

# WebAuthn is coming to town!


bodik@cesnet.cz

With courtesy of reused talks by @herrjemand, @equalsJeffH, @Oskar456 and others from FIDO Alliance ;)

# A non-formal prequel ...

Root.cz • Bezpečnost • Nový standard pro přihlašování: nenutíte uživatele měnit hesla

## Nový standard pro přihlašování: nenutíte uživatele měnit hesla



15. 5. 2017

← Tweet

**Jiří Škrampal** @JiriSkrapal · 3h  
Výborně @spazef0rze .  
Aneb proč do médií neustále o IT bezpečnosti mluví tihle strejcové mentálně zaseklí v roce 2005?

potřebu mít silná hesla a pravidelně je měnit – ať už to od zaměstnání, domu, práce i od karet. A zatřetí je nutné být na internetu stále v pozoru. Znamená to například neotvírat podezřelé e-maily a přílohy od neznámých adresátů,” upozorňuje Hládek.

∩\_(ツ)\_∩

Silná hesla a pravidelně je měnit je něco, co nejde dodržet. Každé heslo zcela jiné a napsané v notýsku (ne na monitoru nebo klávesnici) nebo ve správci hesel a místo hesel věty, to vám pomůže mnohem víc.

Neotvíráte podezřelé e-maily? Jak poznám, že je podezřelý, když ho neotevřu? Neotvíráte přílohy od neznámých adresátů? Pokud chcete zpomalit ekonomiku nebo aby vás šéf jebal, že neděláte svou práci, tak s chutí do toho.

A pak se Česká bankovní asociace diví, že lidi nedodržují jejich poučky. Bud se nedají dodržet nebo jsou k ničemu. Nebo obojí.

ČBA, prozradíte, kdo vám pomáhal tahle pravidla sestavit? Díky.

Přestože se začínají prosazovat alternativní způsob velkou část firem a většinu jednotlivců zásadním se velké firmy přijaly opatření týkající se kvality hesel. I podleňovat. Pravidla se postupně zpřísňují, nároky silnější hesla. Teoreticky.

Liší se vám článěk? Podpořte redakci

Michal Spaček @spazef0rze  
Vynucená změna hesla je spíš nebezpečná, píše @adent a já s ním souhlasím: [kcc.misanthrop.info/2015/03/30/hes...](http://kcc.misanthrop.info/2015/03/30/hes...) nějaký data k tomu

Q-B05:

Is password expiration no longer recommended?

A-B05:

SP 800-63B Section 5.1.1.2 paragraph 9 states:

“Verifiers SHOULD NOT require memorized secrets to be changed arbitrarily (e.g., periodically). However, verifiers SHALL force a change if there is evidence of compromise of the authenticator.”

Users tend to choose weaker memorized secrets when they know that they will have to change them in the near future. When those changes do occur, they often select a secret that is similar to their old memorized secret by applying a set of common transformations such as increasing a number in the password. This practice provides a false sense of security if any of the previous secrets has been compromised since attackers can apply these same common transformations. But if there is evidence that the memorized secret has been compromised, such as by a breach of the verifier’s hashed password database or observed fraudulent activity, subscribers should be required to change their memorized secrets. However, this event-based change should occur rarely, so that they are less motivated to choose a weak secret with the knowledge that it will only be used for a limited period of time.

# A non-formal prequel ...

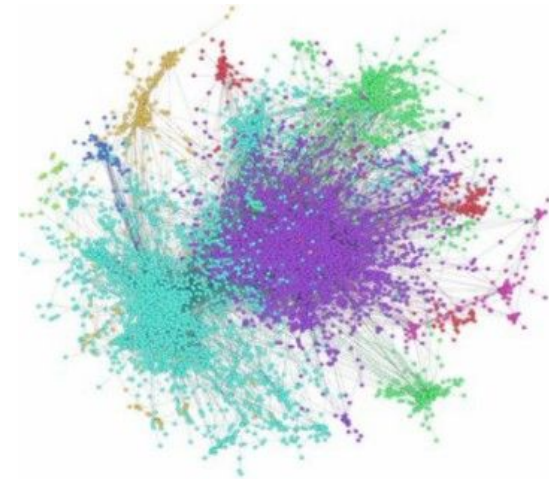
- Do you think that NIST recommendation should be enough for everybody ???



- If password expiration make users to use weak passwords, so the zero password policy requirements would result in usage of strong passwords ?
  - I don't think so

# From a field experience ...

1. Online bruteforce (1st party - login procedure not rate-limited or guarded)
2. Offline bruteforce (1st party - DB leaks)
3. Password reuse (3rd party - DB leaks)
4. Insecure delegation (2nd party - NTLM Wdigest, MSV/Cached Logons, ...)
5. Insecure storage (self - sticky notes, keepass triggers)
6. Phishing (self)



# DCE should be guarded as whole and in depth

- On the contrary to the many other experts ...
- ... I do think that password policy 10 + 3 + 2 is a **good** idea for a common distributed computing environment
  - ... not best idea nor the only one.
  - 10 characters ... average length of a czech word is 5, so it encourages users to use at least two words
  - 3 character classes ... to harden the bruteforce
  - 2 years expiration ... nothing lasts forever

Is there a way out of this ?



**michal.bryxi** 10:27 PM

Tenhle kekel má řešení?

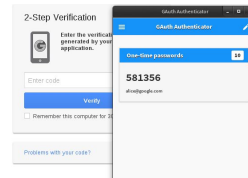
# Black Hat 2019: Every Security Team is a Software Team Now

Start with “yes”

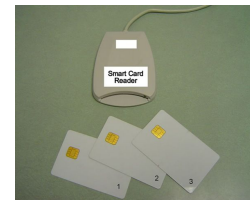
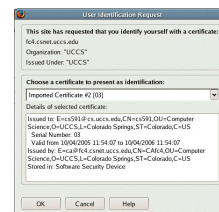


# Current (web) authentication recap

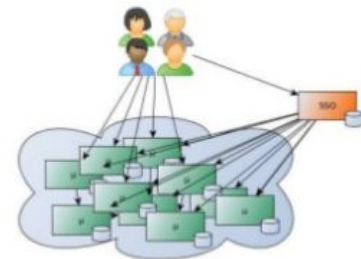
- Username / password
  - + works for everyone (users, developers, business)
  - - password policy
  - - sometimes enhanced by 2FA/MFA
    - - user experience (drivers, special device, recovery, ...)
    - - no real standardization (beyond basic auth and few others ...)



