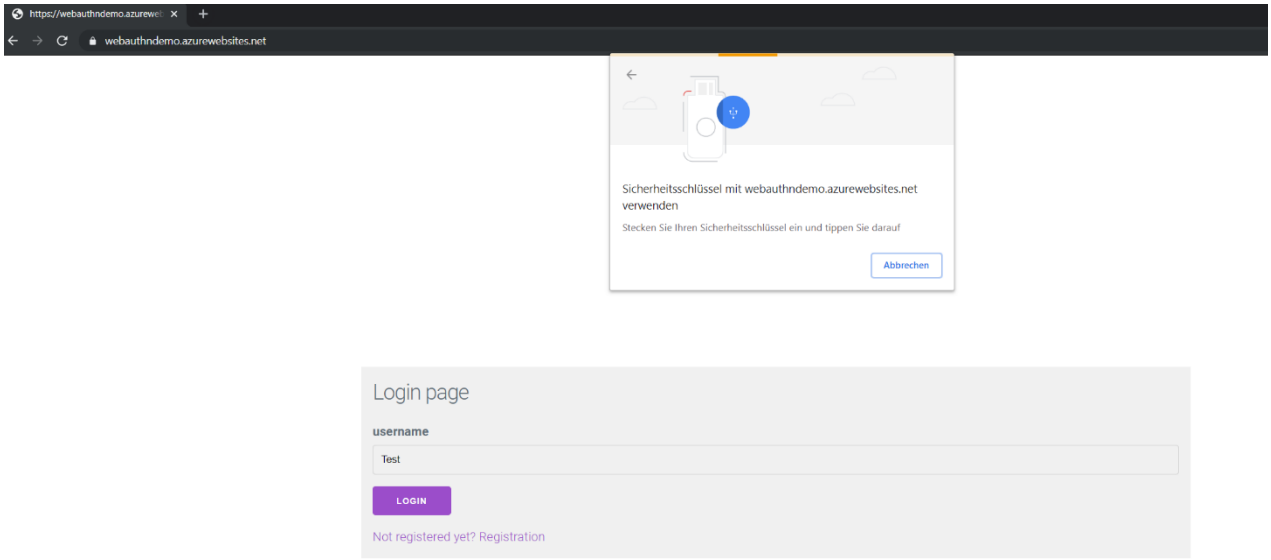

Abbildung 10 Brute-Force mit Benutzername Test (Quelle: eigene Darstellung)

Die Brute-Force Attacke ist bei der WebAuthn Spezifikation nur über die Benutzernamen durchführbar. Man kann Benutzernamen von anderen Authentifizierungssystemen sammeln und diese gegen eine Web Applikation, welche die WebAuthn Spezifikation zur Authentifizierung implementiert hat, ausprobieren. Je nach Implementierung erhält man eine Rückmeldung ob Benutzername des Versuchs überhaupt vorhanden ist, oder keine Information. Bei einem erfolgreichen Versuch mit einem Benutzernamen, muss die Challenge von einem registrierten Authentifikator signiert werden. Dieser Schritt stellt sich mit Brute-Force als schwierig heraus. Man müsste jede Möglichkeit einer erzeugten Signatur ausprobieren. Was bei FIDO2 Authentifikatoren bedeutet, dass Algorithmen aus der SHA-2 Familie geknackt werden müssen. Bei jedem Versuch wird eine neue Challenge erzeugt was die Arbeitslast für den Brute-Force Angriff noch weiter steigert. Bei einer guten Implementierung sollte die Web Applikation einen solchen Angriff melden und einen Logeintrag darüber erzeugen. Die gefundenen Benutzernamen könnten aber für Angriffe auf andere Web Applikationen genutzt werden, die vielleicht nur eine Authentifizierung mittels Passwortes unterstützen.

4.2.1.3. Wörterbuchattacke

Für die Wörterbuchattacke wird eine Liste von Benutzernamen aus anderen fiktiven Web Applikationen angelegt. Diese Liste beinhaltet zwei Benutzernamen:

- Anna
- Test

Die Benutzerin Anna repräsentiert den negativen Versuch. Das heißt diese Benutzerin hat zwar ein Benutzerkonto auf einer anderen Web Applikation aber nicht auf dieser WebAuthn Applikation. Der Test Benutzer wird wieder für den erfolgreichen Eingabeversuch herangezogen.

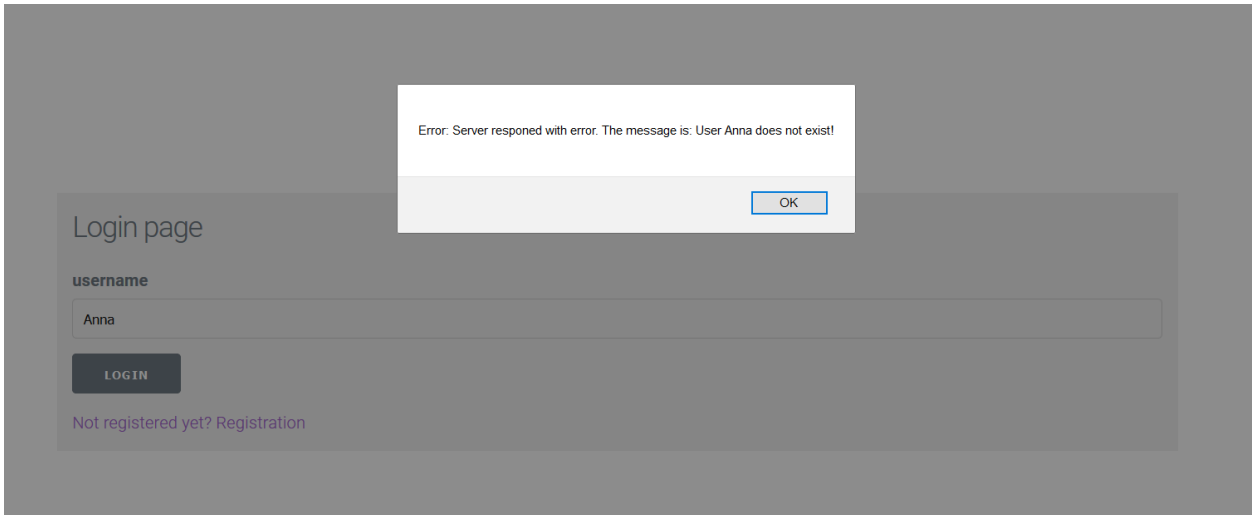

Abbildung 11 Wörterbuchattacke mit Benutzerin Anna (Quelle: eigene Darstellung)

Die Benutzerin Anna konnte beim Anmeldeversuch in der WebAuthn Applikation nicht gefunden werden. Das kann zumindest im Wörterbuch vermerkt werden.

