Master thesis : FIDOLOGY : A Measurement Study of FIDO2 Adoption Across the Web

Auteur : Krumm, Macha
Promoteur(s) : Donnet, Benoît
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en informatique, à finalité spécialisée en "computer systems security"
Année académique : 2025-2026
URI/URL : https://gitlab.uliege.be/Macha.Krumm/tfe_krumm; http://hdl.handle.net/2268.2/26194

University of Liège
Faculty of Applied Sciences

Master's thesis carried out to obtain the degree of

Master of Science in Computer Science and Engineering

FIDOLOGY:

A Measurement Study of FIDO2 Adoption Across the Web

---

submitted by

Macha Krumm

Supervisor : Benoit Donnet

Academic year : 2025-2026

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Professor Benoit Donnet, for his continuous guidance and support throughout this project. I am thankful for the opportunity to work on this topic, which allowed me to gain valuable insight into computer science research and the challenges associated with it.

I also wish to warmly thank Florian Dekinder for his consistent feedback and valuable input during the development and evaluation of this work.

Finally, I am deeply grateful for my family and friends, particularly my parents, Tom Guidi and his parents, for their support and encouragement during this work. A huge thanks to my school friends, Amandine, Fanny, Lola, Sophia and Victoria, without whom these five years would not have been the same.

ABSTRACT

While passwords are outdated and need to be replaced, a new successor is emerging: FIDO2. Developed by the FIDO Alliance and the W3C, this standard enables passwordless user authentication using security tokens and combines the Web Authentication specification (WebAuthn) with the new Client-to-Authenticator Protocol (CTAP2) protocol. Since its launch in 2018, websites have gradually begun to adopt it, but to date, the real state of adoption of FIDO2 in the wild is not known.

This work aims to fill this gap by creating FIDOLOGY, a tool that categorizes the authentication mechanisms implemented by websites into five categories: `None/Error`, `Password-Based`, `Password-Extended`, `Fido2-Native` and `Unknown`, by implementing, in a non-intrusive and deterministic way, an automated pipeline using headless browser and multi-layered signals inference. First applied to a manually crafted groundtruth, FIDOLOGY revealed to be capable of correctly categorizing the authentication methods in more than half of the cases, particularly for the `Password-Based` and `Fido2-Native` categories. Evaluated then against a much larger dataset, this tool has shown that classical password-based mechanisms continue to dominate the world of authentication, whilst FIDO2 clues are detected on a small but non-negligible proportion of websites, resulting in a minority of fully enabled passwordless authentication processes.

In addition to this state of FIDO2 deployment, authentication flows are captured to analyse cryptographic challenges and other metrics, with the aim to evaluate the security level of websites previously identified as belonging to the `Fido2-Native` category. While most of them meet, and and in some cases even exceed, the expected FIDO requirements, critical issues like challenge reuse across sessions are discovered, revealing weaknesses in the implementations and leaving websites vulnerable to replay attacks.

Keywords: FIDOLOGY, FIDO2, password, authentication, signals, classification, challenge

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Context

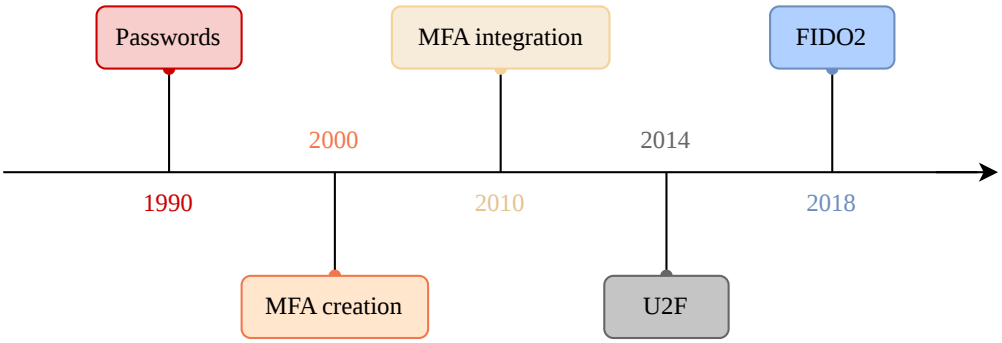


Figure 1.1: Timeline representing the major shifts in terms of authentication mechanisms (taken and adapted from [92]).

As the internet went mainstream in the 1990s, passwords were the dominant form of authentication [92]. Yet even by the early 2000s, it was a well-established fact that passwords and password policies were insufficient in practice [90]. Although these policies can help users create stronger and less predictable passwords [86], they remain highly predictable and are reused far too often [91, 24]. They can be guessed based on one's desktop contents or personality [90]. Furthermore, people in high-profile roles, such as bankers, still use weak passwords [91]. NordPass has compiled a list of the 200 most commonly used passwords, the most common of which appears to have been used at least 21 million times [67].

Besides human factors, there are also server-side associated risks. Vulnerabilities can arise at the server level when security measures are not properly implemented. Santos et al. [83] analysed more than 600 such vulnerabilities and made recommendations to address these security issues. Data breaches, which are becoming increasingly frequent,

[60], also pose a serious threat [12, 55]. In addition to exposing personal information to risk [25], the reuse of passwords also has a significant impact in terms of security [74, 24]. Hackers can gain access to multiple user accounts by obtaining a single leaked personal password. Network attacks [70] and the use of unsecured network communications enabling successful sniffing attacks [95] also lead to theft of credentials.

The list of risks and damages associated with passwords is almost endless. This is why users have turned to multi-factor authentication (MFA). As seen in Figure 1.1, these methods emerged in the early 2000s and became widespread in the 2010s. MFA enhances security by adding an extra layer of security to traditional authentication systems that rely solely on a password, and can offer foolproof protection against various attacks, such as replay attacks [79]. Numerous two-factor authentication (2FA) methods have emerged, the most common being SMS, time-based one-time passwords (TOTP) and push notifications [80]. However, it has been demonstrated that MFA has vulnerabilities and can put users' credentials at risk [88, 8]. Even if security requirements are high, attackers have imagination and develop simple attacks, like relying on social engineering to trick users into revealing their authentication codes [31].

To address these issues relating to passwords and multi-factor authentication, in 2018, the FIDO Alliance and the World Wide Web Consortium (W3C) have created a new standard: Fast IDentity Online 2 (FIDO2). This system builds on the development of Universal 2nd Factor (U2F), which was officially launched in 2014. More specifically, FIDO2 supports passwordless authentication using authentication devices such as security keys. Of all the FIDO standards, FIDO2 is the most widely used [5] and it must meet strict security requirements [4]. This new technology shows promise [53, 52] in the field of passwordless authentication and is the subject of numerous studies aimed at identifying and overcoming the challenges associated with its widespread adoption [49, 56, 113, 106].

1.2 Objectives

The first objective is to be able to categorize the authentication mechanisms found in the wild web. A particular attention should be given to the category of websites revealing FIDO2 clues, in order to classify them in several sub-classes to later focus on them. Categorized websites should allow to assess the global deployment of the passwordless authentication standard FIDO2. Password-based and password-extended methods are also examined to some extent. This will enable a comparison of deployment levels and an accurate assessment of the use of FIDO2.

The second objective is to ensure that the classification process is executed in an ethical way. Signals should be harvested in a non-intrusive manner on websites and combined via a set of heuristics and inference rules. No accounts should be created along the process and interactions with the targeted websites should be kept to a minimum.

The last objective is to provide a preliminary analysis of the security level of websites using FIDO2 as a passwordless authentication method. Cryptographic challenges are a central security requirement in this standard. If they are not correctly generated or handled, it can expose the websites to attacks such as replay attacks, regardless of the strength of the other specifications. Therefore, the focus should be on these cryptographic challenges exchanged between servers and users' browsers during authentication flows. It should suggest whether the FIDO2 implementations are compliant with the specifications.

1.3 Contributions

To fulfil these objectives, this work first contributes to introducing a set of signals necessary to identify and categorize the various authentication methods available. It distinguishes between 5 categories: *(i)* `None/Error`, for websites where no clues could be observed or an analysis error happened, *(ii)* `Password-Based`, for classical authentication flows relying on password, *(iii)* `Password-Extended`, for websites using a second factor in addition to a password, *(iv)* `Fido2-Native`, where websites rely, in some form, on the passwordless authentication of interest and *(v)* `Unknown`, for websites having contradictory or mixed signals that do not allow correct categorization.

Secondly, this work introduces FIDOLOGY, a portmanteau combining 'FIDO' and 'logy' (e.g., the study of ...), which is an automated web scraping tool used to categorize authentication mechanisms. FIDOLOGY uses a headless browser to scrape websites and harvest the set of signals in a non-intrusive and deterministic way. The implementation of heuristics and signals inference allow to categorize the websites into the five previous authentication categories. This whole process is realised with ethical principles, by spending as little time as possible on the targeted websites and keeping interactions to a minimum.

Thirdly, a groundtruth of 1,000 websites was manually crafted, which can be used to evaluate categorization tools like FIDOLOGY. It contains websites from the five categories, with a focus on the `Fido2-Native` category, and each website has its country and IAB category associated to it. This tool is also applied to a larger dataset of 1 million rows, containing websites and their corresponding countries, which allowed to study the adoption level of FIDO2 around the world. Overall, it contributed to showing that the vast majority of websites are still relying solely on passwords, while FIDO2 flows are found in a small portion of the dataset.

Finally, to provide the security level analysis of the FIDO2 implementations, an extension is added that works like FIDOLOGY; using a headless browser, the tool triggers the passkey authentication flow and collects the challenges exchanged between the server and the user's browser. It collects information, compares them against FIDO requirements, computes scores and classifies the website into three levels: strong, moderate and weak, which provides an initial estimation on the FIDO usage of the website.