- PKI based x509 authentication
  - - application support
  - - **user experience** (nor users, nor developers understands the technology well enough)
  - - key/cert provisioning vs multiple devices



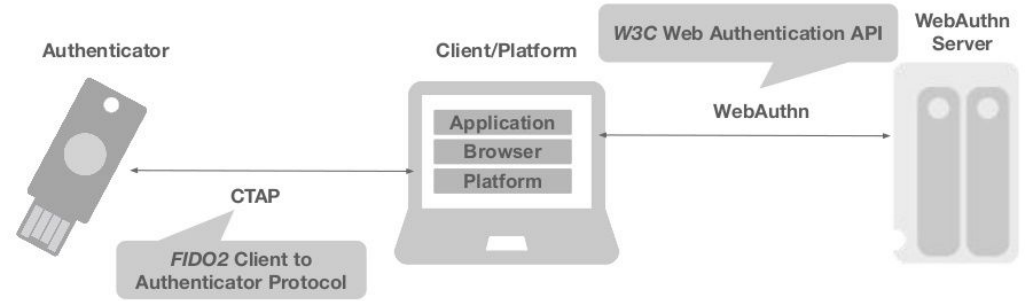
- SSOs / Federations
  - User-centric (mojeid, facebook, ...) / Organization-centric (eduid, ...)
  - Works, but creates a **single point of failure**





And here comes a  
new kid to the block ...

# What WebAuthn IS ?



- WebAuthn (Web Authentication) is a web standard, created by FIDO Alliance and published by the World Wide Web Consortium (W3C).
- The goal of the project is to standardize an interface for authenticating users to web-based applications and services using public-key cryptography.
  - <https://w3c.github.io/webauthn/>
  - <https://webauthn.guide/>
  - <http://slides.com/herrjemand/webauthn-isig>

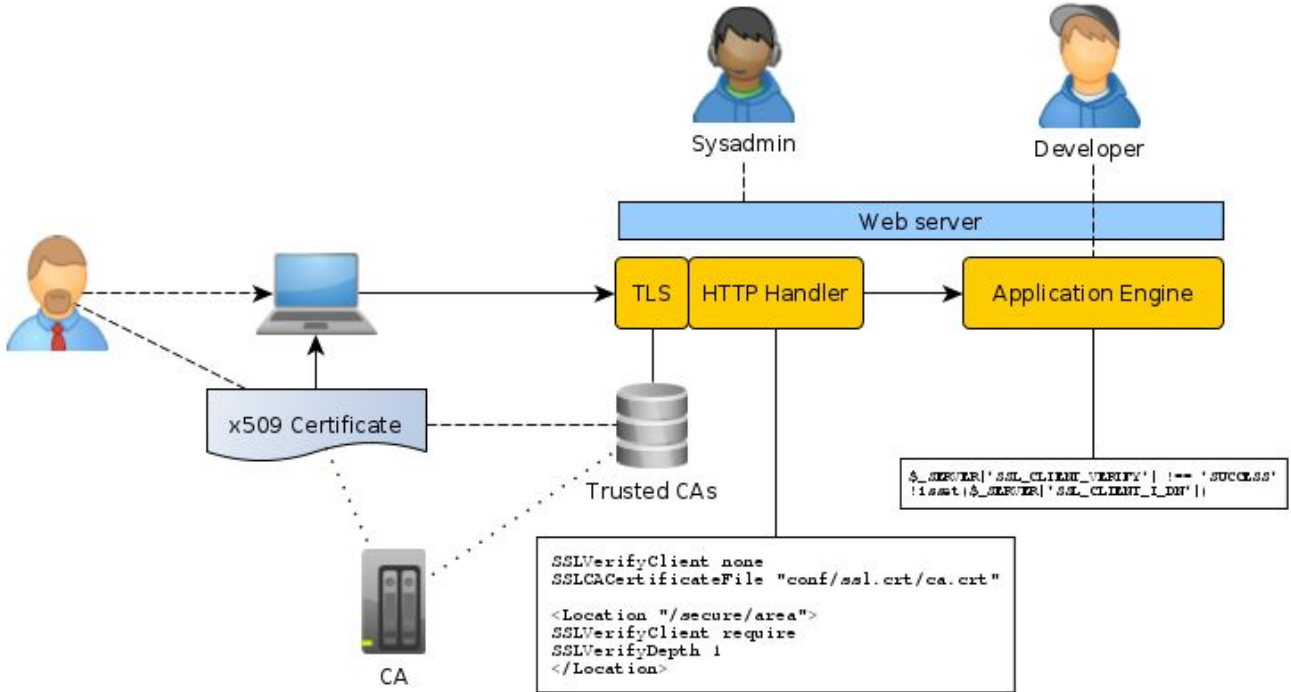
# What WebAuthn **is NOT** ?

- Magic flawless replacement blackbox for currently used authentication schemes.
  
- But it is changing the landscape, because **now there are specs** and implementations in common software
  - Chrome, Firefox, Edge(mium), Windows Hello
  - Java, PHP, Python, Ruby, Go, .NET