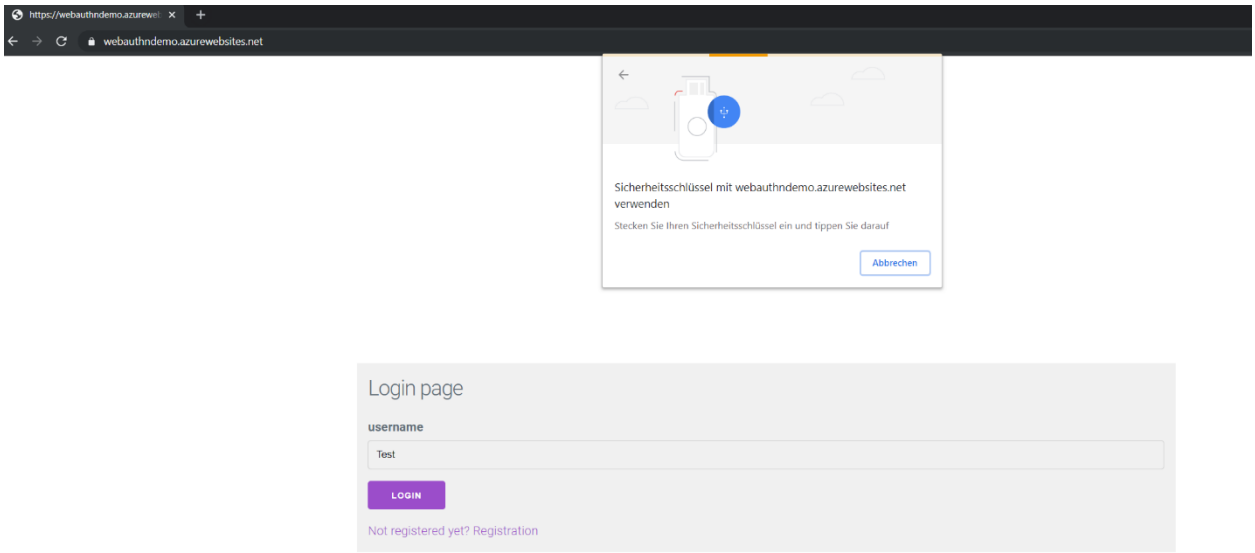

Abbildung 12 Wörterbuchattacke mit Benutzername Test (Quelle: eigene Darstellung)

Der Benutzer Test aus dem Wörterbuch wird zur Authentifizierung mit einem seiner registrierten Authentifikatoren aufgefordert. Da die Angreiferin oder der Angreifer nicht im Besitz von einem dieser Authentifikatoren ist, ist der Angriff damit beendet.

Ähnlich wie bei den Credential Stuffing und Brute-Force Attacken kann hierfür nur der Benutzername herangezogen werden. Um das Wörterbuch zu erzeugen, würde man entweder die Resultate der Brute-Force Attacke ansehen oder sich fertige Listen aus dem Web besorgen welche von Zugangsdaten von anderen Web Applikationen erstellt wurden. Aus diesen können noch Varianten gebildet werden, um mehr Namen abzudecken. Zusätzlich zu den Namen müsste man nun jeden möglichen Signaturwert berechnen, und in das Wörterbuch eintragen. Was die Wörterbuchattacke vor eine unmögliche Aufgabe stellt da der Private-Key unbekannt

ist. Selbst mit einem Clustersystem, das viele Werte gleichzeitig berechnen kann, müsste man jede mögliche Challenge in die Berechnung mit einbeziehen, um den richtigen Wert für die Authentifizierung parat zu haben.

4.2.1.4. Angriff auf ineffektive Speicherung

Um die ineffektive Speicherung beziehungsweise eine SQL Injection Attacke zu simulieren wurden die Daten der Datenbank extrahiert, die eine Angreiferin oder ein Angreifer erhalten würde, wenn er oder sie eine solche Attacke ausführen. Das Resultat wird im JSON Format angeben:


```
{
  "Test": {
    "name": "Test",
    "registered": true,
    "id": "4-XpphAK2yr-uZ-ggY6EIdnIEsfVdS04-eK6XAJU3ic",
    "authenticators": [
      {
        "fmt": "fido-u2f",
        "publicKey": "BEJkp9ldam2zbvHO0TT99VbnAximRZg6LYT6Wyn-
ep4S8DgU370jNCZQPdzoEI6scek4xU13LNL1xGn8b5wJRY0",
        "counter": 1357,
        "credID": "p5D6hn-yqo-pvU6pISMZSj8x3A0oxbkb4AC-
S6063wLxo2UkgUUnXbZ7cfy20DWLxs5c_Udn9y8wKUNrwPnan9wf8uhtBxH5hSyHD36K06jgwzeDx
DG5ncQ1qQUVw52K"
      }
    ]
  },
  "User1": {
    "name": "User1",
    "registered": true,
    "id": "zgCkVavUrVzQ7xyPO32SCJNHSnV54cbJi11lrQ4_NTs",
    "authenticators": [
      {
        "fmt": "fido-u2f",
        "publicKey": "BGZYJAyBaAMWQisi3drHG2yynT55D-
65QTKgvAgoPEjnVohSkbAyNPvUCsrEoX6_RLeJMXdrjRWWN2kONjOlfbo",
        "counter": 1417,
        "credID": "2pc6TSAKbk0wA0hzdByWfFlVQU5KJAPI-
YJwllVpoLKPmyaXGbcadrX4QwYBb6_poKevt-
kCKJ1vjZa6DNqXLo1NklZTPymnDvXVXSJTtdsLa_tvwGymiikDhGyuzLjj"
      }
    ]
  },
  "xyz": {
    "name": "xyz",
    "registered": true,
    "id": "gDwpjRGjQRY6XdIGcizteSPc4N_bLFc1LmbcGiCE2E8",
    "authenticators": [
      {
        "fmt": "fido-u2f",
        "publicKey": "BMEdk1-
_phgXL7eMvRcWwi32buujx9tAWZbSFi78OaC5kUAYet4yoRd2bhL1ggKXWRQ5RizrrupklRKCyyi4
GAQ",
        "counter": 0,
        "credID":
```

```
"KstDEHWAFT9QCejI9reox9kL68mU4UDfscGsVocz_WXzMYKvK_SqFJF5CHaM6xp_E5aRPL_6k5Pg
f9Xfm8j6ivL4Z7xz-CCQNeNWRTFhORB_6wNxhG9Qnai3JySvbbn1"
        }
      ]
    }
}
```


