WebAuthn 101
Demystifying WebAuthn

Christiaan Brand
Blackhat 2019
Agenda

01 02 03

Passwords aren’t enough  MFA - a spectrum of assurance  Enter WebAuthn
Passwords aren’t enough
117% Minimum password reuse rate

4.3B+ Credentials leaked in dumps

110M Accounts proactively re-secured

Data breaches, phishing, or malware? Understanding the risks of stolen credentials (Thomas et al.)
ai.google/research/pubs/pub46437
99.9%
Sources of stolen passwords

Phishing

Keyloggers

Data breach
Stolen credential origin

- **Password reuse is the largest source**: Password breach is the main purveyor of stolen credential with hundreds of millions new credential every year.

- **Phishing is the most dangerous source**: Phished victims are at the highest risk to get their account compromised.

- **The black market fuel account compromise**: There is a whole shadow ecosystem that makes compromised accounts a commodity.
91% of information security attacks start with phishing

80% of attacks on businesses include phishing

Source: PhishMe study, cofense.com/enterprise-phishing-susceptibility-report/
Phishing overtook exploit-based malware in 2016

Source: Safe Browsing (Google Transparency Report)
43% success rate for a well designed phishing page*

76% of account vulnerabilities were due to weak or stolen passwords**

*Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials, 2017
**Verizon 2015 Data Breach Investigations Report
MFA - a spectrum of assurance
Many different types of MFA exist, all providing different levels of assurance and convenience.
Titan Security Key

Enhanced account protection
Phishing-resistant 2nd factor of authentication that verifies user’s identity and sign-in URL

Trusted hardware
Includes a secure element with firmware written by Google to verify the key’s integrity

Open ecosystem
Works with popular browsers and a growing ecosystem of services that support FIDO
Now, your Android phone is also a security key

- Enhanced account protection
  Strongest 2FA protection against phishing

- Easy to use
  Simple, one-time enrollment process, no app required

- Convenient for users
  Use the phone which is already in your pocket

With phone’s built-in security key
How Security Keys work

Challenge was: 123456
Origin was: google.com

{challenge, "google.com"}_{signed}

Alice's Security Key

challenge, "google.com"

Who's calling?

Sign in

Alice Smith
alice@acme.com

Password

Server

https://www.google.com

5

4

3

2

1

6
Created with open standards

https://www.google.com

Who's calling?

USB/NFC/BLE

Server
Created with open standards

WebAuthn API

Who's calling?

USB/NFC/BLE

Server
Created with open standards

WebAuthn API

CTAP API

USB/NFC/BLE

Who's calling?
Enter WebAuthn
Introducing WebAuthn

A W3C specification* (Web API) that allows websites to interact with authenticators

* https://github.com/w3c/webauthn
What is WebAuthn? How does it relate to FIDO2?

- **WebAuthn**
- **FIDO2**
- **CTAP**
- **Remote server** (Website)
- **Client** (Computer, phone)
- **Removable authenticator** (Phone, security key)
- **Built-in authenticator** (fingerprint)
WebAuthn: **two use cases**

1. **“Bootstrapping”** - security key as a 2nd factor

2. **“Re-authentication”** - biometrics as a way to simplify verifying a returning user
Implemented on Android

**Green:** Your app can directly talk to the key store to store and use cryptographic keys

**Red:** Your app can directly talk to the biometric APIs

**OR**

**Blue:** Your app and website can talk to the FIDO/WebAuthn APIs that abstracts the keystore and biometric APIs
Meet Elisa
Elisa wants to sign in to her bank

She starts on her mobile browser and enrolls in fingerprint after sign-in

Registering built-in authenticator for re-auth (mobile web)
Elisa opens launches her mobile browser, Chrome, and goes to Tri-Bank

1. Registering built-in authenticator for re-auth (mobile web)
She signs in with her username and password

1. Registering built-in authenticator for re-auth (mobile web)
Tri-Bank shows a promo asking Elisa if she wants to opt in to fingerprint to sign in

She opts in and continues to her account
What happened behind the scenes?

Silently determined whether a platform authenticator was available:

```javascript
PublicKeyCredential.isUserVerifyingPlatformAuthenticatorAvailable().then(response => {
  if (response === true) {
    // User verifying platform authenticator is available!
  } else {
    // User verifying platform authenticator is NOT available.
  }
})
```
What happened behind the scenes?

Created the credential on the platform authenticator

```javascript
navigator.credentials.create({
   "publicKey": PublicKeyCredentialCreationOptions
});
```
What happened behind the scenes?

With values for `PublicKeyCredentialCreationOptions`

- `excludeCredentials = [ // add any already registered ids ]`
- `authenticatorSelection.authenticatorAttachment = 'platform'
  // other options: ‘cross-platform’`
- `authenticatorSelection.userVerification = 'required'
  // other options: ‘discouraged’ or ‘preferred’`
Elisa comes back to Tri-Bank in another session
The next time Elisa opens Tri-Bank on mobile browser, she gets a fingerprint dialog.

Since the user already signed in on this device, the credential ID is encoded in the cookie and the RP requests the “internal” transport only (since they don’t want the user to see prompts about external authenticators).
Using only her fingerprint, she’s able to sign in without using her username + password on mobile web.
What happened behind the scenes?

Created a signature using the platform authenticator

```javascript
navigator.credentials.get({
   "publicKey": PublicKeyCredentialRequestOptions
});
```

With values for `PublicKeyCredentialRequestOptions`

- `allowCredentials` = `[ // credential associated with session]`
- `userVerification` = `true`
**Elisa** downloads Tri-Bank from the Play Store

She launches the app for the first time to sign in to check her funds
She installs Tri-Bank from Google Play Store and opens the app
Elisa chooses “Sign In” and enters her username.
Elisa is now asked to authenticate with the fingerprint dialog.
What happened behind the scenes?

Created a signature using the platform authenticator
Fido2ApiClient fido2ApiClient = Fido.getFido2ApiClient(this.getApplicationContext());

Task<Fido2PendingIntent> result = fido2ApiClient.getSignIntent(requestOptions);

With values for requestOptions
- allowCredentials = /* credential associated with session */
- userVerification = true
Case study: Yahoo! JAPAN

Reauth using fingerprint reduced time to sign-in by ... 37.5%

comparing to that of using a password.
Case study: Google

98% of biometric reauth users finish in 38s
98% of all users enter password in 150s

Google Internal Data: 2018
Implement WebAuthn today!

- Play with our FIDO server
  webauthndemo.appspot.com

- Implement WebAuthn Create and Get methods
  codelabs.developers.google.com/codelabs/webauthn-reauth/

- Link your Android app for a seamless login experience
  codelabs.developers.google.com/codelabs/fido2-for-android/