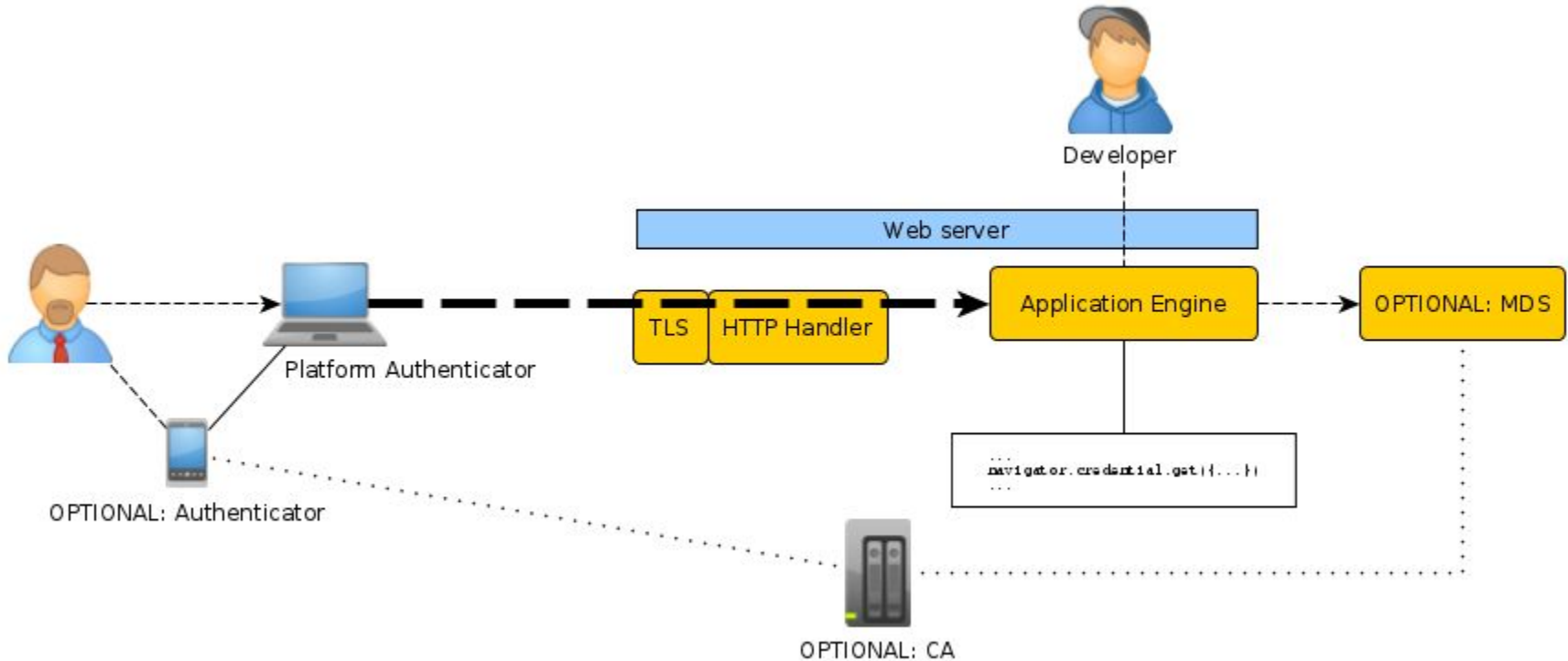
# What WebAuthn **DOES** ?

1. It **replaces** the necessity to care about **passwords** (users are very bad at it) **for** taking care about a **physical token** (users are somewhat good at it).
  - Token can be special device (Yubikey, ...), phone (BLE, NFC, USB) or platform authenticator (TPM with biometrics, ...)
2. **Makes services** itself **more resilient to** a password brute-force **attacks**, eliminating network man-in-the-middle credentials stealing (public key authentication), providing phishing resistant authentication.

# In contrast with x509 authentication ...



... looks easier and end-to-end (user to app)



# An example web application

- <https://github.com/bodik/flask-webauthn-example>
- Components
  - flask, jinja2
  - sqlalchemy, postgresql
  - flask-login
- Quality assurance
  - pylint, flake8
  - pytest, coverage, pytest-selenium
  - travis-ci.org

# FWE branch 10-basic-app

- Features

- User login, logout
  - username/password form based authentication
- Webauthn credential management
  - user can add, remove tokens
- User management
  - user can add, remove users

- Quality assurance

- Test\_login
  - Emulated browser (flask test\_client)
  - Selenium browser (Firefox)



# FWE: Form based authentication

```
@blueprint.route('/login', methods=['GET', 'POST'])
```

```
def login_route():
```

```
    """login route"""
```

```
    form = LoginForm()
```

```
    if form.validate_on_submit():
```

```
        user = User.query.filter(User.username == form.username.data).one_or_none()
```

```
        if user:
```

```
            if form.password.data:
```

```
                if PWS.compare(PWS.hash(form.password.data, PWS.get_salt(user.password)), user.password):
```

```
                    regenerate_session()
```

```
                    login_user(user)
```

```
                    return redirect(url_for('app.index_route'))
```

```
        flash('Invalid credentials.', 'error')
```

```
    return render_template('login.html', form=form)
```

```
<div class="login">
```

```
    <h1>Login</h1>
```

```
    <form class="form-horizontal" method="post">
```

```
        {{ form.csrf_token }}
```

```
        {{ b_wtf.bootstrap_field(form.username, horizontal=True)
```

```
        {{ b_wtf.bootstrap_field(form.password, horizontal=True)
```

```
        {{ b_wtf.bootstrap_field(form.submit, horizontal=True)
```

```
    </form>
```

```
</div>
```

# FWE: Form based authentication tests

```
form = client.get(url_for('app.login_route')).form
form['username'] = test_user.username
form['password'] = tmp_password
response = form.submit()
assert response.status_code == HTTPStatus.FOUND

response = client.get(url_for('app.index_route'))
assert response.xml.xpath('//a[text()="Logout"]')
```

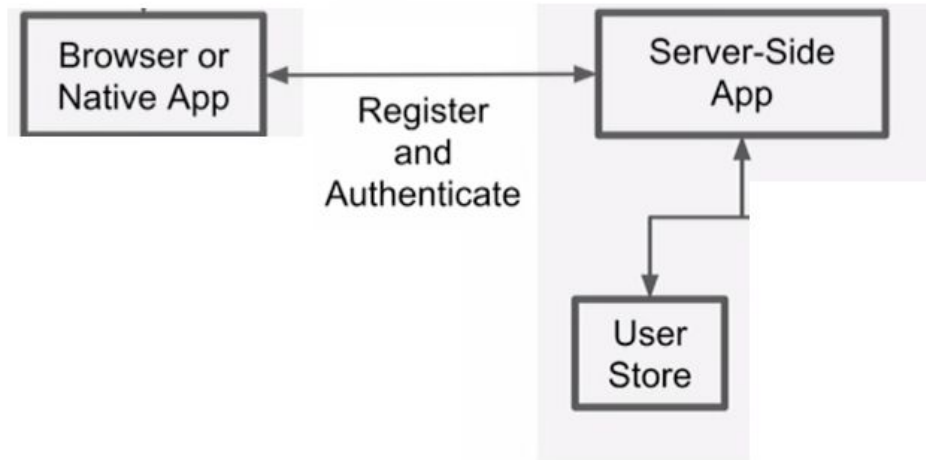
```
selenium.get(url_for('app.login_route', _external=True))
selenium.find_element_by_xpath('//form//input[@name="username"]').send_keys(test_user.username)
selenium.find_element_by_xpath('//form//input[@name="password"]').send_keys(tmp_password)
selenium.find_element_by_xpath('//form//input[@type="submit"]').click()
WebDriverWait(selenium, WEBDRIVER_WAIT).until(EC.presence_of_element_located((By.XPATH, '//a[text()="Logout"]')))
```

