inovex

Built-in security

Sichere Webapps dank moderner Browserfeatures

Clemens Hübner
inovex

Clemens Hübner

Software Security Engineer @ inovex

Helps to secure systems, still hacks them

Located in Munich

@ClemensHuebner

chuebner@inovex.de

@inovexgmbh

@inovexlife


Browsers are the central entrance to the internet

Server and client - it’s complicated

Don’t trust the client!

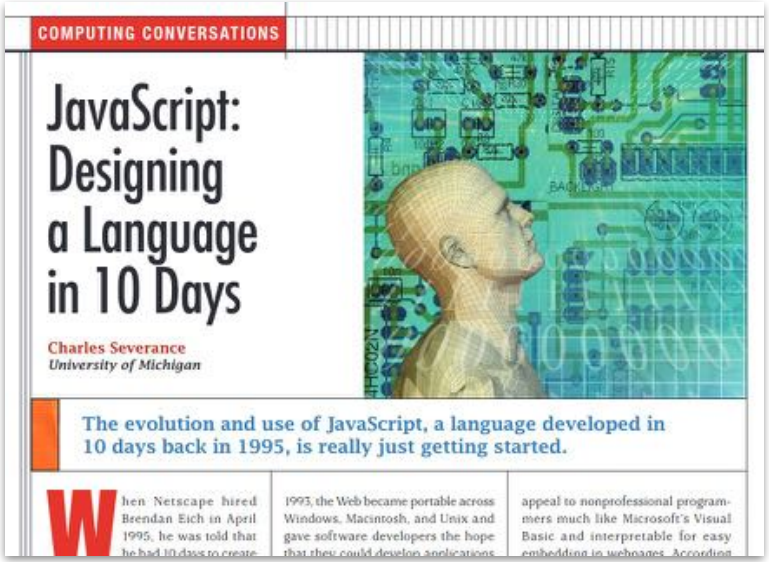
Need (or wish) to do some things in the browser:

- User interaction
- Access to device features
- Strong authentication
- E2E-encryption
- Client-side reporting

inovex

Problems developing client-side webapps

- Limitations of JavaScript
- Problems with 3rd-party dependencies
- Diversity of browsers, systems, devices
- Environment is hard to control and potentially compromised


COMPUTING CONVERSATIONS

JavaScript: Designing a Language in 10 Days

Charles Severance
University of Michigan

The evolution and use of JavaScript, a language developed in 10 days back in 1995, is really just getting started.

When Netscape hired Brendan Eich in April 1995, he was told that

1993, the Web became portable across Windows, Macintosh, and Unix and gave software developers the hope

appeal to nonprofessional programmers much like Macrosoft's Visual Basic and interpretable for easy

Existing security measures in browsers

- Same Origin Policy
- HTTPS
  - Certificate management, warning of unsecure connections
- Cookie Handling
- Security Header
  - X-Frame-Options
  - X-XSS-Protection
  - CSP
- Credential management & checking

