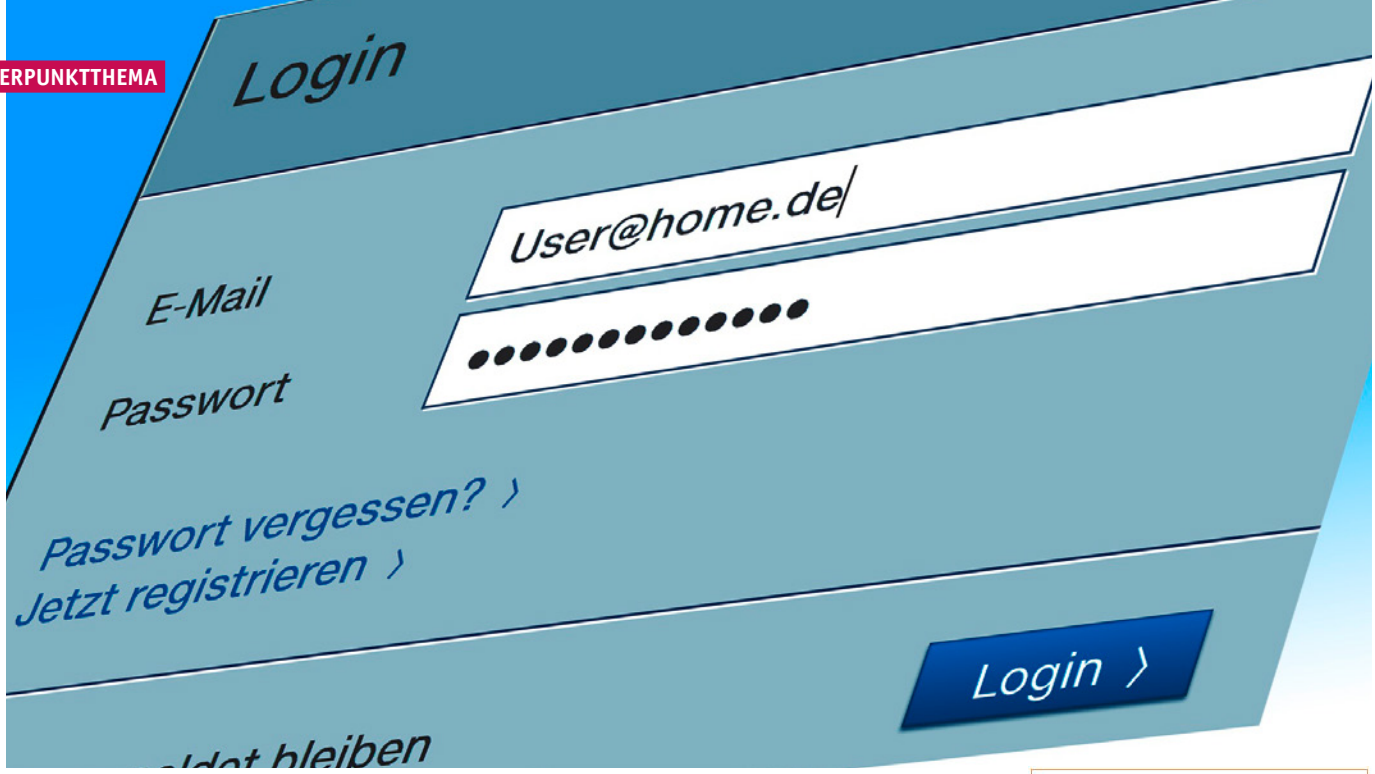


Sonderdruck aus JavaSPEKTRUM

Wer bin ich?

Multifaktor-Authentifizierung – Sicheres Anmelden mit und ohne Passwort

Gregor Tudan, Markus Koschier

Passwörter nerven! „Bitte mindestens 8, höchstens 17 Zeichen, Groß-/Kleinschreibung, mindestens 3 Ziffern, ein kyrillischer Buchstabe und mit Emojis" – so oder so ähnlich klingen die verzweifelten Versuche mancher Webseiten, trotz omnipräsenter Leaks ihren Benutzern noch ein Gefühl von Sicherheit zu vermitteln. Diese sind dann wiederum genervt und verwenden überall dasselbe Kennwort vom Post-it auf dem Monitor. Vielleicht ist der Ansatz „Kennwort" zu sehr in die Jahre gekommen und es ist Zeit für Neues. Der nachfolgende Artikel stellt vor, welche alternativen Authentifizierungsverfahren es gibt und welche APIs und Bibliotheken zur Absicherung von Java-Anwendungen helfen.

