Hochschule für Telekommunikation Leipzig
University of Applied Sciences

Hochschule für Telekommunikation Leipzig (FH)
Institut für Telekommunikationsinformatik

**Abschlussarbeit zur Erlangung des akademischen Grades**
**Bachelor of Science**

| | |
|---|---|
| Thema: | „Inwiefern bietet die Authentifikation ohne Passwort Vor- und Nachteile gegenüber Webanwendungen hinsichtlich Nutzerfreundlichkeit, Datenschutz und Sicherheit?“ |


| | |
|---|---|
| Vorgelegt von: | Ertugrul Sener |
| Geboren am: | 17.10.1998 |
| Geboren in: | Berlin |


| | |
|---|---|
| Vorgelegt am: | 27. Oktober 2020 |


| | |
|---|---|
| Erstprüfer: | Prof. Dr. Erik Buchmann<br><br>Hochschule für Telekommunikation Leipzig<br>Gustav-Freytag-Straße 43-45<br>04277 Leipzig |
| Zweitprüfer: | Juri Lobov<br><br>T-Systems International GmbH<br>Holzhauser Straße 1-4<br>13509 Berlin |

# 1 Vorwort

Vor Ihnen liegt die Bachelorarbeit zur Forschungsfrage „Inwiefern bietet die Authentifikation ohne Passwort Vor- und Nachteile gegenüber Webanwendungen hinsichtlich Nutzerfreundlichkeit, Datenschutz und Sicherheit“.

Diese habe ich in Zusammenarbeit mit der Telekom Security im Rahmen meines dualen Studiums für angewandte Informatik an der Hochschule für Telekommunikation Leipzig angefertigt. Ziel ist es typische Authentifizierungsstrategien kritisch zu bewerten und eine eigene prototypische Lösung zu entwickeln. Diese soll die Anforderungen an Nutzerfreundlichkeit, Datenschutz und Sicherheit bestmöglichst erfüllen und über die selektive Auswahl aus den möglichen Authentifizierungsstrategien gleichzeitig die damit einhergehenden Nachteile minimieren.

Die Fragestellung habe ich zusammen mit meinem Hauptprüfer Prof. Dr. Erik Buchmann von der HfTL und meinem betrieblichen Vorgesetzten Juri Lobov entwickelt. Ich bin davon überzeugt, dass mir die Kombination aus theoretischem Forschungshintergrund und längjähriger praktischer Erfahrung bei einer umfangreichen Beantwortung der Forschungsfrage behiflich sein wird.

Daher möchte ich meinen Begleitern für ihre Unterstützung bei der Erarbeitung meiner Arbeit und der Betreuung danken. Ebenfalls möchte ich an dieser Stelle meinen Dank an meine Kollegen bei der Telekom Security richten, die mir jederzeit Ihre Unterstützung zugesichert haben.

Ich wünsche Ihnen viel Freude beim Lesen.

Ertugrul Sener

27. Oktober 2020

# Inhaltsverzeichnis

# 2 Einleitung

## 2.1 Motivation

Die Motivation für die Erarbeitung der Forschungsfrage ergab sich am 19. Juli 2020, als eine Rundmail des Data Breach Monitoring-Tools von Firefox die Bildschirme erreichte, welches einen neuen Incident bei Wattpad meldete [1]. Nun stellt sich die berechtigte Frage, inwiefern diese in Relation kleine Webseite verwertbare Daten für einen Angreifer oder ein bösartiges Tool liefern kann. Dazu genügt es sich die Daten genauer anzusehen. Kompromittiert wurden neben den klassichen Daten wie Passwörtern, IP-Adressen, E-Mail-Adressen und Geburtsdaten auch sehr persönliche Informationen wie geografische Standorte, ein Kurzprofil des Nutzers und die Social-Media-Profile der Nutzer [1]. Entnehmen kann man dies mehreren Artikeln, bei der die Angabe nach der Größe der Datenmengen zwischen 250 und 280 Millionen Einträgen variiert [2] [3]. Eine genaue Zahl liefert der Artikel der riskbasedsecurity mit 268.830.266 (nach der Entfernung von Duplikaten) kompromittierten E-Mail Addressen, die sich laut des Artikels in einer einzigen Datenbank befanden [4]. Diese Zahl ist dennoch nur als Richtwert zu verstehen, da das Ausmaß des Datenlecks bei Wattpad nicht in Gänze bekannt ist.

## 2.2 Problemdefinition

Mit dem Passwort gelangt der Angreifer an die Identität des Nutzers. Identitätsdiebstahl sei laut BSI ein alltägliches Phänomen, dass jeden Monat eine große Anzahl an digitalen Identitäten von Privatanwendern betrifft [5]. Es ist allgemein bekannt, dass Daten die im Internet landen kaum oder nur mit erheblichem Mehraufwand aus diesem vollständig entfernbar sind. Die Identität kann nun für missbräuchliche Zwecke entfremdet und missbraucht werden.

Auch wenn es bei der Bekanntmachung und Berichterstattung zu solchen Datenlecks manchmal so scheint, ist auch dieses nicht das einzige Beispiel für kompromittierte Unternehmen der letzten Zeit [6]. Gegen Angriffe auf die Technik (beispielsweise Code-Injektionen, 0-Day-Exploits oder Bruteforce-Angriffe) scheint die informationsaffine Menscheit der Neuzeit gut organisiert zu sein, doch dann scheitert es an der Wahl des richtigen Passwortes. Passwörter wichtiger Personen innerhalb eines Unternehmens können nun ohne große Mühen über die Google Suchergebnisse gefunden werden. So liefert die Anfrage 'allintext:password filetype:log after:2019' auf Google alle Logdateien nach 2019, die den Schlüssel "password" beeinhalten. Alternativ kann man über Anfragen folgender Art: 'wattpad data breach download' gezielt nach einem Downloadlink für spezielle Data Breaches suchen. Solche Listen werden häufig in unbekannteren Foren geteilt.Meistens erscheinen Username und Passwort im Format 'Username:Passwort', der erste Doppelpunkt trennt demnach die beiden Zeichenketten. Der Zeilenumbruch trennt die unterschiedlichen Accounts.

Die Passwörter dieser Personen sind im Regelfalle nicht im Klartext aus dem Breach entnommen worden, sondern wurden mittels Hashkillers gebruteforced. Hashkiller sind Passwortcrackingtools die für häufig genutze Hashingalgorithmen wie md5 oder sha256 Milliarden von Ergebnissen vorberechnen um bei entsprechender Suche über den Hash an den Klartext zu gelangen. Dies funktioniert besonders häufig bei sehr einfachen Passwörtern, die in diesen Tabellen sind und am Ende als Klartext dann in solchen Foren verteilt werden.

Ein Grund gegen die Nutzung von externen Geräten für die zwei Faktor Authentifizierung ist die Folge bei Verlust des Gerätes, mit dem die Webseite gekoppelt ist. Dies bedeutet den Verlust des gekoppelten Accounts. Ohne entsprechende Backup-Codes kann womöglich kein Zugang mehr gewährt werden, da das Zurücksetzen des Passwortes meist auch die Authentifizierung durch den zweiten Faktor benötigt so wie zum Beispiel bei Windows Outlook. Ein User besteht auf seine Bequemlichkeit und will so schnell wie möglich und ohne Zwischenschritte Zugang zu seinen Diensten erhalten. Ein zweiter Faktor bedeutet das nach Eingabe des Passwortes das Smartphone gesucht, womöglich noch entsperrt, die zugehörige App geöffnet und die Sicherungen (PIN, Fingerabdruck, Gesichtsmerkmale) an den Server übertragen und verarbeitet werden müssen.

Das Speichern von allen Passwörtern, durch einen Passwort-Manager, an einem gesammelten Ort muss “mit einem gut gewählten Master-Passwort geschützt sein“ [7]. Der Verlust des Masterpasswortes würde den Verlust aller Daten (und den an ihnen hängenden digitalen Identitäten) des Nutzers bedeuten. Die Frage, die sich stellt ist es, ob man mit einer Kombination aus den vorhandenen vielfältigen Authentifizierungsmöglichkeiten eine bequeme aber gleichzeitig sichere Authentifizierungsvariante schaffen kann, bei denen es dem Anwender möglich sein soll, sich gegenüber einer Webseite zu authentifizieren.

## 2.3 Stand der Forschung

### 2.3.1 Bequemlichkeitsproblem

Bei einem Angriff ist es nicht zwingend notwendig, dass die Authentifizierungsquelle, also jene Quelle bei der die Daten persistiert sind und die die Authentifizierung bei Eingabe durchführt, diese Daten durch fehlerhafte Programmierung herausgibt. Durch sogenannte Metadaten, das sind Informationen und Merkmale zu Daten, ist es Angreifern häufig möglich das Passwort zu erraten bzw. zurückzusetzen. Wenn also der Benutzername lautet ’MichaelJacksonFanForever’ ist die Antwort auf die Sicherheitsfrage zum Lieblingskünstler nicht weit entfernt.

Gleichzeitig neigen Menschen aufgrund von Bequemlichkeit zu leicht merkbaren Passwörtern, die sie mehrfach für verschiedene Dienste verwenden. Das die Passwortproblematik allgegenwärtig ist, stützt eine durchgeführte repräsentative Studie der Bitkom Research [8] im Auftrag des Digitalverbands Bitkom. So nutze etwa jeder dritte Onlinenutzer (36%) dasselbe Passwort für mehrere Dienste. Auch wenn gleichzeitig 63% der Befragten angaben, bei der Erstellung von Passwörtern auf “einen Mix aus Buchstaben, Zahlen und Sonderzeichen” [8] zu achten. Eine ähnliche Studie hat die Bitkom zum Thema ’Nachlässigkeit bei Passwörtern’ am 08.11.2016 [9] gemacht, bei der die prozentuale Verteilung an unsicheren Passwortnutzern die befragt wurden nur einen Prozent höher liegt. Das heißt konkret, dass sich innerhalb von 4 Jahren keine messbare Besserung ergeben hat. Das Bewusstsein über die Internetpräsenz und der Schutz dessen scheinen immernoch keine große Aufmerksamkeit vom modernen Nutzer zu erhalten. Das Problem mit unsicheren Passwörtern ist allerdings so alt wie das Internet.

### 2.3.2 Unsichere Passwörter

Der Begriff des Passwortes entstammt dem militärischen Bereich des 16. Jahrhunderts, wobei tatsächlich das einzelne Wort adressiert war, welches einem Zutritt zu Gebäuden verschaffte. Damit verwandt ist das Kennwort, welches nicht das Passieren sondern die Kennung des gemeinsamen Geheimnisses betont. Dies bedeutet, dass der Passierer mit einem Kennwort auch automatisch ein Geheimnisträger ist. Als Computer immer leistungsfähiger wurden, wurde der Begriff der Passphrase etabliert, um die Notwendigkeit längerer Passwörter hervorzuheben. Weitere Schlüsselwörter für das heutzutage bekannte Passwort sind: Schlüsselwort, Kodewort (Codewort) oder die Parole, diese werden im Folgenden als Synonym für die heutige Passphrase verwendet.

User verwenden häufig das selbe leicht zu erratende Passwort für mehrere Accounts auf unterschiedlichen Diensten [10]. Die Länge entscheidet mitunter über die Sicherheit von Passwörtern. Allgemein gillt: je länger desto besser [11]. Dies muss allerdings nicht grundsätzlich der Fall sein, denn laut BSI könne ein starkes Passwort kürzer und komplex oder lang und weniger komplex sein [11]. Anhand sogenannter Passwort-Policies, das sind Regeln für Passwörter, machen die Webseiten das Wählen von Passwörtern komplizierter. Dabei umfasst die Password Policy gängige Festlegungen wie die Mindestlänge, die Komplexität und die Lebensdauer [10]. Außerdem gibt es die Möglichkeit des Verbot von sich wiederholenden Zeichenketten wie ”testtest“, längeren Zahlenreihenwiederholungen wie ”123123“ oder dem Verbot der Nutzung des Usernamen im Passwort. Das gewählte Passwort soll für einen selbst leicht merkbar, für einen Computer oder menschlichen Angreifer schwer zu erraten sein.

Ferner sind der Kreativität von Passwörtern laut Bundesamt für Sicherheit in der Informationstechnik (BSI) keine Grenzen gesetzt [11]. Zum Beispiel könne man einen leicht zu merkenden Satz nehmen, diesen mit Bindestrichen verbinden und von jedem Wort den ersten Buchstaben entfernen. Die Frage die sich dabei stellt ist es, ob dieser Satz dann die Tippgeschwindigkeit des Nutzers beeinträchtigt, weil relativ viele Denkprozesse während des Tippens stattfinden müssen. Zunächst ein Mal müsste sich hierbei der Satz in voller länge gemerkt werden, dies ist noch Recht unkompliziert für den allgemeinen Internetnutzer. Danach muss der Satz während des Tippens bereits mit Bindestrichen verbunden werden, auch dies ist noch kein großen Problem. Zum Problem wird es, sich die ersten Buchstaben beim Tippen automatisch wegzudenken, sodass einem kein Fehler unterläuft. Wenn einem doch ein Fehler unterläuft, ist es anders als bei gängigen Passwörtern, sehr schwer zur Fehlerquelle zu springen und den Fehler zu lokalisieren.

Alternativ bleibt einem nur das Neutippen des Passwortes, welches eine große Zeitverzögerung für den Nutzer bedeutet und auf lange Sicht den Nutzer dazu bringen wird, ein einfacheres und leicht tippbares Passwort zu wählen.

Webseiten nutzen typischerweise standardisierte und gleiche Regeln für jeden Nutzer, doch ignorieren den Fakt, dass jeder Nutzer andere Präferenzen besitzt. Dies macht es schwierig, die Ziele dieser 'strong password goals' zu erreichen. Um die Effektivität für diese Methoden zu steigern, wird eine dynamic personalized password policy (DPPP) empfohlen. Diese kann auf den Charakterzügen des Nutzers basierend personalisierte Password Policies vorschlagen. [12] Diese passen sich dem Nutzer an und lösen das Problem der Usability im Bereich der Password Policies.

Ferner können Passwort Policys wertvolle Informationen für einen potenziellen Angreifer bieten. Denn was Angreifer durch sehr strikte Passwort Policys unteranderem erkennen können, ist die Mindest- und Maximalzeichenlänge. Dabei wird der Angreifer zum Beispiel aus der Regel 'Das Passwort muss mindestens 8 und maximal 16 Zeichen lang sein.' alle Kombinationen für weniger als 8 Zeichen und mehr als 16 Zeichen bei der Erratung eliminieren können. Weitere Regeln wie 'Das Passwort muss mindestens ein Sonderzeichen beinhalten' können zusätzliche Informationen bieten. Daher sind Passwort-Policys zwar ein sehr wichtiges Werkzeug, um Nutzer zu sicheren Passwörtern zu zwingen. Durch die beeinträchtigte Bequemlichkeit in der freien Passwortwahl des Nutzers können dennoch einfach zu erratende Passwörter gewählt werden, die den Kriterien entsprechen. Verhindern lässt sich dies nicht ganz. Die Informationen die Nutzer beim Anmelden bekommen, nutzen Angreifer dann zum Knacken jener Passwörter. Aus Sicht des Angreifers lautet die Faustregel: Je mehr Metadaten, desto besser.

### 2.3.3 Übertragungsproblem

Das Problem mit der Unsicherheit von Passphrasen oder Passwörtern beginnt jedes Mal aufs Neue, sobald man ein Passwort eintippt und ist nicht mit der Wahl eines mathematisch sicheren Passwortes gebannt [13]. Das Verfahren der Passwortauthentifikation bedient sich prinzipiell einer Zeichenkette, die man in ein Feld eintippt und ist per se dann unsicher sobald einer der Geheimnisträger (Menschen mit Kenntniss über die Parole) kompromittiert bzw. infiziert ist. So gibt es verschiedenste Angriffsvektoren um das Passwort eines Users für einen speziellen Dienst herrauszufinden. Von personalisierten (oder auch allgemeinen) Phishing Mails, zu Shoulder Surfing bis hin zu Trojanern und Keyloggern [11] auf dem System Desjenigen, der es zum Zeitpunkt des Angriffs in die Tastatur tippt. Ein weiteres großes Problem ist die Übertragung von Passwörtern über die klassiche User-Browser-Schnittstelle. Dabei wird das Passwort im Browser des Clienten gehasht und dann an den Server übertragen. Die sichere Kommunikation anhand des Hypertext Transfer Protocol Secure (HTTPS) findet erst bei der Übertragung zum Server statt, die Eingabe des Passwortes an den Browser ist ungeschützt. Diese Übertragung von Buchstaben kann mitgelesen werden.

Das Hashen der Zeichenkette kann in einem ungesicherten Netz sicherlich hilfreich sein, da nun ein Angreifer innerhalb des Netzes das Passwort im Klartext nicht lesen kann. Alles was jener Angreifer sieht, ist der Hash, der auf Clientseite erzeugt wurde und im Request steht. Die Nutzung von HTTPS macht es zudem noch schwieriger für Angreifer Kommunikationen zwischen User und Server mitzulesen und infolge dessen zu manipulieren. Bei sicheren Verbindungen ist mitunter das Hashing der Nutzerdaten demnach theoretisch nicht mehr nötig. In der Praxis kommt es häufiger vor, dass Zertifikate auf dem Browser des Users installiert werden, die vom Angreifer selbst ausgestellt wurden. Der Durchschnittsnutzer achtet nicht darauf, wer das Zertifikat ausgestellt hat und bekommt womöglich nichts von einem Angriff mit. Solche Zertifikate können in Massen durch eine kompromittierte Root-CA (Certificate Authority) erstellt werden. Root-CA's stellen im wesentlichen Zertifikate aus und prüfen die Identität (und Angaben) des Anforderers eines Zertifikats. Dabei kann es Monate oder sogar Jahre dauern bis gefälschte Zertifikate von einem Nutzer zufällig erkannt, gemeldet und im Anschluss zurückgezogen werden. Ein solcher Man-in-the-middle (MITM) - Angriff auf SSL-Verbindungen ist sehr typisch und wird im folgenden an einer verschlüsselten Verbindung zu einem Webshop sequenziell erläutert.