**FWE branch 20-webauthn**

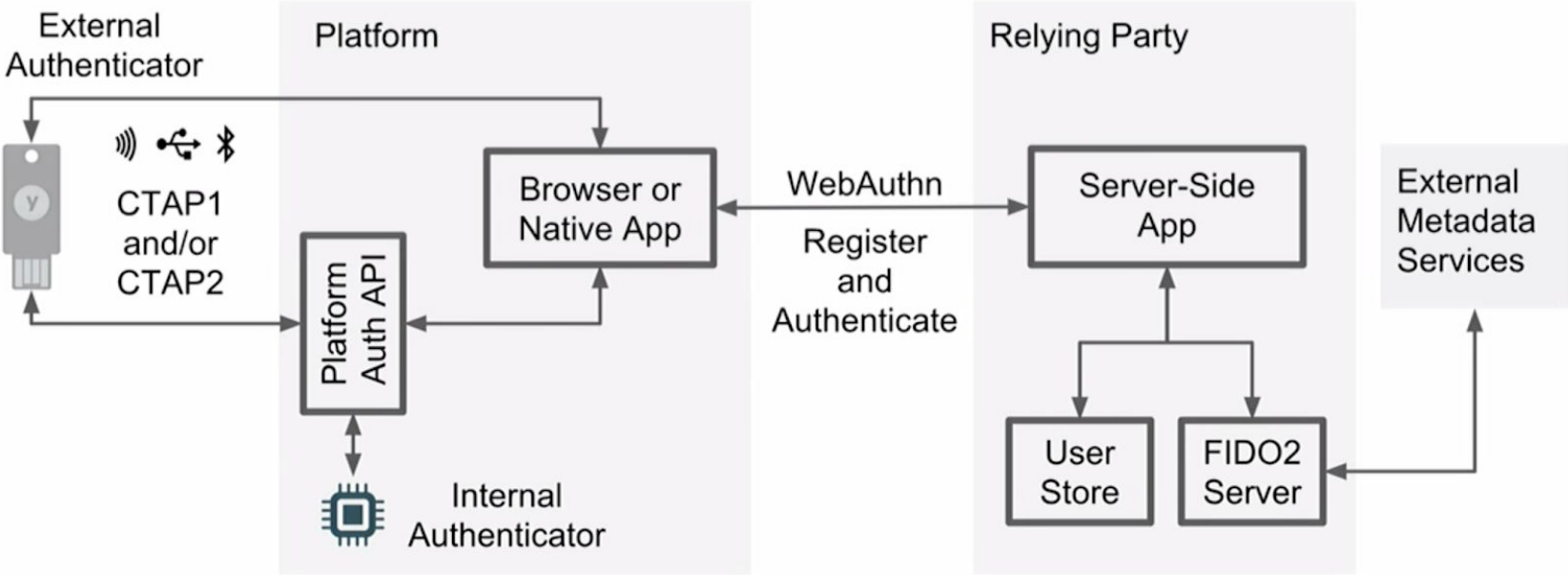
# WebAuthn dictionary

- **User** -- a person
  - User presence -- verification that a person is nearby (touch button).
  - User verification -- verification of the user identity (PIN, biometrics).
- **Client** -- browser
- **Authenticator** -- U2F / FIDO2 token, Platform authenticator, Smartphone
  - **Authenticator transport** -- internal, USB, NFC, Bluetooth Low Energy
  - **CTAP1 / CTAP2** -- Client to authenticator protocol
- **WebAuthn**
  - **CBOR, COSE**
    - Concise Binary Object Representation - structured data binary encoding specification.
    - COSE Object Signing and Encryption - signing and encryption for CBOR objects.
  - **Attestation (data)**
    - Generally an evidence of the origin. In FIDO, a newly generated credential (optionally with proof of the authenticator used).
  - **Assertion (data)**
    - Signed data (RP metadata + challenge) providing data for authentication.