Implementing new features for the client

STRONG AUTHENTICATION

E2E-ENCRYPTION

CLIENT-SIDE REPORTING

inovex

STRONG AUTHENTICATION

inovex

Status quo of authentication

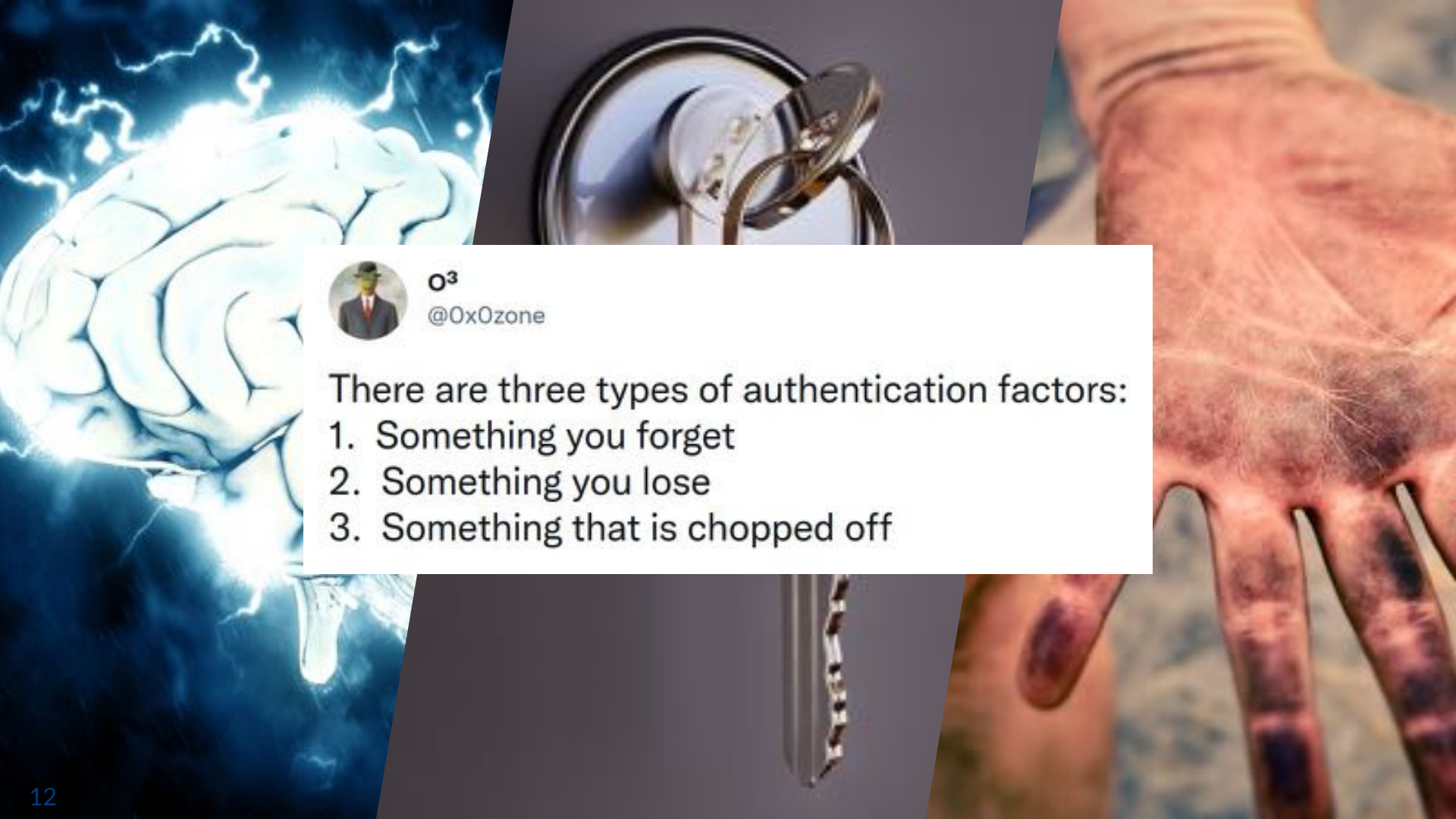
- Authentication with username & password
- sometimes: 2FA (TOTP, Push Notification, ...)

inovex

Problem with password-based authentication

- Human brains are not made for remembering random strings
  → need for password managers
- Threat of phishing, even with 2FA
- Need for transmitting and storing passwords securely




O³
@0x0zone
There are three types of authentication factors:
1. Something you forget
2. Something you lose
3. Something that is chopped off