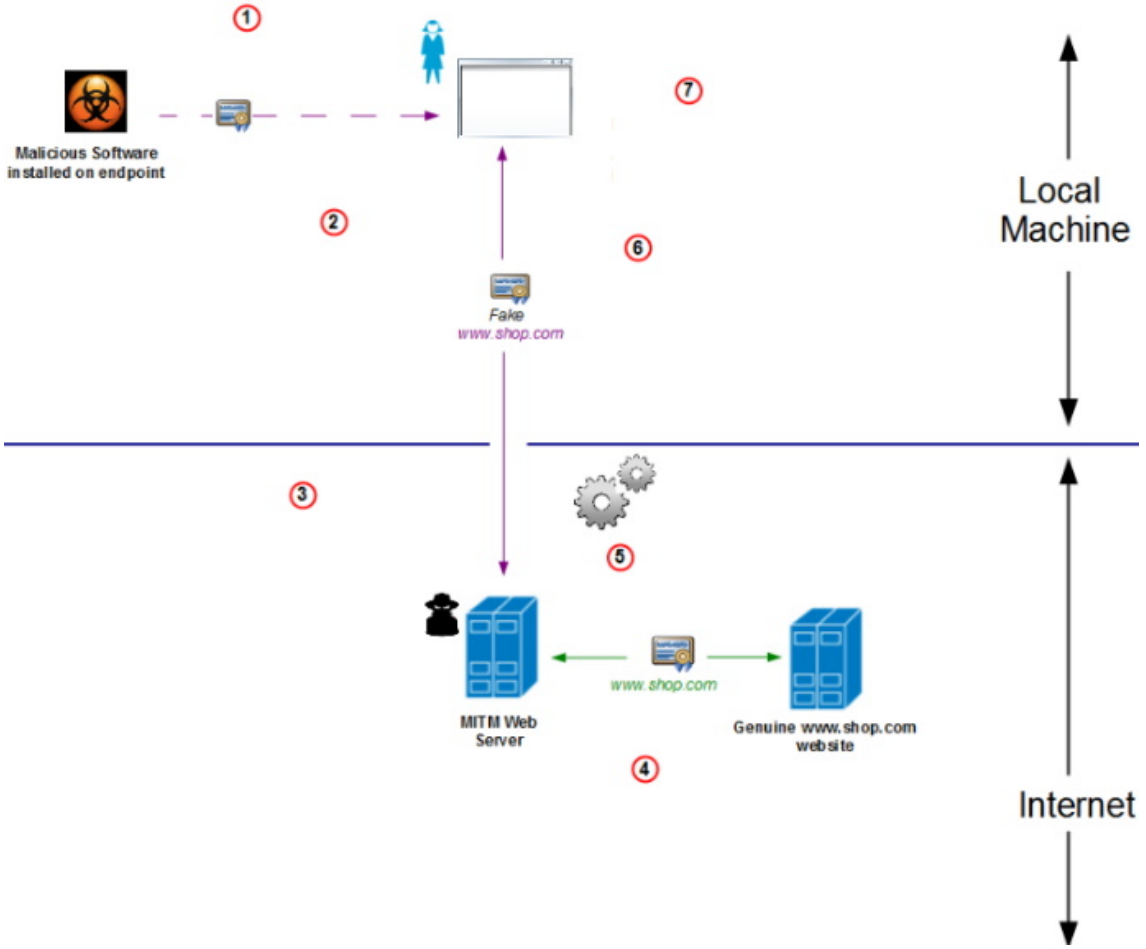



Abbildung 2.1: MITM - Angriff auf SSL verschlüsselte Verbindungen

1. Im ersten Schritt muss der Angreifer ein eigenes Fake-Zertifikat (im folgenden MITM-Zertifikat genannt) im Zertifikatsspeicher des Nutzers installieren. Dies kann eine Malware oder sonstige Fremdsoftware machen. Hat man als Angreifer phsysischen Zugriff zum Rechner, kann man dies innerhalb von wenigen Sekunden selbst machen, ohne dass der Nutzer davon etwas mitbekommt. Der MITM - Angriff basiert auf diesem MITM-Zertifikat, das von einem potenziellen Angreifer im Namen einer Webseite fälschlicherweise ausgestellt wurde. Ohne diesen Schritt funktioniert der Angriff nicht.

2. Der User möchte eine Verbindung zum Webshop (im Schaubild www.shop.com) aufbauen. Hierzu ruft er die Seite mit dem Präfix https:// auf im Glauben, eine sichere Verbindung aufbauen zu können, die von keinem Dritten mitgelesen werden kann. Der MITM fängt die Anfrage ab und leitet ihn zu seinem eigenen Webserver weiter.

3. Der Webserver erkennt einen HTTPS - Request und entschlüsselt mit dem eigenen privaten Schlüssel. Dies kann er durch die Installation der Fake-Root-CA in Schritt 1. Dort wurde die Nachricht mit dem öffentlichen Schlüssel des Angreifers verschlüsselt und abgesendet.

4. Simultan baut der MITM-Webserver eine Verbindung zum echten Webshop auf und leitet die Daten des Nutzers jeweils in beide Richtungen weiter. Dabei simuliert er eine gewöhnliche Nutzeranfrage, die der Webshop nicht als fälschlich erkennen kann. Der User hingegen merkt nichts von dem Angreifer, da er die gewohnte (dies ist Standard) Webseite angezeigt bekommt und keine Annomalien erkennt.

5. Der gesamte Traffic, der eigentlich verschlüsselt sein sollte, kann vom MITM mitgelesen, weitergeleitet oder manipuliert werden. Die Verbindung zwischen User und dem Webshop ist nicht sicher trotz HTTPS Anzeige im Browser.

6. Die Response vom Webshop wird mit dem erstellten MITM-Zertifikat für www.shop.com verschlüsselt und an den User übertragen.

7. Der Browser markiert die Verbindung als sicher und unbedenklich.

Der Nachteil für den Angreifer in diesem Szenario ist die zeitlich limitierte Möglichkeit eines Angriffs. Es muss also auf den initialen Request des Users gewartet werden, um dessen Daten abzufangen und den Prozess des Angriffs in Gang zu setzen. Wiederholungsangriffe (auch: Replay-Attacks) werden in diesem Szenario nicht verhindert. Der Angreifer muss womöglich demnach nicht einmal das Passwort im Klartext besitzen. Es genügt, den Hash und den Benutzernamen im Request abzufangen um diese dann in einem seperaten Aufruf vom eigenen Rechner an die selbe Uniform Resource Locator (URL) zu senden. Es handelt sich bei dieser Art des Angriffs um das Imitieren von Benutzereingaben durch einen Angreifer, bei der der Angreifer das Geheimnis nicht im Klartext kennt.

Durch den Zugang zum Dienst ist es dem Angreifer somit (je nach Implementierung) möglich, sensible Daten des Nutzers einzusehen, die nicht für ihn bestimmt sind. Mit der Identität des Nutzers kann ein Angreifer den Nutzer nun zum Beispiel erpressen um an sensible Daten oder Geld des Nutzers zu kommen.

### 2.3.4 Alternative Methoden

Neben den bereits etablierten Methoden beschäftigt sich die Forschung mit alternativen Authentifizierungsverfahren wie der 'negative authentication' und der 'adaptive mult-factor authentication'. Was diese Verfahren für die Forschung bedeuten und inwiefern sie einen Vorteil bieten wird im Folgenden erläutert.

**Negative Authentication (NA)**

Bei der Negative authentication (NAS) wird die Useranfrage auf Invalidität statt auf Validität geprüft. Die Idee dahinter basiert auf dem 'Negative Selection Algorithm', welcher das menschliche Immunsystem als Insipriator verwendet. T - Helferzellen dienen hierbei der Unterscheidung von Entitäten im Körper nach 'körpereigen' (self) und 'körperfremd' (non-self). [14]

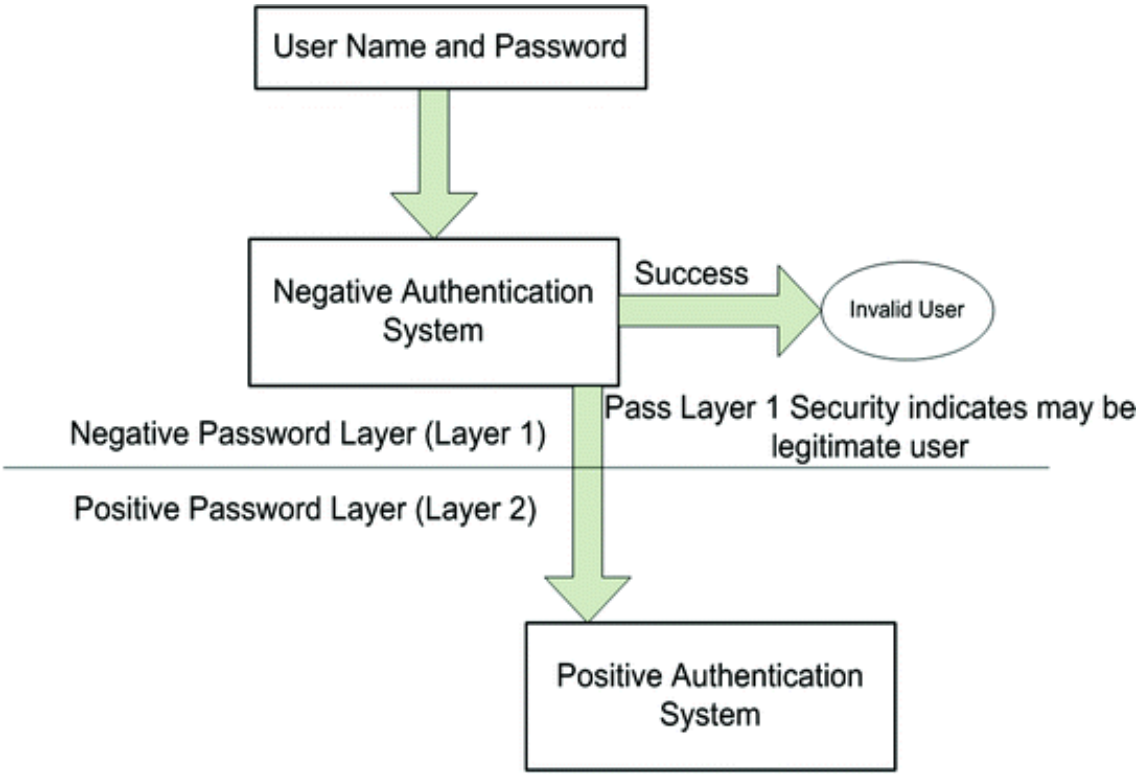



Abbildung 2.2: Architektur zur Authentifizierung mit einem 'Negative Authentication System'

In der self-region sind die Passwortdaten beherrbergt. Diese Daten werden von der weitaus größeren non-self-region 'umschlossen', in der sich die Anti-Passwort-Detectoren befinden. Während man NAS verwendet, ist es angebracht die Passwort-Profile und die Anti-Passwort-Profile auf zwei voneinander getrennten Servern zu

bewahren. Dabei finden wie auf der Abbildung 2.2 zu sehen zwei Authentifizierungen statt. Bei der Ersten wird der Request mit der Datenmenge der Anti-Passwörter verglichen. Gibt es ein Match, ist der eingeggangene Request invalide. Auf diesem liegen die Detektoren für die Anti-Passwörter, die bei der Authentifizierung die Gülltigkeit der Anfrage erkennen, ähnlich wie die T-Helferzellen die Gülltigkeit von anderen Zellen im menschlichen Körper erkennen.

Dieser ”getrennte Server Ansatz“ für die Negativauthentikation besitzt gewisse Vorteile zu der Positivauthentifikation. Zunächst ein Mal ist es in der Lage bereits vor der eigentlichen Prüfung nach der Korrektheit des Passwortes invalide Requests abzufangen, die den zweiten Layer nie erreichen. Dies hat sich bereits als zielbringend erwiesen [15]. Außerdem sind die Daten des Nutzers hinter einem von außen nicht zu erreichenden Server versteckt, der nur mit dem ersten Layer kommuniziert. 'Guessing attacks' also Angriffe, die auf der Erratung von Passwörtern basieren, werden damit zu hoher Wahrscheinlichkeit verhindert [14].

**Adaptive multi-factor-authentication (A-MFA)**

Da Nutzer durch mobile Technologien immer mehr Zugriff auf Online Services erhielten, entwickelte sich der Stand der Forschung immer mehr in Richtung verschiedener Authentifizierungsverfahren, unter denen der User eine Auswahl für sich treffen muss um seine Identität zu bestätigen [14]. Die Auswahl an Authentifizierungsfaktoren die dem Nutzer von Anwendungen präsentiert werden, entscheidet letzlich über die Performance von MFA's [14]. Daraus resultiert, dass die zufällige (oder Selbe) Auswahl von Verfahren für die MFA die Vertrauenswürdigkeit beeinträchtigen, da die Auswahl erratbar wird und bei entsprechenden Exploits auch unsicher.

Die Auswahl der Verfahren findet zunächst aufgrund der Möglichkeiten eines Nutzers statt, demnach wird ermittelt welche Umgebung (welches Betriebssystem, welches Kommunikationsmedium und historische Daten) der Nutzer verwendet [16]. Um eine sichere Lösung für eine adaptive MFA zu entwickeln, muss die Erratbarkeit von dem Set an Authentifikationsmöglichkeiten verhindert werden. Dies bedeutet, dass es nicht möglich sein darf vordefinierte Muster für einen speziellen Nutzer mit einer speziellen Umgebung vorrauszusagen [**11**].

## 2.4 Zielsetzung

Passwörter können der allumfassenden sicheren Authentifikation nicht mehr gerecht werden. Es braucht weitere oder gänzlich andere Methoden, um die Identität eines Nutzers zu schützen. Die Menschheit benötigt einfache und unkomplizierte Methoden, die ihnen nicht das Gefühl geben allumfängliches technisches Know-How besitzen zu müssen, um die eigenen Daten im Internet zu schützen. Es sollte zum konsens werden, dass es bequeme Mechanismen für jeden Nutzer gibt. Je nach Gewichtung der Relevanz der Daten in den Kriterien Sicherheit, Bequemlichkeit und Datenschutz gibt es Möglichkeiten dies zu bewerkstelligen. Dies muss sowohl den Dienstbetreibern als auch dem Endnutzer bewusst werden. Nur die Erkenntnis über ein vorhandenes Problem kann zur Besserung führen und ist der erste Schritt. Nur so kann das Passwortproblem ein für alle Mal gelöst werden.

Ziel dieser Abschlussarbeit ist es eine Möglichkeit der Authentifikation zu spezifizieren, die die Voraussetzungen für eine sichere, bequeme und datenarme Authentifikation erfüllt. Ziel ist es auch, den Leser in die Sicht des Angreifers auf Systeme einzuweisen, sodass im Idealfall automatische Schutzreaktionen wie das Wählen von sicheren Passwörtern hervorgerufen, wenn nicht sogar eine der beschriebenen FIDO2 Verfahren wie der erste oder sogar der zweite Faktor, verwendet werden. Der Prototyp soll die verschiedenen Authentifizierungsmöglichkeiten veranschaulischen und präsentieren, um dem Nutzer die Wahl auf eines der Verfahren zu erleichtern. Dabei ist der Prototyp ein Modul (die Authentifikation) einer vollständigen Webseite, doch wird in der folgenden Arbeit als vollständige Webseite behandelt. Gleichzeitig ist eines der Hauptziele dieser Arbeit auch die Grenzen von 'modernen' Authentifizierungsverfahren aufzuzeigen und auf dessen Nachteile hinzuweisen. Inwiefern eine Kombination dieser vorhandenen Verfahren, Probleme löst und welche minderen Probleme dabei entstehen soll betrachtet werden.

Dennoch sollte erwähnt sein, dass diese Arbeit nicht darauf abzielt jede mögliche vorhandene Authentifikationsstrategie zu durchleuchten. Erläutert werden die klassiche User-ID und Passwort Authentifikation und dem Gegenüber stehen die meistgenutzen Verfahren, die meist eher auf private Schlüssel, Besitz oder Biometrie setzen. Auch ist es ein Nicht-Ziel dieser Arbeit das Resultat auf eine einzige perfekte Lösung zu dezimieren und diese zum neuen Standard zu erklären. Viel mehr soll es dem Leser durch eine tabellarische Auflistung von Vor- und Nachteilen jeder Methodik selbst überlassen sein, welche Authentifikationsmethode ausgewählt wird.

# 3 Grundlagen

Laut des IT-Grundschutz-Kompendiums vom BSI könne ein Benutzer aus Bequemlichkeit oder pragmatischen Gründen bewusst auf komplizierte und unhandliche Kryptomodule verzichten und Informationen stattdessen im Klartext übertragen. [17] Dies stellt ein hohes Sicherheitsrisiko für Unternehmen, aber auch Privatpersonen dar, da Benutzer nicht gewillt sind ihre Passwörter durch komplizierte Verfahren zu erzeugen und in regelmäßigen Abständen vollkommen randomisiert zu setzen. An neuartige Ansätze des Logins in nativen oder webbasierten Anwendungen stellen sich dadurch völlig neue Herausforderungen. So müssen neue Authentifizierungsmöglichkeiten nicht nur sicher sein, sondern auch komfortabel genutzt und bedient werden können, da sie sonst von den Endnutzern gemieden oder umgangen werden. Wichtig ist auch, dass die breite Masse Zugriff auf die Ressourcen hat, die es zur Nutzung dieser Verfahren braucht. Man denke nur an die ganzen betrieblichen Passwörter, bei denen zum Monatsende nur eine Zahl an der letzten Stelle des Passwortes geändert wird. Laut einer Statistik von 2019, der “Global Data Risk Report From the Varonis Data Lab”, gaben 38% aller Nutzer an ein Passwort im Unternehmen zu nutzen, das sie nicht (oder nur geringfügig) ändern. Außerdem wird laut dieser Statistik alle 364 Tage wahrscheinlich ein Data Breach aufgrund von unsicheren Passwörtern in einem mittelständigen Unternehmen stattfinden. Die monatliche Ablaufzeit von Passwörtern in Unternehmen scheint also nicht ganz den Effekt zu erzielen, der ursprünglich damit geplant war, da die Arbeiter die Bequemlichkeit über die Sicherheit stellen.