The code and all datasets used for this thesis are available and documented in this GitLab repository: https://gitlab.uliege.be/Macha.Krumm/tfe_krumm. Furthermore, this work allowed to write and submit an article to the ACM Conference on Computer and Communications Security (CSS) 2026. The Appendix A contains a copy of this article.

1.4 Outline

The rest of this thesis is organized as follows. Chapter 2 positions our work with respect to the state of the art and outlines the challenges one may face when developing such a tool. The technical background on Web authentication related concepts are covered in Chapter 3. The design and implementation principles of FIDOLOGY are described in Chapter 4. Chapter 5 confronts FIDOLOGY with a manually built groundtruth. Chapter 6 applies this scraping tool to a large dataset and discusses results. Further analyses of cryptographic challenges and security level of FIDO2 implementations are presented in Chapter 7. Chapter 8 concludes the report and outlines perspectives for future work.

CHAPTER 2
STATE OF THE ART AND CHALLENGES

This chapter aims to situate our work within the current state of the art by highlighting the differences and the additional contributions with respect to parallel studies. These works can be grouped into three classes: non-technical and human-centred aspects, security and technical aspects, and practical and implementation vulnerabilities. Furthermore, given that a number of challenges were encountered during the development of FIDOLOGY, an analysis is provided to explain these issues and suggest workarounds where appropriate.

2.1 State of the art

The research over authentication protocols and passwordless schemes, such as FIDO2, is quite extensive. First, several studies concerning non technical and human-centred aspects can be put at light. Studies on the usability of passwordless systems, involving real participants, have been carried out by Farke et al. and Sanam et al. [33, 82]. With regard to user behaviour, Huseynov [43] conducted an empirical study on the choice of PIN codes for security keys, concluding that 40% of the deployed security keys had guessable PIN codes. Lassak et al. [56] and Kepkowski et al. [49] identified the main barriers to the adoption of FIDO2 in the corporate environment and formulated recommendations aimed at increasing adoption rate.

The next researches are dealing with the security and technical aspects of FIDO2. Barbosa et al. [7] carried out a provable security analysis of WebAuthn and CTAP2, which was at that moment freshly emerging, followed by a more recent analysis made by Bindel et al. [10] on the newest versions of the two protocols. A similar publication wrote by Guan et al. [40] provides a formal analysis of the FIDO2 protocols and showed that these protocols failed to achieve some strong authentication properties. Kuchhal et al. [54] evaluated the security posture of real-world FIDO2 deployments across the Tranco Top 1K websites. They aim to synthetize the mitigations and improve practical security for that authentication scheme.

Other works investigate practical attacks and implementation-level vulnerabilities. Mahdad et al. [61] analysed the security of FIDO2 hardware keys and their demonstration exposes FIDO2 authentication vulnerabilities in real-world 2FA and passwordless setups, where OS-level security mitigates traditional concurrent attacks. Yadav and Seamons [107] demonstrated the feasibility of seven attacks that exploit four flaws. They build malicious browser extensions and demonstrated the attacks on ten popular web servers that have FIDO2 implemented.

More recently, during the development and writing of this work, Bhardwaj et al. [9] introduced `Fidentikit`. In their study, the authors analysed the Top 100K Tranco websites using a browser-based crawler, `Fidentikit`, which, similarly to FIDOLOGY, relies on heuristics and candidate signals to detect authentication mechanisms. A key difference lies in their analyses of authentication libraries embedded in loaded scripts. However, their dataset is limited to 100K websites, whereas the Tranco list contains up to 1 million domains. This restriction may introduce bias, particularly in ranking distribution and geographical representation.

To the best of our knowledge, only the last highlighted study has conducted a relatively large-scale analysis of authentication mechanisms. However, today, there exists 200 millions active websites on the Internet [23], revealing a huge gab between the Top 100K Tranco websites and the broader web ecosystem. In this context, this work aims to contribute to a more comprehensive assessment of the real-world adoption of FIDO2 by analysing 1 million websites. Furthermore, this work brings a new way to evaluate the quality of deployed FIDO mechanisms. For websites implementing passwordless authentication with FIDO2, it provides an initial analysis of the cryptographic challenges exchanged during authentication flows, offering a first analysis of security level. Finally, it investigates some other requirements that the authentication flows should meet.

2.2 Design and implementation challenges

Each encountered challenge is discussed from the point of view of FIDOLOGY and how they can be surpassed, when possible. It begins by addressing the heterogeneity of authentication mechanisms. It then explains the challenge posed by the dynamic nature of login surfaces. It further describes the challenges associated with delegation, obfuscation, ambiguity and reproducibility. Finally, the ethical considerations are outlined.

2.2.1 Heterogeneity

When a website needs an authentication mechanism, it can make it's own or choose between the standard ones. Currently, as standard authentication mechanism, the followings can be identified: password, one-time password (OTP), single-sign on (SSO), FIDO2 and FedCM. Some combinations of these mechanisms can also be located. For example, it can happen that a website uses password and FIDO2 one after another.

In addition, according to the context, the website could expose a different authentication procedure. As instance, if a site is visited on a smartphone, the FIDO2 mechanism could be preferred, since the smartphone can be used as a passkey. On the contrary, if it is visited with a computer, the website could expose the classical password approach in order to authenticate the user. It may also be that the second factor is only observable after entering a password, making it difficult to directly observe the use of the OTP or FIDO2 (when used as second factor).

This heterogeneity makes it difficult to identify the authentication mechanism used. Therefore, with FIDOLOGY, instead of trying to identify particular implementations precisely, the mechanism are placed into authentication categories. Independent signals are used to cover a wide range of authentication mechanisms and to classify them as accurately as possible. However, given that it is possible to find unique and correct but uncommon login procedures on the web, FIDOLOGY cannot guarantee the completeness of the authentication mechanisms it is able to identify.

2.2.2 Dynamicity

Each website can choose the authentication surface it uses. Some more traditional websites are using standard login page. Others more recent websites shield more the mechanism: it can be behind a modal, a pop-up or in an iframe. And, in particular, there are many ways to access the surface. It can be in one step, with a button that redirects to a login page or opens a modal/pop-up. Or, it can be in multiple steps, with a button that opens a drop-down or redirect to a page with other login options.

In some cases, the website also requires interactions with the user before exposing the surface. There are also cases where the authentication mechanism is hidden in a closed shadow DOM (see subsection 3.1). The DOM is then impossible to examine, which obfuscate the static analysis.

In order to discover most authentication surfaces, FIDOLOGY relies on active and non-intrusive heuristics. It tries to navigate to the login surface, force to expose the UI or to stabilize it. Yet, some interfaces cannot be located and analysed, particularly when they require human interaction or when the authentication surface is too concealed.

2.2.3 Delegation

websites can prevent browser to access the authentication implementation by delegating it to iframes. These iframes can also be cross-origin. These are needed to securely bypass the same-origin policy that is enforced by most modern browsers. In FIDOLOGY, these situations are carefully handed and categorized. When a cross-origin iframe is detected, it is flagged as `unknown_cross_origin` or `none`. It allows to avoid websites categorized in the `None/Error` category despite the effort. It may prevent FIDOLOGY to observe the needed signals to categorize the authentication mechanism used.

2.2.4 Obfuscation

To avoid scrapers and automated analysis, modern websites depend on CAPTCHA, Turnstile, fingerprinting or conditional loading. Some implements there own, such as having to press a button for several seconds, which is not possible for a scraper. In order to produce only meaningful results, FIDOLOGY tries to detect obfuscations and categorize them. When it detects one of these obstacles, the analysis of the website is filtered out.

2.2.5 Ambiguity

Various signals can indicate an active implementation, e.g., detecting a password input on the authentication surface. While others can also point out inactive functionalities, e.g., storage signals. These signals can help detect the usage of FIDO2 as a latent support and not necessarily an effective use.

To face such a challenge, FIDOLOGY adopts a probabilistic approach. It uses a weight for each signal, these are then associated to a confidence score and a diagnostic. It allows to prioritize inference traceability in order to categorize the websites in larger categories, instead of using a potentially erroneous binary classification.

2.2.6 Reproducibility

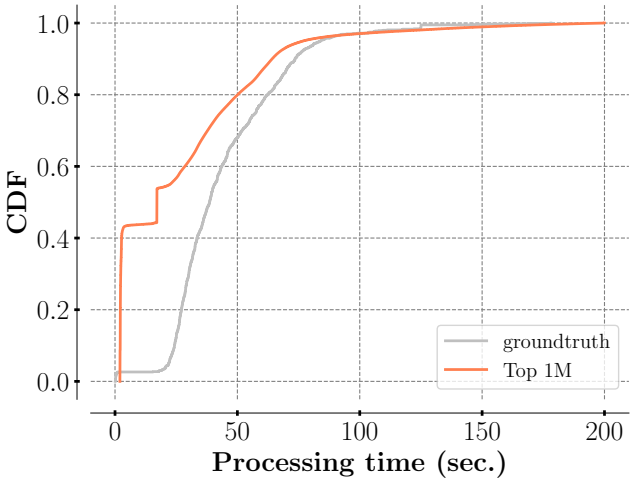
Analyses that are based on dynamic rendering, timing or behavioural heuristics are often non-deterministic. The conditions change each execution. Some elements, like the network connectivity and responsiveness, cannot be controlled. Therefore, non-determinism is always present. However, FIDOLOGY does nothing randomly. It is designed to be deterministic within a constant scope. Between scrapings, the conditions do not change. The same websites are analysed, with the same browser and the same rules. The subsection 5.6 evaluates the level of determinism of FIDOLOGY and attempts to identify the sources of randomness.