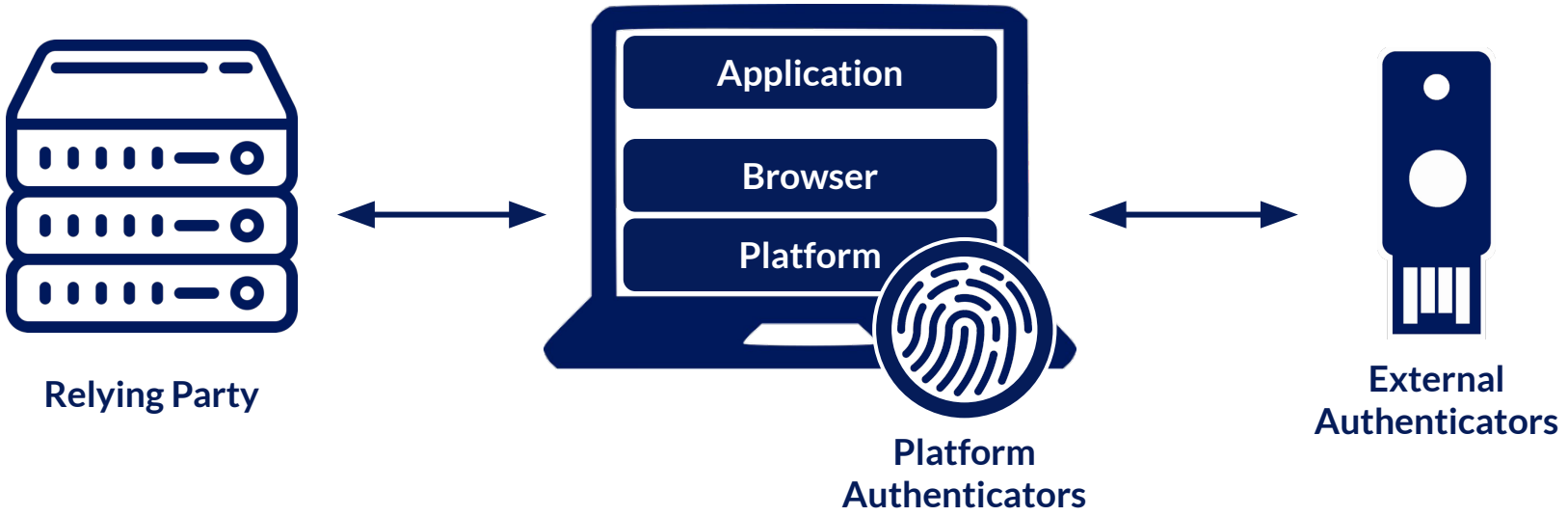
Solution: Less knowledge-based authentication

- use possessed or biometric factors
- use public-key based challenge-response (no leakage of any secret)
- integration into platform for integrated phishing protection

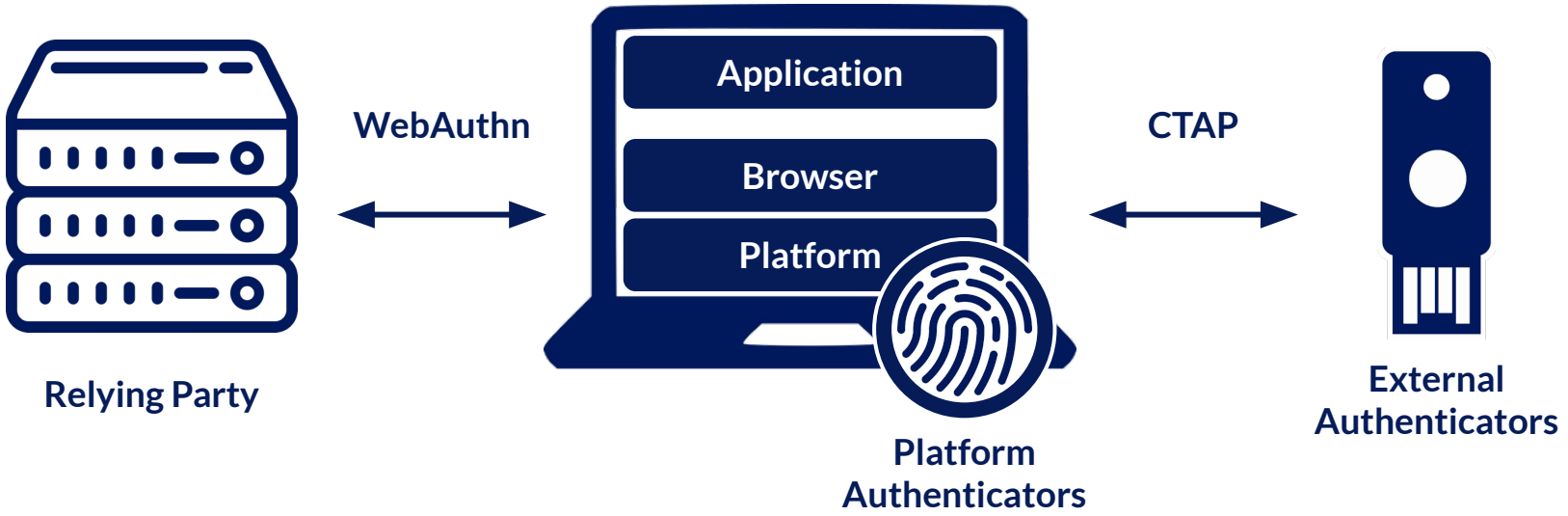
inovex

Demo time!