## 3.1 Standard nach FIDO2

Die genannten Probleme sind den Menschen seit einigen Jahren bekannt und wurden vor allem mit Verfahren gelöst, die keine Passwörter (oder allgemein Verfahren der Wissenskategorie) zur Authentifikation benötigen und diese maximal als ersten Layer der Sicherheit implementieren. Das bekannteste Beispiel für einen Standard zur sicheren und bequemen Authentifikation im Internet findet man unter dem Schlüsselwort FIDO2. FIDO steht dabei für ’Fast Identity Online’ (Schnelle Identität im Netz). Sie ist das Ergebnis einer Kooperation des World Wide Web Consortium (W3C) und der FIDO Alliance.

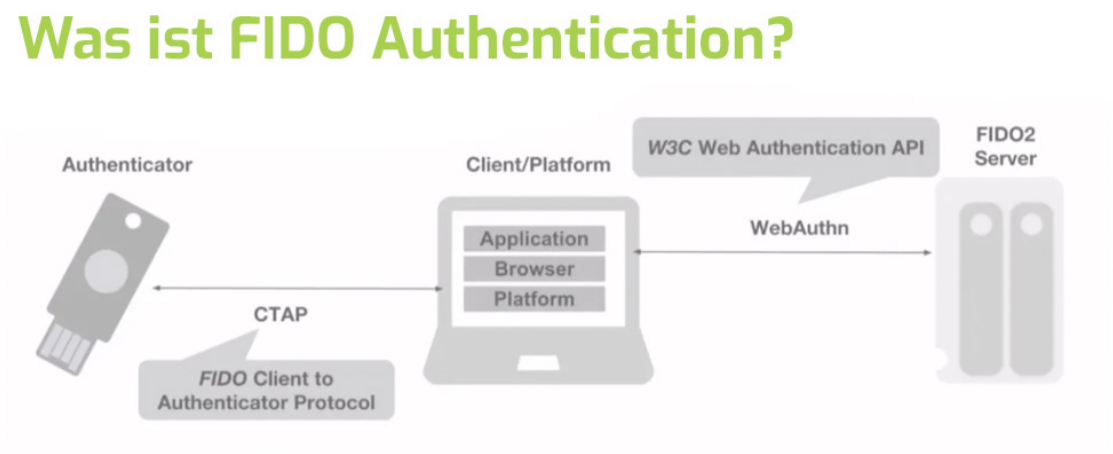



Abbildung 3.1: Bestandteile der FIDO-Authentifikation

Grundsätzlich besteht der Standard aus folgenden Komponenten:

- **Authenticator**

  Der Authenticator kann, wie im Schaubild von der offiziellen Seite der FIDO Alliance zur Frage 'Was ist die FIDO Authentifikation' bereits gezeigt, ein sogenannter "FIDO2-USB" sein. Also ein USB mit der Möglichkeit zur passwortlosen Authentifizierung des Nutzers im Netz. Eine weitere bekannte Möglichkeit für einen Authenticator ist Windows Hello (bestehend aus Hello-PIN, Hello-Fingerabdruck und Hello-Gesichtsmuster) oder Plat für Betriebssysteme vom Hersteller Apple.

  So können also sowohl externe Geräte (mit eingebautem Schlüssel) sowie interne Geräte (sogenannte self-signed-devices, die selbst Schlüssel bei der Registration erzeugen) des Betriebssystems bzw. Rechners zur Authentifikation genutzt werden.

- **CTAP - Client to Authenticator Protocol**

  Das Client-to-Authenticator protocol (CTAP) bzw. genauer CTAP2 ist das Nachfolgeprotokoll zu Universal second factor (U2F) welches mit der Einführung von FIDO2 in CTAP1 umbenannt wurde. Mit FIDO2 wurde die Kommunikation zwischen dem Authenticator und der Plattform (dem Browser, der Applikation) standardisiert. Das Protokoll definiert, dass der Server eine Initialanfrage an den User sendet. Dieser erstellt mittels des Authenticators ein Key-Pair, also einen öffentlichen und einen privaten Schlüssel und übermittelt den öffentlichen Schlüssel an den Server. Der Server persistiert diesen um zukünftige Anfragen des Nutzers zu identifizieren und den Nutzer zu authorisieren. Viele Schlüssel erfordern vor diesem Prozess eine Userverifikation, die zum externen Authenticator (dem USB - Stick) noch eine PIN verlangt, die beim Initiieren gesetzt wurde.

- **Client/Platform**

  Zwischen dem FIDO2-Server und dem Authenticator befindet sich der Client, hier interagiert der Nutzer mit dem Authenticator und verfiziert sich ihm Gegenüber. Über dessen Browser wird dann (über eine durch Webauthn erzwungene sichere Verbindung) ein Request an den FIDO2 - Server gestellt. Über den Client kommunizieren Server und Authenticator. Wichtig für alle diese Vorgänge ist, dass der private Schlüssel bei keiner Anfrage den Client verlässt. Er ist stets sicher auf dessen Rechner (im Trust Store unter Windows, je nach Betriebssystem variierend) lokalisiert.

- **Webauthn**

  Zwischen dem Server und der Plattform befindet sich das WebAuthn - Protokoll, welches in folgenden Kapiteln näher erläutert wird. Das Protokoll basiert auf zwei Hauptbestandteilen: Einem Registrier und einem künftigen Loginvorgang. Das Protokoll vermittelt dem Authenticator unter anderem die Optionen, die der Server für die Authentifikation erlaubt. So kann der Server zum Beispiel externe USB-Sticks verbieten oder eine ausdrückliche Authentifikation mittels NFC erzwingen. Wichtig ist, dass Webauthn keinerlei Details über den User freigibt. Ferner weiß weder der Server noch das Protokoll Webauthn welche der eigentlichen Authentifikationsmethoden vom User genutzt wurden. Webauthn gibt im ersten Schritt dem Nutzer die Optionen bei der Registration und kann im folgenden nur anhand des publicKey's und der credentialID erkennen, dass sich das Selbe gerät authorisieren möchte.

- **FIDO2 - Server**

  Ein FIDO2-Server ist ein durchschnittlischer Server auf dem typischerweise ein Javascript-Framework läuft, das als Backend für die Anfragen fungiert. Der Server verwaltet die öffentlichen Schlüssel des Nutzers und sorgt für die Verifikation der Anfragen für den Registrierungs und Loginprozess. Ferner wird bei sämtlichen Serverprozessen eine 'Relying Party Id' angegeben, welches die IP-Addresse des Servers ist. Teilweise findet man den Server in Abbildungen der FIDO - Alliance auch als 'Relying Party Server'. Erwähnt werden sollte allerdings, dass Client und Server sich die Authentifikation des Users bzw. die Logik hierzu teilen. Es findet im Vorhinein meist eine Userverifikation statt, bevor der Server über einen Loginversuch Kenntnis erlangt. Erst im Anschluss werden verschiedene Challenge-Response Anfragen zwischen Server und Client ausgetauscht.

Unterschieden werden die einmalige Registrierung eines Gerätes für einen Nutzer und die folgende Authentifikation (’der Login’) des Nutzers. Dessen Schritte werden im folgenden dargelegt.

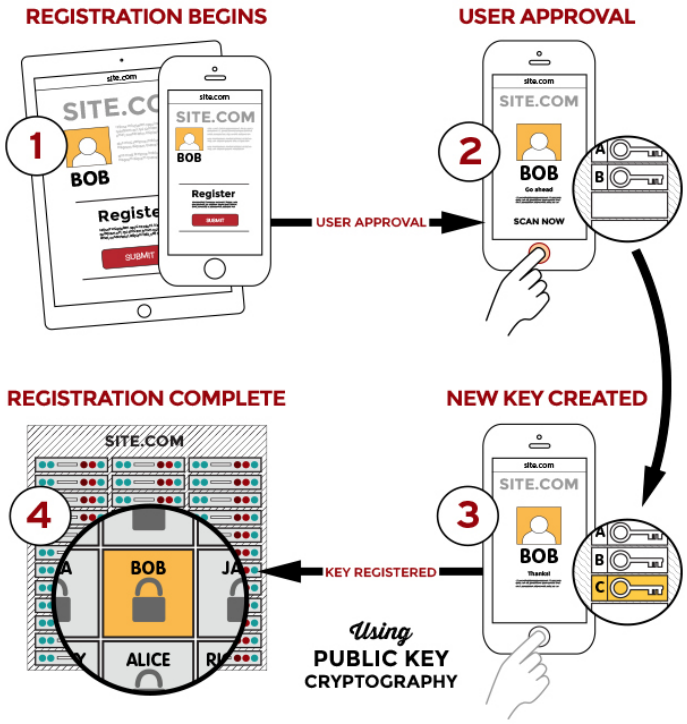



Abbildung 3.2: Userregistration nach FIDO2 mittels Webauthn & CTAP2

1. Der User wählt einen vom Server akzeptierten Authenticator aus. Hier ist es der Fingerabdruck. Auch möglich gewesen wäre ein gesprochenes Wort ins Mikrofon des Nutzers oder ein PIN sowie vieles mehr. Je nach Gerät (und Betriebssystem) können die möglichen Authentifizierungsmechanismen variieren.

2. In der Phase der Userverifikation entsperrt der Nutzer den Authenticator durch seinen Fingerabdruck.

3. Infolge dessen wird im dritten Schritt ein Paar aus privatem und öffentlichen Schlüssel erstellt. Der private Schlüssel bleibt auf dem Gerät des Endnutzers, während der öffentliche Schlüssel an den Server übermittelt wird.

4. Dort wird dieser mit dem Useraccount verknüpft und persistiert um folgende Loginanfragen des Nutzers authentifizieren zu können. Der initiale Nutzer, der die Registrierung angestoßen hat, kann damit erkannt und effektiv authentisiert werden.

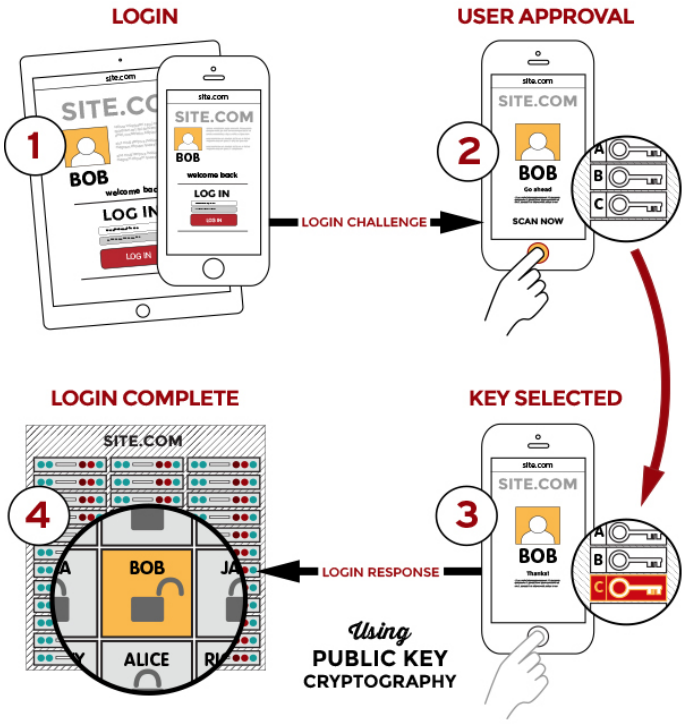



Abbildung 3.3: Userauthentifikation nach FIDO2 mittels Webauthn & CTAP2

1. Bei dem Loginvorgang wird der Nutzer dazu aufgefordert eine Challenge zu lösen.

2. Die Geräteauswahlmöglichkeiten begrenzen sich hierbei auf Diejenige, die zuvor registriert wurden. Wenn der User also sowohl die PIN als auch die Gesichtserkennung eingerichtet hat, kann er sich zwischen den beiden Varianten entscheiden. Dieser Schritt ist identisch zum zweiten Schritt der Registration unter FIDO.

3. Die Challenge des Servers wird mit dem privaten Schlüssel des Nutzers signiert und infolge dessen an den Server übermittelt.

4. Der Server entschlüsselt diese mit dem gespeicherten öffentlichen Schlüssel aus der Registration. Der Nutzer erhält zum Ende der Authentifizierung eine zugehörige Response, die ihm den Login ermöglicht. Die Authentifikation des Nutzers ist damit abgeschlossen.

Zusammengefasst basiert FIDO2 auf vorhandenen Protokollen wie Web Authentication (WebAuthn) für die Browser-Server-Kommunikation und CTAP für die Browser-Authenticator-Kommunikation. Die Yubico, einem der Hauptentwickler des heutigen CTAP1 - Protokolls, fasst FIDO2 wie folgt zusammen: ”an open authentication standard that enables internet users to securely access any number of online services with one single security key [...]. FIDO2 is the latest generation of the U2F protocol“ [5] (übersetzt: Ein offener Authentifikationsstandard der Usern das Erreichen von Online-Services mittels eines einzigen sicheren Schlüssels erlaubt. [...] FIDO2 ist die letzte Generation des U2F - Protokolls). Während das Vorgänger - Protokoll U2F von Google und Yubico ins Leben gerufen wurde, ist FIDO2 ein offener dezentraler Kommunikationsstandart für die passwortlose Kommunikation welches die Authentifizierung für sowohl Privatnutzer als auch Unternehmen beqeuem und gleichzeitig sicher machen soll. Inwiefern die neuen Ansätze nach FIDO2 nun sicherer bzw. bequemer sind, wird im nächsten Kapitel erläutert.

## 3.2 Behandelte Verfahren

### 3.2.1 Username & Passwort

Trotz der bereits erwähnten Probleme des Passwortes, ist es laut eines wissenschaftlichen Artikels von Thomas Maus 2008 “nicht aus unserer Arbeitswelt [...] wegzudenken” [18]. Anzumerken ist, dass der Artikel bereits 12 Jahre in der Vergangenheit liegt und immer noch Relevanz hat. Herr Maus beschreibt alternative Authentifikationsmethoden wie die Einmalkennwörter und biologische Merkmale. Dies ist nur ein weiterer Beweis dafür, dass schon vor mehr als 10 Jahren die Passwortproblematik erkannt wurde. In dem Artikel geht Herr Maus der Hypothese nach, ob Passwörter wirklich per se unsicherer sind als die Authentifikationsmethoden der beiden anderen Kategorien Besitz und biologische Merkmale. So sei das Kernproblem des Passwortes, dass es direkt nach der Eingabe ein geteiltes Geheimnis ist, da alle beteiligten Systeme es mitschneiden konnten und automatisch Geheimnisträger sind. Das Passwort biete demnach sehr viele Angriffsvektoren, so “Shoulder Sourfing, Phishing, Social-Engineering, Man-in-the-Middle Angriffe, schlechte Passwort Qualitäten, das Teilen des Passwortes mit Familie und Freunden” [18] und vielem mehr.

Bei dem Prototyp wird der Ist-Zustand einer Username und Passwort - Authentifikation demonstriert, wie man sie heutzutage auf höchstwahrscheinlich jedem Webdienst vorfinden wird. Teilweise wird sie sogar als einziger Authentifizierungsfaktor verwendet. Während man über alle Schwächen des Passwortes spricht, muss man dennoch anerkennen, dass das Passwort eine gewisse Flexibilität bietet. Es genügt das Wissen über eine bestimmte Zeichenfolge um sich zu authentifizieren, dieses Wissen muss nicht zwangsmäßig auf ein Blatt geschrieben oder in einem Passwort-Manager beherrbergt werden. Das beste Passwort ist jenes, welches nur in dem Gehirn des Nutzers persistiert ist und mit niemandem geteilt wird. Anders als bei neueren Verfahren, die als sicherer gelten, benötigt es keine Smart-Card, USB-Sticks oder anderweitige technische Hilfsmittel um sich zu authentifizieren. Lediglich das Gedächtnis des Nutzers ist gefragt, welches allerdings fortlaufend im digitalen Zeitalter von Social Media und Up-To-Date Informationen, nachlässt. So ist es Usern heutzutage immer schwerer möglich sich die verschiedenen Passphrasen zu merken, ohne sie als Behelf irgendwo zu notieren. Sobald das Passwort auf eine potenziell unsichere Plattform notiert wurde, gilt sie als geteiltes Geheimnis zwischen jedem Leser und dem Initialbenutzer dessen.

### 3.2.2 TOTP

Ein One time password (OTP) kann im Vergleich zu gewöhnlichen Kennwörtern nur ein einziges Mal für eine Authentifizierung genutzt werden. Man unterscheidet drei Arten von OTP's.