2.2.7 Ethics

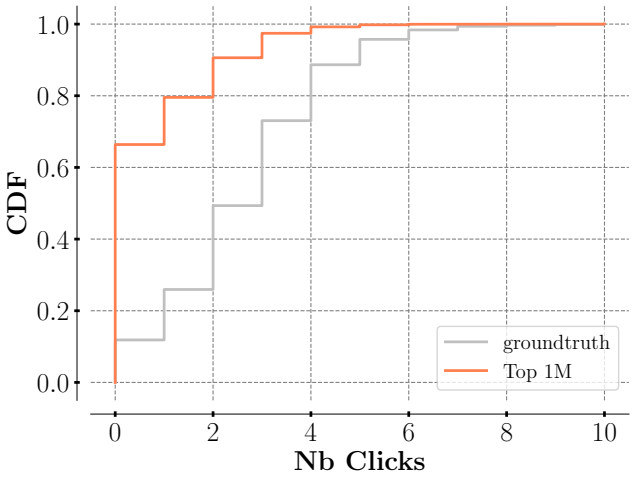
Any real automatic authentication attempt (e.g., form submission, account creation or OTP) would introduce bias in the study, legal risks, and non-reproducibility of results. As such, any automated tool for identifying authentication mechanism must respect a strict ethics in its behaviour [16, 48].

FIDOLOGY is implemented using this mindset. It does not perform any binding action, tries to limit the number of clicks and spends only a limited amount of time on each website. These ethical aspects can be seen in Figure 2.1. It shows how the tool is interacting with the websites. For the processing time, it is constrained to maximum 200 seconds per website. The websites that are taking too much time are often these were FIDOLOGY does not find anything relevant, not even the login page. This time is also required because because some websites have deep Shadow root elements (sometimes

more than 2,000) that need to be investigated. This behaviour suggests a safe and ethical analysis. However, it implies that some mechanisms cannot be observed directly. As said previously, when a certain mechanism is triggered only after entering a password, it is impossible to analyse it in an ethical way. Section 6.5 extends the ethical considerations based on the observed results.



(a) CDF of processing time in seconds for scraping results of the groundtruth and the 1M dataset.



(b) CDF of number of clicks for scraping results of the groundtruth and the 1M dataset.

Figure 2.1: Ethics analysis for the scraping raw results of the groundtruth ($N = 993$) and the 1M dataset ($N = 981,201$).

CHAPTER 3
BACKGROUND

This chapter covers all the technical background on web authentication mechanisms. It focuses on FIDO2 by explaining the three entities and four main properties. It then carries out an explanation of the two protocols composing this standard, which are WebAuthn and CTAP. At the end, it illustrates the difference between U2F and the passwordless authentication scheme of interest and briefly explains FedCM.

3.1 Technical prerequisites

3.1.1 DOM

The Document Object Model (DOM) is a cross-platform and language-independent API that treats an HTML or XML document as a tree structure in which each node is an object representing a part of the document. Each object present in the nodes can be manipulated and modified via programming languages.

If an element needs to be added but concealed, one can use the shadow DOM. It works by allowing to attach a hidden and separate DOM to an element. A shadow DOM can have additional shadow Hosts nested under it. The structure is then called a shadow Tree. This hidden DOM is frequently used in web interfaces [104].

The main objective of shadow DOM is to allow encapsulation. The styles and the scripts inside it are scoped only to that node, and thus will never interfere with the rest of the page. It helps to test individually all parts of a website, without having to deploy the rest of the application. But there can be some potential security concerns with shadow DOM. There could be leakage of information between it and the shadow Host. Some manipulations from the shadow Host, cross-component attacks and exposure to external scripts can also happen. Strategies are developed to mitigate these risks, like sanitizing the content or API security solutions.

However, the most interesting aspect for this work is the closed shadow root. A shadow root is the root element of the shadow DOM tree. Figure 3.1 illustrates the shadow DOM concept. It is associated with a standard DOM element and can be either open or closed. An open root can be accessed and manipulated from outside JavaScript, unlike a closed shadow root. When websites use closed shadow roots, this prevents external scripts from accessing or manipulating shadow DOM components [94].

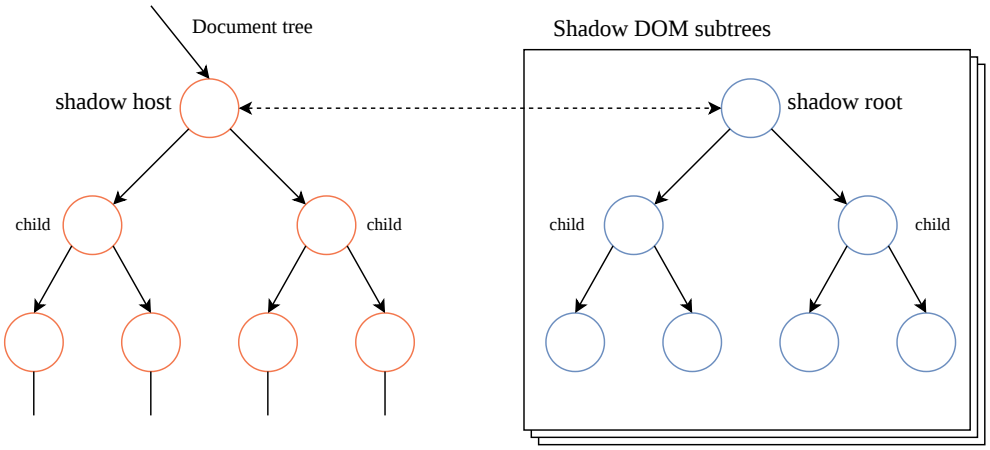


Figure 3.1: Shadow DOM that provides encapsulation for DOM within a component (taken and adapted from [59]).

3.1.2 Web storage

Web applications also maintain client-side state using web storage mechanisms. Storage objects stay intact through page loads and are simple key-value stores. Two facilities are within the web storage. The first is called `sessionStorage` and maintains a separate storage area for each given origin. That area is available for the whole duration of the page session as long as the browser is open. The second is `localStorage`, which is similar to `sessionStorage`, but in this case it persists even when the browser is closed and reopened [62].

With web storage, large amounts of data can be stored locally, without affecting website performance and in a secured way[99]. While it is preferable to not store sensitive data in those areas because of XSS attacks, web developers may use these two facilities in order to cache information relevant to authentication, like sessions or JSON Web Tokens. Storing tokens in `localStorage` is an quick and easy solution that are lot of developers use. As seen in the application window with the highlighted line in Figure 3.2, the Session Storage table could contain a field named 'loginVerify' with the value 'passkey'. This is a clue that the website Binance has passkey as authentication method. Therefore, inspecting these areas is relevant in the context of this study [57, 51].

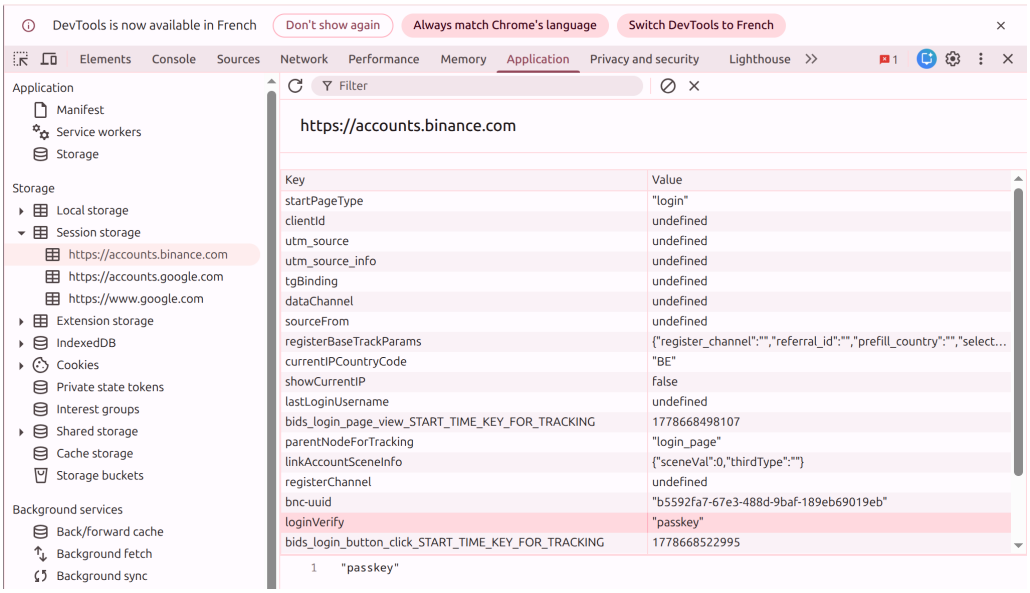


Figure 3.2: Highlighted login field with passkey value in the session storage from the application window seen on the Binance website.

3.1.3 Cookies

In order to retain information across user visits, a website can store in the user's browser small text files called cookies. They are commonly used for authentication in order to identify the user on subsequent requests. Those cookies are accessible by both the client and the server, making them essential for authentication and session management. Thus, inspecting the cookies in the context of identifying authentication mechanisms can help finding relevant clues [38, 85].

3.2 FIDO2

Fast Identity Online 2 (FIDO2) is a passwordless authentication mechanism that aims at strengthening the way people log in online services. This open standard is developed by the FIDO Alliance and W3C. The Fido Alliance is an open industry association focused on reducing the world's reliance on passwords. It includes Microsoft and other technology, business, and government organizations. In 2014, this group released the first version of FIDO that implemented phishing resistant multi-factor authentication. The Word Wide Web Consortium (W3C) develops standards and guidelines to help everyone build a web based on the principles of accessibility, internationalization, privacy and security [100].

