OpenSSH + FIDO workshop

Alan Alvarez & Dennis Hills | Yubico

Open Source Summit North America - May 2026

yubico

About this workshop

- Exercises:
  - **https://github.com/YubicoLabs/fido-openssh-workshop**
- Basic SSH knowledge helps but not required
- Some demos require a GitHub or GitLab account
- If you need one: grab a FIDO Security Key!

Before we start…

- If you haven’t already, download/install:
  - OpenSSH client (version 8.2+, ideally 8.9+)
    $ **ssh -V**
  - Python 3.9+
    $ **python3 -V**
  - Docker Desktop (or something similar)
    https://www.docker.com/products/docker-desktop/
  - An Ubuntu image (latest)
    $ **docker pull ubuntu:latest**
  - FIDO command-line tools
    https://developers.yubico.com/libfido2/
    $ **<pkg-mgr> install libfido2**
    (fido2-tools on debian/ubuntu)

yubico

Overview

- Intro - what is FIDO?
  - Tools: libfido2
- Hardware-backed SSH private keys
  - Exercise: sign in using hardware-backed keys
- Generating SSH signatures using security keys
  - Exercise: git commit signing
- Device attestation and the FIDO metadata service
  - Exercise: generating attestations
- Storing SSH certificates
  - Exercise: largeBlobs
- ssh-agent with security keys
  - Exercise: more secure agent-forwarding

yubico

Introduction

FIDO2 - concepts

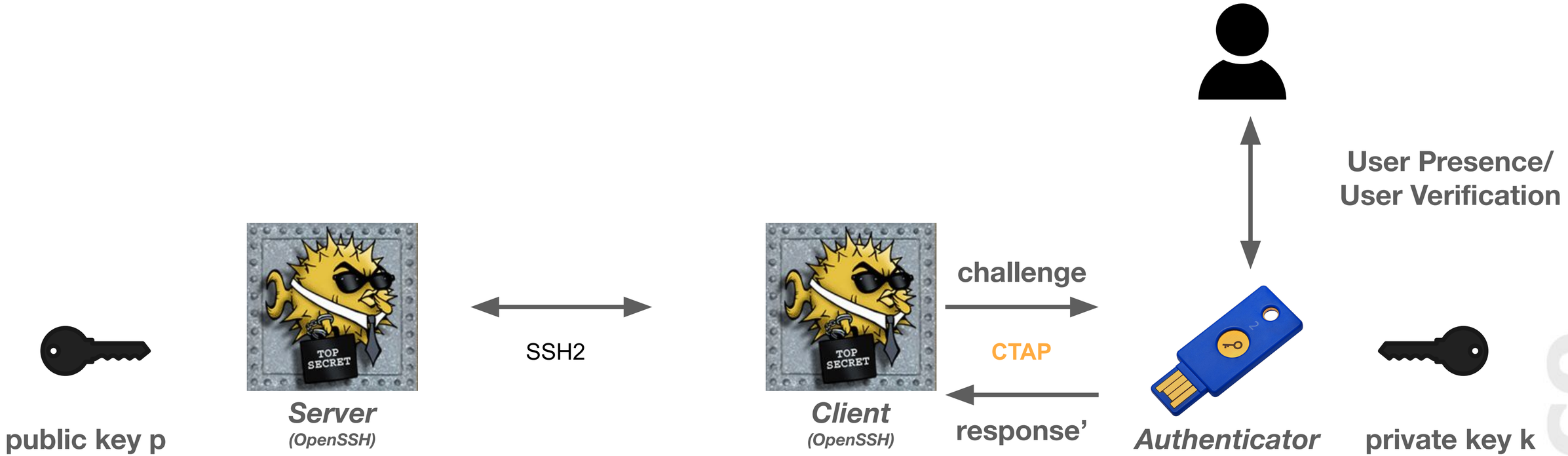
The Problem

- Your SSH private key is a file on disk
- Anyone who gets that file can **impersonate you**
- Passphrases can be **brute-forced offline**
- You have **no proof** of who actually used the key

Why OpenSSH + FIDO2?