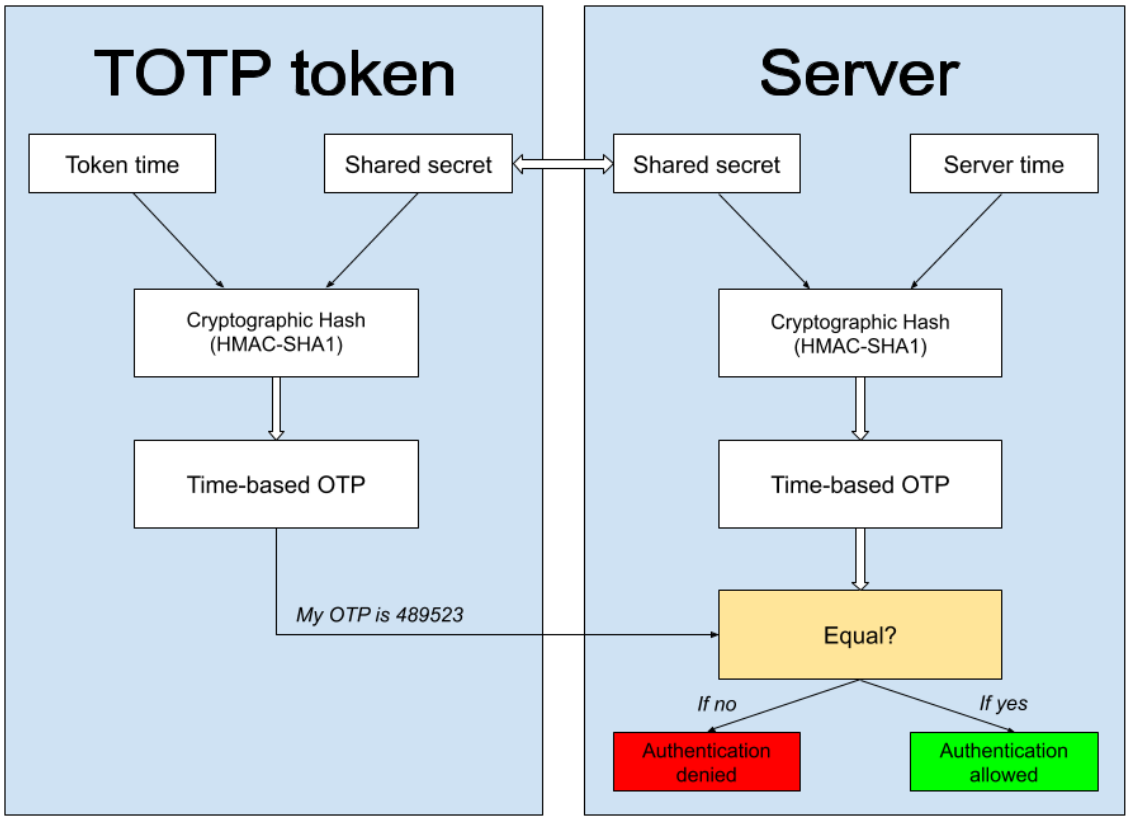



Abbildung 3.4: Authentifikation durch TOTP

Bei der timerbasierten (Time based one time password (TOTP)) Methode wird die aktuelle Systemzeit (Token time) mit dem Geheimnis (Shared secret) anhand eines kryptografischen Verfahrens verschlüsselt. Es entsteht ein kryptografischer Hashwert (Cryptographic Hash). Dieses Verfahren findet gleichermaßen auf dem Server statt. Der einzige Unterschied zum Server besteht darin, dass die Systemzeit des Servers genutzt wird. Findet innerhalb der festgelegten Zeit (standartmäßig nach 30 Sekunden nach RFC6238) ein erfolgreicher Match auf Serverseite statt, wird der User authentifiziert und ihm der Zugang gewährt. [19]

Ein entscheidender Nachteil dieser Methode entsteht, falls der authentifizierende User innerhalb dieser 30 Sekunden (beispielsweise) durch eine Störung des Netzes oder einen kompletten Netzausfall die Verbindung verliert.

Der aktuelle TOTP - Code wird ungültig und muss erneut angefragt bzw. erstellt werden. Apps wie der Google Authenticator lösen dieses Problem, in dem sie einen Timer setzen und alle 30 Sekunden einen automatisch generierten neuen Code mit der zur Verfügung stehenden Zeit anzeigen. Dabei muss beachtet werden, dass Server und Client synchron sind. So gibt es einige ältere TOTP - Generatoren, die einen internen Zähler haben und mit jedem Tick eine Sekunde auf den aktuellen Wert nach UTC draufrechnen. Findet bei Geräteneustart keine erneute Synchronisation (im RFC6238, Resynchronisation) statt, so entstehen fehlerhafte TOTP - Codes auf Clientseite, die der Server ablehnen wird. Dieser Nachteil ist gleichzeitig allerdings ein großer Vorteil in punkto Sicherheit, da ein Angreifer potenziell einen sehr kleinen zeitlich begrenzten Angriffsvektor besitzt, in dem er den TOTP - Code lösen muss. Durch entsprechende Zugriffssicherheitsfeatures, wie das Sperren des Nutzers nach oftmalig wiederholter falscher Eingabe, wird ein Bruteforce-Angriff verhindert.

### 3.2.3 WebAuthn

Webauthn ist kurz für 'Web Authentication'. Zur Verfügung gestellt wurde dieser Standard der Authentifikation 2018 von der FIDO Alliance und dem W3C und ist vollumfänglich im FIDO2 Standard definiert. Sie ermöglicht eine passwortlose Authentifikation durch Nutzung des vorhandenen public-private-key-Verfahrens für Webseiten.

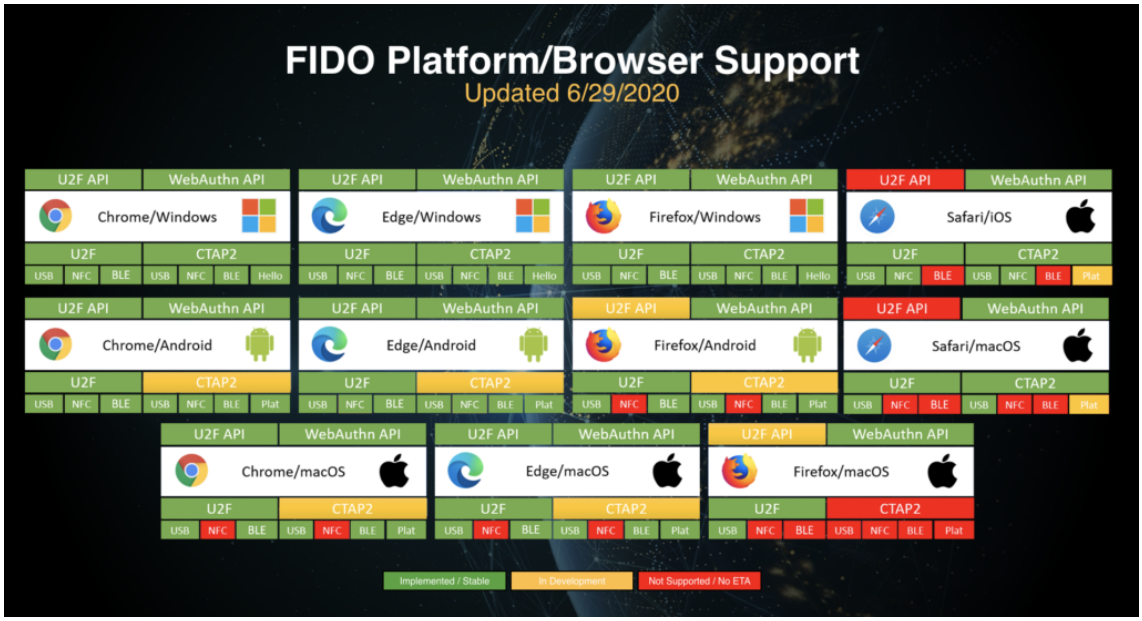



Abbildung 3.5: Plattform/Browser Support von FIDO2, 06.29.2020

Die Abbildung der FIDO - Alliance zeigt die Unterstützung von FIDO2 für verschiedene Browser und Plattformen. So besitzt zwar jeder Browser unter jedem Gerät die Unterstützung für die Webauthn API, allerdings unterstützt zum Beispiel der Browser Firefox unter macOS das CTAP2 Protokoll nicht, welches somit die Untersützung der Webauthn API obsolet macht. Es sollte erwähnt werden, dass in der Abbildung ein gedanklicher Trennstrich zwischen den Beiden Paaren U2F-API und U2F sowie WebAuthn API und CTAP2 fehlt. Sie unterstützen sich nur als Paar gegenseitig allerdings nicht untereinander. Gleichzeitig bedeutet die Unterstützuing der CTAP2 Schnittstelle nicht automatisch, dass auch jede Variante der Authentifizierung möglich ist. So wird unter Windows unter den drei Browsern Google Chrome, Microsoft Edge und Firefox jede der vier Varianten unterstützt. Unter Android fehlt für alle drei Browser die Unterstützung von CTAP2 gänzlich. Unter macOS wiederrum wird CTAP2 im Safari-Browser zwar unterstützt, doch NFC unf BLE zum Stand des Bildes (26.09.2020) nicht.

Die Abbildung verdeutlicht, dass Webauthn zwar auf Windows-Geräten vollumfänglich unterstützt wird, allerdings noch nicht zum Standard für andere Betriebssysteme geworden ist, so wie es ECMAScript6 (bzw. heutiges Javascript) oder Flash ist. Es wird wohl noch einige Jahre andauern, bis alle Browser die CTAP2 Schnittstelle unterstützen und damit eine Authentifikation durch Webauthn ermöglichen.

Der Standard definiert verschiedene Arten von Authentifikationsmöglichkeiten. Neben sogenannten “plattform authenticators“ [20] also jenen Verfahren, die die geräteeigenen Sicherheitsfeatures wie den Fingerabdrucksensor oder der Gesichtserkennung durch eine Kamera nutzen, definiert der Standard auch die Authentifizierung durch externe Geräte wie USB-Sticks, auf denen sich der private Schlüssel des Nutzers (in gesicherter Form) befindet. Grundsätzlich definiert WebAuthn nur Dinge, die man im nativen Bereich bereits kennt, adaptiert diese allerdings für den Webbereich. Dabei basiert Webauthn auf vielen bereits vorhandenen Abhängigkeiten der Informatik wie die Standards von HTML5, ECMAScript, COSE (CBOR Object Signing and Encryption COSE, definiert im RFC8152) oder der Base64url encoding.

Laut dem W3C sei die grundsätzliche Idee hinter diesem Verfahren, dass die Daten des Users zu keinem Zeitpunkt den Rechner verlassen müssen. Die Daten werden vollumfänglich vom jeweiligen Authenticator gemanaged, welches dann mit einer sogenannten ’Relying Party’ interagiert um den Nutzer zu authentifizieren. Eine Relying Party ist grob vereinfacht eine Entität, welche die Web Authentication API nutzt um den Nutzer zunächst einmal zu registrieren und in Zukunft zu authentifizieren. [20]

Auf einer von Codegram organisierten Rede erläutert Suby Raman, seinerseits Softwareentwickler bei Duo Security (einer Firma, die sich auf die Zwei-Faktor-Authentifizierung (2FA) - Authentifikation spezialisiert hat), die bereits bekannten Probleme von Passwörtern, die im Rahmen dieser Arbeit im Kapitel der Einleitung bereits addressiert worden und wie WebAuthn diese löst.

1. **Passwörter sind geteilte Geheimnisse.**

   Der ausgetauschte öffentliche Schlüssel ist kein Geheimnis, der private Schlüssel bleibt stets auf dem Gerät des Nutzers und verlässt diesen zu keinem Zeitpunkt der Authentifikation.

2. **Sichere Passwörter sind schwer zu finden und wenn, dann sind sie nicht leicht merkbar.**

   Der Authenticator auf den Geräten des Nutzers erstellt zufällige und sichere 'credentials' für die Authentifikation. Es liegt nicht mehr in der Verantwortung des Nutzers dies zu tun.

3. **Passwörter können leicht entwendet werden.**

   Sichere interne (oder auch externe) Hardware macht es Angreifern sehr schwer, wenn nicht schon fast mathematisch unmöglich, durch physische Angriffe an die privaten Schlüssel zu kommen. Zum Verfassungszeitpunkt dieser Arbeit sind keine Angriffe auf diese Geräte bekannt, vereinzelt gibt es dennoch Angriffe auf die Software der Hersteller, die sich auf den Geräten teilweise vorinstalliert befindet.

4. **Passwörter haben ein Bequemlichkeitsproblem und sorgen automatisch für die unsichere Wahl dessen.**

   Die 'credentials' des Nutzers sind gebunden an den aktuellen Scope. Das heißt das jede Anfrage einzigartig (Challenge-Response-Verfahren). Dies gillt sowohl für die Registrierung als auch den späteren Login mithilfe der Web Authentication API. Der erwähnte Wiederholungsangriff im Kapitel für unsichere Passwörter wird damit verhindert. Ein mehrmaliges Nutzen der erstellten zufälligen Daten durch den Authenticator ist damit nicht möglich und nur für eine Anfrage valide.

5. **Aus Sicht der Entwickler sind Passwörter schwer zu sichern.**

   Da der öffentliche Schlüssel kein Geheimnis darstellt, hat dieser auch keinen besonderen Wert und kann problemlos im Klartext in Datenbanken persistiert werden.

Zusammengefasst hat der FIDO2 Standard mit CTAP (dem Protokoll für externe Authentifikationen mit Mobilgeräten) und Webauthn (der Schnittstelle bzw. "API") vorhandene Funktionen definiert, mit der native Authentifizierungsmethoden wie das public-private Key Verfahren auf den globalen Webbereich übertragen werden können.

# 4 Konzeption

## 4.1 Sichere Webseiten

Einige wichtige Fakten zu Webseiten, die bei der Konzeption des Prototypen berücksichtigt werden müssen:

- **Immer mehr Nutzer sorgen für mehr Authentifikationen pro Sekunde und damit für eine höhere Auslastung**

  Webseiten besitzen immer mehr und mehr Nutzer [21]. Mittlerweile besitzen über 50% der Menschen weltweit einen Internetzugang und damit Zugriff auf die Vielfalt des World Wide Web [22]. Für Webseitenbetreiber bedeutet diese Entwicklung zunächst einmal das es sehr viel mehr potenzielle Aufrufe auf Webseiten pro Sekunde gibt. Daraus ergibt sich, dass es auf den verschiedenen Diensten auch mehr Authentifikationen pro Sekunde (im Vergleich zu vorher) gibt. Der Prototyp sollte in der Lage sein, Nutzer unabhängig voneinander zu bearbeiten ohne einen bestimmten Service zu blockieren. Dazu müssen von vorneherein begründet die richtigen Werkzeuge (und Programmiersprachen) verwendet werden, die dieses Vorhaben überhaupt erst möglich machen.

- **Webseiten sind über das offene Netz zugänglich und bieten damit eine große Angriifsfläche für Brute-Force Angriffe**

  Die berühmtesten Webdienste wie Netflix, YouTube und Facebook sind aus dem Internet erreichbar. Daher ist es nicht verwunderlich, dass gerade diese Webseiten Ziel von Bruteforce Angriffen werden können. Das gibt einem Angreifer die Möglichkeit, gezielt eine Liste von gebreachten Usernamen und Passwörtern auf verschiedenen Diensten durchzuprobieren. Meist sind diese Dienste in der Lage, die IP-Adresse des Angreifers zu blockieren falls zu viele Anfragen in einer gewissen Zeitspanne eingehen, dies umgehen die Angreifer dann allerdings mit Proxylisten und erzwungenen Wartezeiten vor jedem Angriff (z.B: nur 100 Accounts pro Proxy alle 60 Sekunden ausprobieren). Der Prototyp sollte Brute-Force Angriffe möglichst verhindern, indem vor der Webseite eine Firewall installiert wird, die die eingehenden Anfragen überprüft.

Gehen vermehrt Anfragen aus einem bestimmten Land oder einem bestimmten IP-Bereich ein, muss dies erkannt und gebannt werden.

- **Webseiten besitzen mehr Möglichkeiten als je zuvor um Nutzer zu authentifizieren, von dem muss sie aber erst Gebrauch machen**

  Die alleinige Möglichkeit der Authentifikation durch verschiedene Faktoren genügt heutzutage nicht mehr aus. Wie beim Stand der Forschung zur adaptiven Multifaktorauthentifikation bereits erläutert, schreitet die Digitalisierung rasant voran. Dies zwingt Webseitenbetreiber dazu, die neuen Verfahren wie die TOTP - Authentifikation oder das WebAuthn auch zu unterstützen. Die besten Verfahren sind sonst unbrauchbar, wenn es keine Dienste gibt, die sie umsetzen. Aus dem folgt, dass der Prototyp verschiedene Authentifikationsmethoden anbieten muss, um der Digitalisierung des 21. Jahrhunderts gerecht zu werden. Neben dem Passwort sollte es mindestens ein (besser wären sogar 2 oder mehr) Verfahren geben, die sich an einem der zwei weiteren Kategorien des Besitzes und der Biometrie bedienen.

- **Webanwendungen speichern üblicherweise Nutzerkonten bzw. personenbezogene Daten**

  Der Datenschutz erfordert angemessene Vorsichtsmaßnahmen bei dem Umgang mit diesen Daten. So ist die Passwortdatei gegen unberechtigtes Kopieren zu sichern, Passwörtern müssen auf dem Rechner des Nutzers verschlüsselt und dann übertragen und fehlerhafte Anmeldeversuche protokolliert werden [23].

  Der Prototyp sollte vor allem Passwörter nicht im Klartext übertragen (falls unsichere Verbindung) und in keinem Falle im Klartext in Datenbanken persistieren. Sehr persönliche Daten wie zu Fingerabdrücken sollten möglichst gar nicht in Datenbanken persistiert werden. Dies müssen sie bei der Nutzung entsprechender Schnittstellen wie WebAuthn allerdings auch nicht um eine sichere Authentifikation zu gewährleisten.

- **Webnutzer sind oft unvorsichtig mit Passwörtern und wählen zu leichte**

  Wie bereits gezeigt, neigen Nutzer zu sehr einfachen Passwörtern. Daher müssen einfache Passwörter wie “1234” im Vorhinein bereits ausgeschlossen werden. Man könnte eine Liste mit häufig genutzten Passwörtern verwenden, die den Nutzer vor der Wahl eines zu einfachen Passwortes hindert. Am einfachsten ist es eine Passwort-Policy Strategie zu wählen, die eine Mindestlänge des

Passwortes voraussetzt. Dies macht das Passwort bei missbräuchlicher Nutzung zwar auch nicht sehr viel sicherer, ist aber ein Schritt in die richtige Richtung.