To give an overview about FIDO2, this authentication standard is based on cryptographic key pairs. It is used between a client, an authenticator and a relying party. It introduces 2 protocols: WebAuthn and CTAP2. WebAuthn is an API that allows the browsers and servers to communicate with FIDO2 authenticators. Client To Authenticator Protocol 2 (CTAP2) defines how authenticators talk to the client's devices. All these terms and protocols are defined in the next sections.

3.2.1 Three main entities

As said in the introduction above, FIDO2 is used between three main entities. Before the device can generate a unique set of FIDO2 keys, the user needs to prove that he is not an unauthorized user or a type of malware. This can be done via the first entity, i.e., the authenticator. It is a device capable of accepting a PIN, biometric data or some other gesture from the user. There exists two types of such devices.

The first type is roaming authenticators. These are portable hardware devices that are separated from the users' client devices. It includes security keys or other devices that can connect the clients via USB, NFC or Bluetooth wireless technology. The second type is the platform authenticators. These devices are integrated into the users' client devices. It can be a laptop, a smartphone, or a tablet. It allows to use biometric capabilities such as Apple Touch ID or Face ID [109].

The second entity is simply the relying party (RP). It is the online service on which the user tries to authenticate. For example, it can be Google, Facebook, Amazon and many more. Those websites implement FIDO2 in their servers and allow the users to connect thanks to this passwordless authentication mechanism.

The last entity is the client. The client is composed of the user and the user's agent. The user is the person that tries to authenticate to a given relying party and his agent can be its web browser. In Figure 3.3 can be seen a scheme composed of the three main entities of FIDO2. There are the two types of authenticator, the relying party and the user's agent, also named the platform [6].

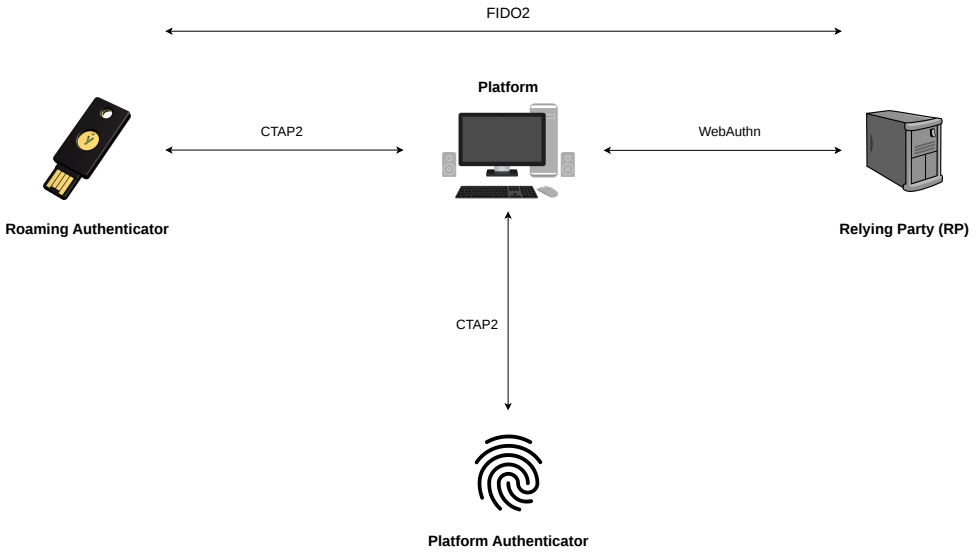


Figure 3.3: The three main entities of the FIDO2 protocol: platform (user's browser), relying party (RP) and the two types of authenticators (roaming and platform) (taken and adapted from [6]).

3.2.2 Main properties

Passwordless authentication with FIDO2 can expose four main properties. The first is the use of public-key cryptography. For each website, a private-public key pair is generated. The most frequent algorithm used are elliptic curve algorithms with P-256 and SHA-256. The private key is either stored on a roaming authenticator, where the key is directly stored on it, either on a platform authenticator, which uses a Trusted Platform Module (TPM). It's a hardware chip attached to the device and it itself uses cryptography to securely store critical information. This private key, once stored in any type of authenticator, cannot be moved between devices. It allows to keep the authentication tied to a user and a specific device. The public key is stored in the server of the website.

Secondly, it mitigates the risk of phishing attacks. Since the private key is bound to both a specific device and an origin, the relying party sending the challenge can be easily verified. Authenticators will only respond to challenges originating from the domain they were registered with. It is therefore not possible to create a fake website and trying to authenticate a user on it using by using the credentials created with the original website.

Thirdly, this authentication mechanism allows to perform user-verification. This is done before using the private key, in order to see if the device is stolen or not. It oblige the correct human to be physically present. The verification is made via PIN or biometric capabilities like face recognition or finger print analysis.

Finally, this reduces the need to store sensitive data on servers, thereby minimising the risk of data breaches and enhancing user security. As there is nothing to be stolen, nothing can be stolen. The important data is stored securely on the authenticators, and can in no way be retrieved. It also removes the danger link to sending OTP over the network to perform multi-factor authentication.

3.2.3 COSE algorithms

FIDO2 uses the CBOR Object Signing and Encryption (COSE) standard to define its cryptographic algorithms. The allowed algorithms are defined in the `pubKeyCredParams` section (see Section 3.3). During registration, the authenticator and the relying party need to agree on an algorithm from the COSE Algorithms registry. The RP will further narrow down the supported algorithms, depending on which are supported on the FIDO2 server [84].

3.3 WebAuthn

WebAuthn, short for Web Authentication, is a browser-based API that allows the creation and usage of authentication credentials that are based on public-key cryptography. In FIDO2, it is the protocol that allows to enable passwordless authentication. As seen in Figure 3.3, it handles the data flow between the client's browser and the relying party.

The API has two main use cases, which are registration and authentication of users, and are explained below [68, 78]. The concept of user verification is first clarified, as it is used in both flows.

3.3.1 User verification

User verification is an optional security measure designed to ensure that it is indeed the person in question who was present during authentication, and not just anyone. When `userVerification` is set to `required`, the RP have to perform it and the authentication fails if the verification is not made by the client. If the UV flag is to `preferred`, which is the default value, the verification is optional thus the process does not fail if not performed. And if it is `discouraged`, the RP does not not want verification. It allows to minimize the operations and disruption of user interaction flow [50].

3.3.2 Registration

Registration makes the authenticator create a new set of public key credentials that can be used to sign a challenge generated by the relying party. The public part of these new credentials, along with the signed challenge, can be sent back to the relying party for storage. The RP can later use these credentials to verify the identity of a user whenever required. This can be done via the registration API call, which is defined as follows:

```
const promise = navigator.credentials.create({
    publicKey: creationOptions
});
```

The `createOptions` object is a dictionary that helps handling the public key signature scheme to use, the location and the type of the new credential, and the metadata stored with the credential, like the user and RP ID. During the registration process, several steps are executed and provided in Figure 3.4.

The client initiates the registration by doing a request for it. When the relying party receives the demand, it creates a `PublicKeyCredentialCreateOptions` instance containing the challenge, the information about the user and the RP (like their IDs). This object is returned to the client. The client's browser calls the `navigator.credentials.create()` function and validate the RP ID against the origin. Then, with the help of the authenticator, the user verification is done. If the user is the one awaited, a new key pair is created by the authenticator and an attestation is signed. An object containing the public key, the credential ID and the attestation data is returned to the browser. The promise made by the `navigator.credentials.create()` function is then resolved by the user's agent and a `PublicKeyCredential` object is created. The server validates this by verifying the signature of the challenge and the attestation [110].

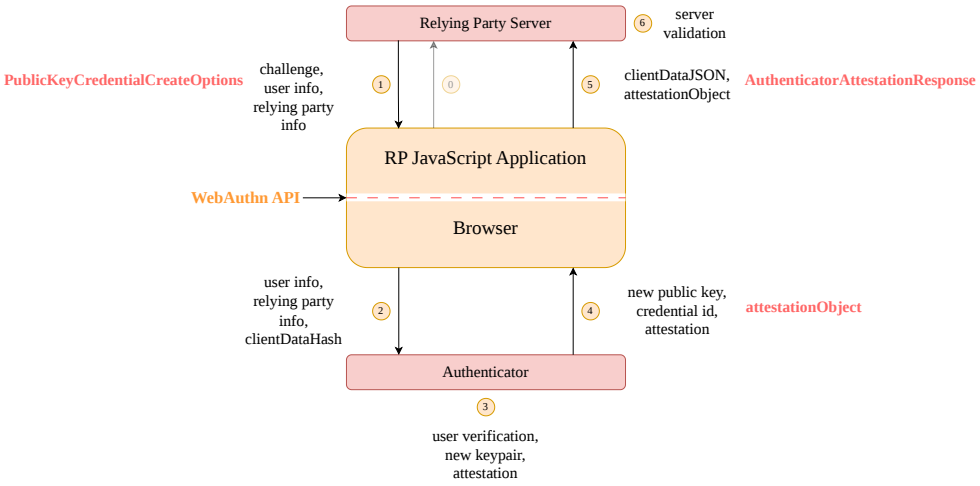


Figure 3.4: WebAuthn registration flow (taken and adapted from [110]).

3.3.3 Authentication

The relying party can also request the user's permission to perform an authentication operation with an existing credential. This can be done via the authentication API call, which is defined as follows:

```
const promise = navigator.credentials.get({
    publicKey: options
});
```

The dictionary contains a challenge (see Section 3.3.4), an RP ID that needs to match values used when the credentials was initially created and an optional `userVerification` field, which value is either `discouraged`, `preferred` or `required` as explained. The authentication flow looks similar to the registration flow. The primary differences are that the authentication does not require user information and the authentication creates an assertion signed by the previously generated private key that is associated with the relying party rather than signing with the attestation certificate. The different steps are illustrated in Figure 3.5 and are explained here after.