- Open standard maintained by Google, Microsoft, Apple, Yubico
- **Private key** is generated on hardware, never leaves it, can't be exported or copied
- Touch = **user presence**, PIN = **user verification**
- 8 wrong PINs and the key locks, no offline brute force
- Sign **git commits** to prove they came from you
- **Attestation** proves your credential was generated on trusted hardware
- Simpler than PGP or PKCS#11, no vendor drivers or middleware, just USB HID

CTAP and SSH



CTAP: operations (simplified)

uID	credId ID	pub
jdoe	0xa1b2	0x41
		…

(RP user store)

RP

Client

Authenticator

credId	uID	rpID	priv
0xa1b2	jdoe	eg.com	0xf1…

(authR credential mapping)

makeCredential(rpId, uID, cData, …)

(priv,pub) = genKeyPair()
credId = genCredId(…)
map {credId, uID, rpID} → priv
signature = sign(…)

{ credId, pub, signature, … }

verify(resp,signature)
store[uID] = {credId, pub}

…

getAssertion(credID, rpId, cData, …)

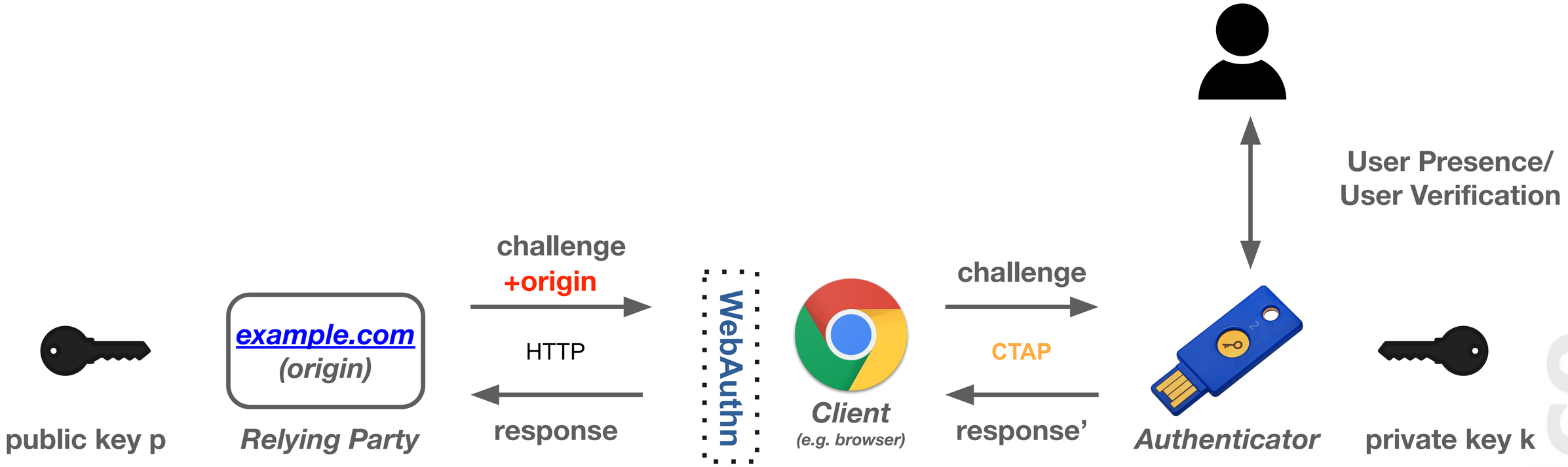
priv = lookup {credID}
signature = sign(priv, rpId, cData…)

{ signature, … }

{credId, pub} = store[uID]

verify(pub, resp, signature)

CTAP and Webauthn (simplified)



Authenticators

Cross-Platform

Roaming Authenticator

Independent of a specific device.

Example:

A USB security key

Transports:

- USB & NFC
- BLE (Bluetooth Low Energy)
- Hybrid

Platform

Built-in Authenticator

Integrated into the user's device.

Examples:

Fingerprint sensors, Facial recognition

FaceID TouchID Windows Hello

FIDO Security Keys: cross-platform authenticators

yubico

User Presence (UP) vs User Verified (UV)

User Presence (UP)

The user is physically present during the operation.

On a security key:

A simple button touch

User Verified (UV)

The user is verified by authenticating to the Security Key.