Das übliche Authentifizierungsverfahren für viele Webseiten ist nach wie vor die Kombination aus Benutzername und Passwort. Im Laufe der letzten Jahre traten die Schwächen und Probleme dieses Verfahrens allerdings immer deutlicher zutage.

Erstens stellt die Verwendung eines (sicheren) Passworts viele Benutzer vor das Problem, sich dieses Kennwort dann auch zu merken. Ein Passwort-Manager kann zumindest für dieses Problem Abhilfe verschaffen.

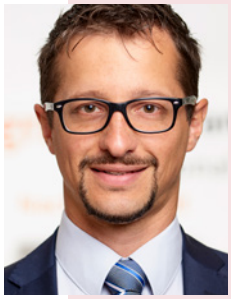
Das zweite Problem betrifft die Komplexität beziehungsweise Sicherheit von Passwörtern: Auch wenn mittlerweile viele Webanwendungen Minimalanforderungen à la „mindestens 8 Zeichen, jeweils ein Groß-/Kleinbuchstabe, Ziffer und Sonderzeichen" stellen, genügen bei Weitem nicht alle eingesetzten Passwörter diesen Kriterien. Die jüngste Veröffentlichung der zehn häufigsten Passwörter in Deutschland vom Hasso Plattner Institut [HPI] fällt diesbezüglich sehr ernüchternd aus. Den meisten Deutschen scheinen die Zifferntasten für ihr Kennwort zu reichen. Derart kurze und wenig sichere Passwörter sind anfällig für Brute-Force- und Wörterbuch-Attacken.

Wer kann sich noch an den Facebook-Vorgänger MySpace erinnern? Bereits vor mehreren Jahren konnten Hacker 400 Millionen Datensätze inklusive Passwörter von der Plattform erbeuten. Diese kursieren immer noch im Netz und stellen ein weiteres Sicherheitsproblem dar. Wieso? Weil mehr als die Hälfte der Deutschen dasselbe Passwort für mehrere Konten verwendet. So kann selbst ein Passwort für einen längst nicht mehr existierenden Dienst zu einem riesigen Problem werden.

Last but not least sitzt eine der größten Sicherheitslücken häufig noch vor der Tastatur. Wer für jeden Dienst ein möglichst sicheres und individuelles Passwort wählt, wird ohne Einsatz eines Passwort-Managers Probleme haben, sich alle Passwörter zu merken. Das führt in der Praxis zu den beliebten Post-its, die auf den Monitor aufgeklebt sind und eines oder mehrere Passwörter enthalten – eventuell sogar noch ergänzt um den Namen des Diensts oder den Benutzernamen. Das öffnet Verfahren wie Social Engineering natürlich Tür und Tor.

Diese vier Beispiele zeigen, dass klassische Authentifizierungsverfahren mit einer Kombination aus Benutzername und Passwort nicht mehr zeitgemäß sind. Gleichzeitig schränkt jede Erhöhung der Security auch die Usability für den Benutzer ein. Um hier einen möglichst guten Kompromiss zu erreichen, bieten sich Verfahren mit Multifaktor-Authentifizierung an. Die Grundidee dahinter ist, dass der Benutzer einen mehrstufigen Anmeldeprozess durchläuft. Im Zuge dessen werden mindestens zwei der nachfolgenden und voneinander unabhängigen Merkmale kombiniert:

Gregor Tudan ist IT-Consultant bei Cofinpro und arbeitet als Entwickler und Architekt mit Schwerpunkt IT-Security an der Webplattform eines großen deutschen FinTechs mit.
E-Mail: gregor.tudan@cofinpro.de



Markus Koschier verfügt als IT-Berater über mehr als zehn Jahre Berufserfahrung im Finanzsektor. Er begleitet aktuell innovative Digitalisierungsvorhaben und setzt dabei das Augenmerk auf agile Vorgehensmodelle bei gleichzeitiger Erfüllung regulatorischer und sicherheitsrelevanter Anforderungen.
E-Mail: markus.koschier@cofinpro.de

- etwas, das der Anwender weiß (beispielsweise ein Passwort),
- etwas, das der Anwender hat (beispielsweise ein Token),
- etwas, das der Anwender ist (biometrische Merkmale).