As with registration, the client initiates the process by requesting authentication. A `PublicKeyCredentialRequestOptions` instance is build by the RP, containing essentially the challenge, and returned to the client. The user's agent then has to call the `navigator.credentials.get()` function and validate the RP ID against the origin. The authenticator finds a credential matching RP ID and prompts the user to consent to the authentication. If those steps are successful, the authenticator creates a new assertion by signing over the `clientDataHash` with the private key generated for this account during registration. Then, the authenticator returns the assertion signature back to the browser. The browser resolves the Promise to a `PublicKeyCredential` that contains the `AuthenticatorAssertionResponse` which is returned to the RP to finalize the authentication [111, 112].

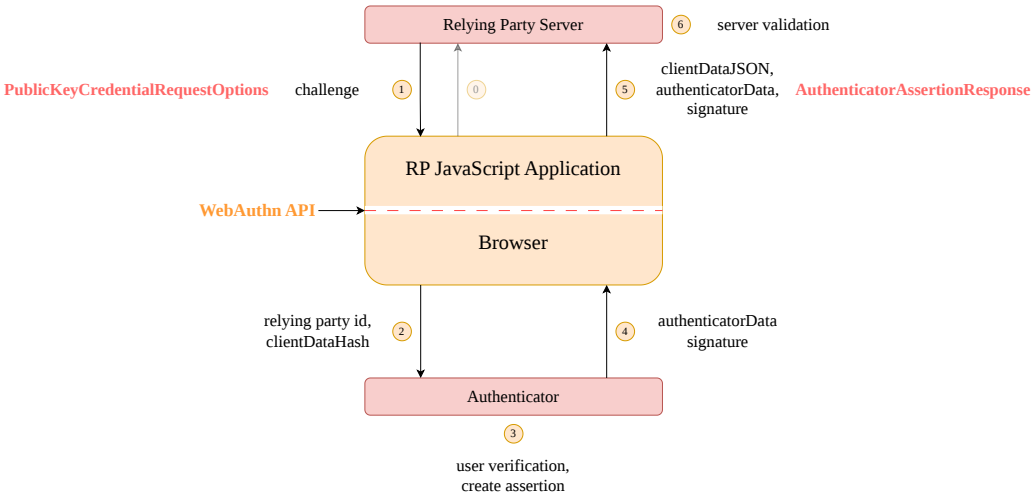


Figure 3.5: WebAuthn authentication Flow (taken and adapted from [111]).

3.3.4 WebAuthn cryptographic challenges

The challenge is a cryptographic nonce used during registration and authentication in order to avoid replay attacks. A challenge needs to be randomly generated by the server in a trusted environment, used only once, be short-lived [44] and it should not be using timestamps [32, 14]. The challenge must contain enough entropy in order to not make it easily guessable. Therefore, to achieve high entropy, the cryptographic challenges used in FIDO2 processes should be at least 16 bytes long [98]. However, satisfying the length requirements is not sufficient to have a strong entropy. For example, a challenge can be 'AAAAAAAAAAAAAAAA', but it has 0 bits of entropy [13]. That is why the random aspect is also important.

Figure 3.6 illustrates the `PublicKeyCredentialRequestOptions` dictionary, which is received from the relying party server when a user triggers the FIDO2 authentication flow on the Binance website. The received cryptographic challenge is highlighted and well above the minimum length of 16 bytes. Other fields like the RP ID, which identifies the Binance domain, and the `userVerification`, set to `preferred`, can also be observed.

3.4 CTAP

CTAP stands for Client To Authenticator Protocol. Its specification describes how an application (e.g., the browser) and OS can establish communications with an authenticator (e.g., a yubikey) over USB, NFC or Bluetooth communications mediums. There are two CTAP protocol versions [108]. The first is the CTAP1 protocol. It is the new name for FIDO U2F, that is given to it after the arrival of FIDO2. It allows to enable a second-factor authentication experience. The second version, the one of interest, is CTAP2. It establishes communication between FIDO2-enabled browsers, OS and authenticators to enable passwordless, second-factor or multi-factor authentication.

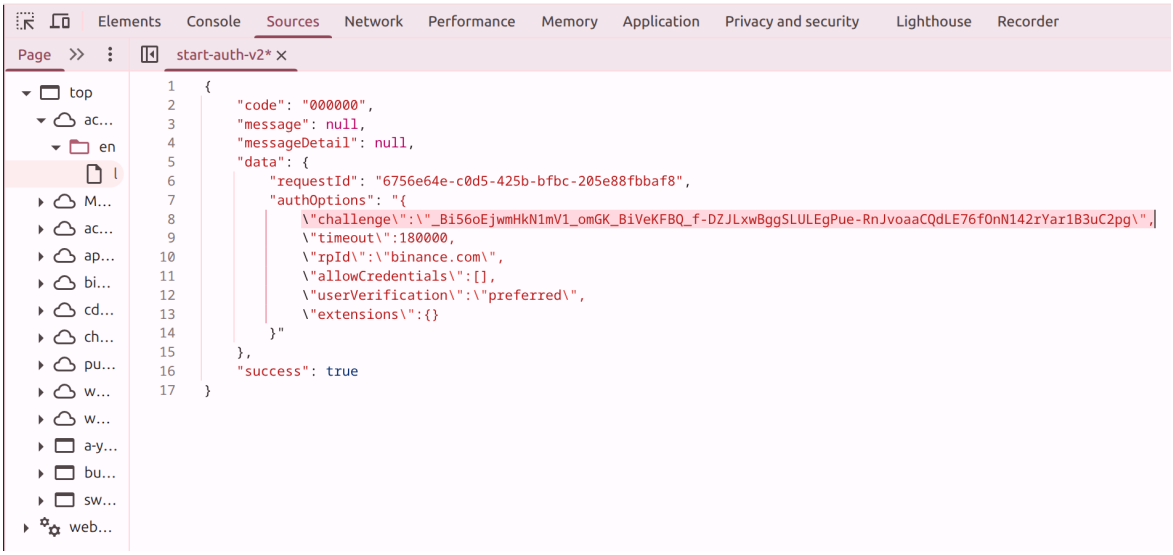


Figure 3.6: Network window of Binance showing the request containing the cryptographic challenge received when triggering the passkey authentication flow.

CTAP2 is the version used in the FIDO2 standard [46]. The general protocol between a platform and an authenticator is as follows. First the platform establishes the connection with the authenticator. Then, it gets information about the authenticator using the `authenticatorGetInfo` command, which helps it determine the capabilities of the authenticator. If the authenticator is capable of supporting a certain operation, the platform sends a command for that operation. At the end, the authenticator replies with response data or an error.

During this protocol, the messages are encoded in the CTAP2 canonical CBOR encoding form. The protocol is specified in 3 parts. First part is the authenticator API. The authenticator operations are similar to an API call, it accepts input parameters and returns an output or an error code. The second is message encoding. Before sending the request to the authenticator, it needs to be encoded. And last part is the transport-specific binding. The requests and responses can be carried over USB, NFC or Bluetooth.

3.4.1 Authenticator API

All operations can be performed independently of others and are asynchronous. There are three main operations. The first operation is `authenticatorMakeCredentials`. It is invoked by the host to request the generation of new credentials in the authenticator. The second is `authenticatorGetAssertion`. That operation is used by a host to request cryptographic proof of user authentication as well as user consent to a given transaction, using a previously generated credential that is bound to the authenticator and relying party identifier. The last operation is `authenticatorGetInfo`. The host can request that the authenticator report a list of all supported protocol versions (FIDO2 or OTP, U2F) and other information, like maximum message size supported.

3.4.2 Message encoding

The encoding of messages is done in Concise Binary Object Representation (CBOR) because the transport can be bandwidth-constrained and serialization formats such as JSON are too heavy-weight. All encoders must serialize CBOR in the canonical form and without duplicate map keys. Thus, decoders must reject non validly encoded messages. In the canonical form, integers are encoded as small as possible, indefinite-length items must be made into definite-length items and map keys must be sorted according to their lengths.

Figure 3.7 illustrates the CBOR encoding from four of the FIDO2 authentication options returned by Binance during the authentication flow. On the left, the JSON format is 178 bytes, while on the right, the canonical CBOR form is 156 bytes long, which is more compact. The canonical CBOR encoding has its keys sorted by length, as indicated by the bytes `64`, `67`, `69`, and `70`, corresponding to key lengths of 4, 7, 9, and 16 bytes respectively. Each group represents a key followed by the value of the field.

```
{
   "challenge":"VvvZIMzJEpQzhiLSQ8
                O1NP2-ZN636H4M1n57
                Dy4DqBygIWuGfF5xKw
                hdg6bmhlx6DZzBlXFD
                gyfNh3NmpnIM3w",
   "timeout": 180000,
   "rpId": "binance.com",
   "userVerification": "preferred"
}
```

```
a4
    64 72704964
    6b 62696e616e63652e636f6d

    67 74696d656f7574
    1a 0002bf20

    69 6368616c6c656e6765
    78 56
       5676765a494d7a4a4570517a68694c53
       51384f314e50322d5a4e36333648344d
       316e353744793444714279674957754 7
       664635784b7768646736626d686c7836
       445a7a426c5846446779664e68334e6d
       706e494d3377

    70 7573657256657269666963617469 6f6e
    69 707265666572726564
```

Figure 3.7: JSON (left) and canonical CBOR (right) representations of four of the FIDO2 authentication options returned by Binance during the authentication flow.

3.4.3 Transport-specific binding

There are three different channels used to carry requests and responses. The first is Universal Serial Bus (USB). Messages are framed for USB transport using the Human Interface Device (HID) protocol. The second is the Near Field Communication (NFC). It is a short range wireless technology that allows devices to exchange data by touching or bringing them close together. And the last one is Bluetooth. Devices are paired and messages are exchanged thanks to the Generic ATTribute profile (GATT) services [11].

