Universal Two-Factor Authentication for the Web And The Mac

Thomas Westfeld
Cocaheads Aachen
2015-03-26
„Treat your password like your toothbrush. Don't let anybody else use it, and get a new one every six months.“

–Clifford Stoll
Who on earth does this?
### Most Used Passwords in 2014

<table>
<thead>
<tr>
<th></th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123456</td>
</tr>
<tr>
<td>2</td>
<td>password</td>
</tr>
<tr>
<td>3</td>
<td>12345</td>
</tr>
<tr>
<td>4</td>
<td>12345678</td>
</tr>
<tr>
<td>5</td>
<td>qwerty</td>
</tr>
<tr>
<td>6</td>
<td>123456789</td>
</tr>
<tr>
<td>7</td>
<td>1234</td>
</tr>
<tr>
<td>8</td>
<td>baseball</td>
</tr>
<tr>
<td>9</td>
<td>dragon</td>
</tr>
<tr>
<td>10</td>
<td>football</td>
</tr>
<tr>
<td>11</td>
<td>1234567</td>
</tr>
<tr>
<td>12</td>
<td>monkey</td>
</tr>
<tr>
<td>13</td>
<td>letmein</td>
</tr>
<tr>
<td>14</td>
<td>abc123</td>
</tr>
</tbody>
</table>

We are doomed!
Best Practices for Passwords

1. Choose long passwords (e.g. use Dicewords™)
2. Do not use keyboard patterns (qwerty, etc.)
3. Do not use consecutive number sequences (1234)
4. Do not reuse passwords along different sites.
5. Change password regularly.
Best Practices for Passwords

1. Choose long passwords (e.g. use Dicewords™

2. Do not use keyboard patterns (qwerty, etc.)

3. Do not use consecutive number sequences (1234)

4. Do not reuse passwords along different sites.

5. Change password regularly.

Almost impossible for every password. Use an offline password manager!
Authentication Is All About Factors

Knowledge Factors

Something only
the user knows
Authentication Is All About Factors

Knowledge Factors

Something only the user knows

Possession Factors

Something only the user has
Authentication Is All About Factors

Knowledge Factors
Something only the user knows

Possession Factors
Something only the user has

Inherence Factors
Something only the user is
Two-Factor Authentication (2F)

************ +

Time-based one time password
TOTP (RFC 6238)

Counter-based one time password
HOTP (RFC 4226), iTAN or SMS
Two-Factor Authentication (2F)

Time-based one time password
TOTP (RFC 6238)
Counter-based one time password
HOTP (RFC 4226), iTAN or SMS
Drawbacks of Two-Factor Authentication

- Need to be able to receive SMS
- Needs to have your authenticator handy
- Transfer received code to login form
- As a fallback application specific passwords may be generated.
- Backup codes have to be stored at a secure location
Drawbacks of Two-Factor Authentication

- Need to be able to receive SMS
- Needs to have your authenticator handy
- Transfer received code to login form
- As a fallback application specific passwords may be generated.
- Backup codes have to be stored at a secure location

You cannot login without the second factor.
The FIDO Alliance

- Founded in summer 2012
- Publicly launched in February of 2013
- Published their first standard 1.0 in 2014-12-09
Universal Second Factor Authentication U2F
FIDO Universal-Two-Factor (U2F)

- Challenge-response public-private key cryptography
- Uses elliptic curve cryptography to minimize key-lengths
- Easy to use, just push a button
How to use U2F?