Der eindeutige JSON-Objektschlüssel ist der Benutzername. Mit diesem erhält man Zugriff auf die Eigenschaften des Objekts. Der Anzeigename der Benutzerin oder des Benutzers wird über die name-Eigenschaft angegeben. Über die Eigenschaft registered kann entnommen werden ob dieser Benutzername zurzeit am System registriert ist. Eine eindeutige Identifikation wird über die id-Eigenschaft erzielt. Das Array-Objekt authenticators, beinhaltet alle Authentifikatoren welche von der Benutzerin oder dem Benutzer für diesen Benutzernamen registriert worden sind. Die Art des Authentifikators wird über die fmt-Eigenschaft beschrieben. Der publicKey wird benötigt, um die signierte Challenge des Clients überprüfen zu können. Die counter-Eigenschaft wurde für den Fall, dass es doch jemanden gelingen sollte einen Authentifikator zu klonen, entwickelt. Damit werden erfolgreiche Anmeldeversuche protokolliert. Wenn nun ein geklonter Authentifikator einen Anmeldeversucht wagt, stimmt der Counterwert nicht überein und ein Angriff kann protokolliert werden. Zuletzt gibt es noch eine eindeutige Credentials Identifikation für den Authentifikator pro Benutzername.

Bei der WebAuthn Spezifikation werden nur die Benutzernamen der Benutzerinnen und Benutzer in der Datenbank abgelegt. Ist es einer Angreiferin oder einem Angreifer nun möglich diese Benutzernamen aus der Datenbank zu extrahieren, stehen die gleichen Möglichkeiten zur Attacke offen wie bei einer Credential Stuffing, Brute-Force oder Wörterbuchattacke. Bei einer Attacke mithilfe von SQL Injection wäre es der Angreiferin oder dem Angreifer aber möglich, die vorhandenen Zugangsdaten einer Benutzerin oder eines Benutzers zu löschen. Dadurch öffnet sich möglicherweise das Tor für eine andere Art von Angriff. Die betroffenen Benutzerinnen und Benutzer müssten sich schließlich erneut in der Web Applikation anmelden, wenn ein Backup nicht eingespielt werden kann.

4.2.2 Schwache Authentifizierungsfaktoren

Für die Rücksetzung von Multi-Faktor Authentifizierung werden Einmal-Codes, E-Mail und SMS Rücksetzung angeboten. Wenn Einmal-Codes ungeschützt gespeichert werden und in die Hände einer Angreiferin oder eines Angreifers gelangen, ist es ein leichtes die Rücksetzung zu vollziehen und die Zugangsdaten einer Benutzerin oder eines Benutzers neu zu setzen. Wird ein Einmal-Code per E-Mail oder SMS angefordert, kann dieser mitgehört und abgefangen werden, um ihn für die Rücksetzung zu verwenden. Für die WebAuthn Spezifikation sind derzeit nur bestimmte Authentifikatoren zugelassen. Diese können aus der folgenden Liste entnommen werden:

- FIDO2 zertifiziertes Smartphone über Bluetooth
- FIDO2 zertifiziertes Smartphone mittels Authentifikator
- USB Security Key

- Eingebaute Authentifikatoren, wie zum Beispiel Windows Hello

Hat man nur einen Authentifikator verloren und will das Benutzerkonto zurücksetzen, ist das sehr schwierig. Die FIDO Alliance empfiehlt immer mehr als einen Authentifikator für ein Benutzerkonto zu registrieren, um bei Verlust oder Diebstahl den verlorenen Authentifikator entfernen zu können und das Konto weiterhin benutzen zu können.

4.2.3 Login Spoofing von WebAuthn

Um dem Opfer eine falsche Anmeldemaske unterjubeln zu können, wurde eine idente Seite mit dem Titel „Malicious Login page“ angelegt. Dieser wird versucht statt der richtigen Seite beim Aufruf der Web Applikation anzuzeigen. Dafür werden die Pakete mithilfe des Werkzeuges Wireshark abgehört. Dann wird nach dem richtigen Paket gesucht, welches der Benutzerin oder dem Benutzer die echte Anmeldemaske anzeigt. Wenn dieses Paket gefunden wurde soll es bei einem erneuten Aufruf der Web Applikation durch die falsche Anmeldemaske ausgetauscht werden.

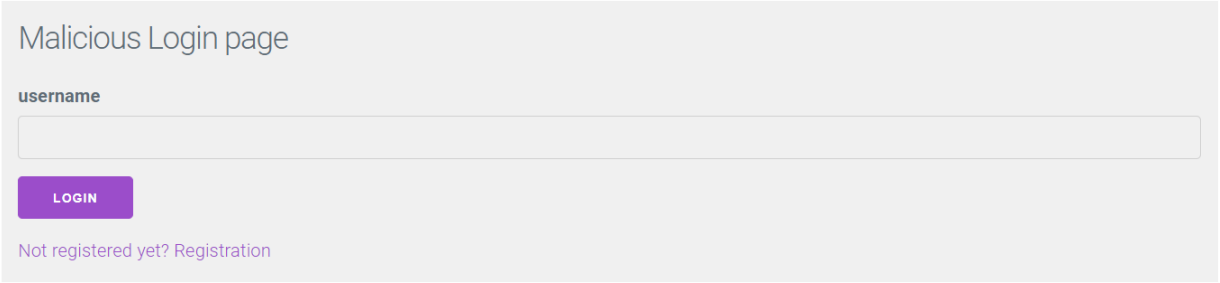

Abbildung 13 Gefälschte Loginmaske (Quelle: eigene Darstellung)

Wireshark zeigt die Datenpakete, welche zwischen der Web Applikation und dem Client ausgetauscht werden. Das Paket, welches die Anmeldemaske für den Client bereitstellt, kann jedoch nicht identifiziert werden, da es über einen verschlüsselten Kanal übertragen wird. Selbst wenn die falsche Anmeldemaske den Weg zur Benutzerin oder Benutzer findet, kann nur der Benutzername von einer Hackerin oder einem Hacker gestohlen werden. Dadurch stehen wieder nur die gleichen Möglichkeiten wie bei den Passwortattacken zur Verfügung.