## 4.2 Auswahl der Authentifizierungsverfahren

Bei der Auswahl der Authentifizierungsverfahren musste vor allem berücksichtigt werden, inwiefern das Verfahren im Rahmen dieser Arbeit umsetzbar ist. Dies bezieht sich einerseits auf die Zeit, die es für die Implementation benötigt und andererseits auch auf die monetäre Machbarkeit. Es existieren für die Webauthentikation zum Beispiel ganz spezielle USB-Sticks von angesehenen Marken, die allerdings einige hunderte Euro kosten. Die Features die sie im Gegensatz zu einem ganz normalen USB-Stick (FIDO USB) liefern, beziehen sich dabei nur auf den Teil der Userverifikation. So besitzen einige USB-Sticks einen eingebauten Fingerabdrucksensor, ein Tastenfeld oder sogar zusätzliche Sicherheitssoftware auf den Massendatenträgern. Innerhalb dieser Arbeit sollte es allerdings um den Einsatz eines solchen Verfahrens für den Durchschnittsnutzer innerhalb von Unternehmen oder einen Casual Surfer gehen. Eine teure Anschaffung für die eigene Sicherheit erscheint dem Durchschnittsnutzer als nicht nötig.

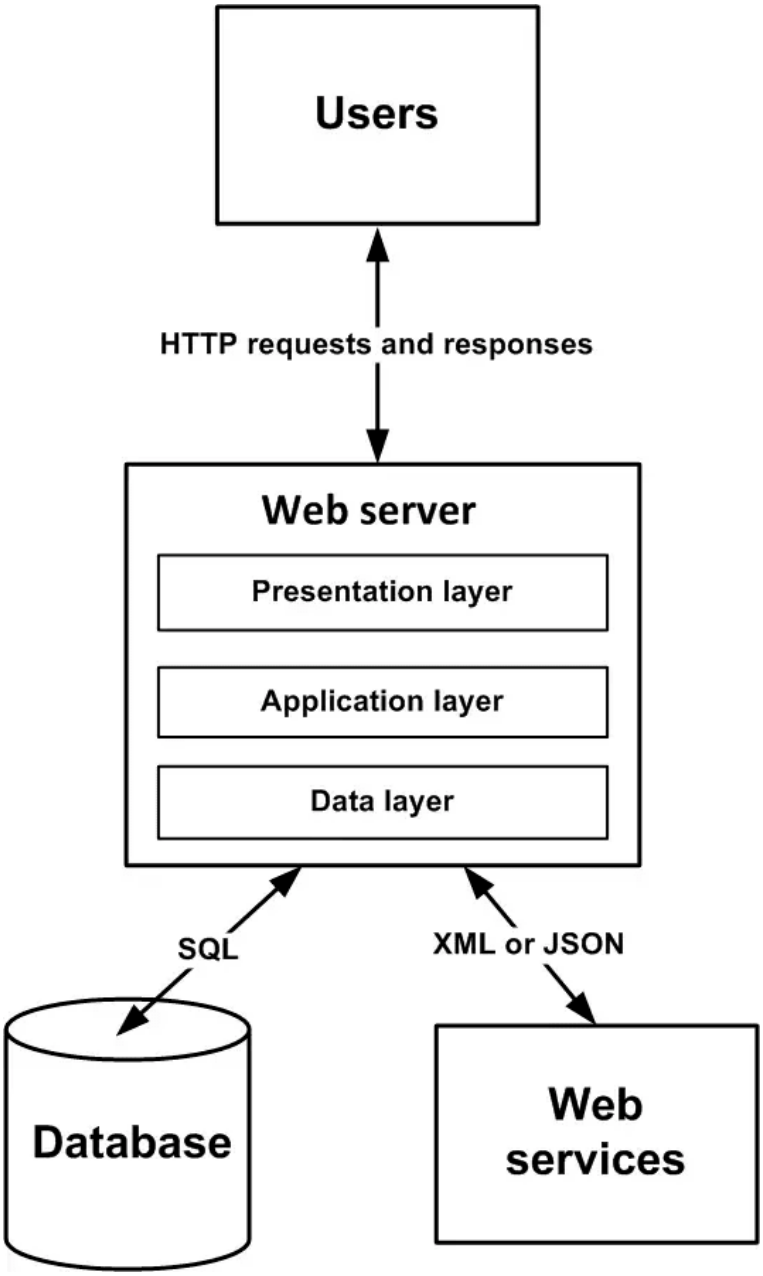
Das Verfahren des Usernamen und des Passwortes wurde gewählt, weil es zum Verfassungszeitpunkt dieser Arbeit die Nutzer von Webseiten meist alleinig sichert. Die Loginverfahren größerer Dienste wie Microsoft Outlook, Google Mail und Facebook besitzen zwar schon seit Jahren die Möglichkeit der Authentifikation mit einem zweiten Faktor, doch das sind eher Ausnahmen. Ein Großteil der Internetpräsenzen ist sich entweder nicht der Risiken von Passwörtern für ihre Nutzer bewusst oder kann sich den Mehraufwand von neuen Verfahren nicht leisten. Gleichzeitig war es im Rahmen dieser Arbeit wichtig eine typische Implementation mit Username & Passwort und ihren Scheinsicherheiten wie Hashes mit Salt und Pepper zu präsentieren. Es ist wichtig, den Ist-Zustand im heutigen Internet zu verstehen, um die Vorteile von neueren Methoden nachzuvollziehen.

Eine dieser neuen Methoden ist das zweite Verfahren. Nach dem Passwort ist das wohl der häufigste zweite Faktor den man findet. Apps wie der 'Google Authenticator' zählen mittlerweile als Synonym für das TOTP - Verfahren wie es im RFC beschrieben ist. Dies liegt unteranderem auch daran, dass die App tatsächlich die recht strikten Vorgaben aus dem RFC befolgt und nur minimale Unterschiede dazu aufweist.

An dieser Stelle stellte sich eher die Frage ob HOTP oder TOTP in den Prototypen implementiert werden sollte. Es wurde sich für das TOTP - Verfahren entschieden, da es zwar den Mehraufwand der Synchronisation zwischen Zeit des Gerätes und des Servers benötigt doch dafür die Sicherheit der begrenzten Zeit bietet wie bereits erläutert.

Das dritte Verfahren, die 'Web Authentication', ist wohl das Erforschungswürdigste der drei Verfahren. Sie ist von allen drei Verfahren am seltensten auf Internetseiten vertreten und da stellt sich nach einer Analyse des Mehrwertes dieses Verfahrens die Frage nach dem Grund dafür. Im Gegensatz zu den anderen beiden Authentifikationsmethoden wurde Dieses nicht für den Prototypen ausgewählt, weil es so weit verbreitet und bereits genutzt wird sondern genau im Gegenteil: Es wurde gewählt, um mögliche Gründe zu suchen weshalb das Verfahren der Webauthn heutzutage kein Standard darzustellen scheint. Es muss untersucht werden ob es starke Nachteile Gegenüber der anderen Verfahren bietet, denn im Ansatz scheint es alle Nachteile des Passwortes auszumerzen.

## 4.3 Authentifikation in drei Schichten




Das Ziel des Prototypen ist es, wie eingangs erwähnt, vorhandene Authentifizierungsverfahren abseits der klassischen UserID / Passwort Methode zu begutachten und dessen Schwächen aufzudecken. Daraus ergeben sich die typischen drei Komponenten von Drei-Tier-Client-Server Architekturen:

Ein Client, der Anfragen an die Applikation (das Backend bzw. den Server) sendet, welches die Daten dann in einer Datenbank mittels eines DBMS (Datenbankmanagementsystems) persistiert und verwaltet. Wie auch im Bild zur 3 Tier Architektur zu sehen, agiert der User auf dem Presentation Layer und stellt anfragen bzw. sendet JSON Objekte an den Server, der die entsprechenden Webservices liefert. Der Server speichert keine States, wodurch der Nutzer typischerweise mit jeder Anfrage alle benötigten Informationen liefern muss. Der Server liefert demnach lediglich die Respresentational State Transfer (REST) Schnittstellen. Implementationsdetails und weitere Informationen zum Prototypen gibt es in folgenden Kapiteln.

## 4.4 Weitere Anforderungen

Auch wenn die Frage: “Ist eine gelungene Authentifikation auch eine erfolgreiche Authentifikation?” eine philosophische zu sein scheint, ist sie dennoch nicht minder interessant. Es wurde bereits gezeigt, dass Nutzer nicht gewillt sind komplizierte und aufwendige Dinge zu tun, um sich selbst und ihre Daten im Internet zu schützen. Sofern also die Methodik der Webseite zu kompliziert ist, sucht der Nutzer sich Ausflüchte um die vorhandenden Verfahren so angenehm wie möglich zu machen, was wiederrum die Sicherheit ihrer Internetpräsenz gefährdet. Das Passwort besitzt zu viele Schwachstellen als das man eine Authentifikation damit als Erfolg verbuchen kann. Es scheint eher dem Zweck ’irgendeiner Schutzbariere zwischen fremder Person und Daten des Nutzers’ zu dienen als wirklich zur ’Sicherheit’.

- **Nutzerfreundlichkeit**

  Angefangen der Nutzerfreundlichkeit, einem der wichtigsten Kriterien für die Authentifikation, welches in direktem Bezug zur Sicherheit steht [24] [25] [26]. Der Prototyp soll eine einfache Authentifikation ermöglichen. Keine häufigen Weiterleitungen, keine Popups, keine langen Ladezeiten und vorallem zwecks Authentifizierung: Keine sinnfreie Aneinanderschaltung von verschiedenen Sicherheitsfaktoren.

Ein Beispiel für solch eine Praktik wäre, dass nach der Bestätigung des Einloggvorgangs durch Fingerabdruck zusätzlich eine PIN und in folge dessen ein Sicherheitsschlüssel (mit zusätzlich benötigtem PIN) verlangt wird, obwohl nach dem ersten Schritt des Fingerabdrucks bereits die Authentizität des Users festgestellt wurde. So muss der Prototyp am Besten aus nur einem Faktor der Sicherheit bestehen und dem Nutzer die Auswahl zur Selbstentscheidung über die verschiedenen Möglichkeiten bieten, so wie es bei der adaptiven MFA (anhand von selektierten Faktoren) auch der Fall ist. Im konkreten Fall des Prototyps soll es demnach auch reichen nur einen der registrierten Geräte für die Web Authentication zu nutzen, so kann der User bequem die PIN zur Authentifizierung nutzen. Falls der Rechner dies nicht unterstützt, kann der User zum Sicherheitsschlüssel greifen oder das Smartphone für das TOTP - Verfahren verwenden.

- **Sicherheit**

  Die Sicherheit einer Webanwendung hängt primär von ihrem Schutzbedarf ab. Viele Softwareentwickler setzen sich nur widerwillig mit dem Thema außeinander, oder in den vergangenen Jahren eher notgedrunken durch die Kunden. Dabei sind die Ursachen teilweise auch UX-Fehlern zuzuschreiben, weshalb der 'Secure by Design' Ansatz befolgt werden sollte [27].

  Der Prototyp sollte zur Sicherung der Kunden möglichst unnötige Metadaten vermeiden. Dazu sollen meist die selben Statuscodes und nur allgemeine Fehlermeldungen bei nicht erfolgreicher Authentifikation zurückgegeben werden. So ist die Nachricht "Das Passwort ist inkorrekt." als obsoluter Fehler zu betrachten, da ein Angreifer nun die Möglichkeit hat, durch einen Bruteforce - Angriff auf das Loginformular das richtige Passwort zu erraten. Diese Nachricht würde nämlich implizieren, dass ein Nutzer mit eingegebenem Nutzernamen in der Datenbank vorhanden ist, doch das Passwort nicht stimmt. Stattdessen werden allgemeine Responsemessages gegeben wie "Der Benutzername oder das Passwort sind falsch.", um dem Angreifer keinerlei Metadaten auszuhändigen. Gleichzeitig muss die Verbindung zum Backend, auch im Prototyp, durch ein eigenes SSL - Zertifikat gesichert sein. MITM - Angriffe sind, wie bereits erläutert, nicht immer verhinderbar. In unserem Falle sind wir selbst der Aussteller des Zertifikats und vertrauen uns selbst. Ein User kann (und wird) allerdings nicht die Vertrauenswürdigkeit seines Zertifikates erfragen.

Wenn man die drei Verfahren nach ihrer Sicherheit ordnet, käme an hinterster Stelle wohl das Passwort. Wieso wurde bereits erläutert. Darauf würde das TOTP - Verfahren kommen, das so lange sicher ist wie das Gerät welches die TOTP - Codes erfragt sicher ist.

Hat man seinen QR-Code also im Google Authenticator eingescannt und lässt sein Smartphone ungesichert an einem öffentlichen Ort liegen, kann ein Angreifer sich das Smartphone nehmen, die App öffnen und den Code auf der Webseite eingeben. Das sicherste Verfahren (aus den ausgewählten im Rahmen dieser Arbeit) ist die Webauthentikation. Sie bietet wenig Angriffsfläche, dadurch das die privaten Schlüssel an einem sicheren Ort im Betriebssystem persistiert werden und es demnach in der Verantwortung des Betriebssystems liegt diese zu sichern. Der öffentliche Schlüssel besitzt keinen Wert und wird vom Authenticator generiert. Dadurch ist kein Zutuen des Nutzers erforderlich wie bei einem Passwort. Dieses Verfahren ist auch relativ sicher gegen MITM - Angriffe, da ein Angreifer maximal die Challenge des Servers abfangen kann, um sein eigenes Gerät für einen Nutzer zu registrieren. Dies erfordert allerdings entweder eine komplette Kontrolle über den Rechner (Remote Access Control) oder phsysischen Zugriff auf den Rechner durch den Angreifer.

- **Datenschutz**

  Der Datenschutz ist ein Element der Sicherheit [28], welches als “Vertraulichkeit” auch in den Grundwerten des IT-Grundschutzes wiederzufinden ist. Die Forschungsfrage dieser Arbeit ist überhaupt erst entstanden, durch einen Datenschutzvorfall bei Firmen. Auch wenn der Datenschutz bis jetzt nicht viel Erwähnung fand, ist er neben der Sicherheit und der Nutzerfreundlichkeit kein minderes Kriterium für eine erfolgreiche Authentifikation. Zum Schutz der Daten gehört zum Beispiel, SQL - Injektionen durch das Benutzen von ’Prepared SQL Statements’ [28] zu verhindern. Dies bedeutet, dass Userdaten an keiner Stelle unformatiert in ein SQL Schnipsel eingebaut werden. Gleichzeitig muss sichergestellt werden, dass der Prototyp bei erfolgreichem Login keinerlei Metadaten über den Nutzer preisgibt. So wäre es ein Datenschutz-Problem wenn bei der Authentifikation über Webauthn zusätzliche Daten wie die E-Mail des Nutzers, sein Passwort und die Zeit der Erstellung des Accounts mitgesendet werden. Das Prinzip sollte stets lauten: So wenig Daten wie nötig. Das Recht auf informelle Selbstbestimmung [29] ist hier insofern gewährleistet, dass der User selbst entscheiden kann, ob er einen TOTP registriert oder welches Gerät er für die Webauthn nimmt.

Ein großer Vorteil neuerer Verfahren ist, dass hinter einer TOTP oder Webauthn - Geräteregistration keinerlei persönliche Daten stehen. Selbst wenn Angreifer also den gesamten Registrationsprozess auf dem Gerät mitschneiden, können sie daraus keine verwertbaren Metadaten für zukünftige Angriffe erlangen.

# 5 Prototypischer Lösungsansatz

## 5.1 Implementationsdetails

1. **Client / Webseite**

   Die Webseite besteht aus einer zur Single Page Application (SPA) ähnlichen Architektur, die ausschließlich Javascript und JQuery, also clientseitige Programmiersprachen nutzt. SPA's sind Webapplikationen, bei denen der Nutzer eine Seite betritt und diese nie wieder vollständig laden muss. Frameworks wie Angular, React oder VueJS basieren auf dieser Mechanik und nutzen Javascript und JQuery um die Ladezeiten einer Webseite durch Module zu reduzieren. Anstatt also die gesamte Webseite neuzuladen, laden diese Frameworks nur spezielle Bereiche der Webseite asynchron nach. In dem Prototyp ist dies größtenteils der Fall, da neben der Hauptseite **index.html** eine weitere Seite existiert, auf die man bei erfolgreichem Login weitergeleitet wird: **secret_panel.html**. Auch werden keine Module nachgeladen, sondern bei entsprechender Aktion nur Elemente innerhalb des DOM - Baums per id identifiziert und versteckt.

   Beim Betreten von beiden Seiten der Anwendung findet eine Prüfung nach dem Session-Cookie (**user_sid**) statt, die vom Server bei einer erfolgreichen Authentifikation im Header über 'Set-Cookie' als Response zurückkommt und vom Browser infolge dessen gesetzt wird. Der Aufruf der **index.html** Seite ist nicht möglich mit gesetztem Cookie und leitet den User auf **secret_panel.html** weiter und umgekehrt genauso. Die Sicherung dieses Cookies auf Serverseite durch Prüfungen ist nicht Bestandteil dieser Arbeit, hier wird sich lediglich auf den Loginprozess von Anendungen konzentriert um dessen Schwächen und die Vorteile neuer Verfahren aufzuzeigen. Aktuell könnte ein Angreifer demnach selbst einen entsprechenden Cookie mit einem Wert setzen und das 'geheime Panel' erreichen. Bei weiter ausgereiften Webapplikationen wird bei jedem Request an den Server der Session Cookie mit einer temporären Map innerhalb des Backends abgeglichen. Befindet sich der Session Cookie nicht in dieser Map oder hat nicht genügend Rechte für diese Anfrage, erhält der Nutzer den Statuscode 401 (Unauthorized) zurück. So haben es etablierte Dienste wie 'Netflix' und 'Microsoft' bereits umgesetzt.