Register U2F device

Authenticate w/ U2F device
Registration

U2F device

Client

Relying Party

U2F dongle

Browser

Webservice
Registration

U2F device

Client

Relying Party

generate challenge

U2F dongle

Browser

Webservice
Registration

U2F device

U2F dongle

Client

challenge c, appID a

generate challenge

Relying Party

Webservice
Registration

U2F device

Client

challenge c, appId a

Relying Party

generate challenge

U2F dongle

Browser

Webservice
Registration

U2F device

Client

Relying Party

challenge c, appID a

generate challenge

Check appID
c, a, origin

U2F dongle

Browser

Webservice
Registration

U2F device

Client

challenge c, applID a

Relying Party

generate challenge

U2F dongle

Browser

c, a, origin

Webservice

Generate $k_{pub}, k_{priv}, handle h$
Registration

Relying Party

Client

U2F device

challenge c, appID a

generate challenge

Webservice

Browser

U2F dongle

Generate $k_{pub}, k_{priv}, handle h$

Check appID

c, a, origin

$k_{pub}, h, att. cert, signature (a, c, h, k_{pub})$
Registration

U2F device

- Generate $k_{pub}, k_{priv}, \text{handle } h$

Client

- Check appId
- $c, a, \text{origin}$

- $k_{pub}, h, \text{att. cert, signature } (a, c, h, k_{pub})$

- $c, k_{pub}, h, \text{att. cert, signature } (a, c, h, k_{pub})$

Relying Party

- generate challenge

Web_service

- challenge $c, \text{appId } a$

U2F dongle

Browser
Registration

U2F device

Generate $k_{pub}$, $k_{priv}$, handle $h$

U2F dongle

Check appID

c, a, origin

Browser

$k_{pub}$, $h$, att. cert, signature (a, c, h, $k_{pub}$)

Webservice

challenge c, appID a

c, $k_{pub}$, $h$, att. cert, signature (a, c, h, $k_{pub}$)

Relying Party

generate challenge

$k_{pub}$, h stored for user
Authentication

U2F device

Client

Relying Party

U2F dongle

Browser

Webservice
Authentication

U2F device

Client

Relying Party

generate challenge

U2F dongle

Browser

Webservice
Authentication

U2F device

Client

challenge c, appID a, handle h

generate challenge

Relying Party

Webservice

U2F dongle

Browser
Authentication

U2F device

Client

challenge c, appId a, handle h

Relying Party

generate challenge

Browser

Check appId

Webservice

U2F dongle
Authentication

U2F device

Check appID

c, a, h, origin

Client

challenge c, appID a, handle h

generate challenge

Relying Party

Webservice

U2F dongle

Browser
Authentication

U2F device

Client

Relying Party

generate challenge

c, a, h, origin

Check appID

challenge c, appID a, handle h

Lookup $k_{priv}$ from handle h

counter++

U2F dongle

Browser

Webservice
Authentication

Webservice

Client

challenge c, appID a, handle h

c, a, h, origin

counter, signature (a, c, counter)

Relying Party

generate challenge

U2F device

Lookup k_{priv} from handle h

counter++

Browser

U2F dongle

lookup k_{priv} from handle h

counter++

check appID
c, a, h, origin
Authentication

U2F device

Lookup
$k_{priv}$
from
handle $h$

counter++

Client

challenge $c$, appId $a$, handle $h$

Check
appId $a$

c, a, h, origin

counter,
signature (a, c, counter)

signature (a, c, counter)

counter

Relying Party

generate challenge

Webservice

Browser

U2F dongle
Authentication

- U2F device
  - Lookup $k_{priv}$ from handle $h$
  - counter++

- Client
  - check appID
    - c, a, h, origin
  - counter,
    - signature (a, c, counter)

- Relying Party
  - generate challenge
  - check sign.
    - verify c
      - check and store new counter

- Webservice
  - challenge c, appID a, handle h

- Browser
  - signature (a, c, counter)
How many accounts on the U2F dongle?

• As many as you like!

• The private key is NOT stored on the device in Yubico’s implementation. Instead it is encrypted with a 256bit AES key on the secure element on the dongle.
U2F Summary

- Every account on every website gets a new public-private key pair.
- The dongle has no UUID and cannot be tracked between different sites.
- No passcode has to be entered manually.
- No shared secret - even if key handle and public keys leaked one cannot copy the U2F dongle.
References

- Yubico U2F demo server [https://demo.yubico.com/u2f](https://demo.yubico.com/u2f)
- The U2F specifications [https://fidoalliance.org/specifications/download/](https://fidoalliance.org/specifications/download/)
- Yubico U2F C host library [https://github.com/Yubico/libu2f-host](https://github.com/Yubico/libu2f-host)