Dieser Angriff scheitert meist schon an der erzwungenen Übertragung über einen verschlüsselten Kanal. Im besten Fall kann ein Benutzername gestohlen werden.

4.2.4 Sniffing der WebAuthn Authentifizierung

Die Sniffing Attacke wird mithilfe des Werkzeugs Wireshark durchgeführt. Dieses bietet die Möglichkeit die Netzwerkpakete in einem Netzwerk abzufangen und mitzulesen. Dadurch sollen die benutzten Zugangsdaten für die Authentifizierung sichtbar gemacht werden.

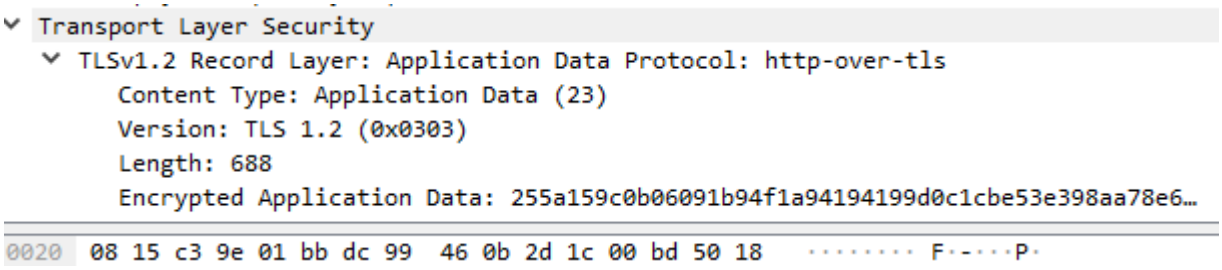

Abbildung 14 Erfasste Übertragung via Wireshark (Quelle: Eigene Darstellung)

Die Datenpakete werden über einen sicheren Kanal übertragen und können von einer Angreiferin oder einem Angreifer nicht lesbar gemacht werden. Sollten Daten aus irgendeinem Grund im Klartext übertragen werden, könnte die Angreiferin oder der Angreifer nur Informationen wie Benutzername, Public-Key des Authentifikators, Art des Authentifikators und ID herausfinden.

Fazit:

Damit könnte man sich zwar auf eine andere nicht technische Art von Attacke vorbereiten, für die Sniffing Attacke wären diese Daten jedoch nutzlos.

4.2.5 Übernahme einer mit WebAuthn authentifizierten Sitzung

Zur Durchführung der Übernahme einer WebAuthn Sitzung, muss eine HTTPS Downgrading oder auch SSL Stripping Attacke durchgeführt werden. Laut Spezifikation wird für die Authentifizierung mit WebAuthn ein HTTPS verschlüsselter Kanal vorausgesetzt. Wie Wireshark bei den vorherigen Attacken schon gezeigt hat können die Pakete nicht lesbar gemacht werden. Schafft man es die Web Applikation über das HTTP-Protokoll laufen zu lassen, gibt es ein anderes Problem für Hackerinnen und Hacker. Wenn ein create oder get mit der publicKey Eigenschaft auf das credentials-Objekt angewandt wird, die Web Applikation aber über HTTP ausgeführt wird, hat das credentials-Objekt den Wert null und kann nicht verwendet werden.

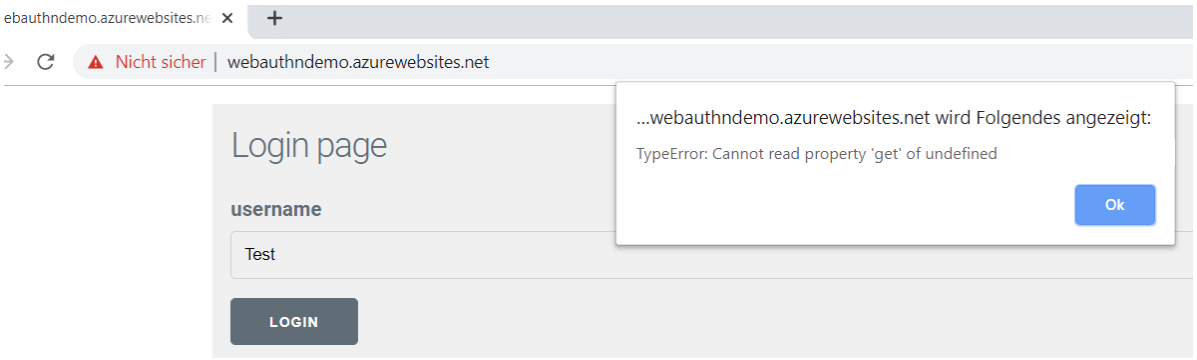

Abbildung 15 Fehler bei unsicherer Verbindung (Quelle: eigene Darstellung)

4.2.6 Social Engineering

Eine Social Engineering Attacke kann eine der effektivsten Attacken auf die WebAuthn Spezifikation darstellen. Da es man in diesem Fall nur wissen möchte, welche Authentifikatoren die Benutzerin oder der Benutzer zur Anmeldung in Verwendung hat. Da man sich leicht in Sicherheit wiegt, zeigt man diese offen. Angreiferinnen oder Angreifern müssen, falls nicht vorab schon passiert, die Web Applikationen und Seiten des Opfers ausfindig machen. Danach können sie einen oder mehrere der verwendeten Authentifikatoren stehlen. Wenn der Angriff nun schnell genug stattfinden kann und etwaige Authentifikatoren, welche sich noch in Besitz des Opfers befinden, aus dem Benutzerkonto entfernt werden können, ist es dem Opfer nicht mehr möglich ihr oder sein Benutzerkonto wiederherzustellen, wenn von der Relying Party kein anderer Weg zur Wiederherstellung bereitgestellt wird.

Am besten ist es, wenn die Relying Party gar keine Möglichkeit zulässt, dass eine Benutzerin oder ein Benutzer einen Authentifikator löschen kann. Falls ein Authentifikator kompromittiert wird, sollte es der Relying Party gemeldet werden, und diese soll sich darum kümmern. Natürlich muss damit ein Verifikationsprozess verbunden werden, denn sonst könnte auch der Angreifer diese Prozedur für die übrigen Authentifikatoren des Opfers durchführen. Als Gegenmaßnahme kann in diesem Fall nur empfohlen werden, immer auf Authentifikatoren aufzupassen und niemandem zu zeigen welche verwendet werden. Meist werden Authentifikatoren am Schlüsselbund oder ähnlichem befestigt und können so ein leichtes Ziel darstellen, wenn dieser am Tisch gelegt wird.