Aus der Forschungsfrage ergibt sich, dass die Webseite ausschießlich Javascript (oder auf Javascript basierende Programmiersprachen wie JQuery) zur Implementation der Programmlogik verwendet. Javascript selbst ist eine clientseitige Programmiersprache und gibt dem Nutzer den gesamten Code auf Webseiten preis. Demnach wurde bei der Anwendung darauf geachtet, alle clientseitigen Prüfungen, auch serverseitig zu implementieren. Zumindestens alle wichtigen Prüfungen, die sonst den Programmablauf verhindern würden oder sogar Serverabstürze zur Folge hätten. Nicht nur gibt Javascript den Code preis, über die Konsole (Unter Google Chrome und Firefox in den Entwickleroptionen des Browsers zu finden) lassen sich ganze Funktionen oder globale Variablen manipulieren. Variablen innerhalb von Funktionen können vom Angreifer nicht so leicht manipuliert werden.

Möglich ist es dennoch, den Inhalt dieser Funktion in eine andere Funktion zu schreiben, um die Variable dort auszutauschen. Um dem Nutzer nicht jede Funktion problemlos erreichbar zu machen, wurde bei der Implementation das Revealing-Module-Pattern angewendet, über die gewisse Variablen nicht im globalen Scope (window) sondern im Scope einer Funktion bleiben, die nur gewisse andere Funktionen in sich exportiert und als eine Art Schnittstelle fungiert. Auf dieses Pattern wird im Folgenden näher eingeggangen.

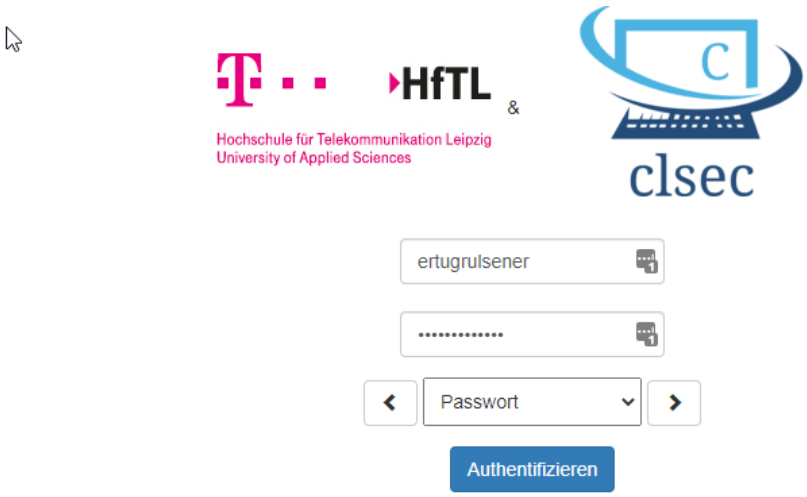



Abbildung 5.1: Design des Prototypen (Codename: clsec) - Anklicken zum Abspielen

Das Design der Webanwendung ist kein wesentlicher Punkt dieser Arbeit und deshalb schlicht gehalten. So besteht der Prototyp aus einer einfachen Webseite, die aus dem globalen Internet erreichbar ist und die zwei Eingabefelder und einen Loginbutton besitzt. Um sich die Designarbeit oder das Suchen von Icons und passenden Buttondesigns zu sparen wurde auf bekannte Frameworks wie Bootstrap4 und Funktionalitäten wie Flexbox gesetzt, die eine automatisches Rescaling und Positioning von Webseitenelementen umsetzen. So war es nicht nötig ein Loginformular zu designen sondern vorhandene Strukturen von Bootstrap für Buttons aller Art zu nutzen. Die unterschiedlichen Methoden der Authentifizierung wählt man über ein Dropdownmenü über dem Login - Button. Je nach Authentifizierungsverfahren werden andere Elemente sichtbar, die die weiteren Schritte für die Authentifikation erläutern. Die Methode kann sowohl über das Dropdownmenü als auch über einen Klick auf die Pfeile links und rechts neben dem Dropdownmenü gewählt werden, die als Pagination zwischen den verschiedenen behandelten Verfahren fungiert. Ist man am Ende der drei Methoden, springt man wieder zur ersten Methode und andersherum.

Bei der Web Authentication gibt es eine Besonderheit. Die Webseite muss auf die Schnittstellen des Betriebssystems zugreifen, um den Nutzer zu verifizieren. Dieser Teil kann von dem Prototyp nicht beeinflusst werden und wird vom CTAP2 Protokoll innerhalb des FIDO2 Standards definiert. Dadurch entstehen teilweise merkwürdige und aus UX (User Experience) - Sicht höchst fragwürdige Interaktionen. So fragt das Betriebssystem zunächst (bei entsprechender Möglichkeit) nach einem PIN um den Nutzer zu registrieren. Drückt man nun die Escape-Taste erscheint ein Dialog um einen Sicherheitsschlüssel (ein externes Gerät) einzurichten. Beim Login wiederrum ist dies durch eine Dropdownliste schöner gelöst worden, wo der User alle möglichen Loginmethoden auf einem Blick sieht und diese Wählen kann. Während er bei der Registration keine Chance hat dies zu tun und immer erst ein PIN - Feld angezeigt wird. Auf die einzelnen behandelten Verfahren wird in späteren Kapiteln noch genauer eingeggangen, da werden solche Schwierigkeiten aufgegriffen da dies nur eines von vielen 'Problemen' neuerer Verfahren ist: Die Abhängigkeit vom Betriebssystem.

2. **Server**

Der NodeJS - Server besteht aus einer REST Api, die keine Zustände speichert. Neben der Aufgabe des Cookie Managements und der Generierung von UUID's liefert er zudem die Schnittstelle zur Datenbank und verschiedene Funktionalitäten für die behandelten Verfahren Username & Passwort, TOTP und Webauthn:

- **/logout** - POST - Löscht den Session-Cookie (und damit die Session des Nutzers) und leitet ihn auf die Loginseite **index.html** weiter.

- **/get_public_key** - GET - Liest den öffentlichen Schlüssel des Servers ein und gibt diesen in einer JSON - Struktur zurück. Ist wichtig für die Username & Passwort - Authentifizierung.

- **/password/login** - POST - Nimmt einen mit dem öffentlichen Schlüssel des Servers verschlüsselten Usernamen und Passwort entgegen und entschlüsselt diese mit dem privaten Schlüssel. Im Anschluss darauf wird in der Datenbank über einen gepoolte Query nach dem Nutzer gesucht. Wurde dieser gefunden, erstellt der Server einen SHA512 - Hash aus dem Passwort (aus dem Request) und dem Salt des Users (aus der Datenbank) indem die **hashString** - Methode aufgerufen wird. Diese bezieht den Pepper des Servers mit ein. Sofern die Hashes aus der Datenbank und der soeben Generierte übereinstimmen, erhält der Nutzer die Response 200 und den Text “OK”. Gleichzeitig wird wie bei jeder anderen Authentifizierungsmethode die **createSessionCookie** - Methode aufgerufen um den Session-Cookie (eine zufällige UUID) im Header zurück an den Nutzer zu senden, sodass dessen Browser ihn setzt..

- **/password/create_password_hash** - GET - Nimmt eine Zeichenkette des Nutzers entgegen und erstellt einen Password-Hash für den Nutzer, der manuell in die Datenbank persistiert werden kann. Ist als 'Quality of Service' Funktion zu verstehen, um es dem Administrator einfacher zu machen, Passwörter zu erstellen, die bei Vergleich einen gülltigen Hash ergeben.

- **/totp/check_username** - POST - Diese Methode dient lediglich der Prüfung, ob der Nutzer die TOTP Authentifikation mittels zufälligem SECRET bereits gegenüber der Webseite validiert hat. Das Datenbankfeld 'totp_activated' wird hier geprüft, ist der Wert 1 (für das boolsche 'wahr') liefert der Server den Text "Success" zurück, das der Client deutet um nur das Feld für den sechsstelligen TOTP Code anzuzeigen. Hat das Feld allerdings den Wert 0, wird (sofern noch nicht vorhanden im Feld 'totp_secret') ein neues Secret erstellt und in der Datenbank für diesen Nutzer persistiert. Gleichzeitig wird eine URL beginnend mit otp:// erstellt, um diesen zusammen mit einem Base64 enkodierten QR Code an den Client zu schicken, der dies dem User präsentiert. Es wurde sich hier bewusst dazu entschieden, diese Schritte auf Serverseite vorzunehmen. Möglich wäre es auch auf Clientseite gewesen, wäre aber durch die Suche nach passenden Javascript Modulen mit einem Mehraufwand verbunden gewesen, welches verhindert werden sollte.

- **/totp/check_token** - POST - Nimmt den Usernamen des Nutzers und einen sechsstelligen TOTP Token entgegen. Sucht im Anschluss darauf in der Datenbank nach dem Nutzer und prüft über die Library 'otplib' ob der eingegebene TOTP - Token zum secret in der Datenbank valide ist. Dieser Schritt findet sowohl bei der Registration als auch beim Login statt. Wenn das Feld 'totp_activated' also 0 ist, wird es bei der ersten Authentifikation auf 1 gesetzt und der Nutzer wird eingeloggt. (Session-Cookie wird gesetzt und Nutzer weitergeleitet)

- **/webauthn/generate-attestation-options** - POST Liefert dem Nutzer die verschiedenen Registrieroptionen für die Web Authentication mittels WebauthnAPI. Dabei wurden verschiedene Optionen gesetzt, die im folgenden erklärt werden und die der User im Body der Response als JSON - Objekt erhält. Zur Vereinfachung werden hier nur die Optionen aus der abstrahierten Library und dessen Möglichkeiten erklärt. Dies ist zum Verständnis der Arbeit und des Prototypen ausreichend.

```
1  {
2      rpName: 'clsec',
3      rpId: 'localhost',
4      userID: 4,
5      userName: 'username',
6      userDisplayName: 'username',
7      attestationType: 'none',
8      authenticatorSelection: {
9          requireResidentKey: false,
10         userVerification: "discouraged"
               ,
11     },
12     excludedCredentialIDs:
           userAuthenticators.map(dev => dev
           .credentialID),
13 }
```

**rpName:** rp steht hier bei für Relying Party. Das ist der Name, der später auch im Registrierungsdialog als 'Webseiteninhaber' gelistet wird.

**rpId:** Die Relying Party Id ist eine URL, auf der sich der Nutzer registrieren möchte. In unserem Falle ist dies 'localhost' ohne Zusätze wie das Protokoll oder der Port. Diese Variable ist vor allem für Webseiten im Netz wichtig, sodass ein Angreifer nicht per Pishing den Nutzer zur Registrierung auf einer anderen Webseite bringt. Die Registrierung schlägt clientseitig fehl, wenn die rpId und die Webseitenaddresse (Damit ist nicht die Domain sondern die wahrliche IP-Addresse gemeint) nicht übereinstimmen. Beim Wert 'localhost' sendet der Server diesen Teil der JSON - Abfrage nicht mit, da es localhost aus Testgründen nicht prüft. Gleichzeitig erzwingt Webauthn ausschließelich für diese rpId keine sichere Verbindung.

**userID:** Das ist die ID des Nutzers die verifiziert wird, meist eine fortlaufende Ganzzahl.

**userDisplayName:** Das ist der Name, der beim Registrieren als Username gelistet wird.

**attestationType:** Der Server definiert, wie viele Informationen über den Authenticator er im attestation statement haben möchte. Das attestation statement erhält er nach dem der User sich mit einem Authenticator registriert hat und ist ein wichtiger Bestandteil des Objekts welches den Server im Nachhinein über eine erfolgreiche Registration benachrichtigt. Ferner ist der öffentliche Schlüssel des Nutzers in diesem Objekt lokalisiert, den der Server dann in der Datenbank persistiert. 'none' bedeutet hierbei, dass keinerlei Informationen erwünscht sind. 'indirect' als Option würde dem Nutzer erlauben selbst zu entscheiden wie viele Informationen er preisgibt bzw. könnte er eine anonymisierte CA verwenden um sein Zertifikat auszustellen und 'direct' als Option würde die Daten direkt vom Authenticator ohne einen Eingriff des Nutzers erwarten.

**authenticatorSelection:** Das sind verschiedene Optionen um den verwendeten Authenticator zu bestimmen. Das Argument 'requireResidentKey' bestimmt hierbei darüber, ob ein Authenticator benutzt werden darf, der selbst nicht in der Lage wäre einen privaten Schlüssel auf dem Betriebssystem zu sichern und dem Server infolge dessen einen öffentlichen Schlüssel zu senden. Die 'userVerification' ist im Prototyp nicht erzwungen also 'required' sondern 'discouraged', es obliegt also dem Betriebssystem und dem zu authentifizierenden Gerät / und der initialen Konfiguration zu entscheiden, ob der Nutzer sich gegenüber seines Authenticators verifizieren muss. In dem Beispiel eines bereits sicheren FIDO2 - USB Sticks könnte dies zum Beispiel nicht vom Nutzer gewünscht sein.

**excludedCredentialIDs:** Dies ist eine Liste von credentialID's die in der Datenbank im Feld 'webauthn_authenticator_data' als JSON - Struktur persistiert sind. Sie dient dazu, bereits registrierte Methoden nicht erneut anzuzeigen, funktioniert dennoch nicht zuverlässig.

3. **Datenbank**

Das gewählte Datenbankmanagementsystem ist PosgreSQL 12, dies hat einen speziellen Grund. Das Datenbankmanagementsysten besitzt fortgeschrittene Funktionalitäten um JSON Strukturen zu persistieren und zu bearbeiten. Gleichzeitig wäre es möglich Constraints zu erstellen, die bei anderen Managementsystemen wie MYSQL nur bei speziellen Tabellentypen wie InnoDB und das auch nicht vollständig verfügbar sind. Der Prototyp benötigt nur eine einzige Tabelle namens **users**, dessen Struktur anhand des folgenden SQL Statements erläutert wird.

```
1  CREATE TABLE public.users
2  (
3      id bigint NOT NULL,
4      username text COLLATE pg_catalog."default" NOT NULL,
5      password text COLLATE pg_catalog."de-DE-x-icu" NOT NULL,
6      salt text COLLATE pg_catalog."default",
7      totp_secret text COLLATE pg_catalog."default",
8      totp_activated integer NOT NULL DEFAULT 0,
9      webauthn_register_challenge text COLLATE pg_catalog."default"
           NOT NULL DEFAULT ''::text,
10     webauthn_login_challenge text COLLATE pg_catalog."default" NOT
           NULL DEFAULT ''::text,
11     webauthn_authenticator_data jsonb,
12     CONSTRAINT users_pkey PRIMARY KEY (id)
13 )
```

Er speichert typische Daten um den Nutzer zu authentifizieren. Jeder Nutzer besitzt eine eigene einzigartige ID, die eine Ganzzahl ist. Dann besitzt jeder User einen Nutzernamen im Feld 'username'. Die Felder 'password' und 'salt' besitzen logischerweise nur Nutzer, die die Passwort Authentifikation unterstützen. Die Felder 'totp_secret' und 'totp_activated' werden einerseits genutzt um das Geheimnis (welches der User beim TOTP Setup einscannt) zu erstellen und den Nutzer damit später zu authentifizieren und andererseits um dem Frontend zu signalisieren, ob der Nutzer das Setup vollzogen hat. Die Challenge-Felder für Webauthn 'webauthn_register_challenge' und 'webauthn_login_challenge' sind als eine Art Zwischenablage zu verstehen und hätten theoretisch auch im Cache des Backends persistiert werden können, da bei jeder neuen Registration und jedem neuen Login durch Webauthn eine neue Challenge vom Server erstellt und vom Nutzer nach dem Signieren zurückkommt. Interessant ist das Feld 'webauthn_authenticator_data', das in einer JSON-Liste alle registrierten Geräte für die Web Authentication beinhält.

```
[
    {
        "counter": 0,
        "publicKey": "pAEDAzkBACBZAQCgna71QX...",
        "credentialID": "HEeJWgh9QOWURee7dF..."
    },
    {
        "counter": 2,
        "publicKey": "pQECbt1via1qrycev0vyc...",
        "credentialID": "_HbcWx_XWd6Pyjyh..."
    },
    {
        "counter": 5,
        "publicKey": "pQECAyYgASFYIGpEwj...",
        "credentialID": "45nEVRuXWky8CyvW79e..."
    }
]
```

Dabei besitzt jedes Element jeweils einen counter, um die Anzahl an Einloggversuchen zu zählen, den öffentlichen Schlüssel und eine einmalige credentialID, die bei jeder Registration als 'exludedCredentialID' an den Clienten übermittelt wird, um zu signalisieren das diese credentialID sich nicht erneut registrieren kann. (Um doppelte Registrationen von ein und dem selben Gerät zu verhindern für den selben Nutzer)

## 5.2 Fehlerbetrachtung

Die Fehlerbetrachtung ist essenziell um nun auf die Probleme einzugehen, die während der Implementation bereits ersichtlich wurden.

1. **UX-Probleme bei Geräteregistrierung für Webauthn unter Windows**

   Bei der Implementation des Webauthn - Protokolls schien es zunächst ein Mal nicht ersichtlich ob die unerklärlichen Dialogboxen nun ein Implementationsproblem darstellten oder tatsächlich so gewollt sind. Erst die Prüfung durch das offizielle 'Playground-Tool' der FIDO Alliance, zu finden unter 'https://webauthn.io/' wurde es ersichtlich, dass die verwirrenden UX Entscheidungen von Windows kein Fehler des Prototyps sind.

Dennoch tragen sie zur allgemeinen Unverständnis bei. Bei diesem Problem geht es um folgende Abfolge an Dialogboxen, die bei der Registration eines neuen Geräts erscheinen:

a) Der Nutzer klickt unter Angabe seines Nutzernamens auf den Button 'Gerät registrieren' und erwartet eine Auswahl an vorhandenen Geräten, die registriert werden können. In diesem Beispiel stehen ein physischer Sicherheitsschlüssel und Windows-Hello-PIN zur Verfügung.

