Les passkeys: WebAuthn

Què és WebAuthn?

WebAuthn (Web Authentication) és una API de navegador escrita en JavaScript que permet a les aplicacions web utilitzar mètodes d'autenticació forts i basats en criptografia de clau pública. Forma part d'un projecte més gran anomenat **FIDO2**, que és el fruit de la col·laboració entre l'aliança **FIDO** (Fast Identity Online) i el **W3C** (World Wide Web Consortium).

1. Els tres actors del sistema

Perquè WebAuthn funcioni, hi intervenen tres elements:

1. **El relying party (RP):** És el servidor (l'aplicació **CodeIgniter 4**). És qui "confia" en l'autenticació.
2. **El client (User Agent):** El navegador web (Chrome, Firefox, Safari). Actua d'intermediari i garanteix que la web que demana la clau és qui diu ser.
3. **L'authenticator:** El maquinari que genera les claus. Pot ser:
   - **Roaming:** Una clau USB (com una YubiKey).
   - **Platform:** El mateix dispositiu (el lector d'empremtes del MacBook, el FaceID de l'iPhone, el PIN de Windows Hello).

Els pilars del Standard

A. Criptografia de clau pública (Asimètrica)

A diferència de les contrasenyes o el TOTP, aquí no s'envia cap "secret" al servidor.

- Quan l'usuari es registra, l'**Authenticator** crea un parell de claus: una **pública** (que s'envia al servidor) i una **privada** (que no surt mai de l'Authenticator).
- L'autenticació es resol mitjançant un repte matemàtic: el servidor envia una dada, l'usuari la "signa" amb la clau privada, i el servidor verifica la signatura amb la clau pública.

B. Protecció contra el phishing (Origin-Bound)

Aquesta és la característica més potent. El navegador vincula la credencial al **domini (Origin)**.

> **Exemple:** Si un usuari rep un correu fals que el porta a `amazon-fals.com`, el navegador buscarà una clau per a aquest domini. Com que la clau real es va crear per a `amazon.com`, el navegador ni tan sols oferirà la possibilitat de signar. El phishing esdevé impossible.

C. Verificació de l'usuari (User verification)

L'estàndard permet que el servidor exigeixi una prova que l'usuari està present i és qui diu ser. Això es fa mitjançant l'autenticació local (biometria o PIN). El servidor rep un "flag" (bit) confirmant que la identitat de la persona s'ha verificat localment.

Flux de dades o protocol

Per explicar-ho a nivell de codi, WebAuthn defineix dos mètodes principals de l'API `navigator.credentials`:

1. `create()` **(Registre):** El servidor envia opcions (com el nom d'usuari i un challenge aleatori). El navegador genera la clau i retorna el "Attestation Object" (la clau pública).
2. `get()` **(Login/2FA):** El servidor envia el challenge. El navegador demana l'empremta/PIN, signa el challenge i retorna una "Assertion".

Per què s'anomena "passkey" ara?

Mentre que **WebAuthn** és el nom de l'estàndard tècnic, el terme **Passkey** és el nom comercial que Apple, Google i Microsoft han donat a les credencials WebAuthn que se sincronitzen al núvol (iCloud, Google Password Manager).

> **Diferència clau:** Una "Passkey" permet que si l'usuari crea la clau al seu iPhone, la pugui fer servir també al seu iPad o Mac sense haver de tornar-se a registrar, ja que la clau privada viatja de forma segura entre els seus dispositius.

Com funciona?

Per explicar el protocol pas a pas, ens centrarem en el diàleg que es produeix entre el **Servidor (PHP)**, el **Navegador (Client)** i la **YubiKey (Autenticador)**. L'important aquí és entendre que les dades que viatgen són objectes complexos (binaris), però la lògica és un "repte-resposta".

1. El registre / Enrollment

L'objectiu d'aquesta fase és generar la parella de claus i que el servidor guardi la **Clau Pública**.

Pas 1: Sol·licitud (Servidor PHP)

L'usuari ja ha fet login amb user/pass i demana "Afegir YubiKey". El codi PHP genera un objecte de configuració:

- **Challenge:** Una cadena de bytes aleatòria (molt important per evitar atacs de repetició).
- **User ID:** L'ID de l'usuari a la BD (per exemple, l'ID 42).
- **RP ID:** El domini de la web (ex: `la-teva-web.cat`).

Pas 2: Creació (Navegador → YubiKey)

El navegador rep aquestes dades i executa la funció JS: `navigator.credentials.create()`.

1. El navegador demana a l'usuari que insereixi la YubiKey i la toqui.
2. La **YubiKey** genera internament un nou parell de claus (clau privada + clau pública) específic per a aquest domini.
3. La YubiKey signa el Challenge amb la seva nova clau privada.

Pas 3: Verificació i emmagatzematge (Servidor PHP)

El navegador envia el resultat al servidor. El PHP rep:

- La **Clau pública**.
- L'**ID de la credencial** (necessari per saber quina clau demanar després).
- Una signatura que demostra que la YubiKey realment ha creat això.

Què guardes a la teva BD? Guardaràs a una taula (ex: `user_passkeys`) l'**ID de la credencial** i la **Clau Pública** associats a l'usuari.

2. El login / Autenticació 2FA

L'usuari ja té la clau registrada. Ara ve un dia normal de login.

Pas 1: El repte (Servidor PHP)

L'usuari posa el seu usuari i contrasenya. El servidor detecta que té 2FA activat amb Passkey.