# Implementation WebAuthn for Web application

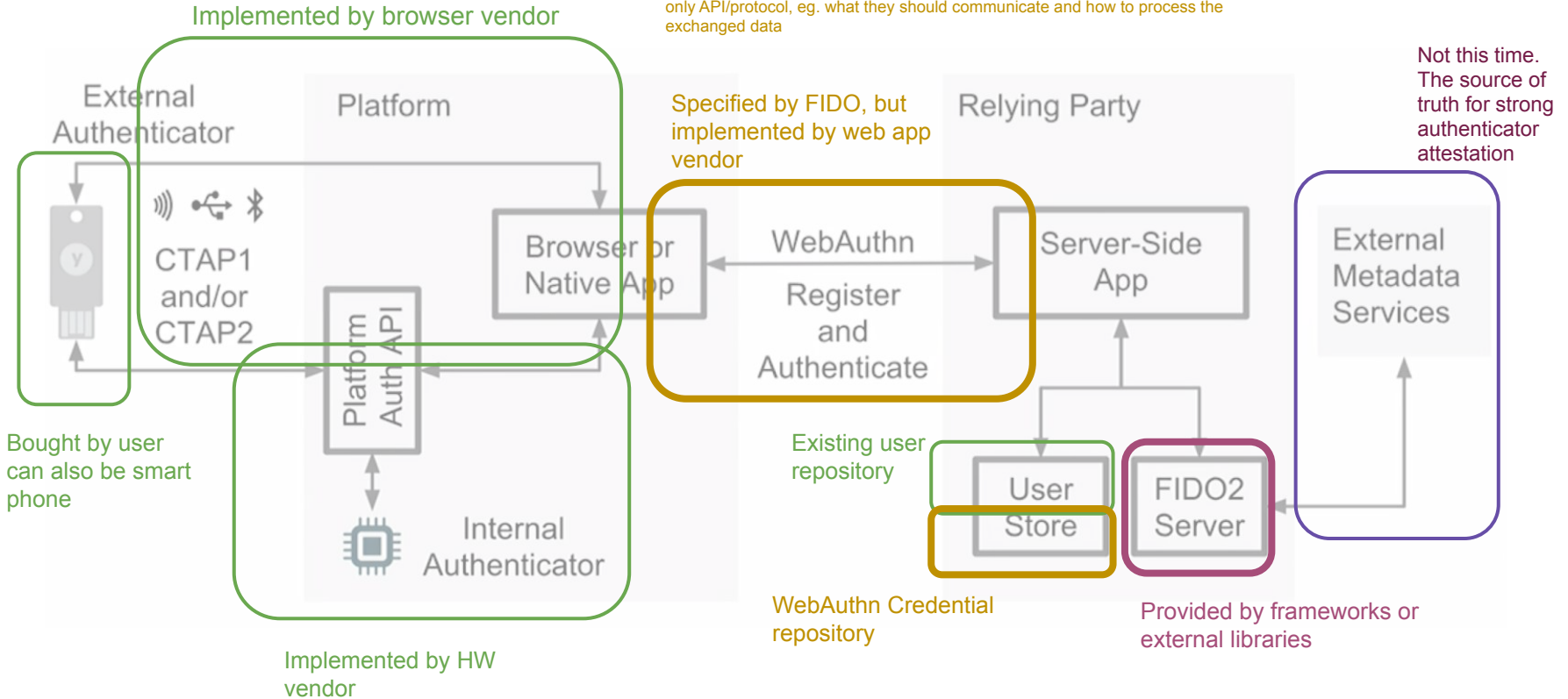


# Implementation WebAuthn for Web application



# Implementation WebAuthn for Web application

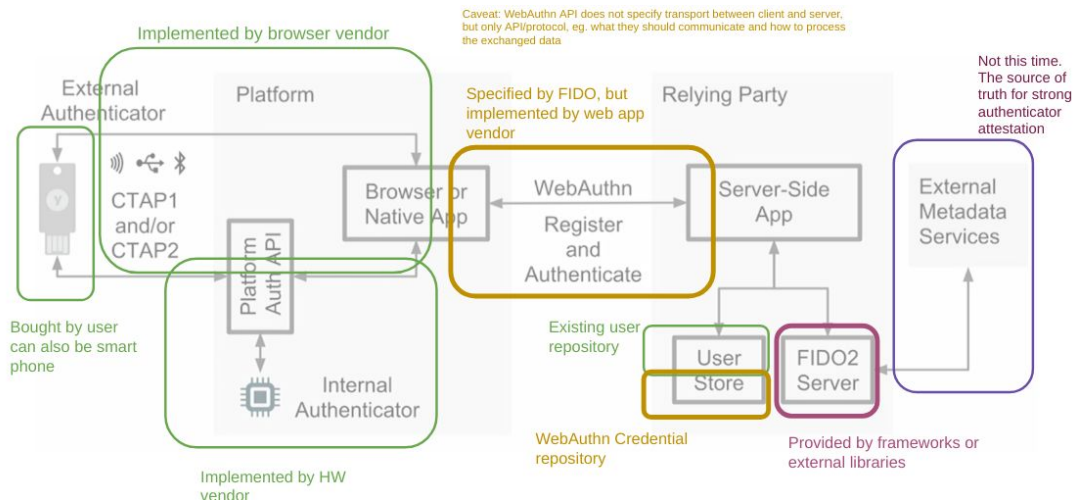
Caveat: WebAuthn API does not specify transport between client and server, but only API/protocol, eg. what they should communicate and how to process the exchanged data



# WebAuthn implementation steps

1. Update user registry to hold registered credentials
  2. Add fido2 server (fido library) for webauthn functions
- Update frontend and backend to perform

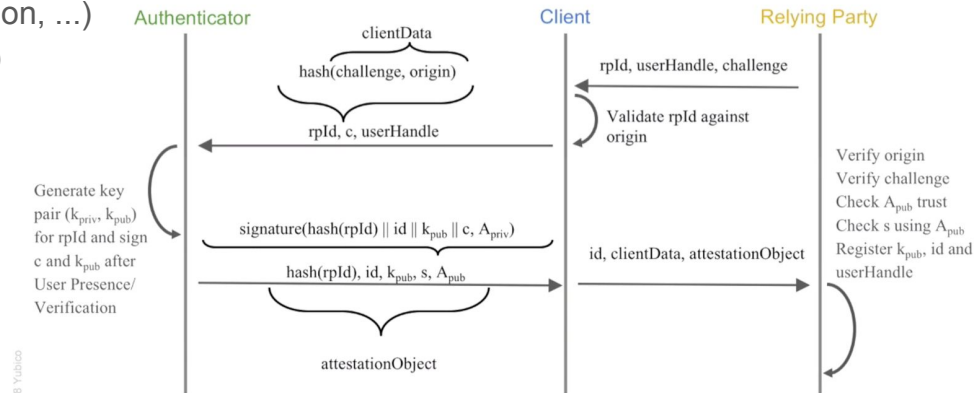
3. Registration flow
4. Authentication flow
5. Implement tests





# How WebAuthn **WORKS** ? - Registration flow

1. User is logged in application
2. User starts a registration flow (clicks “Register new credential”)
3. Application frontend (browser running javascript)
  - a. Requests **PublicKeyCredentialCreationOptions** from backend by ajax call
    - i. Backend returns a challenge along with other parameters (app can request credential with various properties: resident key, user presence, attestation proof, ...)
  - b. Browser calls authenticator with PKCCO parameter
  - c. Authenticator generates a new credential and returns attestation data
  - d. Browser passes the response to backend
4. Application backend
  - a. Unpacks the attestation data (credential, attestation, ...)
  - b. Verifies the response (if attestation was required)
  - c. Stores the credential for later authentication