3.5 FIDO2 vs U2F

FIDO U2F is a mechanism providing a strong second factor for user login. This older standard was already using security keys, but only after a password. When FIDO2 arrived, U2F has been integrated as CTAP1. Now, it allows to use it, like Yubikeys, as second factor on FIDO2-enabled browsers. As explained above, it is the CTAP2 protocol that implements all the passwordless capabilities [39]. The difference is in the goal. U2F is made to strengthen password-based authentication methods. FIDO2 is made to remove all password and create a passwordless authentication [34].

3.6 Federated Credential Management

Federated Credential Management (FedCM) is a web API and new standard that allows users to sign in to websites using their federated identity providers (IdPs). An IdP could be for example Facebook or Google. The browser itself mediates the login flow. In consequence, there is less cross-site tracking information. Federation is used to avoid using a new password on every site and therefore helps with account creation. The fact that it is transferable across devices offers a real advantage. Thanks to FedCM, websites do not rely on third-party cookies anymore.

Concretely, it works in 5 steps. First the user initiates the sign-in. This can be done by clicking on a 'Sign-in with [IdP]' button on a website. Then, the user's browser uses the FedCM API call to request an identity assertion from the IdP. The third step consist in getting user consent. The browser have to prompt the user to consent it to sharing their identity with the RP. Following, the IdP provides an identity assertion to the website. Finally, the website submits the identity to create an ID and a session [66].

FIDO2 and FedCM can work together in order to reach a federated passwordless authentication. In that case, the first will handle the authentication and the second the federation. For example, on a website, a user can choose to sing-in with Google. The browser will thus use FedCM. The RP in question, here Google, is going to authenticate the user via a passkey. Via FedCM, Google is going to return an identity token [81].

In the context of this report, inspecting FedCM clues can help because it reveals the federated identity provider used by a website. Checking if those identified IdP authenticate users with FIDO2/WebAuthn passkeys can suggest that FIDO2 is part of the login process [97].

CHAPTER 4

FIDOLOGY

This chapter starts with an overview of FIDOLOGY (Section 4.1). The three steps, mapping, building targets and scraping, are then explained in details. The code is available and referenced in Section 1.3. Finally, the chapter shows a small example of how FIDOLOGY classifies a website into its authentication category (Section 4.5).

4.1 Overview

FIDOLOGY is an automated tool, which searches for clues of the different authentication methods, focusing on FIDO indicators. Figure 4.1 illustrates an overview. It is first composed of a mapping step (Section 4.2), which maps some websites to their corresponding country and IAB category. Then, it gets to the building targets step (Section 4.3). This combines the CSV files outputted by the previous step and a list of URLs in order to prepare the data to be scraped. This brings to the last step, the scraping part (Section 4.4). That one uses a CSV file as input and outputs an enriched CSV file containing all the scraped information.

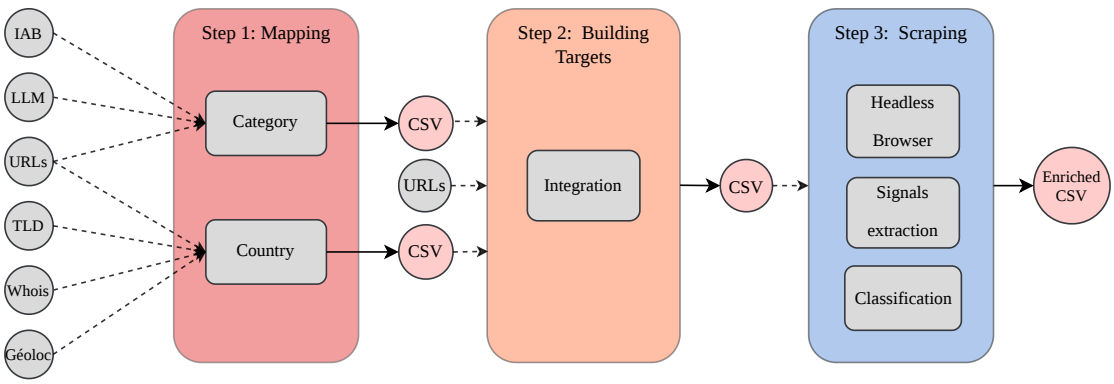


Figure 4.1: FIDOLOGY pipeline overview.

4.1.1 Technology choices

This tool is fully implemented in Python. This was an obvious choice, as it already contains a lot of libraries and functions allowing to do web scraping easily. Another factor that could have an impact on the tool is the headless browser. The choice could be between Selenium or Playwright. However, Playwright seems to be quicker, more suitable to analyse modern websites and offers built-in browser contexts allowing to have a clean state for each scraping test. The Chrome browser from Playwright is an obvious choice, as it is enabled by default and is the most popular browser [77].

4.1.2 Parallelization

The scraping step has been parallelized using the multiprocessing module from Python [28]. It allows to spawn 40 independent FIDOLOGY processes per machine. Each process is launched with their own arguments and part of the CSV input list. Each worker aggregates its results in a file, and at the end, all the CSV result files are concatenated to get the final enriched CSV. A keyboard interrupt detector has been added in order to cleanly exit all the workers.

4.2 Mapping

4.2.1 Category Mapping

For each URL, it is necessary to find the corresponding IAB category. IAB categories are an industry standard and are widely used in services such as advertising, internet security and filtering systems. These categories consist of Tier-1 and Tier-2 categories, with Tier-2 being a sublevel of a parent category [101]. Only Tier-1 categories were found in this study, as their identification was already complex. The following two subsections describe the two methods explored: the first proved unsuccessful, while the second was successful.

The idea was to map domains to an IAB category database. But first, to obtain this database, the 100 GB of Wikidata had to be analysed [103]. This is done using a script that scans all the lines of information of the compressed Wikidata file. It checks the QID associated with the different domains and their properties. Once all the QIDs and properties are collected and placed in a dictionary, a mapping between the Wikidata properties and the IAB categories is performed using a `WIKIDATA_TO_IAB` dictionary that translates one into the other.

After this analysis of the Wikidata database, the output file contained 150 000 lines. The number was not expected to be this low, since the objectives was to scrape at least 500 000 websites. However, to assess the reliability of these categories, a match was made against a list containing 1318 websites that were already categorized [102]. The results are discouraging: only 472 domains were in the IAB category database, and only 23 of

these matched the expected category. In addition, when inspecting the results, it could be seen that almost all the categories were IAB 3 (Business). When manually checking some well known websites, like Netflix or Amazon, the category was clearly off. These results demonstrate that the Wikidata-based approach is insufficiently reliable for large-scale website categorization, and an alternative method was therefore required.

The second method involved querying free LLMs. A script was developed to extract the HTML code from a website and query an LLM to determine its category. To retrieve the HTML code, a simple request is made using the `requests` library. The HTML code is then parsed using the `BeautifulSoup` library. All the text from the various tags is retrieved and truncated to 80,000 tokens if the amount of information is too large. Once this operation is performed, the text is sent to an LLM via the corresponding API. The LLMs used were *gemini-2.5-flash-lite*, *mistral-medium{ ,-2508}*, *mistral-small-{2503,2603}* and *open-mistral-7b* [37, 64, 36, 65, 45, 63]. It was asked to categorize the given website based on the provided text into one of the IAB Categories provided in Table 4.1.

IAB Code	IAB Label	IAB Code	IAB Label
IAB 1	Arts & Entertainment	IAB 14	Society
IAB 2	Automotive	IAB 15	Science
IAB 3	Business	IAB 16	Pets
IAB 4	Careers	IAB 17	Sports
IAB 5	Education	IAB 18	Style & Fashion
IAB 6	Family & Parenting	IAB 19	Technology & Computing
IAB 7	Health & Fitness	IAB 20	Travel
IAB 8	Food & Drink	IAB 21	Real Estate
IAB 9	Hobbies & Interests	IAB 22	Shopping
IAB 10	Home & Garden	IAB 23	Religion & Spirituality
IAB 11	Law, Government & Politics	IAB 24	Uncategorized
IAB 12	News	IAB 25	Non-Standard Content
IAB 13	Personal Finance	IAB 26	Illegal Content

Table 4.1: IAB Label and Code for the 26 IAB Tier-1 categories.

4.2.2 Country Mapping

A matching method to identify countries from websites was needed. An important aspect to consider was the top-level domain extension (TLD). URLs like 'amazon.be' should be identified as 'United States of America' rather than 'Belgium.' Therefore, a preprocessing step was implemented. This step aimed to separate websites appearing in multiple countries from those appearing only once. The first category is called *multi-country* and the second *single-country*. For example, the website 'amazon' falls under the

multi-country label because it can appear multiple times in the dataset, such as: 'amazon.be, amazon.com, amazon.au, amazon.it.' In comparison, 'lesoir.be' appears only once in the dataset and is classified as *single-country*.

Once done, three different strategies were applied to get the corresponding countries. The first was about using the *Whois* protocol. It is a service that allows to analyse a domain and get the country associated to it. The *Whois* Python library [71] is used and the *whois* method allowed to quickly look up the domain. For the second method, a free geolocation API was found. It is the Geolocation-db service and offers country information about domains [75]. The last strategy is to extract the TLD extensions from the URLs and perform a matching between a file containing the TLDs and the corresponding countries [105].

For *multi-country* labelled websites, the strategies are applied in this order: whois, geolocation and TLD. For *single-country* labelled websites, the TLD strategy is applied first, then the two other in the same order as previously. The whole process is parallelized with the Pandarallel library in order to make it more efficient. It still required an important amount of time to get the 1 million countries.

