

FIDO and Webauthn on BSD: Authentication for the easily distracted

Taylor R Campbell
riastradh@NetBSD.org

EuroBSDcon 2023
Coimbra, Portugal
September 17, 2023

FIDO and Webauthn on BSD

[https://www.NetBSD.org/gallery/presentations/
riastradh/eurobsdcon2023/fidobsd.pdf](https://www.NetBSD.org/gallery/presentations/riastradh/eurobsdcon2023/fidobsd.pdf)



Why do we need a new authentication system?

Hook

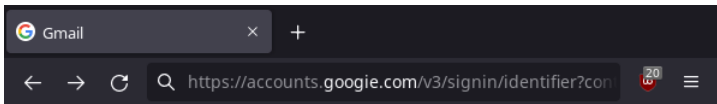
Date: Sun, 17 Sep 2023 13:20:59 +0000
From: "EuroBSDcon (via Google Drive)"
<eurobsdcon@gmail.com>
To: riasradh@gmail.com
Subject: Folder shared with you: "Conference program"

I've shared an item with you:

Conference program
[https://drive.google.com/drive/folders/
Gb5Z_sYiHuiqUClpeCISutMRc3rMmzZAg?
usp=sharing&invite=vigcIJy&ts=6ff7f21e](https://drive.google.com/drive/folders/Gb5Z_sYiHuiqUClpeCISutMRc3rMmzZAg?usp=sharing&invite=vigcIJy&ts=6ff7f21e)

It's not an attachment -- it's stored online.
To open this item, just click the link above.

Line



Google

Sign in

to continue to Gmail

Email or phone

riastradh@gmail.com

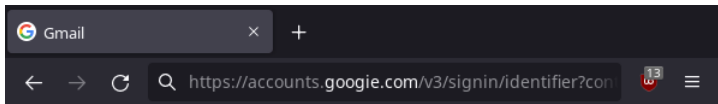
[Forgot email?](#)

Not your computer? Use a Private Window to sign in. [Learn more](#)

[Create account](#)

Next

Sinker



Google

Welcome

 riastradh@gmail.com ▾

Enter your password

hunter2|



Show password

[Forgot password?](#)

Next

You've been phished!

Two-factor authentication

Prove at least two:

- ▶ something you know (password, security question)
- ▶ something you have (phone, USB token, smart card)
- ▶ something you are

Two-factor authentication

Prove at least two:

- ▶ something you know (password, security question)
- ▶ something you have (phone, USB token, smart card)
- ▶ something you are (a BSD nerd)

Two-factor authentication

Prove at least two:

- ▶ something you know (password, security question)
- ▶ something you have (phone, USB token, smart card)
- ▶ something you are (a BSD nerd)

Two-factor authentication

Prove at least two:

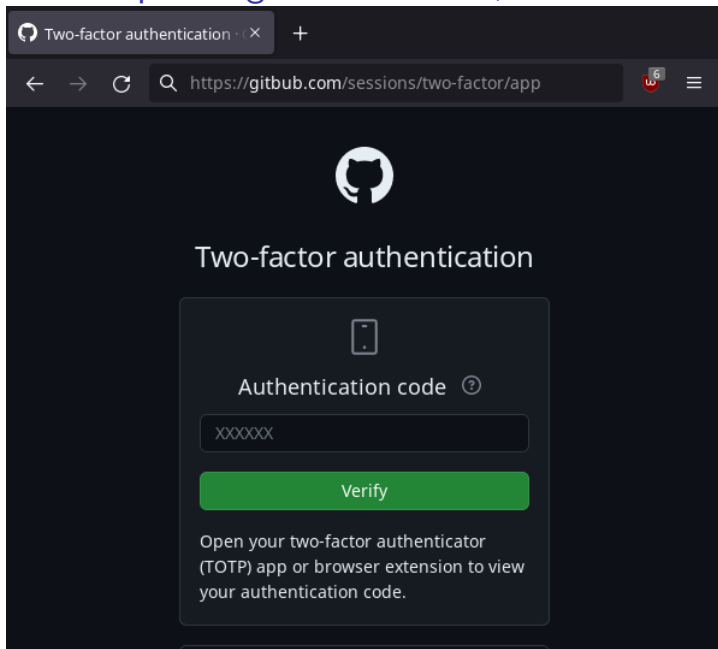
- ▶ something you know (password, security question)
- ▶ something you have (phone, USB token, smart card)
- ▶ something you are (biometrics: retina, fingerprint, ...)

Two-factor authentication

Typical 2FA:

- ▶ 2FA codes sent over SMS to your phone
- ▶ Authenticator app, usually meaning TOTP (RFC 6238/4226) stored on phone
- ▶ Push notifications to your phone, usually Microsoft or Duo proprietary

Two-factor phishing: TOTP codes, SMS 2FA codes



The image shows a browser window with a dark theme. The address bar contains the URL `https://github.com/sessions/two-factor/app`. The page features the GitHub logo at the top center, followed by the heading "Two-factor authentication". Below this is a dark grey rounded rectangle containing a mobile phone icon, the text "Authentication code" with a help icon, a text input field with "XXXXXX", a green "Verify" button, and a paragraph of instructions: "Open your two-factor authenticator (TOTP) app or browser extension to view your authentication code." The browser's taskbar is visible at the bottom.