# How WebAuthn **WORKS** ? - Registration flow

## PublicKeyCredentialCreationOptions

```
{
  publicKey: {
    challenge: Uint8Array(32) [ 176, 120, 37, ... ],
    rp: {
      id: "webauthntest.cesnet.cz"
      name: "webauthntest.cesnet.cz"
    },
    user: {
      displayName: "fwe"
      id: Uint8Array(32) [ 75, 83, 55, ... ]
      name: "fwe"
    },
    pubKeyCredParams: [
      { alg: -7, type: "public-key" },
      { alg: -8, type: "public-key" },
      { alg: -37, type: "public-key" }
      { alg: -257, type: "public-key" }
    ],
    excludeCredentials: Array [],
    authenticatorSelection: {
      requireResidentKey: false,
      userVerification: "preferred"
    },
    attestation: "none",
    timeout: 30000
  }
}
```

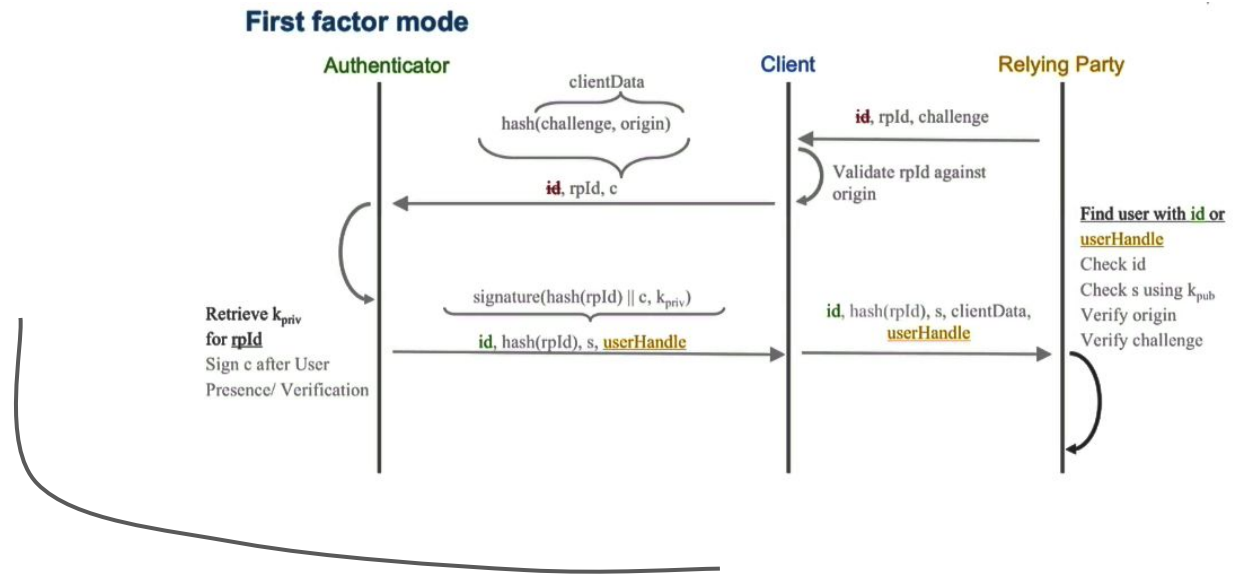
## PublicKeyCredential

```
{
  id: "W1Hq-y-hnRpxgx_Ciu39X5Y...",
  rawId: ArrayBuffer { byteLength: 64 },
  response: {
    attestationObject: ArrayBuffer { byteLength: 226 },
    clientDataJSON: ArrayBuffer { byteLength: 180 }
  },
  type: "public-key"
}
```

# How WebAuthn **WORKS** ? - Authentication flow

1. User is NOT logged in application
2. User starts an authentication flow (clicks “Login” w/o username)
3. Frontend application part (browser running javascript)
  - a. Requests `PublicKeyCredentialRequestOptions` from backend by ajax
    - i. Backend returns a challenge along with other parameters (timeout, allowed credentials, transport requirements, ...)
  - b. Browser calls authenticator with PKCRO parameter
  - c. Authenticator verifies the user, signs challenge and returns assertion data
  - d. Browser passes the response to backend
4. Backend
  - a. Unpacks assertion data
  - b. Verifies the response
  - c. If valid, logs the user in

# Authentication flow (frontend)



```
$(&document).ready(function() {
  console.log(window.PublicKeyCredential ? 'WebAuthn supported' : 'WebAuthn NOT supported');

  get_pkcro()
  .then(pkcro => navigator.credentials.get(pkcro))
  .then(assertion_response => authenticate_assertion(assertion_response))
  .catch(function(error) {
    toastr.error('Webauthn authentication failed.')
    console.log(error.message);
  });
});
```

# Authentication flow (frontend)

Conveying of public key options, attestation and assertion data between RP and browser is not part of the specs. The specs says **what** but **not how** to transport the data ...

```
/**
 * pack and submit credential/assertion object for authentication
 *
 * @param {object}          assertion credential assertion object returned by navigator.credential.get()
 * @return {Promise<undefined>}      A promise that resolves with undefined
 */
function authenticate_assertion(assertion) {
  console.debug('authentication assertion', assertion);

  var assertion_data = {
    'credentialRawId': new Uint8Array(assertion.rawId),
    'authenticatorData': new Uint8Array(assertion.response.authenticatorData),
    'clientDataJSON': new Uint8Array(assertion.response.clientDataJSON),
    'signature': new Uint8Array(assertion.response.signature),
    'userHandle': new Uint8Array(assertion.response.userHandle)
  };

  var form = $('#webauthn_login_form')[0];
  form.assertion.value = array_buffer_to_base64(CBOR.encode(assertion_data));
  form.submit();
}
```