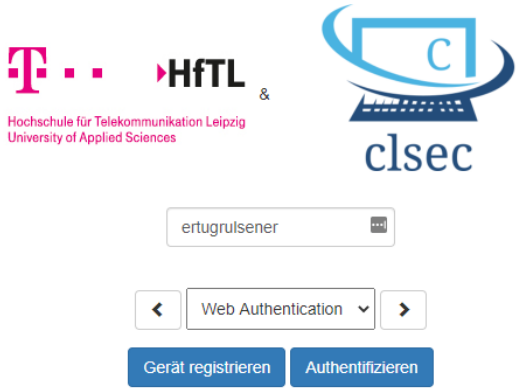



Abbildung 5.2: UX-Problem bei Webauthn Geräteregistrierung (1)

b) Statt der erwarteten Auswahlbox erhält der Nutzer die Möglichkeit auf die Windows-Hello-PIN und geht nicht von weiteren Auswahlmöglichkeiten aus.

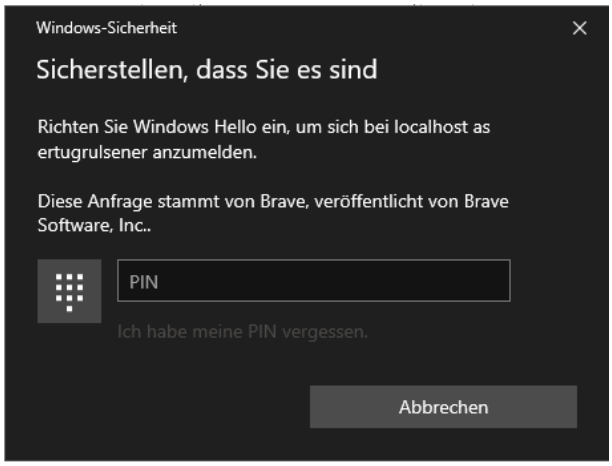



Abbildung 5.3: UX-Problem bei Webauthn Geräteregistrierung (2)

c) Beim Betätigen der Escape-Taste erscheint plötzlich der Dialog, um einen Sicherheitsschlüssel zu registrieren. Dies ist bereits ein Problem, weil der User sich zum Zeitpunkt der Registration nicht über diese Möglichkeit bewusst ist. Bei der Implementation ist diese Option auch nur durch Zufall aufgefallen.

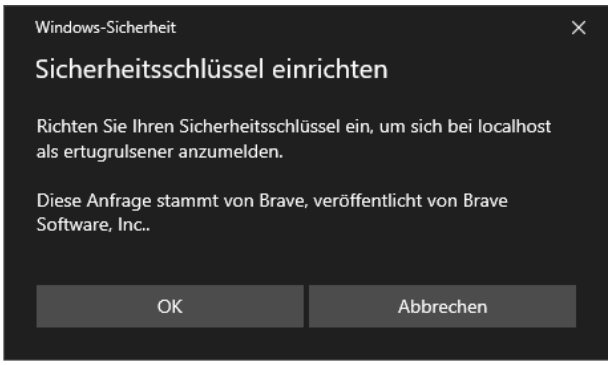



Abbildung 5.4: UX-Problem bei Webauthn Geräteregistrierung (3)

d) Im Vergleich dazu sieht der Dialog für den Login, wie vom Nutzer erwartet, durch ein Gerät folgend aus (sofern das Feld 'webauthn_authenticator_data' nicht leer ist):

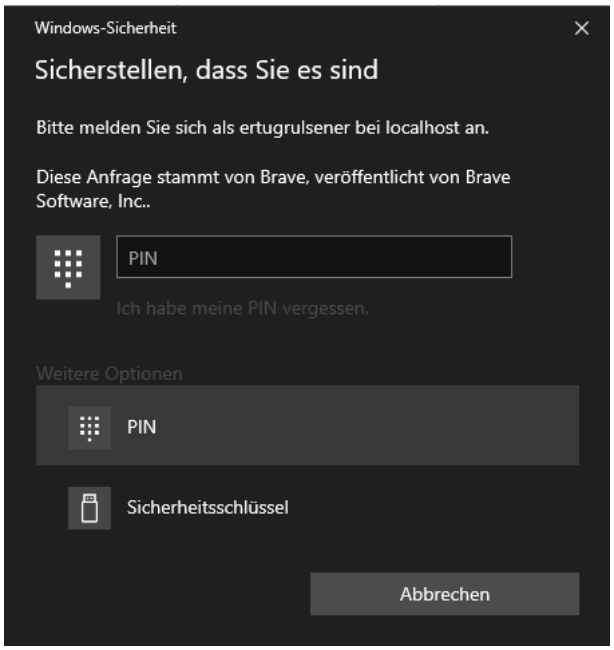



Abbildung 5.5: UX-Problem bei Webauthn Geräteregistrierung (4)

2. **Unerwartete Auswahlmöglichkeiten bei keinem registrierten Authenticator**

   Sollte das Feld ’webauthn_authenticator_data’ allerdings leer sein, erscheint folgender Dialog:

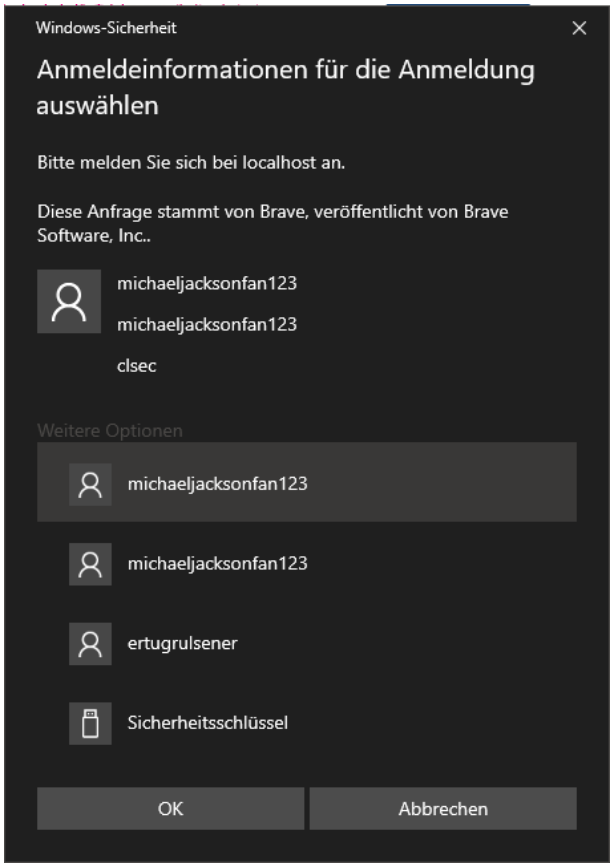
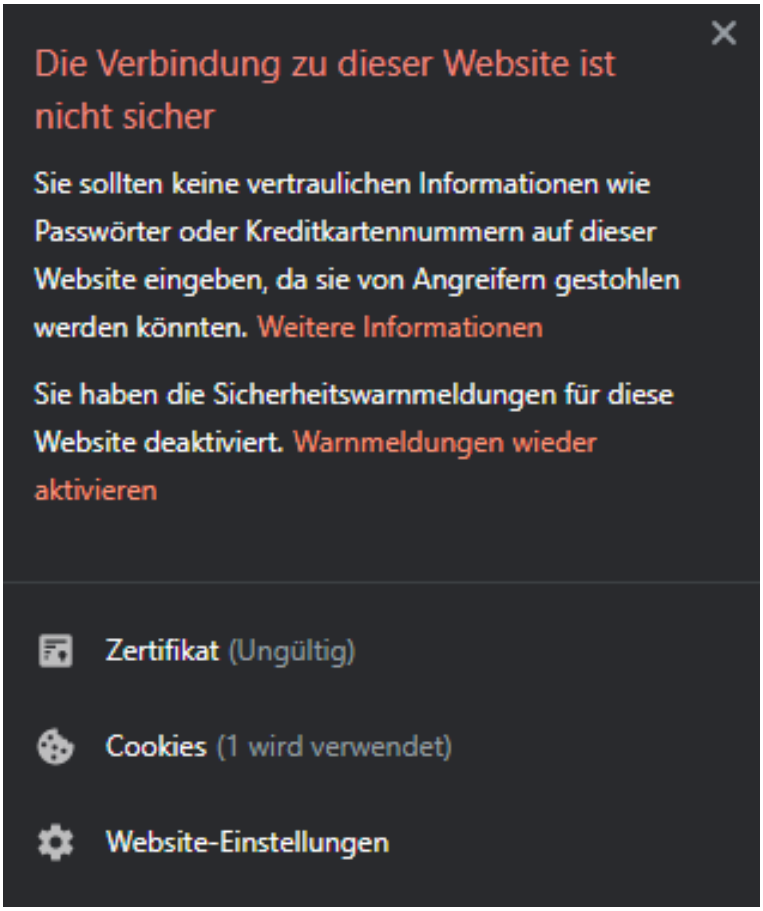



Abbildung 5.6: UX-Problem bei Webauthn Geräteregistrierung (5)

Dieser ist insofern ein Fehler, da kein Sicherheitsschlüssel registriert wurde und bei Auswahl dessen nichts passiert. Gleichzeitig werden hier von Windows alle erstellten ’Profile’ während der Testphase angezeigt, dessen Name gleich verwirrend für den Nutzer ist. Wenn also das Profil mit dem Namen ’michaeljacksonfan123’ versucht einzuloggen, erhält er einen Statuscode 401 (Nicht autorisiert) als Response. Der Fehler hierbei ist die Anzeige einer Einloggoption durch Windows-Sicherheit, die nicht valide ist. Selbst bei bestätigtem und korrektem PIN. Die Felder ’exludedCredentialIDs’ für die Registration und ’allowedCredentialIDs’ für den Login scheinen keine Wirkung zu zeigen, dies könnte einerseits an der WebauthnAPI liegen, die ein Modul für NodeJS ist. Andererseits könnten nicht richtig konfigurierte Optionen für den Login / die Registration der Grund sein.

Außerdem ist die genutzte Libary im Backend nicht vollständig, bei einer eigenen Implementation des Webauthn - Protokolls nach dem W3C könnte dieser Punkt womöglich (durch erheblichen Mehraufwand) gelöst werden.

3. **Selbsterstelltes Zertifikat wird vom Browser als unsicher eingestuft**




4. **Bruteforce-Angriffe können Datenbank überlasten**

   Der Prototyp besitzt keinerlei Zugriffskontrolle. Demnach ist es zum jetzigen Stand möglich, unendlich Anfragen an das Backend zu senden, welches z.B: SQL Anfragen an die Datenbank stellt. Dies könnte die Datenbank überlasten oder sogar den Server crashen. Sollte nämlich kein Pooling mehr möglich sein, wird der Server einen Fehler schmeißen und abstürzen. Dies sollte in einer im Business-Umfeld verwendeten Anwendung implementiert werden, um sowohl den Crash des Servers als auch die Integrität der Daten zu gewährleisten. Ohne solche Abfragen ist das Erraten der Daten des Nutzers (vorallem des Passwortes) nur eine Frage der Zeit und nicht mehr der Möglichkeiten.

# 6 Auswertung

Das Passwort war jahrzehntelanger Vorherrscher in der Authentifikation im Web und schien eine kurze Weile auch sicher. Nach dem die ersten Bruteforce-Angriffe auf Formulare angeggangen wurden, mussten neue Regeln für Passwörter her. Der Nutzer durfte sie nicht mehr nach eigenem Ermessen wählen, da der Imageschaden durch z.B: die Infiltrierung einer wichtigen hochrangigen Person eines Unternehmens zu groß wurde. Die Fälle der Data-Breaches und Leaks wurden trotz Passwort-Policy Ansätzen und der Einführung des zweiten Faktors nicht weniger sondern im Gegenteil: Sie häuften sich mehr und mehr. Passwörter wurden theoretisch durch erzwungene Sonderzeichen und einer Mindestlänge komplexer, allerdings wurde genau hier nicht mit der Natur des Menschen gerechnet. Da Formulare nun mehr und mehr Passwörter abzulehnen schienen, beggangen Menschen die Sicherheitsfeatures instinktiv zu umgehen. Darauf folgten die 'zeitlich begrenzten Passwörter', welche aus dem selben Grund, dem Bequemlichkeitsproblem, scheiterten. Neuere Verfahren sollten die alteingesessenen ersetzen oder zunächst ein Mal unterstützen. Die neue Zeit des Smartphones ermöglichte den Nutzern ihre Geräte zur Authentifikation zu nutzen, wofür sie davor kostspielige Geräte kaufen hätten müssen. Die Zeit der Einmalkennwörter brach an und Apps wie der Google Authenticator oder der Microsoft Authenticator gewannen mehr und mehr an Popularität. Mit der Zeit wurden allerdings auch diese 'weiteren Schritte' zur Authentifikation dem Nutzer zu lästig. Vor allem wichtige Dienste wie Banken führen kein Sessionmanagement und nötigen den User häufig zur mehrmaligen Authentifikation während einer Sitzung. Und genau an diesem Standpunkt setzen die neuesten Verfahren an. Die Webauthentikation ist dem Passwort in vielen Dingen überlegen. Zunächst ein Mal werden nur noch zufällig generierte Schlüssel ausgetauscht. Außerdem geschieht das über Challenge-Response-Verfahren wodurch Wiederholungsangriffe vermieden werden. Der Nutzer ist nicht mehr der Kern der Authentifikation sondern der Authentifikator (das Gerät) selbst. Bei neueren public-private-key Verfahren gibt es keine Geheimnisse, die auf einfachste Art und Weise (z.B: über Keylogger, Trojaner oder Shoulder Surfing) entwendet werden könnten. Die Verantworung zur sicheren Aufbewahrung von Passwörtern hat sich vom Nutzer auf die Betriebssysteme verschoben, die die privaten Schlüssel sichern müssen. Die MITM - Problematik schien immernoch nicht gelöst. So ist die Kommunikation dann durch den MITM angreifbar, wenn der Aussteller des Zertifikats nicht bekannt ist. Die Webauthentikation bietet einem Mann in der Mitte allerdings keine einfache Möglichkeit, die Identität des Nutzers anzunehmen.

Abseits der Problematik bietet der FIDO2 - Standard mit Webauthn und CTAP2 die Kommunikation über verschiedenste Authentifikatoren und bietet damit potenziell sehr vielfältige Lösungsansätze für Authentifikationen im Webbereich. Dennoch ist die Umsetzung des Hauptbetriebssystems Windows aus UX-Sicht nur suboptimal. Gleichzeitig ist die Implementation von Webauthn auf Clientseite zwar größtenteils schon möglich, auf Serverseite fehlt es bei Javascript basierenden Webseiten aber an ausgereiften Web-API's. Die Abstraktion der Vorgänge im Webauthn - Protokoll durch vorhandene API's (wie die genutze im Prototyp) hat den Vorteil, dass der Implementationsaufwand durch eine Senkung der Komplexität sinkt. Gleichzeitig muss bei Fehlern in der abstrahierten Bibliothek ein Mehraufwand an Verständnis dessen, was die Bibliothek macht, aufgetrieben werden. Gemessen daran, dass man das Protokoll allerdings nur einmalig implementieren muss und die FIDO-Alliance große Unterstützung für Webseiten für alle möglichen Programmiersprachen bietet, scheint dieses Argument entkräftigt.

Der Prototyp beweist zweierlei Dinge. Zunächst ein Mal ist es möglich, einen Nutzer bereits mit einem einzigen sicheren Faktor und dessen Nutzernamen zu authentifizieren. Das Passwort sollte im Jahre 2020 eher zur Ausnahme in der Sicherung und wenn überhaupt nur mit einem zweiten Faktor verwendet werden dürfen, sofern es dem User nicht möglich ist die sichereren Verfahren zur Authentifikation zu nutzen. Dies könnte zum Beispiel dann der Fall sein, wenn es dem Nutzer an den benötigten Geräten (der Hardware) fehlt oder die Hardware nicht mit dem Betriebssystem kompatibel ist. Außerdem beweist die Webseite, dass die Implementation der neueren Verfahren keine große Hürde darstellt und einem potenziellen Unternehmen auf lange Sicht viele Supportanfragen zwecks vergessenen Passwörtern oder gekaperten Accounts ersparen kann.

Es gillt noch zu schauen, inwiefern die Anforderungen aus dem Kapitel für sichere Webseiten vom Prototypen erfüllt werden konnten:

- **Immer mehr Nutzer sorgen für mehr Authentifikationen pro Sekunde und damit für eine höhere Auslastung**

  Bei der Wahl der Umgebung des Backends wurde bewusst NodeJS gewählt, das im Kern bereits eine asynchrone Verarbeitung hat. Das heißt das diese Programmiersprache und die genutzte Library 'ExpressJS' in der Lage sind, viele Anfragen verschiedenster Nutzer parallel zu bearbeiten. Die Anfragen blockieren sich nicht untereinander. Dazu besitzt der Prototyp bewusst keine statische Verbindung zur Datenbank sondern ein Pooling. Es baut also bei Serverstart einige (nicht spezifiziert) Verbindungen zur Datenbank auf. Sollte es mal sein, dass eine dieser Verbindungen abbricht, dann stürzt der NodeJS

Server nicht ab sondern kann bei der nächsten Anfrage einfach eine neue Verbindung aus dem Pool für die Anfrage verwenden. Die Authentifikation von vielen verschiedenen Nutzern ist dem Prototypen dadurch problemlos möglich.

- **Webseiten sind über das offene Netz zugänglich und bieten damit eine große Angriifsfläche für Brute-Force Angriffe**

  Der Prototyp besitzt keine konkreten Vorsichtsmaßnahmen gegen Bruteforce - Angriffe. Dies müsste in Folgeversionen implementiert werden, da sonst eben genau die geschilderte Gefahr der Erratung von User Credentials über das Durchprobieren von Passwörtern groß ist. Bei den beiden dem Verfahren der Webauthn ist kein Schutz gegen Bruteforce - Angriffe nötig, da diese keine 'offenen Credentials' wie dem Passwort existieren das erraten werden könnte. Anders sieht es bei der TOTP-Authentifikation aus, hier ist eine Brute-Force Gefahr sehr hoch, da der TOTP Code nur aus einer sechsstelligen Zahl besteht. Das sind genau 1.000.000 Möglichkeiten, die der Angreifer innerhalb von 30 Sekunden (alle 30 Sekunden zyklisch) ausprobieren muss. So würde er die Authentifikation durch ein Gerät umgehen können, dies muss in Folgeversionen verhindert werden und wurde zeitbedingt nicht implementiert. Auch war es geplant diesen Schutz anhand einer Firewall vor der Webseite zu stützen, wodurch dies programmatisch vernachlässigt werden könnte.