4.2.7 Shoulder Sniffing

Shoulder Sniffing lässt sich hier nur abgewandelt durchführen. Man kann eine Benutzerin oder einen Benutzer dabei beobachten mit welchem Benutzernamen sie oder er sich an einer Web Applikation anmeldet und welchen Authentifikator sie oder er dafür verwendet. Dieser Authentifikator kann ein einem unaufmerksamen Moment gestohlen werden, um die Authentifizierung durchführen zu können.

Wenn kein bestimmtes Zielsystem attackiert werden soll, muss die Angreiferin oder der Angreifer zuerst herausfinden auf welcher Web Applikation sich die Benutzerin oder der Benutzer mit einem Authentifikator anmeldet.

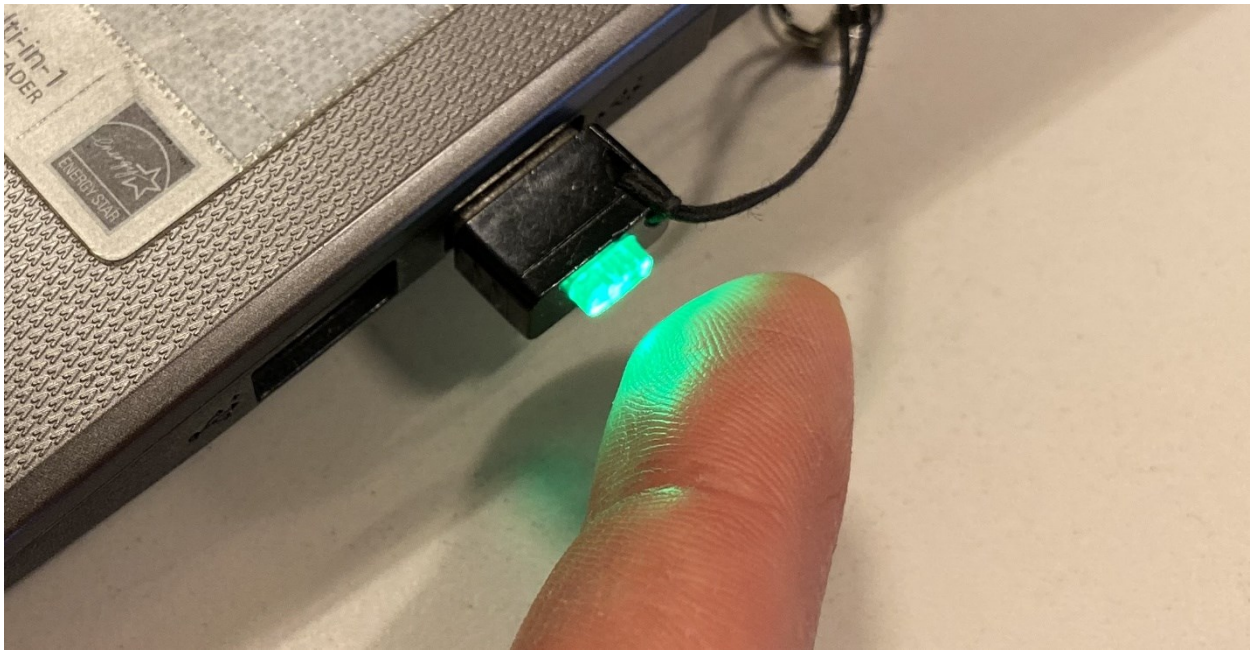

Abbildung 16 Drücken des Authentifikators (Quelle: eigene Darstellung)

Sieht eine Angreiferin oder ein Angreifer das ein Authentifikator verwendet wird der nur über einen Tastendruck verifiziert wird, ist es leicht diesen zu verwenden, wenn man ihn stiehlt.

Abbildung 17 Verwendung eines biometrischen Sensors (Quelle: eigene Darstellung)

Bei einem biometrischen Sensor muss man überlegen ob es überhaupt Sinn macht diesen zu stehlen, da die Verwendung von einer unautorisierten Person sehr schwierig ist. Daher sollte man vorsichtig mit Authentifikatoren umgehen. Werden die genutzten Web Applikationen und die dafür verwendeten Authentifikatoren von einer Angreiferin oder einem Angreifer herausgefunden und gestohlen, kann damit das Benutzerkonto eines Opfers übernommen werden.

4.2.8 Phishingattacken

Der folgende Abschnitt werden Attacken der Phishinggruppe gegen die WebAuthn Applikation geprüft.

4.2.8.1. Phishing

Für die Phishingattacke wird normalerweise versucht dem Opfer einen Link zu einer gefälschten Web Applikation unterzujubeln, um dann an dessen Zugangsdaten oder andere persönliche Informationen zu kommen. Zu diesem Zweck wurde eine zweite Web Applikation, die der echten Experimentapplikation sehr ähnlich ist zur Infrastruktur hinzugefügt. Nun stellt man die WebAuthn Spezifikation auf die Probe. In diesem Szenario besitzt die Benutzerin oder der Benutzer bereits ein Benutzerkonto auf der echten Web Applikation. Nun wird der Link zur falschen Web Applikation angeklickt, welche sich für das Experiment demonstrativ nur über den Hintergrund und natürlich der URL unterscheidet.