# Authentication flow (backend)

```
if form.validate_on_submit():
    try:
        assertion = cbor.decode(b64decode(form.assertion.data))
        webauthn.authenticate_complete(
            session.pop('webauthn_login_state'),
            webauthn_credentials(user),
            assertion['credentialRawId'],
            ClientData(assertion['clientDataJSON']),
            AuthenticatorData(assertion['authenticatorData']),
            assertion['signature'])
        regenerate_session()
        login_user(user)
        return redirect(url_for('app.index_route'))

    except (KeyError, ValueError) as e:
        current_app.logger.exception(e)
        flash('Login error during Webauthn authentication.', 'error')
```

# Quality Assurance

WebAuth flows can be tested with software tokens

```
(venv) conda activate /opt/fwe/venv/
python -m flake8 fwe tests
python -m pylint fwe tests

-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

-----
coverage run --source fwe -m pytest tests/app -x -vv
----- test session starts -----
platform linux -- Python 3.7.3, pytest-3.10.1, py-1.8.0, pluggy-0.13.0 -- /opt/fwe/venv/bin/python3
cachedir: .pytest_cache
driver: Firefox
sensitiveurl: .*
metadata: {'Python': '3.7.3', 'Platform': 'Linux-4.19.0-5-amd64-x86_64-with-debian-10.0', 'Packages': {'pytest': '3.10.1', 'py': '1.8.0', 'pluggy': '0.13.0'}, 'Plugins': {'base-url': '1.4.1', 'html': '1.22.0', 'Flask': '0.15.0', 'selenium': '1.17.0', 'metadata': '1.8.0', 'variables': '1.8.0'}}
rootdir: /opt/fwe, inifile: pytest.ini
plugins: base-url-1.4.1, html-1.22.0, flask-0.15.0, selenium-1.17.0, metadata-1.8.0, variables-1.8.0
collected 23 items

tests/app/test_auth.py::test_login_route PASSED [ 4%]
tests/app/test_auth.py::test_logout_route PASSED [ 8%]
tests/app/test_auth.py::test_not_logged_in PASSED [ 13%]
tests/app/test_commands.py::test_dbinit_command PASSED [ 17%]
tests/app/test_commands.py::test_dbremove_command PASSED [ 21%]
tests/app/test_default.py::test_index_route PASSED [ 26%]
tests/app/test_default.py::test_user_list_route PASSED [ 30%]
tests/app/test_default.py::test_user_add_route PASSED [ 34%]
tests/app/test_default.py::test_user_delete_route PASSED [ 39%]
tests/app/test_models.py::test_models PASSED [ 43%]
tests/app/test_password_supervisor.py::test_all -- fwe/password_supervisor.py PASSED [ 47%]
tests/app/test_password_supervisor.py::test_password_supervisor PASSED [ 52%]
tests/app/test_sessions.py::test_timeout_session PASSED [ 56%]
tests/app/test_sessions.py::test_notexist_session PASSED [ 60%]
tests/app/test_sessions.py::test_gc_session PASSED [ 65%]
tests/app/test_webauthn.py::test_webauthn_list_route PASSED [ 69%]
tests/app/test_webauthn.py::test_webauthn_delete_route PASSED [ 73%]
tests/app/test_webauthn.py::test_webauthn_register_route PASSED [ 78%]
tests/app/test_webauthn.py::test_webauthn_pkcco_route_invalid_request PASSED [ 82%]
tests/app/test_webauthn.py::test_webauthn_register_route_invalid_attestation PASSED [ 86%]
tests/app/test_webauthn.py::test_webauthn_login_route PASSED [ 91%]
tests/app/test_webauthn.py::test_webauthn_pkcro_route_invalid_request PASSED [ 95%]
tests/app/test_webauthn.py::test_webauthn_invalid_assertion PASSED [100%]

----- 23 passed in 6.26 seconds -----

-----
coverage report --show-missing --fail-under 100
Name Stmts Miss Cover Missing
-----
fwe/_init_.py 29 0 100%
fwe/commands.py 15 0 100%
fwe/controller.py 133 0 100%
fwe/forms.py 18 0 100%
fwe/models.py 28 0 100%
fwe/password_supervisor.py 84 0 100%
fwe/sessions.py 71 0 100%
fwe/wrapped_fido2_server.py 8 0 100%
-----
TOTAL 386 0 100%
python -m pytest -x -vv tests/selenium
----- test session starts -----
platform linux -- Python 3.7.3, pytest-3.10.1, py-1.8.0, pluggy-0.13.0 -- /opt/fwe/venv/bin/python
cachedir: .pytest_cache
driver: Firefox
sensitiveurl: .*
metadata: {'Python': '3.7.3', 'Platform': 'Linux-4.19.0-5-amd64-x86_64-with-debian-10.0', 'Packages': {'pytest': '3.10.1', 'py': '1.8.0', 'pluggy': '0.13.0'}, 'Plugins': {'base-url': '1.4.1', 'html': '1.22.0', 'Flask': '0.15.0', 'selenium': '1.17.0', 'metadata': '1.8.0', 'variables': '1.8.0'}}
rootdir: /opt/fwe, inifile: pytest.ini
plugins: base-url-1.4.1, html-1.22.0, flask-0.15.0, selenium-1.17.0, metadata-1.8.0, variables-1.8.0
collected 4 items

tests/selenium/test_auth.py::test_login PASSED [ 25%]
tests/selenium/test_default.py::test_index_route PASSED [ 50%]
tests/selenium/test_webauthn.py::test_webauthn_register_route PASSED [ 75%]
tests/selenium/test_webauthn.py::test_login_webauthn PASSED [100%]

----- 4 passed in 27.19 seconds -----
```

# soft-webauthn -- python software token

- <https://pypi.org/project/soft-webauthn/>
  - <https://github.com/bodik/soft-webauthn>
  - Based on Yubico python-fido2 library

## 🔗 Python software webauthn token

build passing

Package is used for testing webauthn enabled web applications. The use-case is authenticator and browser emulation during web application development continuous integration.

