

FIDO2

UAF, U2F, W3C und nun...

Security Meetup Kassel, 20.02.2019

Cornelius Kölbel

Wer bin ich

- Cornelius Kölbel
- 2FA seit 2005
- U2F seit 2013
- cornelius.koelbel@netknights.it



[BEST PRODUCTS](#)[REVIEWS](#)[NEWS](#)[VIDEO](#)[HOW TO](#)[SMART HOME](#)[CARS](#)[DEALS](#)[JOIN / SIGN IN](#)[SECURITY](#) | [LEER EN ESPAÑOL](#)

Massive breach leaks 773 million email addresses, 21 million passwords

The best time to stop reusing old passwords was 10 years ago. The second best time is now.

BY ALFRED NG | JANUARY 17, 2019 8:40 AM PST



source: <https://www.cnet.com/news/massive-breach-leaks-773-million-emails-21-million-passwords/>

Probleme

- SQL Injection und was auch immer.
- **Phishing.**

Meet the FIDO Alliance

- <https://fidoalliance.org>

Simpler, Stronger Authentication

Solving the World's Password Problem

EXPLORE THE BENEFITS

What?

standards that enable
secure user
experiences across many
services.



FIDO2

Introducing a new era of ubiquitous,
phishing-resistant, strong authentication to
protect internet users worldwide.

LEARN MORE >



2019 State of Strong Authentication

Javelin Strategy & Research's report
contains an analysis on the state of
consumer and enterprise authentication
and the role of strong authentication.

DOWNLOAD >

Videos

Standards Now Adopted As ITU

FIDO, Financial Inclusion, and Digital Financial Services

Case Studies



































Google Case Study

White Papers

Recommended Account Recovery

Meet the FIDO Alliance /2

- Will eine sichere Authentisierungsmöglichkeit für den Enduser im Internet schaffen.
- "Fast IDentity Online"

Board Level Members				
				
				
				
				
				
				
				

Geschichte

- 2012: Start
- 2013: Gründung der FIDO Alliance.
- 2013 bereits erste Drafts zu U2F. Treiber Google und Yubico
- 2014: Offizielles Release der Spec zu U2F und UAF.
- seit 2015: Arbeit an FIDO2

Wiederholung U2F



- Ein **Schlüsselpaar** wird **pro Dienst** generiert und dort registriert.
- Ein Dienst **weiß nichts** vom anderen.
- Privater Schlüssel "bleibt auf dem Authenticator".
 - => Kompromittierung nicht "ansteckbar".

Wiederholung U2F /2



- Schlüsselpaar ist an die Domäne gebunden (AppID, Trusted Facets).
 - => Kein Phishing möglich

Technik U2F

Registrierung

- "Authenticator" erzeugt Schlüsselpaar.
- Der private Schlüssel wird an AppID oder Trusted Facets gebunden.
- **Key-Handle** und **Public-Key** werden vom "Authenticator" zurückgegeben und vom Dienst gespeichert.

Spezialfall Yubikey

- Schlüsselpaar wird abgeleitet aus:
 - **Masterkey**
 - **AppID** oder Trusted Facets
 - **Nonce**, die als Key-Handle fungiert

Bonus: Attestation Certificate

- Zertifikat des Herstellers
- Wird bei der Registrierung mitgesendet (mit Signatur einer Challenge)
- Soll die Güte des Gerätes beglaubigen

Trivia: Gleiches Zertifikat, Laufzeit u2fzero, nur ausgewählte Hersteller registrieren

Technik U2F

Authentifizierung

- Server sendet:
 - Challenge
 - Key-Handle (oder mehrere)
 - App-Id (oder trusted facets)
- Anhand des **Handles** findet der "Authenticator" den privaten Schlüssel
- prüft, ob dieser für die **AppID** gültig ist
- ...und signiert die **Challenge**.