- El PHP busca a la BD la **Clau Pública** i el **Credential ID** de l'usuari.
- Genera un nou **Challenge** (aleatori).
- Envia al navegador el Challenge i una llista dels Credential IDs permesos.

Pas 2: La signatura (YubiKey)

El navegador crida a `navigator.credentials.get()`.

1. Apareix una finestra del navegador: "Si us plau, toca la teva clau de seguretat".
2. L'usuari toca la **YubiKey**.
3. La YubiKey busca la clau privada que correspon a l'ID enviat, agafa el Challenge i el **signa** criptogràficament.
4. El navegador torna aquesta signatura al servidor.

Pas 3: La validació final (Servidor PHP)

Aquesta és la part "màgica" on no hi ha contrasenyes compartides:

1. El PHP agafa el Challenge que va enviar originalment.
2. Agafa la **Clau Pública** que tenia guardada a la base de dades.
3. Agafa la **Signatura** que acaba d'arribar de la YubiKey.
4. Utilitza una funció criptogràfica (com `openssl_verify`) per comprovar si la signatura és vàlida per a aquest challenge i aquesta clau pública.

Resultat:

- **Si la signatura és vàlida:** L'usuari és qui diu ser. Sessió iniciada.
- **Si no:** Accés denegat.

Que viatja per la xarxa?

Moment	Què surt del Servidor?	Què surt del Navegador/ YubiKey?
Registre	"Crea una clau per a aquest ID i signa aquest Challenge"	"Aquí tens la Clau Pública i la signatura del Challenge"
Login	"Signa aquest nou Challenge amb la teva clau privada"	"Aquí tens la signatura feta amb la meva clau privada"

Per què la YubiKey?

A diferència d'una Passkey al mòbil, la YubiKey és un dispositiu físic independent. Si l'usuari perd la YubiKey i no té un mètode de recuperació, **no podrà entrar**, ja que la clau privada no es pot extreure del xip de la YubiKey ni tan sols amb microscopis electrònics. És el nivell màxim de seguretat actual.

I si ho canviem per windows Hello o Bitwarden?

Quan substituïm una YubiKey per **Windows Hello** o un gestor de contrasenyes com **Bitwarden**, el protocol WebAuthn és exactament el mateix, però canvia el que anomenem el **contenidor** de la clau privada.

On resideix el "Secret"? L'autenticador

La diferència principal és on es genera i on es guarda la clau privada:

- **Windows Hello (Platform Authenticator):** La clau es genera al xip **TPM** (Trusted Platform Module) de la placa base de l'ordinador. La clau està lligada al hardware físic de la màquina.
- **Bitwarden (Cross-platform / Software Authenticator):** La clau es genera mitjançant programari i es guarda xifrada al núvol de Bitwarden (protegida per la teva "Master Password").

El flux de registre i login

El protocol PHP no canvia, però l'experiència d'usuari i la seguretat interna sí:

- Windows Hello

1. **Registre:** El navegador demana permís. Windows obre una finestra emergent pròpia: "Vols utilitzar el teu rostre, empremta o PIN?".
2. **Verificació d'usuari (UV):** Windows Hello garanteix al servidor que l'usuari és realment qui diu ser (biometria).
3. **Lligat al dispositiu:** Si l'alumne registra el seu portàtil amb Windows Hello, **no podrà fer servir aquesta passkey al mòbil**. La clau privada no surt mai d'aquell xip TPM.

- Bitwarden

1. **Registre:** Quan el servidor envia el challenge, l'extensió de Bitwarden al navegador l'intercepta i s'ofereix per guardar la Passkey.
2. **Sincronització:** Bitwarden puja la clau pública i la privada (xifrada) als seus servidors.
3. **Portabilitat:** A diferència de Windows Hello, si l'usuari va a un altre ordinador i té instal·lat Bitwarden, la Passkey hi serà. Això és el que anomenem una **Synced Passkey**.

Comparativa de seguretat

Característica	YubiKey (Hardware)	Windows Hello (Platform)	Bitwarden (Synced)
On és la clau privada?	Dins la clau USB física.	Al xip TPM de l'ordinador.	Al núvol de Bitwarden (xifrada).
Què passa si perds el PC?	No passa res (la clau és fora).	Perds l'accés (has de fer recovery).	No passa res (està al núvol).
És fàcil de copiar?	Impossible.	Molt difícil (cal accés físic).	Possible si algú et roba el Master Pass.
Factor de forma	Poses la clau i toques el botó.	Rostre, empremta o PIN de Windows.	Clic a l'extensió del navegador.

Què veu el servidor PHP?

A nivell de codi, pràcticament no hi ha diferència, però hi ha un detall tècnic anomenat **Attestation**:

Quan la YubiKey es registra, envia un certificat que diu: "Hola, soc una YubiKey model 5". Quan ho fa Windows Hello o Bitwarden, el certificat diu una altra cosa o, en el cas de moltes Passkeys modernes, s'envia un "None Attestation" per privadesa (perquè el servidor no pugui saber quin model exacte de mòbil o PC fas servir).

> **Punt clau:**

Amb **Windows Hello**, el segon factor és "alguna cosa que TENS" (el PC) + "alguna cosa que ETS/SAPS" (biometria/PIN).

Amb **Bitwarden**, s'assembla més a un gestor de contrasenyes tradicional, però amb la immunitat al phishing que ofereix el protocol WebAuthn.