Architecture



Architecture



inovex

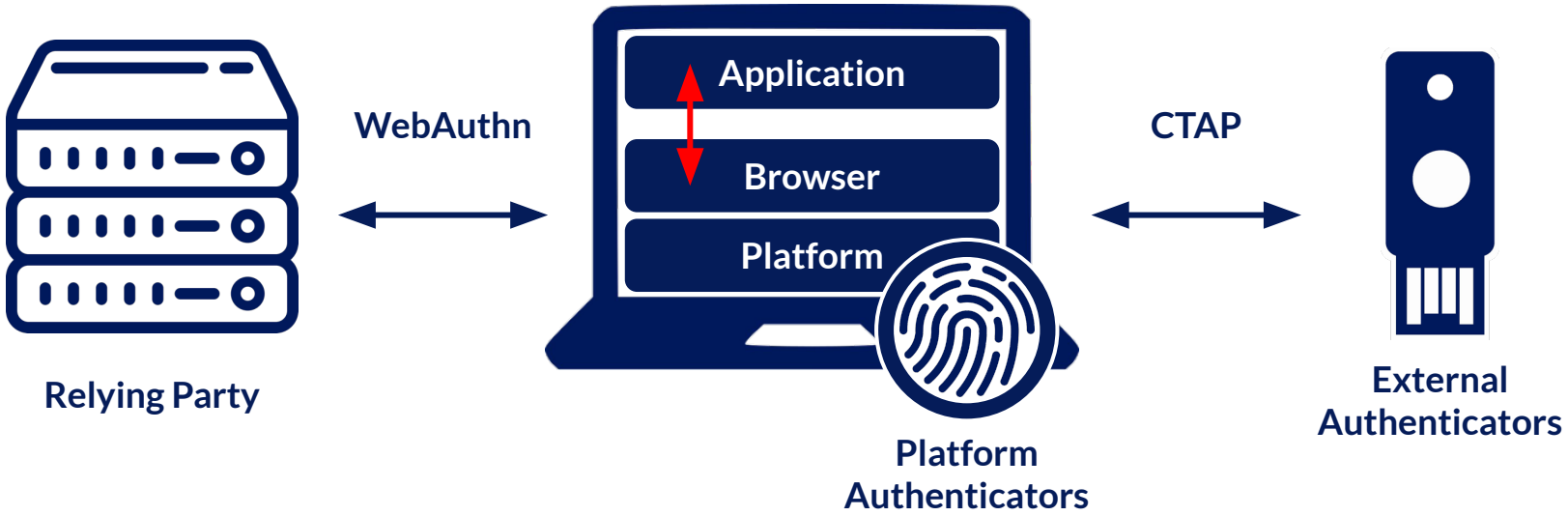
The two WebAuthn Ceremonies

Registration Ceremony	Authentication Ceremony
Creating a public key credential, scoped to a Relying Party with a user's account	**Proving** the presence and consent of the user that registered the public key credential

Attention: That's all!

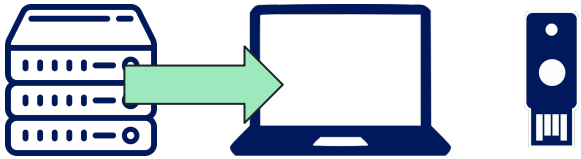
inovex

Architecture



inovex

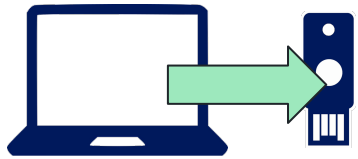
Registration flow: Initiation



```
{
      "rp":{
         "name":"WebAuthn demo server", "id":"webauthndemo.inovex.de"
      },
      "user":{
         "id":"CCDcj5Dx", "displayName":"Clemens Hübner"
      },
      "challenge":"Q7MBekjcE9LlIwcyskj_Dj_SHtJkfe1QemS8HhoRRrA",
      "pubKeyCredParams":
         [{ "alg":-7, "type":"public-key" }, { "alg":-257, "type":"public-key" }],
      "authenticatorSelection":{
         "authenticatorAttachment":"cross-platform"
      },
      "attestation":"direct"
}
```