4.3 Building Targets

This step takes as input a mapping between URLs and IAB categories, a mapping between a domain and a country, and a list of URLs. It constructs a FIDOLOGY target by merging the files, and the output file is used during the scraping process.

4.4 Web Scraping

This step takes a CSV file and scraps the web for FIDO2/WebAuthn adoption clues. The identification of such usages relies on a series of signals captured during the automated scraping of each website. These signals are next weighted and aggregated to produce a classification of usage, a confidence score between 0.0 (no confidence) and 1.0 (full confidence) and a diagnostic. The output is a CSV file used for later analysis.

4.4.1 Overview

The global overview of the scraping part of FIDOLOGY can be seen in Figure 4.2. It takes as input a CSV originated from the previous step, containing URLs, countries and possibly an IAB category. The first step starts with an initialisation process. This process checks the URL validity, and if it is valid, continues by creating the context for the Playwright headless browser and a corresponding `Page` object. That object will be updated during the whole website analysis.

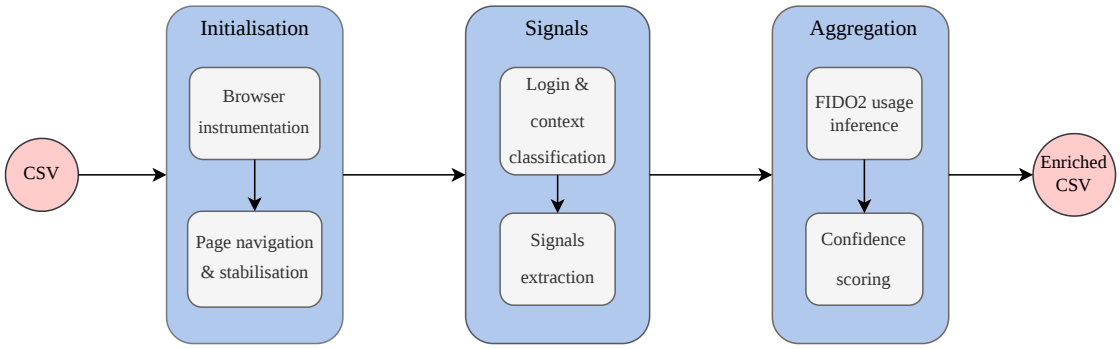


Figure 4.2: Overview of the third step of FIDOLOGY (scraping step): taking a CSV as input, it initialises the browser, does signal scraping, aggregates everything and outputs an enriched CSV with the resulting information.

The browser context is created with a hook injected in order to intercept authentication-related API calls at runtime. This task will monitor signals that cannot be captured while inspecting the DOM or the network layer communication. As instance, it can captures the usage of `navigator.credential.get()`. Following that, it navigates to the website. If the website is unreachable, for network reasons or timeout execution, the analysis of that URL will stop there. If the targeted website is reached, it then proceeds to handle consent banners, like cookies, language and eventual advertisement pop-ups. It includes those that could be hidden inside iframes. Next, it stabilizes the page context. A correct stabilization allows the scripts of the page to be fully loaded and executable.

The second part is signal capture. It starts by trying to locate and reach authentication entry points. These can be login pages or dialogues. It can be achieved by attempting to navigate to the authentication surface. This search is therefore based on standard buttons, common login paths and keywords that are typically used to access login surfaces. It also relies on a heuristic way to detect if whether a page is the targeted authentication surface. Pages are also classified into an anti-bot class if one is detected. This process allows us to determine whether the following analysis is relevant or not. It then proceeds to harvest signals from the DOM, shadow DOM, storage, network requests, and runtime APIs. These signals are explained in detail in Section 4.4.3. The resulting output of this signal capture step is a multi-dimensional representation of the authentication behaviour of the website.

The final step of scraping is the integration of all signals to being able to infer a potential FIDO2 usage. The authentication category is found by a deterministic combination of the collected signals. The websites can be categorized as `Fido2-Native`, `Password-Based`, `Password-Extended`, `Unknown` or `None/Error`. These categories are described in Section 4.4.4. A confidence score is then associated by the means of a weighted heuristic model. Weights reflect the relative strength and reliability of signals. The score reflects the degree of certainty of the inferred category (see Section 4.4.5). For native FIDO2-classified sites, a validation of the usage is attempted (detailed in Section 4.4.2).

That scraping process is, as mentioned in Section 2.2, is limited to a functional scope. It attempts to collect only relevant information, while staying on the same domain as the original website. Cross-origin domains are only accepted when the redirection to a certain authentication surface is linked through a standardized identity protocol, which could be FedCM or OAuth.

4.4.2 Validation of `Fido2-Native` classes

For websites that are categorized as `Fido2-Native`, e.g., `full_fido2` or `webauthn`, a validation is attempted. The authentication surface has a passkey option and the validation process is able to click on it in order to interact with the WebAuthn/passkey flow. In the case of `full_fido2` websites, the validation should be successful. An example showing the passkey button is provided in Section 4.5.

However, this step is limited. In certain cases, like `latent_usage` or `latent_support`, the FIDO2 validation is obviously less effective since the passwordless standard is not completely implemented. It also happens that a user account on a website is required before being offered passkey options, or the option needs to be explicitly enabled it in the parameters.

As an example, Figure 4.3 represents the login page of Facebook, and it can be seen that no passwordless option is available. It needs to be enabled in the parameters after creating an account. Figure 4.3 Therefore, even though the validation process is robust, it is not always productive. In addition to FIDO2 usage, when the validation is successful, FIDOLOGY is theoretically able to record the COSE algorithm (see Section 3.2.3) accepted during the authentication flow.

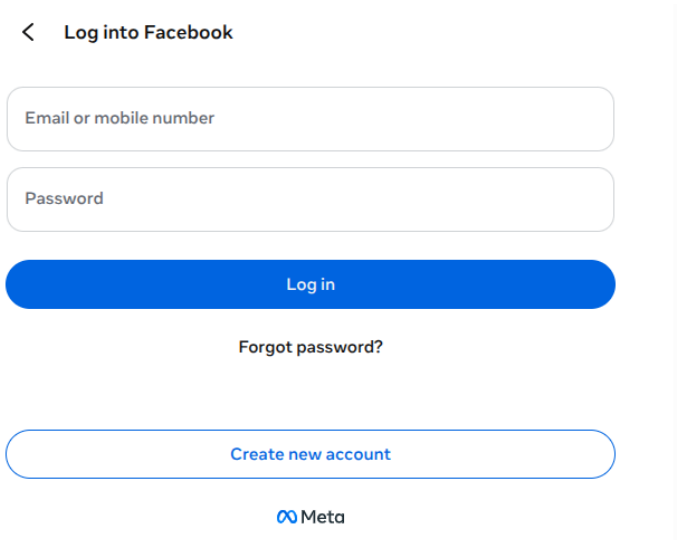


Figure 4.3: Facebook login page, which is not explicitly offering FIDO2 login option, although implementing it.

4.4.3 Signals

The identification of FIDO2/WebAuthn usages relies on 18 signals captured during the automated scraping of each website, with the lowest possible interaction with it. These signals are distributed between three types, the first being technical. These signals are direct factual observations. They provide an objective proof of a certain mechanism usage. In other words, these are direct, verifiable evidences of authentication mechanisms observed through browser-level inspection. They originate from explicit implementation traces of FIDO2/WebAuthn or classical login flows. For example, a signal can be the presence of a password input field in the DOM. Figure 4.4 represents the DOM of the Facebook login page, with the password input field highlighted. This field is going to be detected by FIDOLOGY. The WebAuthn API, a call to `navigator.credentials.create()` or to `navigator.credentials.create()` is a technical signal and is illustrated in the exhaustive example in Section 4.5.

Figure 4.4: Password input field present in the DOM of the Facebook login page.

The second type heuristic and includes indirect contextual clues (interpretation based on UI or storage data). Namely, these are indirect clues suggesting the use of a given authentication mechanism. They rely on observed patterns or keywords in UI text, stored data, or structural hints. While they are not conclusive on their own, they help strengthen inferences when combined with technical signals. As an example, FIDO-related terms in `localStrorage` or `sessionStrorage` are heuristic signals.

The last type is synthetic. The synthetic signals represent logical inferences and are deduced from correlations between signals. In other words, these are derived or inferred signals, built by correlating multiple technical and heuristic observations. They do not come directly from the page, but result from reasoning logic implemented in the analysis engine. An example is the signal `fido2_indirect_possible`. It is used to show that the FedCM provider likely supports FIDO2.

Two tables are used to present the signals. Table 4.2 lists each signal along with its ID, type, and weight, while Table 4.3 provides the corresponding descriptions.

ID	Signal	Type	Weight
ST_1	password_input_present	Technical (DOM)	0.02
ST_2	password_input_in_shadow_dom	Technical (DOM)	0.02
ST_3	credentials_api_used	Technical (API)	0.3
ST_4	credentials_create_summary	Technical (API)	0.2
ST_5	network_webauthn	Technical (Network)	0.25
ST_6	network_password	Technical (Network)	0.06
ST_7	passkey_setup_endpoint_present	Technical (Network)	0.07
ST_8	fedcm_present	Technical (Federated API)	0.05
ST_9	fedcm_detected_via_api	Technical (Federated API)	0.1
SH_1	local_storage_contains_fido	Heuristic (Storage)	0.1
SH_2	session_storage_contains_fido	Heuristic (Storage)	0.05
SH_3	cookies_contain_fido	Heuristic (Storage)	0.02
SH_4	shadow_dom_webauthn	Heuristic (UI/Structure)	0.08
SH_5	ui_webauthn_keywords_present	Heuristic (UI)	0.02
SH_6	otp_indicators_present	Heuristic (UI/Text)	0.1
SH_7	fedcm_provider	Heuristic (Provider)	0.05
SH_8	auth_js_supports_passkey	Heuristic (JS Bundle)	0.05
SS_1	fido2_indirect_possible	Synthetic (Inductive)	0.05

Table 4.2: The 18 signals extracted by FIDOLOGY, along with their ID, type and weight.