Security Key

Entering a PIN or using a built-in fingerprint scanner

Platform Authenticator

Typically a biometric (e.g., Touch ID / Face ID)

Resident vs Non-resident Keys

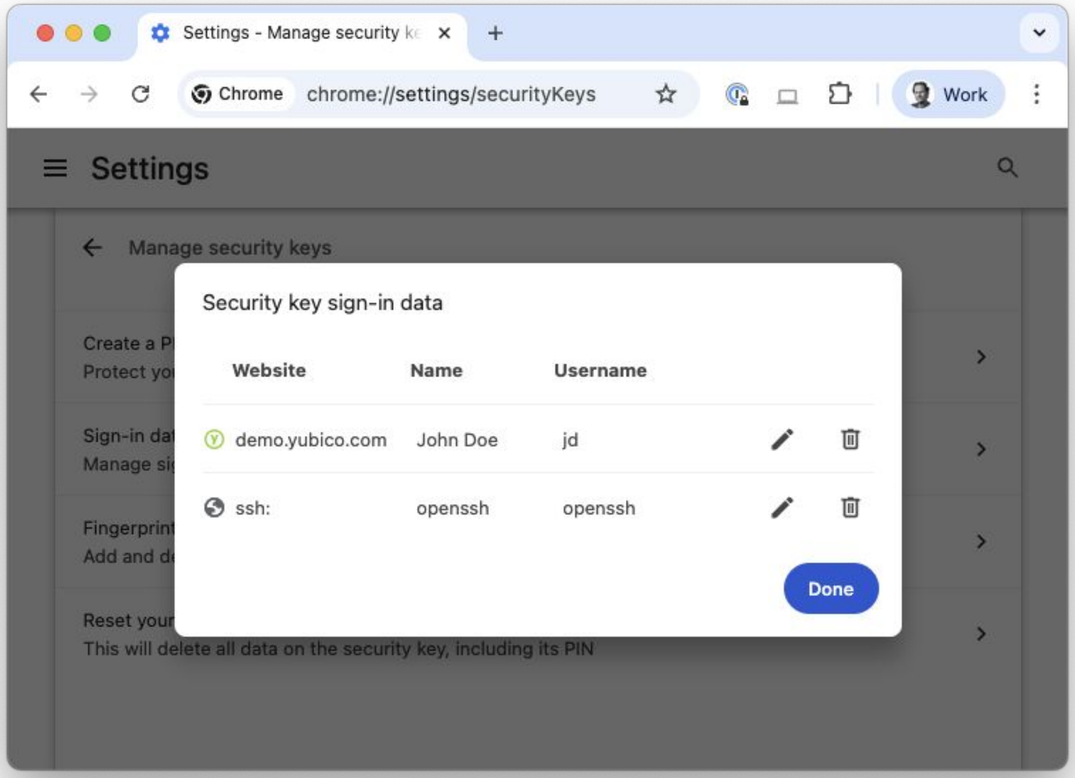
Resident Keys

Stored in a security key's internal memory. Also known as ***Discoverable Credentials*** since CTAP v2.1.

Non-resident Keys

Encrypted with a wrapping key and stored off-key. They do not occupy internal storage slots.

A security key can store a limited amount of resident keys.