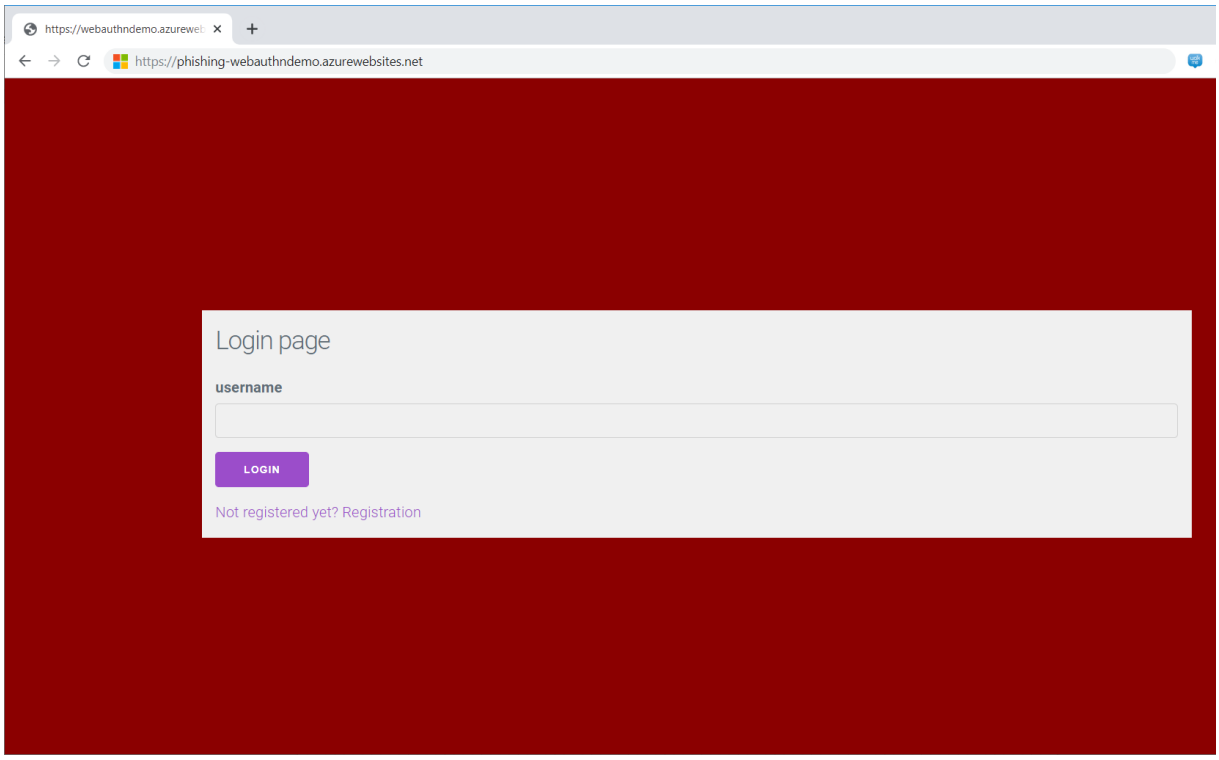

Abbildung 18 Phishing Web Applikation (Quelle: eigene Darstellung)

Das Opfer versucht sich an der Web Applikation mit ihrem oder seinem Benutzernamen anzumelden. Nun wird ihr oder ihm aber eine Fehlermeldung angezeigt, dass der Benutzername in der Applikation nicht gefunden werden konnte. Damit scheitert auch der klassische Phishingangriff.

4.2.8.2. Vishing

Da es für WebAuthn derzeit keine Authentifikatoren mit Stimmerkennung gibt, ist dieser Angriff für das Experiment nicht relevant. Man könnte dem Opfer vielleicht nur entlocken welche Authentifikatoren es im Einsatz hat und bei welchen Web Applikation es Benutzerkonten damit gesichert hat. Damit ließe sich eine Social Engineering Attacke starten.

4.2.8.3. Spear Phishing

Durch Spear Phishing könnte man gezielt die Authentifikatoren eines bestimmten Opfers ausfindig machen. Dazu noch die nötigen Benutzernamen sammeln und zuletzt einen oder am besten, falls möglich, alle Authentifikatoren stehlen. Dadurch wäre das Opfer nicht mehr in der Lage die kompromittierten Authentifikatoren von den, damit gesicherten, Benutzerkonten zu entfernen. Spear Phishing ist in diesem Fall die Informationsbeschaffung über das Opfer und dessen Authentifizierungsmethoden. Als Abschluss davon müssen die Authentifikatoren gestohlen werden.

4.2.9 Attacken auf Authentifikatoren

In diesem Abschnitt werden die Angriffe auf Authentifikatoren beschrieben.

4.2.9.1. Biometrische Sensoren

Aus der Literaturrecherche ist hervorgegangen, dass schon Versuche gelungen sind biometrische Sensoren mithilfe von Gummibonbons zu überlisten. Zur Durchführung dieser Attacke wird ein FIDO zertifizierter biometrischer Authentifikator von Kensington verwendet. Dieser wurde auf den Test-Benutzer registriert. Anschließend wurde versucht die Fingerabdrücke auf verschiedene Gummibonbons zu drücken und damit den biometrischen Sensor zu entsperren.

Der Angriff wurde folgendermaßen durchgeführt:

- Fingerabdruck in Applikation registriert
- Finger auf Gummibonbon gepresst bis Abdruck sichtbar war
- Gummibonbon in verschiedenen Winkeln auf biometrischen Sensor gedrückt, um Verifizierung durchzuführen

Da dieser offenbar mehr als nur den Fingerabdruck der Benutzerinnen und Benutzer misst, schlägt dieses Experiment fehl. Als Vermutung bleibt jedoch offen, dass es stark von den jeweiligen Sensoren abhängt ob diese, dieser Attacke standhalten können.

4.2.9.2. USB Security Key

Angriffe auf andere Arten von Authentifikatoren sind nicht einfach durchzuführen. Da jeder Authentifikator, einschließlich biometrischer Sensoren, über das CTAP Protokoll mit dem Computer interagiert, müsste dieses eine Schwachstelle aufweisen, um einen Angriff auf den Authentifikator selbst durchführen zu können. Da die Analyse und Schwachstellensuche des CTAP Protokolls aber den Rahmen dieser Arbeit sprengen würde, wird davon ausgegangen, dass die Sicherheit von Authentifikatoren unter anderem von der Implementierung des CTAP Protokolls am jeweiligen System abhängig ist.

4.2.10 Cross-Site Request Forgery Attacke auf WebAuthn

Die CSRF-Attacke ist im Fall von WebAuthn der Login Spoofing Attacke sehr ähnlich. Man versucht ein Bild oder ein unscheinbares HTML-Element in eine Seite der Benutzerin oder des Benutzers einzuschleusen. Dieses soll dann Schadcode ausführen damit eine Angreiferin oder ein Angreifer Zugriff auf das Benutzerkonto oder die Zugangsdaten dieses Benutzerkontos erlangen. Da die die WebAuthn Spezifikation aber nur über HTTPS ausgeführt werden kann, kann kein Element in die Seite eingeschleust werden.