Two-factor authentication

Authentication code ?

XXXXXX

Verify

Open your two-factor authenticator (TOTP) app or browser extension to view your authentication code.

Two-factor phishing: push notifications

(screenshot of notification left as an exercise for the reader)

Two-factor phishing

- ▶ 2FA codes sent over SMS
- ▶ TOTP codes
- ▶ Push notifications

Two-factor phishing

- ▶ 2FA codes sent over SMS
 - ▶ ... are gathered by the same phishing page and relayed on by the attacker
- ▶ TOTP codes

- ▶ Push notifications

Two-factor phishing

- ▶ 2FA codes sent over SMS
 - ▶ ... are gathered by the same phishing page and relayed on by the attacker
- ▶ TOTP codes
 - ▶ ... are gathered by the same phishing page and relayed on by the attacker
- ▶ Push notifications

Two-factor phishing

- ▶ 2FA codes sent over SMS
 - ▶ ... are gathered by the same phishing page and relayed on by the attacker
- ▶ TOTP codes
 - ▶ ... are gathered by the same phishing page and relayed on by the attacker
- ▶ Push notifications
 - ▶ ... are sent when the password you entered into the phishing page is relayed on by the attacker

Two-factor phishing

- ▶ 2FA codes sent over SMS
 - ▶ ... are gathered by the same phishing page and relayed on by the attacker
- ▶ TOTP codes
 - ▶ ... are gathered by the same phishing page and relayed on by the attacker
- ▶ Push notifications
 - ▶ ... are sent when the password you entered into the phishing page is relayed on by the attacker
 - ▶ ... lead to notification fatigue

Two-factor phishing

Main problem: copying & pasting secrets not bound to origin

Threat models

Threat models

1. Phishing

Threat models

1. Phishing
2. Phishing

Threat models

1. Phishing
2. Phishing
3. Phishing

Threat models

1. Phishing
2. Phishing
3. Phishing
4. User fatigue and circumvention

Threat models

1. Phishing
2. Phishing
3. Phishing
4. User fatigue and circumvention
 - ▶ Message to security people: Be an enabler. Don't get in the way; enable people to get their work done with less risk.

Threat models

1. Phishing
2. Phishing
3. Phishing
4. User fatigue and circumvention
 - ▶ Message to security people: Be an enabler. Don't get in the way; enable people to get their work done with less risk.
5. Hardware theft, MITM attacks, shoulder surfing, . . .

Hardware tokens

- ▶ RSA SecurID—proprietary version of TOTP on a gizmo with an LCD display
- ▶ Old Yubikeys—USB keyboard that types a proprietary version of TOTP token
- ▶ PKCS#11, PKCS#15, OpenPGP, ...

Legacy crypto tokens

- ▶ Software stack

1. security/pcsc-lite—daemon that talks to USB smartcard-like reader (pcscd)
2. security/opensc—library and tools that talk to smartcard through pcsc-lite
3. security/ccid—opensc driver that talks to chip/smart card interface driver devices

- ▶ proprietary magic protocols and file layout:

<https://github.com/OpenSC/OpenSC/pull/2097>

- ▶ limited number of keys per device

- ▶ state management

- ▶ privacy leaks across sites

https://wiki.NetBSD.org/tutorials/howto_bootstrap_the_ePass2003_smartcard/

FIDO will protect us from the phish

Live demo

Protocol flow—Registration

1. Server at example.com asks to make a credential
2. Browser asks user to tap button to approve
3. Device generates credential id and key pair for 'example.com'
4. Device returns credential id and public key
5. Server stores credential id and public key for later use

Note: Every registration creates an independent random key pair—key generation with elliptic-curve crypto is cheap!

Protocol flow—Authentication

1. Server at example.com sends a challenge and stored credential ids and asks for proof of one of them
2. Browser asks user to tap button to approve
3. Device re-derives key pair from credential id for 'example.com'
4. Device returns signature on challenge
5. Server verifies signature with stored publickey

Properties

- ▶ Independent keys for each site—no cross-site tracking
- ▶ No special software, drivers, configuration tools needed
- ▶ No user-visible state to manage on device
- ▶ Unbounded number of credentials
- ▶ Used as 2FA: vendor is not single point of failure

Privacy leaks

Privacy leaks are much more limited than traditional hardware tokens with X.509 client certificates:

- ▶ On registration: device may send attestation of manufacturer and batch number (not serial number!)—up to browser
- ▶ On authentication: device may send signature count—up to device
- ▶ Server can tell if same device is used for multiple accounts

Recommendations for users

Get two devices:

- ▶ Primary on keychain or always plugged into laptop
- ▶ Backup in desk or somewhere safe

If you lose one, no big deal—get a new one and use the backup to log in and register it.

Recommendations for users

Get two devices:

- ▶ Primary on keychain or always plugged into laptop
- ▶ Backup in desk or somewhere safe

If you lose one, no big deal—get a new one and use the backup to log in and register it.