`SoftWebauthnDevice` class interface exports basic navigator interface used for webauthn features:

- `SoftWebauthnDevice.create(...)` aka `navigator.credentials.create(...)`
- `SoftWebauthnDevice.get(...)` aka `navigator.credentials.get(...)`

To support authentication tests without prior registration/attestation, the class exports additional functions:

- `SoftWebauthnDevice.cred_init(rp_id, user_handle)`
- `SoftWebauthnDevice.cred_as_attested()`



# FWE: WebAuthn registration tests

```
def test_webauthn_register_route(cl_user):
    """register new credential for user"""

    device = SoftWebauthnDevice()

    response = cl_user.get(url_for('app.webauthn_register_route'))
    # some javascript code must be emulated
    pkcco = cbor.decode(b64decode(cl_user.post(url_for('app.webauthn_pkcco_route'), {'csrf_token': get_csrf_token(cl_user)}).body))
    attestation = device.create(pkcco, 'https://%s' % webauthn.rp.ident)
    attestation_data = {
        'clientDataJSON': attestation['response']['clientDataJSON'],
        'attestationObject': attestation['response']['attestationObject']}
    form = response.form
    form['attestation'] = b64encode(cbor.encode(attestation_data))
    # and back to standard test codeflow
    form['name'] = 'pytest token'
    response = form.submit()

    assert response.status_code == HTTPStatus.FOUND
    user = User.query.filter(User.username == 'pytest_user').one()
    assert user.webauthn_credentials
```

# FWE: WebAuthn registration tests

## Webauthn registration selenium test

```
def test_webauthn_register_route(live_server, sl_user): # pylint: disable=unused-argument
    """register new credential for user"""

    device = SoftWebauthnDevice()

    sl_user.get(url_for('app.webauthn_register_route', _external=True))
    # some javascript code must be emulated
    WebDriverWait(sl_user, WEBDRIVER_WAIT).until(js_variable_ready('window.pkcco_raw'))
    pkcco = cbor.decode(b64decode(sl_user.execute_script('return window.pkcco_raw;').encode('utf-8')))
    attestation = device.create(pkcco, 'https://%s' % webauthn rp.ident)
    sl_user.execute_script('pack_attestation(CBOR.decode(base64_to_array_buffer("%s")));' % b64encode(cbor.encode(attestation)))
    # and back to standard test codeflow
    sl_user.find_element_by_xpath('//form[@id="webauthn_register_form"]//input[@name="name"]').send_keys('pytest token')
    sl_user.find_element_by_xpath('//form[@id="webauthn_register_form"]//input[@type="submit"]').click()


    user = User.query.filter(User.username == 'pytest_user').one()
    assert user.webauthn_credentials
```


# There's a lot more ...


- U2F vs. Resident keys
- Password-less vs. Username-less authentication
- User presence vs. verification (touch, pin, biometrics, ...)
- User best practices (lost token procedure, account recovery, ...)
- Real world support in mobile devices
- Enterprise grade deployment
  - Windows Hello and beyond


# Current support for WebAuthn and FIDO2


<https://fidoalliance.org/fido2/fido2-web-authentication-webauthn/>


U2F API			WebAuthn API				
 Chrome Desktop							Windows MacOS & Linux
CTAP1 / U2F			CTAP2				
USB	NFC	BLE	USB	NFC	BLE	Win10	

U2F API			WebAuthn API				
 Firefox							Windows MacOS & Linux
CTAP1 / U2F			CTAP2				
USB	NFC	BLE	USB	NFC	BLE	Win10	

U2F API			WebAuthn API				
 Safari macOS							
CTAP1 / U2F			CTAP2				
USB	NFC	BLE	USB	NFC	BLE	os	

U2F API			WebAuthn API				
 Chrome Android							
CTAP1 / U2F			CTAP2				
USB	NFC	BLE	USB	NFC	BLE	Android	

U2F API			WebAuthn API				
 Edge							
CTAP1 / U2F			CTAP2				
USB	NFC	BLE	USB	NFC	BLE	Win10	

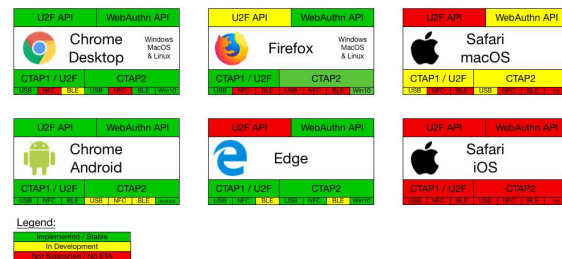
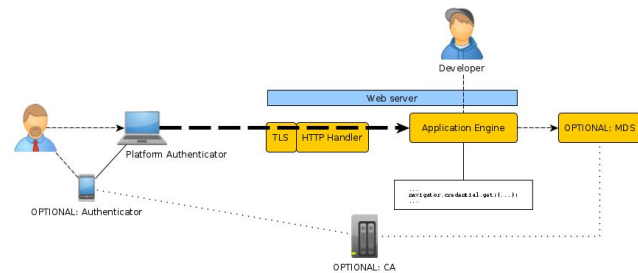
U2F API			WebAuthn API				
 Safari iOS							
CTAP1 / U2F			CTAP2				
USB	NFC	BLE	USB	NFC	BLE	os	

Legend:

Implemented / Stable
In Development
Not Supported / No ETA

# Wrap up

- There is an alternative for password authentication
  - Based on public key cryptography
  - Available in currently used software
- It is (will be) more convenient for users
  - Platform authenticators
  - Smartphones
  - External key
- If done properly
  - Less passwords to remember
  - No bruteforce on services login
  - Database leaks does not compromise long term secrets
  - No credential reuse
  - Phishing resistant authentication





Let's make  
authentication great  
again !

# References

- <https://w3c.github.io/webauthn/>
  - <https://webauthn.guide/>
  - <https://webauthn.io/>
  - <http://slides.com/herrjemand/webauthn-isig>
  - <https://slides.com/fidoalliance/jan-2018-fido-seminar-webauthn-tutorial>
  - <https://hybrismart.com/2019/05/23/authentication-with-hardware-security-keys-via-webauthn-in-sap-commerce-cloud/>
  - <https://www.imperialviolet.org/2019/08/10/ctap2features.html>
  - <https://fidoalliance.org/securing-a-web-app-with-passwordless-web-authentication/>
  - <https://medium.com/@herrjemand/introduction-to-webauthn-api-5fd1fb46c285#2bd3>
- 
- <https://github.com/bodik/flask-webauthn-example>
  - <https://github.com/bodik/soft-webauthn>