Durch dieses mehrschichtige Verfahren sind Systeme robuster gegenüber Angriffen wie Brute-Force-Attacken. Außerdem wird im Gegensatz zu Passwörtern der Diebstahl früher bemerkt und Phishing-Attacken sind für Angreifer deutlich schwerer.

Versand von Tokens

Zu den einfachsten mehrschichtigen Verfahren gehört der Versand von Codes per SMS oder E-Mail: Nutzer von Slack oder WhatsApp kennen diese „Magic Links" oder SMS-Codes. Benutzer können sich damit einloggen, sofern sie Zugriff auf die Mobilfunknummer oder das E-Mail-Postfach haben.

Solche Verfahren sind leicht zu implementieren: Nach dem Login mit Benutzername und Kennwort generiert der Auth-Server in einem zweiten Schritt einen zufälligen Code und verschickt ihn. Erst nach dem Klick auf den Link oder der korrekten Eingabe des Codes ist die Anmeldung abgeschlossen.

Hierbei ist meist das Telefon der zweite Faktor – und da liegt auch das Problem: Webseiten werden schon lange nicht mehr nur am Desktop benutzt. Bei Verlust des Smartphones mitsamt gespeicherten Kennwörtern, SIM-Karte mit der für den SMS-Empfang genutzten Rufnummer und konfiguriertem E-Mail-Account heißt es: Game over.

Beim Einsatz in besonders kritischen Umfeldern sollte auch bedacht werden, dass hier oft die Infrastruktur Dritter im Spiel ist – zum Beispiel ein SMS-Gateway oder der E-Mail-Hoster. Diesen müssen alle Beteiligten vertrauen und Angreifer erhalten einen weiteren möglichen Punkt zum Abgreifen von Daten.

Apps und TOTP

Etwas ausgefeilter sind App-basierte Verfahren. Hier gibt es mit dem „*T*ime-based *O*ne-time *P*assword"-Algorithmus (TOTP) einen Standard [IETF], den viele große Unternehmen wie Google, Twitter, GitHub oder Amazon nutzen. Hierbei tauschen Anbieter und Benutzer einmal einen geheimen Schlüssel aus, meistens durch Scannen eines QR-Codes.

Die Apps generieren Anmeldecodes, indem sie aus dem Schlüssel und der aktuellen Uhrzeit ein Hash erzeugen. Die Uhrzeit wird dabei auf beispielsweise 30 Sekunden gerundet, wodurch der Code auch genau so lange gültig ist. Der Benutzer gibt den Code in einem zweiten Schritt bei der Anmeldung mit ein. Errechnet der Server mit den gleichen Parametern (Uhrzeit und Geheimnis) denselben Code, ist die Anmeldung erfolgreich.

TOTP ist ein Standard und bietet daher Vorteile für Anbieter und Nutzer: Anbieter profitieren von fertigen und geprüften Bibliotheken für fast alle Frameworks und Sprachen, Benutzer verwalten alle Codes für verschiedene Seiten in einer App (s. Abb. 1). Auch viele Passwort-Manager unterstützen TOTP bereits. Da das Verfahren keine Kommunikation zwischen Telefon und Anwendung erfordert, funktioniert es auch ohne Mobilfunkempfang oder Zugang zum Firmen-VPN. Verloren gegangene Geräte lassen sich durch Ändern des Geheimnisses auf Serverseite aussperren.

Der Request for Comments [IETF] zu TOTP enthält auch eine Referenzimplementierung in Java. Es gibt aber auch fertige Plug-ins für fast alle populären Auth-Server und -Frameworks wie Keycloak, Spring-Security oder Aerogear.