Beispiel Github

- Anmeldung an Github mit einem U2F Token
- Blick in Trusted Facets:
https://github.com/u2f/trusted_facets

```
156 <div class="auth-form-body u2f-auth-form-body js-u2f-auth-form-body mt-3">
157   <span class="u2f-enabled">
158     <!-- '"` --><!-- </textarea></xmp> --></option></form><form class="js-u2f-auth-form" data-app-
159 id="https://github.com/u2f/trusted_facets" data-challenge="EK0rxi_Sucmyo8eqTTmgVhahxzfU9GV0w2paas3GBx4" data-sign-
requests="[{"&quot;version&quot;:&quot;U2F_V2&quot;;&quot;keyHandle&quot;:&quot;Rv0eVm9sU9606ivpI95oTQl-RmpGCwb0-
HEZkB7AiaT-j89iC-8zJHbSeq0UbXeML9BKGX1wd5iFaF-HVSNr5g&quot;;},
{"&quot;version&quot;:&quot;U2F_V2&quot;;&quot;keyHandle&quot;:&quot;4b2WhRcW00k2gfdPk9C00sSdCmFrV__4GwuvzxuDT-
Vqelp2hCuLIp0RvmehGsvvgvQ4-tPqewoL8sXFzt-hYg&quot;;}]" action="/sessions/two-factor/security-key" accept-
charset="UTF-8" method="post"><input name="utf8" type="hidden" value="&#x2713;" /><input type="hidden"
name="authenticity_token"
value="o3ilfqILLkudkfVn3EwzEJKexU69WKAzoni0k9/i4zXbfwec+m3otbUaikqWBTRAt+4lCw8UGy+UvsaWf7SNmQ==" />
```

































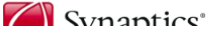



Gedanken zur AppID und Trusted Facets

- Konzept der AppID: Einmal registrieren, einmal nutzen.
 - Gut gegen Phishing
 - Gut für Endanwender
 - Schlecht für Enterprise
- Trusted Facets: Einmal registrieren, mehrmals nutzen.
 - Gut, aber:
 - Nur für Hosts innerhalb einer Domäne möglich (github.com, garage.github.com, admin.github.com)

Beschränkungen oder Probleme mit U2F

- No Passwordless
- Immer ein Benutzername
- Gräßliche Dokumentation
- Keine einheitliche Implementierung

You had one job and failed

<div><div><div><div><div><div>fido</div><div>alliance</div></div><div><div>member</div><div>authentication</div></div></div></div><div><div>THE ALLIANCE</div><div>STANDARDS & TECHNOLOGY</div><div>DISCOVER FIDO</div><div>FIDO® CERTIFIED</div><div>NEWS & EVENTS</div></div><div><div>Q</div></div></div></div>					
<div><div>FIDO MEMBERS</div><div>LIAISON PARTNERS</div><div>WORKING GROUPS</div><div>COMMITTEES AND STUDY GROUPS</div><div>Board Company Quotes ></div></div>					
<div><div>Home / FIDO Members</div><div>FIDO Members</div></div>					
<div>Board Level Members</div>					
					<div>DOWNLOAD SPECS</div>
					
					
					
					
					
					

Everybody gets a second chance

FIDO2

FIDO2

- Genauere Standards
- Kein Dokumentationswust
- Direkte Browser-Unterstützung für leichtere Implementierbarkeit
- U2F (USB) und UAF (Bio) getrennte Protokolle, nun unter FIDO2 vereinheitlicht
- Schützenhilfe vom W3C
- Buzz-Word: Passwordless!

FIDO2 Top View

Es gibt zwei Protokoll Bereiche:

1. Zwischen **Browser und Server** (Relying Party)
 - **WebAuthN**, Spec vom W3C
 - Gut für Web-Entwickler
2. Zwischen **Browser und Gerät** (Authenticator)
 - Kann U2F Geräte, FIDO2-Geräte, TPM, Key-Chain (Fingerprint)
 - **CTAP1** API (U2F)
 - **CTAP2** API (neu)
 - Gut für Hardware-Entwickler
 - Password on Device

CTAP Was?