→ **PublicKeyCredentialCreationOptions**

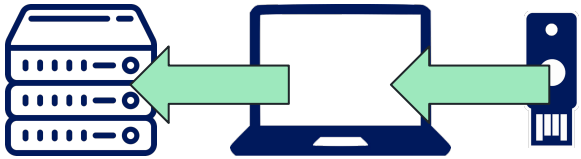
Registration flow: API-Call



```
navigator.credentials.create(
    { publicKey: PublicKeyCredentialCreationOptions }
);
```

inovex

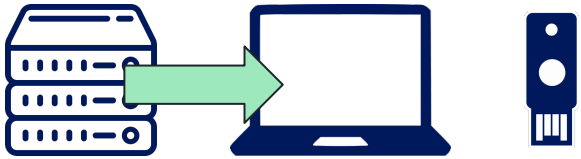
Registration flow: Response




```
{
    "attestationObject": "o2NmbXRoZmlkby11MmZnYXR0U3RtdKJjc2lnWEgwRgIhAP9qYQY...",
    "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJyZTlKSUNUX1VXZUJ2ekhETkJRd2tlWU9DTF..."
}
```

```
{
    "challenge": "Q7MBekjcE9LlIwcyskj_Dj_SHtJkfe1QemS8HhoRRrA",
    "hashAlgorithm": "SHA-256",
    "origin": "https://webauthndemo.inovex.de",
    "type": "webauthn.create"
}
```

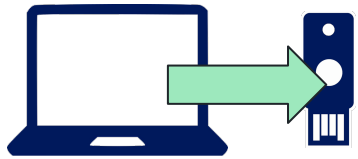
Login flow: Initiation




```
{
   "challenge":"fyKHLm3Rqt3jntx7XEXOA5x3uJb1yjmv20aod3gvj8Y",
   "rpId":"webauthndemo.inovex.de",
   "allowCredentials":[
      {
         "type":"public-key",
         "id":"1eVJX8Po7SSASUnDGnHcV_I03zLMaOXvn89jHUDvfqTdU6hJ9AJO9XWh",
         "transports":"usb,nfc"
      }
   ],
   "userVerification":"preferred”
}
```


→ **PublicKeyCredentialRequestOptions**

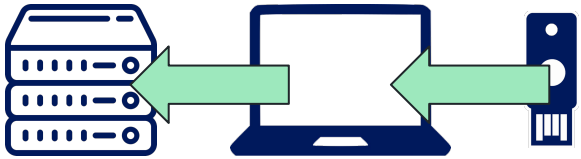
Login flow: API-Call



```
navigator.credentials.get(
    { publicKey: PublicKeyCredentialRequestOptions }
);
```

inovex

Login flow: Response




```
{
    "authenticatorData": "SZYN5YgOjGh0NBcPZHZgW4_krrmihjLHmVzzuoMdl2MBAAAAnQ",
    "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJFWmtXN2ZKSjV4RFo5NFdFT1l1emt4UW1jeVpo...",
    "signature": "MEUCIQCEdWLdMFw-F5RdYpNA8-ZFgZbhqS49foTZNVXvGE5GygIgLyWPXk6lXxR2e8yY"
}
```

```
{
  "challenge": "fyKHLm3Rqt3jntx7XEXOA5x3uJb1yjmv20aod3gvj8Y"
  "hashAlgorithm": "SHA-256",
  "origin": "https://webauthndemo.inovex.de",
  "type": "webauthn.get"
}
```