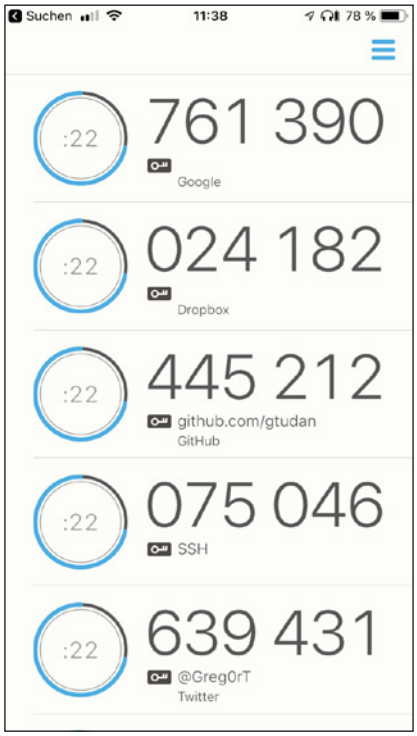


Abb. 1: TOTP-App mit Keys unterschiedlicher Anbieter

Hardware-Tokens

Ähnliche Verfahren gibt es auch in Hardware: Wer schon einmal in einem ICE mit Pendlern gesessen hat, hat bestimmt beim einen oder anderen Fahrgast RSA-Tokens gesehen. Sie eignen sich insbesondere für interne Anwendungen und funktionieren ähnlich wie TOTP – auch hier werden ein ab Werk festgelegtes Geheimnis (auch Seed genannt) und eine Uhr zur zufälligen Erzeugung von Codes verwendet.

Der Vorteil von Hardware-basierten Verfahren liegt in ihrer Unempfindlichkeit gegen Softwareangriffe. Sie lassen sich nicht kopieren und sind deutlich schwerer zu manipulieren, da sie bei dem Versuch meist zerstört werden. Gegen sie spricht ihr Preis und dass es nicht besonders bequem ist, immer ein separates Gerät mit sich tragen zu müssen. Zudem können sie ohne Schutzmaßnahmen wie PIN-Codes leicht gestohlen werden.

Der fest eingebrannte Seed wurde Hardware-Tokens bereits zum Verhängnis: Im Jahr 2011 bot der Marktführer RSA seinen Kunden den Austausch von SecurID-Tokens an, nachdem bei einem Hackerangriff mutmaßlich Informationen zum Verfahren für die Generierung dieser Secrets entwendet wurden [ARS11].

Etwas moderner geht es mit FIDO-kompatiblen Hardware-Tokens, wie von YubiKey, oder den gerade eingeführten Titan-Keys von Google. FIDO (steht für Fast IDentity Online) ist eine Allianz aus verschiedenen Unternehmen mit dem Ziel, offene Standards für Authentifizierung voranzubringen. Diese Tokens werden zum Beispiel per USB mit dem Rechner verbunden, wobei es un-

terschiedliche Bauformen gibt: Manche sind als Schlüsselanhänger konzipiert, andere sind so klein, dass sie komplett in der Buchse verschwinden (s. Abb. 2). Für mobile Anwendungen gibt es auch Ausführungen mit NFC oder Bluetooth, die sich mit dem Handy koppeln lassen.

Die Kernidee zeigt Abbildung 3: Der Token generiert bei der Registrierung einen öffentlichen und einen privaten Schlüssel. Der öffentliche Schlüssel wird an den Dienst geschickt, der private verbleibt auf dem Token und verlässt diesen nie. Bei einer Anmeldung schickt der Dienst einige zufällige Bytes (die sogenannte Challenge) und lässt diese vom Token mit dem privaten Schlüssel signieren und zurückgeben. Mithilfe des gespeicherten öffentlichen Schlüssels kann der Dienst dann die Echtheit des Tokens überprüfen.