- ▶ ... And don't use PINs: bad user experience, limited software support, requires special tooling

How to add web application support—Registration

```
const credential = await navigator.credentials.create({
  publicKey: {
    challenge: ...,
    rp: {name: "Example GmbH", id: "example.com"},
    pubKeyCredParams: [{alg: -7, type: "public-key"}],
    authenticatorSelection: {
      authenticatorAttachment: "cross-platform"
    },
    excludeCredentials: [...],
    timeout: 60000,
    ...
  }
})
```

Returns structure with credential id, public key, optional device attestation.

More info: <https://webauthn.guide>

How to add web application support—Authentication

```
const credential = await navigator.credentials.get({
  publicKey: {
    challenge: ...,
    allowedCredentials: [{
      id: credential_id0, ...
    }],
    ...
  }
})
```

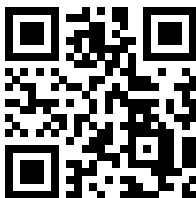
Returns structure with proof of ownership of one of the allowed credentials.

More info: <https://webauthn.guide>

How to add web application support

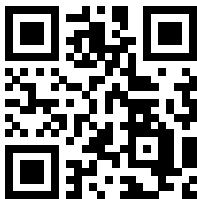
Various existing Webauthn libraries to handle data structures and verify credentials on the server side

More info: <https://webauthn.guide>



Sites that support Webauthn

`https://dongleauth.com`



FIDO on BSD

BSD support in kernel: USB HID

- ▶ Main transport: USB HID, like USB keyboard/mouse devices
- ▶ No special drivers needed—simple input/output ‘report’ pipes
- ▶ Other transports: smartcard, NFC—kind of works on BSD but requires pcsc
- ▶ (unsure if FIDO over Bluetooth works on BSD)

BSD support in userland: libfido2

- ▶ libfido2: C library for talking to FIDO devices
- ▶ Maintained by Yubico
- ▶ Supports NetBSD, OpenBSD, FreeBSD
- ▶ libfido2 available in pkgsrc/ports, shipped in NetBSD base

BSD support in browser: Firefox, authenticator-rs

- ▶ authenticator-rs: Rust crate for talking to FIDO devices
- ▶ Maintained by Mozilla
- ▶ Used by Firefox
- ▶ Supports NetBSD and FreeBSD out of the box
- ▶ OpenBSD support may be broken, needs maintainer:
<https://github.com/mozilla/authenticator-rs/pull/234>



FIDO in OpenSSH

- ▶ Different from usual FIDO—similar to usual OpenSSH
- ▶ `$ ssh-keygen -t ecdsa-sk`
- ▶ Keep `id_ecdsa-sk` private as usual
- ▶ Copy `id_ecdsa-sk.pub` to `~/.ssh/authorized_keys` on server to register as usual
- ▶ Tap device to authenticate on login

FIDO in OpenSSH

- ▶ Different from usual FIDO—similar to usual OpenSSH
- ▶ `$ ssh-keygen -t ecdsa-sk`
- ▶ Keep `id_ecdsa-sk` private as usual
- ▶ Copy `id_ecdsa-sk.pub` to `~/.ssh/authorized_keys` on server to register as usual
- ▶ Tap device to authenticate on login
- ▶ Alternative—resident keys/discoverable credentials:
 - ▶ No need to keep `id_ecdsa-sk`
 - ▶ Requires newer FIDO keys
 - ▶ Limited storage per device

Other platforms

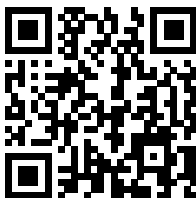
All major desktop and mobile operating systems and browsers support FIDO out of the box!

Other cool things with FIDO

'Storing' disk encryption keys—fidocrypt(1)

<https://github.com/riastradh/fidocrypt>

- ▶ Enroll multiple devices to have access to a secret
- ▶ Works with legacy U2F devices and modern FIDO2 devices
- ▶ Needs no storage on device—stored as per-device ciphertexts in a cryptfile
- ▶ (Might change file format to lighten executable, will provide upgrade path)



Using FIDO to sign messages—fidosisig(1)

`https://github.com/riastradh/fidosisig`

- ▶ `fidosisig(1)`: Sign arbitrary messages with FIDO devices
- ▶ Easily configurable threshold signature policies
- ▶ Binary format, no temptation to act on unauthenticated data
- ▶ [Experimental]



Using FIDO to sign messages—OpenSSH

- ▶ Use ecdsa-sk, ed25519-sk keys with OpenSSH:
`ssh-keygen -Y sign`

Using FIDO with Age to send encrypted messages

`https://github.com/riastradh/age-plugin-fido`

- ▶ Plugin for Age encryption tool:
`https://age-encryption.org`
- ▶ Requires newer FIDO2 devices (but no state or PINs)
- ▶ [Experimental]



Kerberos and FIDO

- ▶ Traditional Kerberos single-sign-on uses password to get SSO tickets
- ▶ New PA-REDHAT-PASSKEY preauthentication protocol adds 2FA step with FIDO
- ▶ Very new, not widely supported, maybe soon in Heimdal and MIT Kerberos!

Questions?