WebAuthn

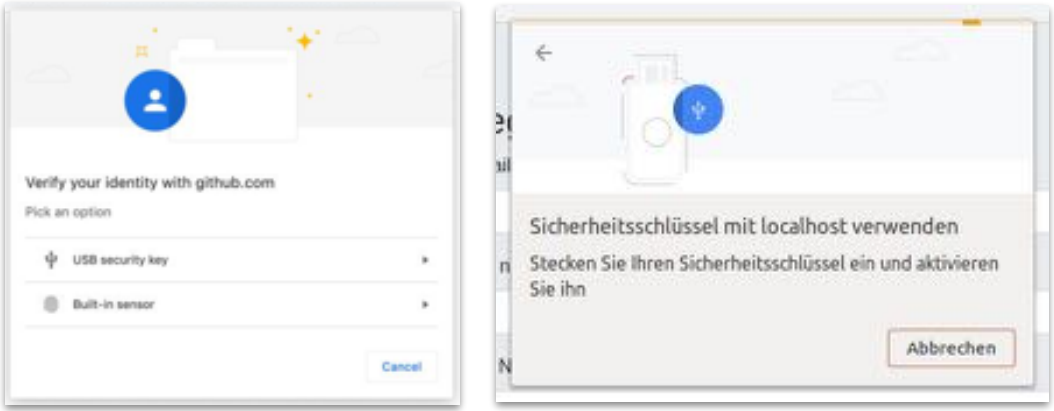
- W3C Recommendation since March 2019 (Level 1) resp. April 2021 (Level 2)
- today, >90% global usage possibility

IE	Edge *	Firefox	Chrome	Safari	Safari on iOS *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			100		15.3				
	101	100	101	15.4	15.4				16.0
11	102	101	102	15.5	15.5	all	102	12.12	17.0
		102	103	16.0	16.0				
		103	104	TP					
			105						

caniuse.com, 21.06.2022

Different UI on different platforms

Firefox 69 / 89



Chrome 92

Android 8 / 11

inovex

Takeaways WebAuthn

- promising standard, yet not widely implemented
- platform-dependent UI and behaviour
- basic functions, needs processes and concepts around it
- implementation requires user education and guidance

inovex

E2E ENCRYPTION

Problems with cryptography in the browser

- no native implementation of cryptographic operations
  - no cryptographic randomness
- wide range of libraries with varying quality
- difficulty to protect against different attack vectors, e.g. side-channel attacks
- bad performance
- need to maintain critical dependencies

***Anyone** can design a cipher that he **himself cannot break**. This is why you should uniformly **distrust amateur cryptography**, and why you should **only use published** algorithms that have **withstood broad cryptanalysis**.*

\- BRUCE SCHNEIER

Use cases for client-side cryptography

- end-to-end encryption for cloud storage
- improve authentication methods
- secure transport of sensitive data
- integrity protection of locally stored data

Web Crypto API

- asynchronous, platform-independent JavaScript-API
- interface for low-level cryptographic operations
  - cryptographically secure random numbers
  - key generation and handling
  - symmetric and asymmetric crypto systems