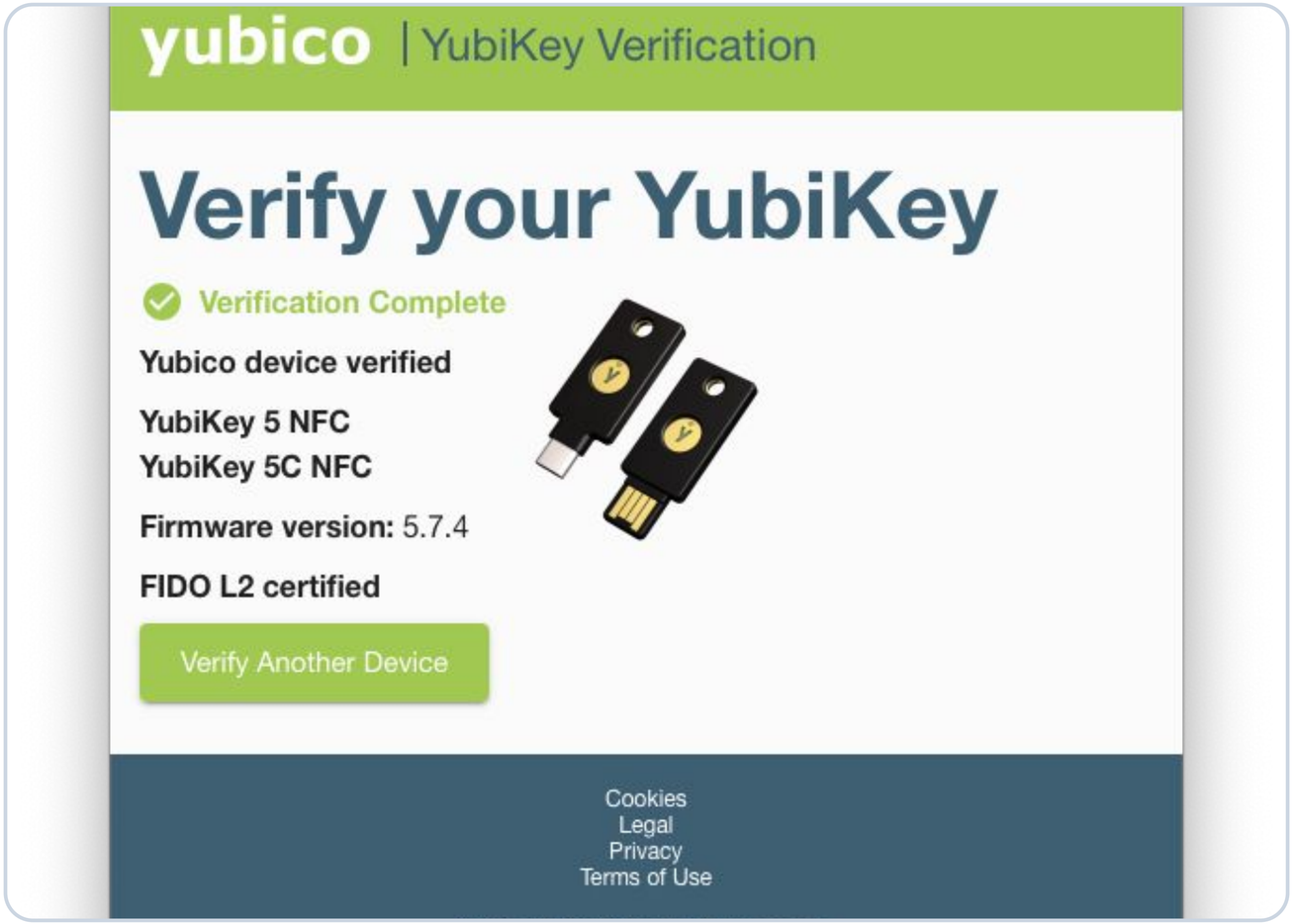
Attestation

Identity Verification

Proves which make and model of security key generated a credential.

Policy Enforcement

Useful for policies, only allow SSH keys generated on approved hardware.



yubico

FIDO2 Token Information

Terminal — fido2-token

```
$ fido2-token -L
ioreg://4296874505: vendor=0x1050, product=0x0407 (Yubico YubiKey OTP+FIDO+CCID)
$ fido2-token -I ioreg://4296874505
proto: 0x02
major: 0x05
minor: 0x07
build: 0x04
caps: 0x05 (wink, cbor, msg)
version strings: U2F_V2, FIDO_2_0, FIDO_2_1_PRE, FIDO_2_1
extension strings: credProtect, hmac-secret, largeBlobKey, credBlob, minPinLength
transport strings: nfc, usb
algorithms: es256 (public-key), eddsa (public-key), es384 (public-key)
aaguid: d7781e5de35346aaafe23ca49f13332a
options: rk, up, noplat, noalwaysUv, credMgmt, authnrCfg, clientPin, largeBlobs, pinUvAuthToken, setMinPINLength, makeCredUvNotRqd
fwversion: 0x50704
maxmsgsiz: 1536
maxcredcntlst: 8
maxcredlen: 128
maxcredblob: 32
maxlargeblob: 4096
maxrpids in minpinlen: 1
remaining rk(s): 98
minpinlen: 4
pin protocols: 2, 1
pin retries: 8
pin change required: false
uv retries: undefined
```


yubico

Resources

Developer Program

Access documentation, SDKs, and community support.

https://dev.yubi.co/

fido2-token Commands

Detailed manual for libfido2's *fido2-token* commands.

https://developers.yubico.com/libfido2/Manuals/fido2-token.html

SSH Authentication Guide

Securing SSH with FIDO2 Security Keys.

https://developers.yubico.com/SSH/Securing_SSH_with_FIDO2.html

OpenSSH Workshop

Hands-on training materials from Yubico Labs.

https://github.com/YubicoLabs/fido-openssh-workshop/

Get in Touch: alan.alvarez@yubico.com | dennis.hills@yubico.com

yubico

Questions?

Let’s Exercise

https://github.com/YubicoLabs/fido-openssh-workshop