Abb. 2: Ein Yubikey – der Metallbügel am oberen Ende ist berührungsempfindlich

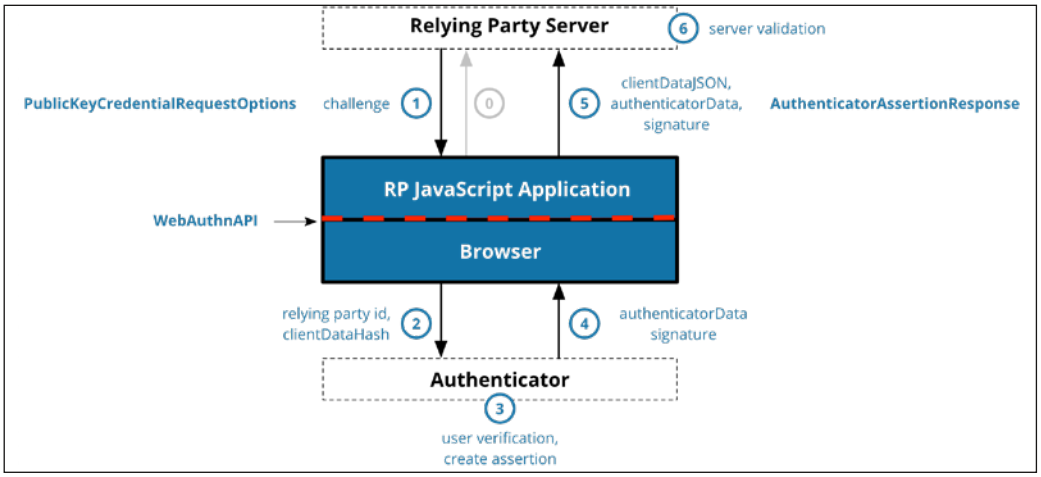


Abb. 3: WebAuthn Authentication flow

Da nur der öffentliche Schlüssel auf dem Server der Betreiber gespeichert wird, nimmt der Schaden von Datenleaks erheblich ab – selbst wenn bei einem Leak der Schlüssel bekannt wird, ist noch nicht alles verloren. Das ist ein klarer Vorteil gegenüber TOTP, wo der Betreiber das gemeinsame Geheimnis kennen muss.

Die direkte Verbindung mit dem Rechner hat den Vorteil, dass diese Art von Authentifizierung robuster gegen Social Engineering ist: Es gibt keinen Code, den die Benutzer verraten können. Allerdings sind die Tokens damit wieder anfällig gegen Angriffe durch Malware. Zur Abmilderung dieses Nachteils schreibt der Standard vor, dass der Benutzer zur Aktivierung des Tokens mit diesem interagieren muss. Die meisten Tokens lösen dies durch einen Knopf, oder eine berührungsempfindliche Metallfläche. Das soll verhindern, dass Webseiten oder Software ohne Zustimmung des Benutzers mit dem Key interagieren und so massenhaft Antworten des Tokens sammeln, um damit unbemerkt Codes abzugreifen oder Rückschlüsse auf den privaten Key zu ziehen.

```
let credential = await navigator.credentials.create({ publicKey: {
  challenge: Uint8Array(32) [12, 105, 53, 102, 32, 212, …]  // (1)
  rp: { id: "cofinpro.de", name: "Cofinpro AG" },           // (2)
  user: {                                                   // (3)
    id: Uint8Array(8) [129, 101, 79, 71, 54, 209, 102, 58]
    name: "gtudan",
    displayName: "Gregor Tudan"
  },
  pubKeyCredParams: [{                                      // (4)
    type: "public-key",
    alg: -7
  }]
}});
```

Listing 1: Registrierung von Credentials, (1) Challenge, bestehend aus 32 zufälligen Bytes, (2) Informationen zur „Relaying Party" (Dienstanbieter), (3) Benutzerdaten, (4) Parameter zur Erzeugung des Public-Keys (hier mit dem Algorithmus „ES256")

Signaturgestützte Verfahren in Hardware sind nicht neu und in Form von Smartcards schon länger verbreitet, zum Beispiel für die Anmeldung bei Windows. Doch WebAuthn, ein neuer Standard des W3C [W3C], bringt Tokens zur Zwei-Faktor-Authentifizierung nun auch ins Web. Dieser enthält auch ein JavaScript-API, das die Interaktion von Websites mit Security-Tokens über den Browser ermöglicht. Der Standard löst die bisherige Programmierschnittstelle „U2F" (steht für Universal Second Factor) der FIDO-Allianz ab und ist zu dieser rückwärtskompatibel.

Der WebAuthn-Standard ist zwar noch vergleichsweise jung, findet allerdings schnell Verbreitung. Viele große Dienste unterstützen ihn oder seinen Vorgänger U2F bereits, darunter Google, Microsoft, Amazon-AWS, Dropbox, GitHub und GitLab.