- WebAuthN abstrahiert das: Nicht relevant für die Einbindung in Web-Dienste
- Lediglich notwendiges Wissen: Es gibt "Authenticator" mit unterschiedlichen Eigenschaften!

WebAuthN

- Basiert auf der Javascript Navigator Eigenschaft

```
navigator.credentials
```

- Für Passwörter gab es die Methoden store und get.
- Neu:
 - create zum Erzeugen von Schlüsselpaaren
 - get kann nun auch signieren.

WebAuthN

Registrieren

```
navigator.credentials.create({ publicKey: { ...options... } })
```

- Wichtige Parameter sind
 - challenge
 - rp: Relying Party, quasi der Hostname des Servers
 - user
 - keyParameters (ECC... bits...)
- Weitere Parameter ermöglichen bspw. auch gezielt spezielle "Authenticator" auszuwählen.

WebAuthN

Registrieren /2

- Was passiert:
 - Schlüsselpaar wird im "Authenticator" erzeugt
 - U.a. wird zurückgegeben:
 - Key ID
 - Public Key
 - attestationObject
- und an den Server gesendet.

Attestation bei der Registrierung

Demo mit netknights.webauthn.org

- Server sendet: "Hey ich bin webauthn.org"
- Im Browser wird aber netknights.webauthn.org aufgerufen.
 - Genau das signiert der Authenticator
- Dieses attestationObject wird an den Server gesendet
- Erst auf dem Server wird geprüft, dass "effective origin" nicht dem "expected origin" entspricht.

WebAuthN

Authentisieren

- Server sendet "challenge" und "allowedCredentials".
- Client ruft auf

```
navigator.credentials.get({publickey: { challenge: ...,  
                                     allowedCredentials ... }})
```

- ...und sendet die Antwort zurück an den Server.
- Die Antwort enthält die Signatur der Challenge und die Signatur der "effective origin" (die aufgerufene URL).

Origin-Prüfung bei der Authentisierung

Demo mit netknights.webauthn.org

- Server sendet: "Hey ich bin webauthn.org, ich akzeptiere die folgenden Schlüssel..."
- Der Authenticator sucht einen Schlüssel für die "effective origin" und findet keinen.
- Keine Server-Kommunikation, Fehler direkt im Authenticator

Bonus: User Verification

- Die RP kann "User Verification" fordern, d.h. nur "Authenticator" mit

```
{ "userVerification": "required" }
```

- PIN Eingabe
 - oder Biometriewerden akzeptiert.
- => Kein Yubikey
- Betrachtung: *Passwordless*, vgl. Smartcards

Bonus: Userless Authentication

- Bei der Registrierung kann ein "Resident Key" gefordert werden.
- Während der Authentifizierung:
 - RP sendet lediglich Challenge
 - und fordert "Resident Credentials"
 - Es wird keine userid mitgesendet!
- ■ Der Client beantwortet die Challenge mit dem "Resident Key"
- ■ ...und schickt seine userid mit.
- => Anmeldung ohne Eingabe des Benutzernamens!

Bonus: Attestation

- Bei der Registrierung kann die RP das Attestation Certificate mit Signatur fordern.
- Es werden nur spezielle "Authenticator" akzeptiert, bzw. von vertrauenswürdigen CAs.

Die ID der Relying Party

- Bei der Registrierung wird geprüft, ob die Relying Party

```
"rp": {"name": "webauth.org"}
```

zur aufgerufenen Webseite passt.

- Die "effective origin" muss zur "expected origin" passen.
- Es scheint keine Trusted Facets zu geben! :-/

Fazit

- End-User: Haben wollen!
- Unternehmen: Cool, aber an der Realität vorbei
- Wie seht Ihr das?

Links

- <https://www.youtube.com/watch?v=eihy1kPM4gQ>
- <https://medium.com/@herrjemand/introduction-to-webauthn-api-5fd1fb46c285>
- <https://w3c.github.io/webauthn/>