Universal Second Factor authentication or why 2FA today is wubalubadubdub
Yuriy Ackermann
Sr. Certification Engineer
@FIDOAlliance
twitter/github: @herrjemand
Today we will learn

- Why passwords not enough
- Why 2FA has not succeeded
- Introduction to U2F
- DEMO
- Q&A
Why not just passwords?

Typical passwords life cycle

Weak → Reuse → Phishing → pwned

SOLUTION!

Two Factor Authentication - aka 2FA

haveibeenpwned.com
What is 2FA?

Passwords verify

2FA authenticate
Do you use 2FA?
What does 2FA looks like?

Three main types

- **Apps**
  - (TOTP and HOTP)

- **Tokens**
  - (PKI and OTP)

- **SMS**
So we solved it?

Right?
# Why 2FA has not succeeded?

<table>
<thead>
<tr>
<th>Apps</th>
<th>Tokens</th>
<th>SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phishing!!</td>
<td>Cost</td>
<td>Still phishable</td>
</tr>
<tr>
<td>UX</td>
<td>DRIVERS</td>
<td>UX</td>
</tr>
<tr>
<td>Shared key</td>
<td>Phishing</td>
<td>Privacy</td>
</tr>
<tr>
<td>Synced time</td>
<td>UX</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td>Centralised</td>
<td>  SIM reissue</td>
</tr>
<tr>
<td></td>
<td>Fragile</td>
<td>  SIM spoof</td>
</tr>
</tbody>
</table>

- Coverage
- NIST Ban
Two factor sucks, so... why not just point a webcam at your token?

shodan.io/host/198.2.49.... via @djvc1993
Current state of 2FA

WUBA LUBA DUB DUB....

I am in the deep pain, please help!
So how do we solve it?

We need:

- Easy to use
- Open
- Secure
- Standardized protocol.
Introducing
Universal Second Factor
aka FIDO U2F
How does U2F works?
User layer
Browser layer

User side

FIDO U2F Authenticator
- User action
- Secure Element
- NFC
- USB(HID)
- BLE

FIDO Client i.e. Browser
- JS API
- NFC
- USB(HID)
- BLE

Relying party
- U2F Lib
- PK, KeyHandles, Certificates
Protocol Layer
Step one: Challenge-Response

- **U2F Device**
  - Sign with $k_{priv}$
- **Challenge**
- **Client**
  - Signature(challenge)
  - $s$
- **Relying Party**
  - Challenge
  - $s$
  - Lookup $k_{pub}$
  - Check $s$ using $k_{pub}$
Step two: Phishing protection

U2F Device

challenge, origin, channel id

challenge

signature(c)

Client

challenge

c, s

Relying Party

Lookup

k_{pub}

Check s using k_{pub}

Verify origin and channel id

Sign with k_{priv}
Step three: **Application-specific key-pair**

- **U2F Device**
  - Lookup the $k_{priv}$ associated with $h$
  - $h$, $a$; challenge, origin, channel id, etc.
  - Signature$(a,c)$
  - $c$

- **Client**
  - Check app id
  - $h$, $a$
  - $c$, $s$

- **Relying Party**
  - Lookup the $k_{pub}$ associated with $h$
  - Check $s$ using $k_{pub}$
  - Verify origin and channel id

**Diagram Notes:**
- The diagram illustrates the steps of using an application-specific key-pair in a U2F authentication flow.
To Wrap, or not to Wrap?
Step four: **Replay Attack Protection**

- **U2F Device**
  - Lookup the $k_{priv}$ associated with $h$
  - $counter++$

- **Client**
  - $h, a; \text{challenge, origin, channel id, etc.}$
  - $c$
  - $counter, \text{signature}(a, c, counter)$
  - $s$

- **Relying Party**
  - $\text{handle, app id, challenge}$
  - $\text{Check app id}$
  - $h$
  - $a$
  - $\text{Lookup the } k_{pub} \text{ associated with } h$
  - $\text{Check } s \text{ using } k_{pub}$
  - $\text{Verify origin, channel id, counter}$
  - $\text{counter, c, s}$
Step five: Device attestation

U2F Device

Generate: 
k_{pub}
k_{priv}
handle h

Client

Check app id

app id, challenge

a

Relying Party

Associate k_{pub} with handle h for user

c, k_{pub}, h, attestation cert, s

k_{pub}, h, attestation cert, signature(a, c, k_{pub}, h)

a; challenge, origin, channel id, etc.

c
Metadata service

Direction of data flow
Direction of connection

FIDO Server
1. Download & verify Metadata TOC from well-known URL
2. Download & verify Metadata Statement from URL supplied in Metadata TOC

Download Metadata Statement from Authenticator Vendor

Authenticator Vendor

Provide Metadata as part of FIDO Certification

Metadata Service
Step five and a half: **Key exercise protection**

User must confirm their decision to perform 2FA, by performing user gesture

- Pressing button
- Fingerprint
- Retina scan
- Pincode
- Remembering your wife's birthday.
- Solving Rubik's cube
- ...anything you want.

[fido ALLIANCE logo]
Multiple identifiers

Web
- mail.google.com

Android
- apk-key-hash:FD18FA

iOS
- com.google.SecurityKey.dogfood

How do we deal with it?
Application Facets

```
{
  "trustedFacets": [{
    "version": { "major": 1, "minor" : 0 },
    "ids": [
      "https://accounts.google.com",
      "https://myaccount.google.com",
      "https://security.google.com",
      "android:apk-key-hash:FD18FA800DD00C0D9D7724328B6D...",
      "android:apk-key-hash:/Rj6gA3QDA2ddyQyi21JX1y6gw9D...",
      "ios:bundle-id:com.google.SecurityKey.dogfood"
    ]
  }
}

MUST be served over VALID HTTPS!

...no self signed certs.
```
Implementations
@NotThatNiemand @jessysaurusrex - YubiKeys make nice earrings as well. :)

Показать перевод
Current users
Browser support

- Chrome: Yes
- Firefox: Yes* (Nightly)
- Internet Explorer: No* (Soon...)
- Safari: Yes

Maybe?
WebAuthN

A W3C standard for PublicKey credential authentication

https://www.w3.org/Webauthn/
Today we learned

- Passwords are hard
- 2FA is wubalubadubdub, and we need to do something about it.
- FIDO U2F is sweet.
  - Protocol is cute
  - You can have multiple identities
  - There are existing solutions...
  - ...and people do use it
DEMO
Security considerations

- You must use HTTPS
- Start using TLS Channel ID's
- U2F is just 2FA. Don't use as primary factor.
Things to play with

- https://github.com/Yubico/pam-u2f
- https://github.com/Yubico/python-u2flib-server
- https://github.com/Yubico/python-u2flib-host
- https://github.com/herrjemand/flask-fido-u2f
- https://github.com/gavinwahl/django-u2f
- https://github.com/google/u2f-ref-code
- https://github.com/conorpp/u2f-zero

Specs and data

- https://developers.yubico.com/U2F/
- https://fidoalliance.org/specifications/download/
- https://github.com/LedgerHQ <- JavaCard
- FIDO Dev (fido-dev) mailing list
What's next?
Questions?
twitter/github: @herrjemand
Quick thanks to Feitian and Yubico for swag!
Thank you

OWASP!