Wie das genau aussieht, zeigt Listing 1: Der Server des Dienstanbieters erzeugt eine Challenge und fordert den Browser über das JavaScript-API dazu auf, ein neues Schlüsselpaar zu erzeugen. Nachdem der Benutzer zugestimmt hat (zum Beispiel durch Berühren des Tokens), wird der Public Key zurückgegeben und auf dem Server im Profil des Benutzers gespeichert. Ein Benutzer kann mehrere Tokens registrieren, was auch eine gute Idee ist: So gibt es bei Beschädigung oder Verlust des Tokens Ersatz.

Der Login läuft analog dazu: Der Server erzeugt wieder eine Challenge und übergibt die Liste mit registrierten Credentials an den Browser. Dieser wählt eine ID aus, die gerade mit dem Rechner verbunden ist und lässt diesen die Challenge mit dem Private Key signieren, nachdem der Benutzer bestätigt hat. Die signierte Challenge wird an den Server übertragen, der wiederum mit dem gespeicherten Public Key die Echtheit überprüft.

Im Moment unterstützen die Browser Chrome, Edge und Firefox den WebAuthn-Standard. Im Safari-Technology-Preview ist WebAuthn auch schon enthalten – ob und wann es in die reguläre Version einzieht, darüber gibt sich Apple gewohnt schweigsam. Da Apple ebenfalls Mitglied der WebAuthn-Gruppe ist, besteht aber durchaus Hoffnung.

Google und Yubico (Hersteller der Yubikeys) bieten Java-Bibliotheken für die Integration von WebAuthn auf Serverseite an. Außerdem gibt es Standalone-Server zum Anbinden per REST-API. Einige Identity-Provider-Produkte wie beispielsweise ForgeRock haben WebAuthn bereits integriert, andere werden in absehbarer Zeit nachziehen.

Doch werden Hardware-Tokens weite Verbreitung außerhalb des professionellen Umfelds finden? Vermutlich nicht, denn wenn es um das bekannte Spannungsfeld Bequemlichkeit versus Sicherheit geht, entscheiden sich viele doch lieber für den bequemen Weg. Das führt nahtlos zum Thema Biometrie – denn was ist bequemer als ein Schlüssel, den man ohnehin immer dabei hat?

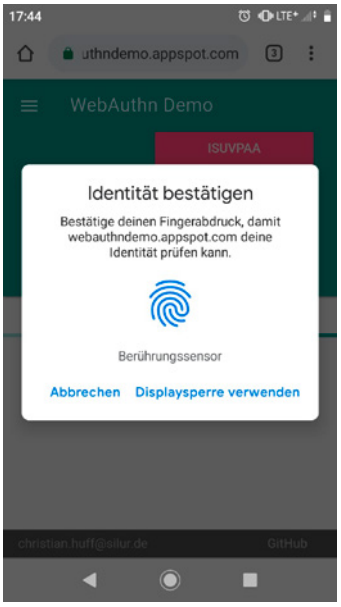


Abb. 4: Authentifizierung per Fingerabdruck in Android

Biometrie

In vielen Telefonen stecken mittlerweile Fingerprint-Sensoren und Apple macht mit FaceID die Gesichtserkennung massentauglich. Dies ist sicher kein Verfahren für jedermann – nicht jeder fühlt sich damit wohl, von der Kamera beobachtet zu werden oder seine Fingerabdrücke zu hinterlassen.

Bisher war biometrische Authentifizierung allein Desktop-Anwendungen vorbehalten. Anwendungen im Web, wie Gesichtserkennung durch die WebCam, Spracherkennung über das Mikrofon oder Erkennung der Benutzer über das Tippverhalten blieben trotz existierender APIs in den Browsern Nischenprodukte. Auch das könnte sich mit WebAuthn jetzt ändern: Der Standard sieht nämlich nicht nur Hardware-Tokens vor, sondern unterstützt auch sogenannte Soft-Tokens: Hier übernimmt das Betriebssystem oder der Browser das Erzeugen und Speichern der Schlüssel. Zur Freigabe wird dabei der Fingerprint-Sensor des Telefons verwendet, bei neueren Macs mit TouchBar alternativ die TouchID. Vermutlich wird auch die Gesichtserkennung mit FaceID bald möglich sein, falls WebAuthn in Safari unterstützt wird. Bisher sind Microsoft Edge (mit Authentifizierung über Windows Hello) und Chrome unter Windows, Mac und Android dabei (s. Abb. 4).