- **Webseiten besitzen mehr Möglichkeiten als je zuvor um Nutzer zu authentifizieren, von dem muss sie aber erst Gebrauch machen**

  Die Verfahren des 21. Jahrhunderts werden dem Nutzer wunderbar präsentiert und können von ihm für die Authentifikation genutzt werden. Bei der Username & Passwortauthentifikation fehlt es an einer Registrierung. Abseits dessen, sind die TOTP und Webauthn Registration bereits sicher und bequem implementiert. Auch werden alle drei Kateogorien durch ihre entsprechenden Verfahren bedient.

- **Webanwendungen speichern üblicherweise Nutzerkonten bzw. personenbezogene Daten**

  Personenbezogene Daten werden vom Prototypen nicht gespeichert, da es sich hierbei nur um ein Modul einer Webanwendung handelt, ist dies allerdings auch nicht nötig. Andere Daten wie Passwörter werden mit einem sicheren Hashingverfahren (Salted and Peppered SHA256) in der Datenbank persistiert und sogar auf Clientseite wie empfohlen zusätzlich verschlüsselt und dann übertragen. Wie bereits erwähnt ist die Speicherung von Biometriedaten nicht

nötig, da Webauthn nur einen zufällig generierten Hash als 'credentialID' persistieren muss um die Folgeauthentifikation des Nutzers zu erkennen.

- **Webnutzer sind oft unvorsichtig mit Passwörtern und wählen zu leichte**

  Dieser Punkt war recht schwierig zu verhindern, obwohl die Verfahren wie man ihn umsetzen kann bereits gut erforscht sind. Hier wird die Hauptproblematik von Passwörtern und Passwort Policies gut zusammen. Egal welche Policy für Passwörter gewählt wird, dem Nutzer ist es möglich dennoch unsichere Passwörter zu wählen. Wählt man zu viele Richtlinien wird der Nutzer sich aufgrund seine Beqeuemlichkeit auf kurz oder lang bewusst für diese Unsicherheit entscheiden. Nutzt man gar keine Policy dann ist dem Nutzer gar keine Grenzen an einfachen Passwören wie "1234" gesetzt. Der Prototyp hat nur eine simple Policy: Passwörter und Nutzernamen müssen eine Mindeslänge von 8 Zeichen besitzen. Die Länge ist zwar nicht der einzige Faktor von Unsicherheiten bei Passwörtern, allerdings sind erzwungene Sonderzeichen, wie vom BSI zuvor beschrieben, nicht zielbringend. Dieser Punkte kann von keiner Webseite so richtig so bearbeitet werden, da ein Nutzer im Zweifelsfalle immer seine Bequemlichkeit vor seine Sicherheit stellt was dann zu Identitätsdiebstahl usw. führt.

Zum Ende der Facharbeit muss noch geklärt werden, welches Verfahren für welchen Nutzertypus geeignet ist. Auch wenn man dies nicht pauschalisieren kann, sollte man zunächst ein Mal klären welcher Nutzer welche Anforderungen stellt. Alle Internetnutzer, ob es nun ein erfahrener Entwickler, ein Casual Erwachsener oder Jugendlicher ist, haben Interesse an dem Schutz ihrer Daten. Auch wenn ihnen die Wichtigkeit ihrer Daten und die Gefährdung größenteils nicht bewusst sind. Jugendliche haben heutzutage nur in seltenen Fällen kein Smartphone und können somit das TOTP-Verfahren oder die Webauthn (falls von der Internetseite umgesetzt und untersützt) verwenden. Älteren Personen fehlt es womöglich an technischer Hardware, doch hier kommt die Authentifikation über PIN (sogenanntes self-signed-device) oder die Stimme am Rechner ins Spiel. Auch wenn kein externes Gerät wie ein Smartphone vorhanden ist, kann häufig bereits der eigene Rechner einige Authentifikationsmethoden unterstützen. Sollte man, aus gegebenen und teilweise genannten Gründen, keine Möglichkeit auf Alternativen zum Passwort besitzen, sollte das Passwort nicht als alleiniger Authentifikator verwendet werden dürfen. Ein zweiter Faktor ist aber nur strikt für das Passwort erforderlich, denn die anderen Verfahren sind nicht so großen Gefahren ausgesetzt wie es das Passwort ist. Die Öffentlichkeit muss sich der Wichtigkeit ihrer Identität bewusst werden, um die Entscheidung über Passwörter und dem erzwungenen zweiten Faktor nachzuvollziehen. Zuletzt bleibt noch zu erwähnen: Der Codename des Prototypen 'clsec' ist ein Kürzel

der Wortschöpfung 'Clientless Login' und bedeutet so viel wie "Die Authentifikation ohne Passwort" und ist gleichzeitig ein Beweis dafür, dass diese auch Zukunft hat.

# 7 Ausblick und Fazit

Die Userverifikation ist eine der größten Herausforderungen an Webanwendungen der Neuzeit und wird es voraussichtlich auch bleiben. Das Verfahren der Negativauthentifikation zeigt in welche Richtung zukünftige Verfahren wohl gehen: Man versucht die alteingesessenen Passwörter durch neue Denkansätze und Verfahren besser zu schützen. In diesem konkreten Beispiel macht man wortwörtlich das Gegenteil einer gewöhnlichen Authentifikation und versucht dadurch falsche Anfragen zu erkennen bevor der Server sie überhaupt prüfen muss. Zusätzlich scheint der Zufall in den Fokus zu kehren, während die gewöhnliche Multi-Fator-Authentifikation feste Auswahlmöglichkeiten bot, ist die A-MFA darauf spezialisiert dem User seinen Parametern zugeschnittene Möglichkeiten zu bieten. Das macht es Angreifern ungemein schwer, einen bestimmten Nutzer konkret anzugreifen - Da jeder Nutzer ein potenziell anderes Preset bzw. 'Profil' besitzen wird, das seine Authentifizierungsmöglichkeiten ergibt. Neben neuen Ansätzen muss sich die zukünftige Forschung wohl vermehrt mit der Haltung der Nutzer zum Thema Bequemlichkeit widmen. Neuere Verfahren, die aus technischer Sicht sehr sicher sind, können nicht bestehen wenn sie vom Nutzer nicht als angenehm bzw. bequem empfunden werden. Die Digitalisierung zwingt Webseitenbetreiber (durch öffentlichen Druck und Konkurrenz) dazu, alle vorhandenen Möglichkeiten zur Authentifikation anzubieten und dem Nutzer die Wahl dessen zu lassen. Das Selbe macht der Prototyp, der eine zufällige Webseite als Authentifikationsmodul absichern kann. Um die Verfahren (MFA) noch mehr abzusichern, wäre es in Zukunft nötig die Adaptivität zu steigern, indem die Verfahren dem Nutzer nur präsentiert werden, falls das Userprofil und das vorhandene Betriebssystem / Gerät dies hergibt. Die Authentifikation gelingt den Menschen der Zukunft recht gut, sie muss dem Nutzer nur noch bequem gemacht und angeboten werden. So kann das Passwortproblem mit neuen Verfahren gelöst werden.

# Selbstständigkeitserklärung

Hiermit erkläre ich, Ertugrul Sener, dass die von mir an der *Hochschule für Telekommunikation Leipzig (FH)* eingereichte Abschlussarbeit zum Thema

**„Inwiefern bietet die Authentifikation ohne Passwort Vor- und Nachteile gegenüber Webanwendungen hinsichtlich Nutzerfreundlichkeit, Datenschutz und Sicherheit?“**

selbstständig verfasst wurde und von mir keine anderen als die angegebenen Quellen und Hilfsmittel verwendet wurden.

Leipzig, den 27. Oktober 2020

Ertugrul Sener

# Literaturverzeichnis


[1] Firefox Monitor. *Datenleck einer Webseite: Wattpad*. Deutsch. Internetartikel. URL: https://monitor.firefox.com/breach-details/Wattpad (zitiert auf Seite 7).

[2] L. Abrams. *Wattpad data breach exposes account info for millions of users*. Deutsch. Pressemitteilung. Berlin: Bitkom, Juli 2020. URL: https://www.bleepingcomputer.com/news/security/wattpad-data-breach-exposes-account-info-for-millions-of-users/ (zitiert auf Seite 7).

[3] *WATTPAD DATA BREACH LEADING TO THE LEAK OF 270 MILLION USER RECORDS – ONE OF THE LARGEST DATA BREACHES EVER RECORDED IN THE BOOKS OF HISTORY*. Deutsch. Pressemitteilung. Berlin: Cybleinc, Juli 2020. URL: https://cybleinc.com/2020/07/15/wattpad-data-breach-leading-to-the-leak-of-270-million-user-records-one-of-the-largest-data-breaches-ever-recorded-in-the-books-of-history/ (zitiert auf Seite 7).

[4] *Personal Data and Credentials of 268 Million Users Exposed In Recent Wattpad Hack*. Englisch. Internetartikel. Juli 2020. URL: https://www.riskbasedsecurity.com/2020/07/23/personal-data-and-credentials-of-268-million-users-exposed-in-recent-wattpad-hack/ (zitiert auf Seite 7).

[5] Bundesamt für Sicherheit in der Informationstechni. *Die Lage der IT-Sicherheit in Deutschland 2014*. Deutsch. Dez. 2014. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf?__blob=publicationFile (besucht am 15. 09. 2020) (zitiert auf Seite 7).

[6] *The Biggest Data Breaches in the first half of 2020*. Deutsch. Blog. URL: https://www.keepnetlabs.com/the-biggest-data-breaches-in-the-first-half-of-2020/ (zitiert auf Seite 8).

[7] R. Porath. „Tipps und Tricks für die eigene IT-Sicherheit“. de. In: *Internet, Cyber- und IT-Sicherheit von A-Z*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, S. 369–373. ISBN: 978-3-662-60910-1 978-3-662-60911-8. DOI: 10.1007/978-3-662-60911-8_31 (zitiert auf Seite 9).

[8] L. Gentemann. *Jeder dritte Onliner nutzt dasselbe Passwort für mehrere Dienste*. Deutsch. Pressemitteilung. Berlin: Bitkom, Jan. 2020. URL: https://www.bitkom-research.de/de/pressemitteilung/jeder-dritte-onliner-nutzt-dasselbe-passwort-fuer-mehrere-dienste (zitiert auf Seite 9).

[9] *Jeder Dritte nachlässig bei Passwortwahl*. Deutsch. Pressemitteilung. Berlin: Bitkom, Nov. 2016. URL: https://www.bitkom.org/Presse/Presseinformation/Jeder-Dritte-nachlaessig-bei-Passwortwahl.html (zitiert auf Seite 9).

[10] G. Beuchelt. *Schwache Passwörter — Nutzer spielen weiterhin Vogel Strauß*. de. In: *Wirtschaftsinformatik & Management* 10.5 (Okt. 2018), S. 18–21. ISSN: 1867-5905, 1867-5913. DOI: 10.1007/s35764-018-0094-x (zitiert auf Seite 10).

[11] Bundesamt für Sicherheit in der Informationstechnik. *Sichere Passwörter erstellen*. Deutsch. Internetartikel. Publication Title: Sichere Passwörter erstellen. Mai 2020. URL: https://www.bsi-fuer-buerger.de/BSIFB/DE/Empfehlungen/Passwoerter/passwoerter_node.html (zitiert auf den Seiten 10, 12).

[12] Y. Guo, Z. Zhang, Y. Guo und X. Guo. *Nudging personalized password policies by understanding users' personality*. en. In: *Computers & Security* 94 (Juli 2020), S. 101801. ISSN: 01674048. DOI: 10.1016/j.cose.2020.101801 (zitiert auf Seite 11).

[13] Dennis K. Branstad und Frederick Gallegos. *Auditing Password Usage*. Deutsch. White Paper. Sep. 1988. URL: https://csrc.nist.gov/publications/detail/white-paper/1988/09/01/auditing-password-usage/final (zitiert auf Seite 12).

[14] D. Dasgupta, A. Roy, A. Nag und Springer International Publishing. *Advances in User Authentication*. German. OCLC: 1003245462. 2017. ISBN: 978-3-319-58808-7. URL: https://doi.org/10.1007/978-3-319-58808-7 (besucht am 13. 10. 2020) (zitiert auf den Seiten 15, 16).

[15] D. Dasgupta und R. Azeem. *An Investigation of Negative Authentication Systems*. In: (Okt. 2020) (zitiert auf Seite 16).

[16] A. Nag, D. Dasgupta und K. Deb. *An Adaptive Approach for Active Multi-Factor Authentication*. In: Juni 2014 (zitiert auf Seite 16).

[17] Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutz-Kompendium*. Internetartikel. 2018. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Kompendium/IT_Grundschutz_Kompendium_Edition2018.pdf?__blob=publicationFile&v=7 (besucht am 24. 05. 2020) (zitiert auf Seite 19).

[18] T. Maus. *Das Passwort ist tot — lang lebe das Passwort!* Deutsch. In: *Datenschutz und Datensicherheit - DuD* 32.8 (Aug. 2008), S. 537–542. ISSN: 1614-0702, 1862-2607. DOI: 10.1007/s11623-008-0127-3 (zitiert auf Seite 26).

[19] D. M'Raihi, S. Machani, M. Pei und J. Rydell. *TOTP: Time-Based One-Time Password Algorithm*. RFC 6238 (Informational). RFC. Fremont, CA, USA: RFC Editor, Mai 2011. DOI: 10.17487/RFC6238 (zitiert auf Seite 27).

[20] Dirk Balfanz (Google) et al. *Web Authentication: An API for accessing Public Key Credentials Level 1*. Deutsch. Spezifikation. URL: https://www.w3.org/TR/webauthn/ (zitiert auf Seite 30).

[21] *Anzahl der Internetnutzer weltweit in den Jahren 2013 bis 2016 sowie eine Prognose bis 2021*. Deutsch. Statistik. Weltweit: Statista, Nov. 2019. URL: https://de.statista.com/statistik/daten/studie/369356/umfrage/prognose-zur-anzahl-der-internetnutzer-weltweit/ (zitiert auf Seite 33).

[22] *Anteil der Haushalte weltweit mit Internetzugang von 2002 bis 2019*. Deutsch. Statistik. Weltweit: Statista, Nov. 2019. URL: https://de.statista.com/statistik/daten/studie/187116/umfrage/anteil-der-haushalte-mit-internetzugang/ (zitiert auf Seite 33).

[23] Landesbeauftragter für Datenschutz und Informationsfreiheit Baden-Württemberg. *Umgang mit Passwörtern*. Deutsch. Internetartikel. Juni 2010. URL: https://www.baden-wuerttemberg.datenschutz.de/umgang-mit-passwortern/ (zitiert auf Seite 34).

[24] M. Hub, Renáta Myšková und Jan Čapek. *Relationship between security and usability – authentication case study*. Deutsch. Pressemitteilung. 2011. URL: http://www.universitypress.org.uk/journals/cc/19-853.pdf (zitiert auf Seite 39).

[25] C. Braz und J.-M. Robert. *Security and usability: the case of the user authentication methods*. en. In: *Proceedings of the 18th international conference on Association Francophone d'Interaction Homme-Machine - IHM '06*. Montreal, Canada: ACM Press, 2006, S. 199–203. ISBN: 978-1-59593-350-8. DOI: 10.1145/1132736.1132768 (zitiert auf Seite 39).

[26] M. Keith, B. Shao und P. J. Steinbart. *The usability of passphrases for authentication: An empirical field study*. en. In: *International Journal of Human-Computer Studies* 65.1 (Jan. 2007), S. 17–28. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2006.08.005 (zitiert auf Seite 39).

[27] M. Rohr. *Sicherheit von Webanwendungen in der Praxis: Wie sich Unternehmen schützen können – Hintergründe, Maßnahmen, Prüfverfahren und Prozesse*. German. OCLC: 1030597976. 2018. ISBN: 978-3-658-20145-6 (zitiert auf Seite 40).

[28] *Grundlagen der Sicherheit bei Webanwendungen*. Deutsch. Techn. Ber. Stuttgart: Hochschule für Technik Stuttgart, Nov. 2006. URL: https://static.twoday.net/ChristianG78/files/neueRohfassung.pdf (zitiert auf Seite 41).

[29] H. Garstka. „Informationelle Selbstbestimmung und Datenschutz“. de. In: *Bürgerrechte im Netz*. Hrsg. von C. Schulzki-Haddouti. Wiesbaden: VS Verlag für Sozialwissenschaften, 2003, S. 48–70. ISBN: 978-3-8100-3872-2 978-3-322-92400-1. DOI: 10.1007/978-3-322-92400-1_5 (zitiert auf Seite 41).

# Abkürzungsverzeichnis

**2FA** Zwei-Faktor-Authentifizierung . . . . . 30

**BSI** Bundesamt für Sicherheit in der Informationstechnik . . . . . 10, 19

**CTAP** Client-to-Authenticator protocol . . . . . 21, 25, 29, 30, 31, 45

**DPPP** dynamic personalized password policy . . . . . 11

**HTTPS** Hypertext Transfer Protocol Secure . . . . . 12, 14

**MITM** Man-in-the-middle . . . . . 12, 13, 14, 40, 41, 57

**NAS** Negative authentication . . . . . 15

**OTP** One time password . . . . . 27

**REST** Respresentational State Transfer . . . . . 39

**SPA** Single Page Application . . . . . 43

**TOTP** Time based one time password . . . 27, 28, 36, 37, 40, 41, 42, 45, 47

**U2F** Universal second factor . . . . . 21, 25, 29

**URL** Uniform Resource Locator . . . . . 14

**W3C** World Wide Web Consortium . . . . . 19, 29, 30, 55

**WebAuthn** Web Authentication . . . . . 25

# Abbildungsverzeichnis