inovex

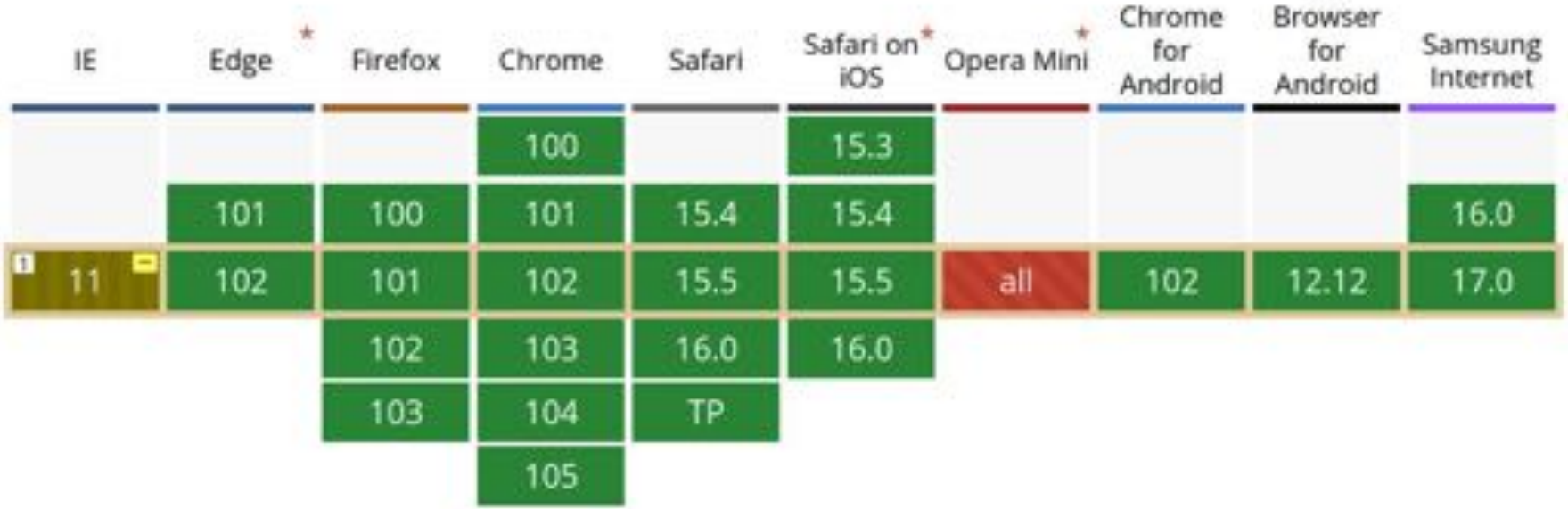
Operations and algorithms

- `generateKeys()`
  - AES, RSA, ECDSA, ...
- `deriveKeys()`
  - ECDH, HKDF, PBKDF2
- `importKeys()`, `exportKeys()`
- `wrapKeys()`, `unwrapKeys()`
- `digest()`
  - SHA-1, SHA-2
- `sign()`, `verify()`
  - RSA, ECDSA, HMAC
- `encrypt()`, `decrypt()`
  - RSA, AES
- `getRandomValues()`

inovex

Web Cryptography API

- W3C Recommendation since January 2017
- today, >95% global usage possibility



IE	Edge	Firefox	Chrome	Safari	Safari on iOS	Opera Mini	Chrome for Android	Browser for Android	Samsung Internet
			100		15.3				
	101	100	101	15.4	15.4				16.0
11	102	101	102	15.5	15.5	all	102	12.12	17.0
		102	103	16.0	16.0				
		103	104	TP					
			105						

caniuse.com, 21.06.2022

Example

```
iv = window.crypto.getRandomValues(new Uint8Array(16));
encryptedMessage = window.crypto.subtle.encrypt(
    {
        name: "AES-CBC",
        iv
    },
    key,
    encodedMessage
);
```

Takeaways Web Crypto API

- easy, standardized access to cryptographic operations
- performant calculations
- context-defined access to the key storage

- still: requirement of cryptographic knowledge
- even greater need to thoroughly secure webapp

CLIENT-SIDE REPORTING

inovex

Logging must not be limited to your server

- diversity of browsers, systems, devices
- errors and crashes will happen in the wild
- extensive clientside logging is hard

→ How to obtain information about browser errors?

```
Content-Security-Policy:
    default-src 'self';
    img-src *;
    script-src my.analytics.com;
```

→ What happens if some subpage wants to load a video from YouTube?

```
Content-Security-Policy:
    default-src 'self';
    img-src *;
    script-src my.analytics.com;
    report-uri https://example.com/reports;
```

```
Content-Security-Policy:
    default-src 'self';
    img-src *;
    script-src my.analytics.com;
    report-to csp-endpoint;

Reporting-Endpoints:
    csp-endpoint="https://example.com/reports"
```

Scope of the Reporting API

The API allows to get reports about

- CSP violations
- Deprecation reports
- CORS errors
- Crash reports
- Network Error Logging (NEL)
- Permission Policy violations

The Reporting API defines two interfaces

Reporting-Endpoints

- HTTP-Header
- allows to define endpoints for reporting
- reports are delivered out-of-band by POST-request of type

  `application/reports+json`

ReportingObserver

- JavaScript API
- allows capturing and handling of reporting events in the client code

```
const observer = new ReportingObserver(
  (reports, observer) => { ... },
  {types: ['deprecation']}
);
```