Da Benutzer keine Extra-Geräte kaufen müssen, sondern ihre vorhandene Hardware nutzen, könnte dies die Verbreitung von Biometrie noch einmal deutlich voranbringen. Auch hier kommt wieder das Verfahren mit Public und Private Key zum Einsatz – der Fingerabdruck dient nur zum Freischalten des Schlüssels und bleibt auf dem Gerät. Geht das Smartphone verloren, hilft auch der Fingerabdruck allein nicht mehr weiter. Das ist auch im Sinne der Nutzer, schließlich lässt sich ein Fingerabdruck nicht mehr tauschen, falls ein Foto davon im Internet landet.

Unternehmen mit gesteigertem Sicherheitsbedürfnis können Hardware-Tokens mit Biometrie kombinieren: Der erzeugte Public Key auf den Tokens wird nämlich mit einem sogenannten Attestation-Certificate signiert. Das wird bei der Herstellung des Geräts eingebracht und identifiziert das Modell. Bei der Registrierung kann der Serverbetreiber entscheiden, nur bestimmte an die Mitarbeiter ausgegebene Geräte zuzulassen. Statt einer einfachen Berührung benötigen diese dann einen Fingerabdruck zur Freigabe. Samsung erlaubt in seinen Smartphones auch die Freigabe per Iris-Scan mit der Front-Kamera.

Biometrie kann als zusätzliche Hürde bei der Authentifizierung durchaus sinnvoll sein – gänzlich auf sie verlassen sollte man sich aber derzeit lieber nicht. Bisher wurde noch jedes Verfahren mit zum Teil einfachsten Mitteln wie Fotos ausgetrickst. [STBG]

Fazit

Auch wenn sich die klassische Anmeldung mit der Kombination aus Benutzername und Passwort nach wie vor hoher Beliebtheit erfreut, lässt die Sicherheit solcher Verfahren stark zu wünschen übrig. Zwei- beziehungsweise Mehrfaktor-Authentifizierungsverfahren schaffen hier Abhilfe. Durch die Kombination mehrerer Sicherheitsmerkmale erhöht sich die Robustheit von Anwendungen gegenüber Angriffen.

Die Bandbreite dieser Verfahren ist in den letzten Jahren immer größer geworden: Von passwortlosen Magic Links über App-basierte TOTP-Verfahren und Hardware-Tokens bis zu Biometrie ist sowohl Hardware- als auch Software-basiert die Auswahl enorm. Auch die großen Software- und Browser-Hersteller haben nachgerüstet und unterstützen standardisierte Verfahren.

Dennoch gibt es auch hier die altbekannte Herausforderung, dass sich Usability und Sicherheit nicht immer unter einen Hut bringen lassen. Und wer glaubt, dass durch neue Verfahren allein maximale Sicherheit erreicht wird, dem sei noch eine alternative Definition der verschiedenen Sicherheitsmerkmale mitgegeben:

- etwas, das man vergessen kann,
- etwas, das man verlieren kann,
- etwas, das man nicht verstecken kann.

Literatur und Links

[ARS11] P. Bright, RSA finally comes clean: SecurID is compromised, ars Technica, 2011, https://arstechnica.com/?post_type=post&p=41753

[GOGL] Google, WebAuthn-Demo, https://github.com/Yubico/java-webauthn-server

[HPI] Die Top Ten deutscher Passwörter, Hasso Plattner Institut, 2018, https://hpi.de/pressemitteilungen/2018/die-top-ten-deutscher-passwoerter.html

[IETF] Internet Engineering Task Force, RFC-6238, TOTP: Time-Based One-Time Password Algorithm, https://tools.ietf.org/html/rfc6238

[STBG] Starbug: Ich sehe, also bin ich … Du, auf dem 31C3, https://media.ccc.de/v/31c3_-_6450_-_de_-_saal_1_-_201412272030_-_ich_sehe_also_bin_ich_du_-_starbug

[W3C] W3C, Web Authentication: An API for accessing Public Key Credentials, https://www.w3.org/TR/webauthn/

[YUBI] Yubico, Java-Webauthn-server: Server-side Web Authentication library for Java, https://github.com/Yubico/java-webauthn-server