Durch die erzwungene verschlüsselte Übertragung von WebAuthn, lässt sich kein Paket angreifen beziehungsweise Schadcode einschleusen. Daher schlägt dieser Angriff, gleich wie die Login Spoofing Attacke, fehl.

4.3 Gegenmaßnahmen

Für die meisten Arten von Angriffen bietet WebAuthn einen robusten Schutz des Benutzerkontos. Die gefährlichsten Attacken bleiben nach wie vor verschiedene Arten von Social Engineering Attacken. Grundsätzlich kann gesagt werden, dass man immer ein Auge auf seine verwendeten Authentifikatoren haben sollte. In einem öffentlichen Umfeld sollte man darauf achten, ob jemand die Eingabe überwacht und es auf die Authentifikatoren abgesehen hat. Bei der Auswahl der verwendeten Authentifikatoren sollte darauf geachtet werden, keine zu alten Geräte zu kaufen oder Geräte zu verwenden, welche bereits eine bekannte Sicherheitslücke aufweisen. Im Fall von beispielsweise biometrischen Sensoren ist es stark vom Prüfungsalgorithmus abhängig, ob dieser zukünftigen Angriffsarten noch standhalten kann. Authentifikatoren, welche eine biometrische Benutzerverifizierung erfordern, bieten jedoch immer einen besseren Schutz als Authentifikatoren, die nur durch einen einfachen Tastendruck verifiziert werden können. Programmierfehler in Schnittstellen können immer auftreten. Als Benutzerin oder Benutzer kann man dabei nur auf die Entwicklungsfirmen von WebAuthn Applikationen achten und diesen vertrauen. Man kann jedoch grundsätzlich sagen, dass die WebAuthn Spezifikation bessere Sicherheit als herkömmliche Passwortauthentifizierung bietet. Daher sollte, wenn möglich, WebAuthn immer vorgezogen werden. Selbst wenn sich ein Fehler

in der CTAP Implementierung befindet, ist es schwieriger diese Lücke auszunützen als verschiedene Arten von Passwortattacken durchzuführen.

5 CONCLUSIO

In dieser Arbeit wurden verschiedene Attacken auf unterschiedliche Arten von Authentifizierungsmethoden aus der Literaturrecherche untersucht. Dabei wurden Schwächen von passwortgestützten Methoden erläutert. Des Weiteren wurden ineffektive Speicherungs- und Verschlüsselungsansätze von Passwörtern diskutiert. Selbst die sicher geglaubte Multi-Faktor Authentifizierung weist Schwächen in den Rücksetzungsmethoden auf. Herkömmliche Authentifizierungsmethoden weisen Schwächen gegen allerlei Arten von Phishing Angriffen auf, welche durch die WebAuthn Spezifikation ausgemerzt wurden. Die gefährlichsten Arten von Angriffen bleiben jedoch nach wie vor verschiedene Social Engineering Attacken. Diesen Arten von Attacken kann mit Technologie nur schwer entgegengewirkt werden, da man es im Fall von WebAuthn auf die Authentifikatoren selbst abgesehen hat. Natürlich muss zuvor auch der Benutzername und die zugehörige WebAuthn Applikation herausgefunden werden, damit man keinen Brute-Force Angriff auf verschiedene WebAuthn Applikationen im Internet durchführen muss. Als Benutzerin oder Benutzer sollte man sich vor dem Kauf von Authentifikatoren gut informieren, um keine Authentifikatoren zu kaufen, die bereits eine Sicherheitslücke aufweisen oder mit einem nicht behebbaren Programmierfehler ausgeliefert wurden. Grundsätzlich sollten aber immer Authentifikatoren verwendet werden, welche eine zusätzliche Interaktion vom Benutzer erfordern. Ein Beispiel dafür wäre ein biometrischer Sensor. Selbst bei Diebstahl ist es fast unmöglich, diesen zu verwenden. Bei Authentifikatoren, welche nur einen Tastendruck benötigen, um die Authentifizierung durchzuführen, ist man bei Diebstahl nicht weiter geschützt.

Abschließend ist zu sagen, dass die WebAuthn Spezifikation eine Bereicherung für die Sicherheit von Applikationen darstellt. Aufgrund der Diebstahlgefahr jedoch sollten die genutzten Authentifikatoren immer im Auge behalten werden oder eng am Körper getragen werden und keinesfalls offen zugänglich gemacht werden. An öffentlichen Orten, wie beispielsweise Flughäfen, sollte man sich vor Beobachtung bei der Eingabe eines Benutzernamens auf einer WebAuthn Applikation und der Nutzung eines Authentifikators schützen. Bei der Wahl von Authentifikatoren sind jene zu bevorzugen, welche eine biometrische Verifizierung erfordern. WebAuthn bietet guten Schutz gegen fast alle evaluierten Angriffe. Verbesserungswürdig ist noch die Handhabung des Rücksetzungsprozesses von Authentifikatoren.

ABKÜRZUNGSVERZEICHNIS

API ..... *Applikationsprogrammierschnittstelle*
CAPTCHA..... *Completely Automated Public Turing to tell Computer and Humans Apart*
Challenge..... *Forderung*
CORS ..... *Cross-Origin Resource Sharing*
CSRF ..... *Cross-Site Request Forgery*
CTAP ..... *Client to Authenticator Protocol*
FAR..... *false-acceptance rate*
FIDO ..... *Fast Identity Online*
GPUs ..... *Graphical Processing Unit*
HTML ..... *Hypertext Markup Language*
HTTP ..... *Hypertext Transfer Protocol*
HTTPS ..... *Hypertext Transfer Protocol Secure*
IEEE..... *Institute of Electrical and Electronic Engineers*
IP Internet Protocol
JSON ..... *JavaScript Object Notation*
JWT ..... *JSON Web Token*
Message Header ..... *Nachrichtenkopf*
MitM ..... *Man-in-the-middle*
NFC ..... *Nahfeldkommunikation*
OTP ..... *One-Time-Passwort*
OWASP ..... *Open Web Application Security Project*
Password-Based Key Derivation Function 2 ..... *Password-Based Key Derivation Function 2*
Public Keys ..... *öffentliche Schlüssel*
Relying Party ..... *Vertrauenspartei*
Request Data..... *Anfragedaten*
Response Body ..... *Antwortkörper*
Secret ..... *Geheimnis*
Session ..... Sitzung
Session ID ..... *Sitzungsidentifikation*
SSL ..... *Secure Socket Layer*
TLS ..... Transport Layer Security
TUP..... *Test of User Presence*
U2F ..... *Universal Second Factor*
UAF..... *Universal Authentication Framework*
URI..... *Uniform Resource Identifier*
URL..... *Uniform Resource Locator*
USB ..... *Universal Serial Bus*
WAF ..... Web-Application-Firewall
WLAN ..... *Wireless Local Area Network*