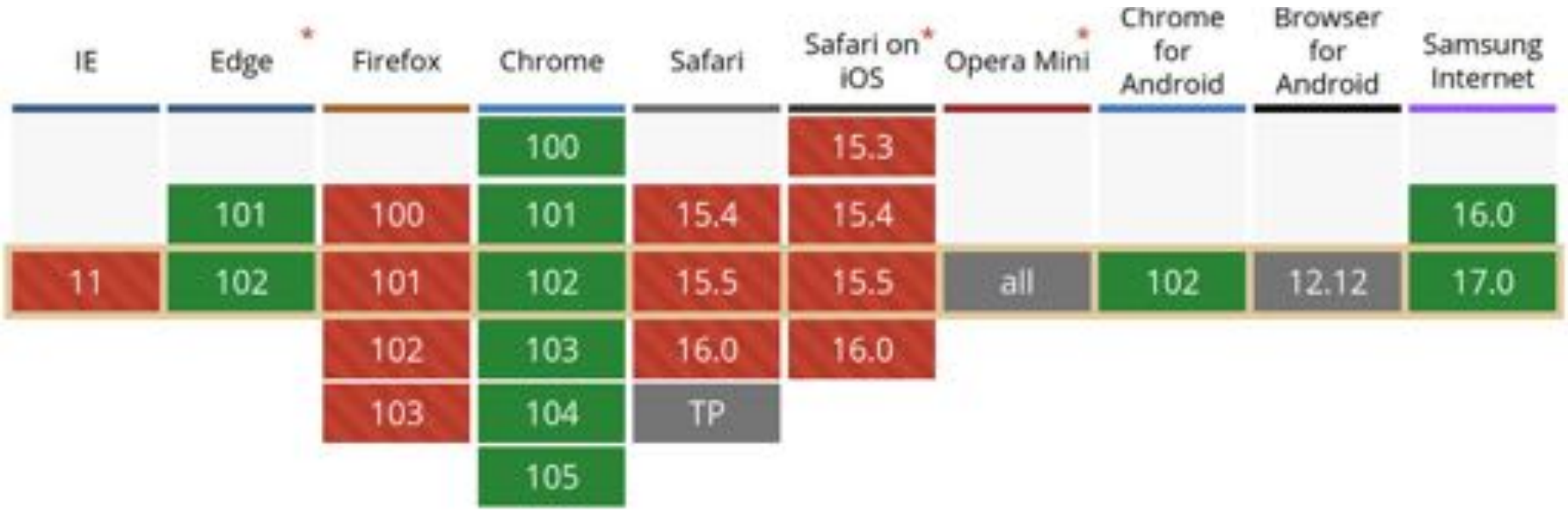
inovex

Receiving reports

- own implementations
- *Sentry*
  - support for CSP, Expect-CT, HSTS reports
- *report-uri.io*
  - wide support for different report types
- only for CSP violations: services like *CSPer*

Reporting API

- Working Draft, latest version April 2022
- today, >70% global usage possibility already



IE	Edge *	Firefox	Chrome	Safari	Safari on iOS *	Opera Mini *	Chrome for Android	Browser for Android	Samsung Internet
			100		15.3				
	101	100	101	15.4	15.4				16.0
11	102	101	102	15.5	15.5	all	102	12.12	17.0
		102	103	16.0	16.0				
		103	104	TP					
			105						

caniuse.com, 21.06.2022

inovex

Takeaways Reporting API

- Reporting API is not standardized yet, still work in progress
- mostly driven and implemented by Google Chrome
- potentially helpful insights into the deployed application

Implementing new features for the client

STRONG AUTHENTICATION

E2E-ENCRYPTION

CLIENT-SIDE REPORTING

inovex

Takeaways

Trend towards standardization
allow usage of browser features

Need to handle the small
differences in the implementations

Obligation to thoroughly analyze
your threat model for the client

inovex

Further resources

WebAuthn

- Specifications: Level 1, Level 2, latest draft
- Simple demo, extensive demo

Web Crypto API

- Specification: latest
- Documentation, Proof-of-concept app

Reporting API

- Specification: Working Draft;
- Demo 1, Demo 2, blog article

Vielen Dank

Clemens Hübner

@ClemensHuebner

chuebner@inovex.de

@inovexgmbh

@inovexlife

inovex