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

6 REFERENCES

Abhay Bhargav (2019). What is and how to prevent Broken Access Control | OWASP Top 10 (A5). Retrieved from https://hdivsecurity.com/owasp-broken-access-control

Ackermann Yuriy (2019). Introduction to WebAuthn API – Ackermann Yuriy – Medium. Retrieved from https://medium.com/@herrjemand/introduction-to-webauthn-api-5fd1fb46c285

Alexander Lauert (1999). Elektronisches Bezahlen: Proseminar "eCommerce" SS 1999. Retrieved from http://ddi.cs.uni-potsdam.de/Lehre/e-commerce/elBez2-5/page08.html

APWG (2019). Phishing Activity Trends Report: 1st Quarter 2019. Retrieved from https://docs.apwg.org/reports/apwg_trends_report_q1_2019.pdf

Bertocci, V. (2016). *Modern authentication with Azure Active Directory for web applications*. Redmond, Washington: Microsoft Press.

Boyd, R. (2012). *Getting started with OAuth 2.0*. Sebastopol, CA: O'Reilly.

Burnett, M. (2005). *Password roulette: Beating the authentication game*. Rockland, Mass., Oxford: Syngress; Elsevier Science [distributor].

Chapman, N. (2012). *Authentication and authorization on the web*. [Place of publication not identified]: Macavon Media.

Colin Watson, & Tin Zaw (2018). OWASP Automated Threat Handbook: Web Applications. Retrieved from https://www.owasp.org/images/3/33/Automated-threat-handbook.pdf

Dawn M. Turner (2016). Digital Authentication - the basics. Retrieved from https://www.cryptomathic.com/news-events/blog/digital-authentication-the-basics

Diogenes, Y., & Ozkaya, E. (2018). *Cybersecurity, attack and defense strategies: Infrastructure security with Red Team and Blue Team tactics*. Birmingham, UK: Packt Publishing. Retrieved from http://proquest.tech.safaribooksonline.de/9781788475297

Jad Karaki (2019). Passwordless future using FIDO2 & WebAuthn – The Startup – Medium. Retrieved from https://medium.com/swlh/passwordless-future-using-fido2-webauthn-9342aaa7d7f7

Jared Kee (2008). Social Engineering: Manipulating the Source. Retrieved from https://www.sans.org/reading-room/whitepapers/engineering/social-engineering-manipulating-source-32914

Joyce Tammany (2018). The OWASP Top 10: Sensitive Data Exposure – The SiteLock Blog. Retrieved from https://www.sitelock.com/blog/owasp-top-10-sensitive-data-exposure/

Larry Loeb (2017). Forgot Password? Man-in-the-Middle Attack Can Perform a Password Reset, Researchers Warn. Retrieved from https://securityintelligence.com/news/forgot-password-man-in-the-middle-attack-can-perform-a-password-reset-researchers-warn/

LeBlanc, J., & Messerschmidt, T. (2016). *Identity and data security for web development: Best practices* (First edition). Sebastopol, CA: O'Reilly Media.

Margaret Rouse (o. J.). Was ist Asymmetrische Kryptografie (Public-Key-Kryptografie)? - Definition von WhatIs.com. Retrieved from https://www.computerweekly.com/de/definition/Asymmetrische-Kryptografie-Public-Key-Kryptografie

Margaret Rouse (2006). Social Engineering. Retrieved from https://www.computerweekly.com/de/definition/Social-Engineering

McDonough, B. R. (2019). *Cyber smart: Five habits to protect your family, money, and identity from cyber criminals*. Indianapolis, IN: John Wiley & Sons, Inc.

Meier, J. D. (2003). *Improving Web application security: Threats and countermeasures. - "This guide was produced by the following.NET development specialists: J.D. Meier …"--P. lxi. - Includes index*. *Patterns & practices*. [Redmond, Wash.?]: Microsoft.

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (2001). *Handbook of applied cryptography* (rev. reprint with updates, 5. printing). *CRC Press series on discrete mathematics and its applications*. Boca Raton: CRC Press.

OWASP (2017). OWASP Top 10 - Broken Authentication. Retrieved from https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication

Paul Ducklin (2013). Serious Security: How to store your users' passwords safely. Retrieved from https://nakedsecurity.sophos.com/2013/11/20/serious-security-how-to-store-your-users-passwords-safely/

Peter Ray Allison (2016). The problem of passwords and how to deal with it. Retrieved from https://www.computerweekly.com/feature/The-problem-of-passwords-and-how-to-deal-with-it

Richer, J., & Sanso, A. (2017). *OAuth 2 in action*. Shelter Island, NY: Manning Publications.

Scambray, J., Liu, V., & Sima, C. (2010). *Hacking Exposed Web Applications (3rd Edition)* (3rd ed.). New York, USA: McGraw-Hill Professional Publishing.

Schneier, B. (2015). *Applied Cryptography: Protocols, Algorithms and Source Code in C*. New York: John Wiley & Sons Incorporated. Retrieved from https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4875261

Shema, M. (2012). *Hacking web apps: Detecting and preventing web application security problems*. Amsterdam, Boston: Syngress.

Sherry Hsu (2018). Session vs Token Based Authentication - Sherry Hsu - Medium. Retrieved from https://medium.com/@sherryhsu/session-vs-token-based-authentication-11a6c5ac45e4

Shin, B. (2017). *A Practical Introduction to Enterprise Network and Security Management* (First edition). Boca Raton, FL: CRC Press.

Shostack, A. (2014). *Threat modeling: Designing for security*. Indianapolis, IN: Wiley.

Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada, & Satoshi Hoshino (2010, October 28). Impact of Artificial "Gummy" Fingers on Fingerprint Systems. Retrieved from http://cryptome.org/gummy.htm

Tyra Appleby (2018). A Guide to Preventing Common Security Misconfigurations. Retrieved from https://resources.infosecinstitute.com/guide-preventing-common-security-misconfigurations/#gref

Woodward, J. D., Orlans, N. M., & Higgins, P. T. (2003). *Biometrics: Identity assurance in the information age*. New York, NY: McGraw-Hill/Osborne.