4.4.4 Classification of usage

Once the signals have been captured, they are first put into 11 groups (represented in 4.1) in order to have more concise inference rules. These groups of signals are evaluated and the result is used to derive one of the 12 different classes of FIDO2/WebAuthn authentication usage. These classes are aggregated into the 5 authentication categories. An overview of this aggregation is provided in Figure 4.5.

The classes are grouped by security level. It starts with red, meaning no authentication mechanism is found, then orange for classic password authentication, followed by yellow for MFA, green represents the category with FIDO2/WebAuthn-based logins and finally, grey for unknown methods. Hereafter is provided a description of the 5 authentication categories along with a presentation of each of the 12 classes. An inference rule is provided for each class.

ID	Description
ST_1	Presence of a visible password field in the main DOM
ST_2	Password field hidden inside a shadow DOM
ST_3	Explicit use of *navigator.credentials.** (JavaScript API)
ST_4	Details of parameters passed to *navigator.credentials.create()* (publicKey)
ST_5	Network requests targeting WebAuthn endpoints (e.g., /WebAuthn, /assertion)
ST_6	Network request explicitly transmitting a password (e.g., form POST, JSON data)
ST_7	Backend exposes passkey
ST_8	FedCM detected through console logs
ST_9	FedCM directly detected via intercepted *navigator.credentials.get()* API calls
SH_1	FIDO/WebAuthn keywords found in localStorage
SH_2	Same FIDO/WebAuthn keywords found in sessionStorage
SH_3	Cookies containing FIDO/WebAuthn-related strings
SH_4	WebAuthn-related keywords detected inside shadow DOMs
SH_5	Visible UI keywords (e.g., security key or passkey)
SH_6	Detected FedCM identity provider (e.g., Google, Apple)
SH_7	Evidence of an OTP flow (e.g., SMS, email, TOTP).
SH_8	Frontend plans passkey
SS_1	Inference that FedCM could trigger FIDO2 (based on known IdP)

Table 4.3: ID and description for the 18 signals extracted by FIDOLOGY.

$$
\begin{aligned}
P &= \texttt{password\_input\_present} \lor \texttt{password\_input\_in\_shadow\_dom} \\
P_n &= \texttt{network\_password} \land \neg P \\
O &= \texttt{otp\_indicators\_present} \\
F_s &= \texttt{credentials\_api\_used} \lor \texttt{credentials\_create\_summary} \\
&\quad \lor \texttt{network\_webauthn} \\
F_w &= \texttt{ui\_webauthn\_keywords\_present} \lor \texttt{shadow\_dom\_webauthn} \\
S &= \texttt{cookies\_contain\_fido} \lor \texttt{local\_storage\_contains\_fido} \\
&\quad \lor \texttt{session\_storage\_contains\_fido} \\
\text{FedCM} &= \texttt{fedcm\_present} \lor \texttt{fedcm\_detected\_via\_api} \\
&\quad \lor \texttt{fido2\_indirect\_possible} \lor \texttt{fedcm\_provider} \\
J &= \texttt{auth\_js\_supports\_passkey} \\
U &= \texttt{auth\_surface\_unobservable} \\
A &= \texttt{auth\_form\_appeared} \\
E &= \texttt{analysis\_error}
\end{aligned}
\tag{4.1}
$$

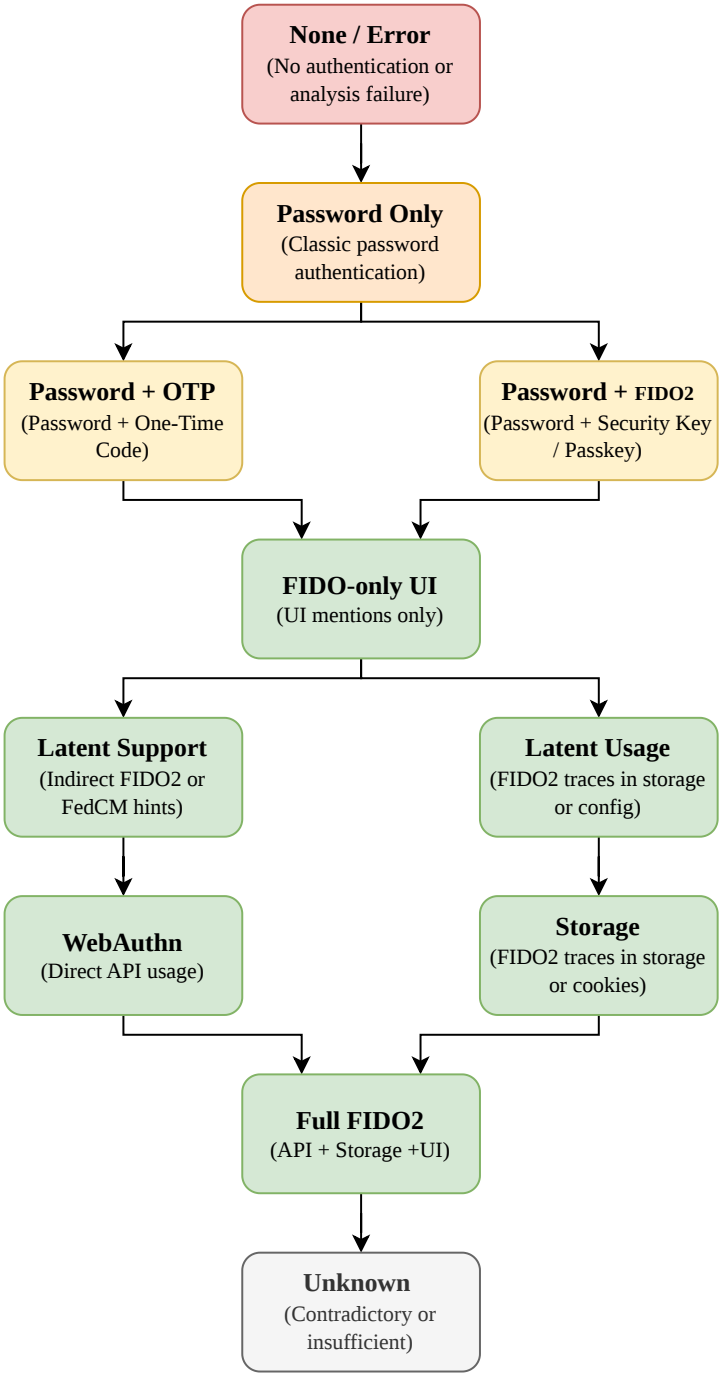


Figure 4.5: Classes grouped by security level: red indicates unsecured (no authentication), orange through green represent increasing levels of trust, and grey denotes an unknown authentication method.

The `None/Error` category indicates that no authentication mechanism could be determined. In practice, it means that no authentication clues could be observed. Particularly, the `none` class happens when no signals or login surface could be discovered during the analysis of a website. It does not imply directly that the website had no authentication mechanism, it rather indicates that FIDOLOGY could not found the login page, expose login UI or find clues related to signals. The `error` class is used when an error is raised during the analysis. Such an error can occur when the website actively deny access to automated browser and is a bot detection issue. For example, navigating to the website Adobe results in a HTTP/2 protocol error [72]. In that context, the mechanism is explicitly classified as `error`. It is more precise and thus indicates that the negative outcome cannot be improved. Therefore, two classes are inferred in the `None/Error` category through the rules 4.2.

$$\begin{aligned} \texttt{none} &\Leftrightarrow \neg P \land \neg P_n \land \neg O \land \neg F_s \\ &\quad \land \neg F_w \land \neg S \land \neg \text{FedCM} \\ \texttt{error} &\Leftrightarrow E \end{aligned} \tag{4.2}$$

The `Password-Based` category corresponds to the classical authentication schemes, which rely only on passwords. It implies that during the scraping, a usage of password is detected while no other additional protection like OTP or FIDO2 could be detected. This does not directly indicate that the site only uses passwords, but simply that scraping did not detect any further information. A single class is considered in this category, and is obtained via the inference rule 4.3.

$$\begin{aligned} \texttt{password\_only} \Leftrightarrow (P \lor P_n) \land \neg O \land \neg F_s \\ \land \neg F_w \land \neg S \land \neg \text{FedCM} \land \neg J \end{aligned} \tag{4.3}$$

The `Password-Extended` category indicates two-factor or transitional mechanisms. They combine passwords with another factor. This second factor can be a One-Time password (e.g., TOTP, SMS, or email) or FIDO2/WebAuthn-based methods. This authentication system represents a gradual adoption of enhanced security. In this category, two classes are distinguished, `password+otp` and `password+fido`. They are determined by the two rules in 4.4.

$$\begin{aligned} \texttt{password+otp} &\Leftrightarrow P \land O \land \neg F_s \\ \texttt{password+fido} &\Leftrightarrow P \land (F_s \lor \text{FedCM}) \end{aligned} \tag{4.4}$$

The `Fido2-Native` category represents passwordless or FIDO2-centric mechanisms. It refers to modern authentication flows based primarily on FIDO2/WebAuthn. It includes direct use with the WebAuthn API and latent/indirect indicators, such as FedCM, user interface clues, or stored credentials. Six classes, `webauthn`, `fido_only_ui`, `storage`, `latent_support`, `latent_usage` and `full_fido2`, can be inferred based on the rules 4.5.

$$
\begin{aligned}
\texttt{fido\_only\_ui} &\Leftrightarrow F_w \wedge \neg P \wedge \neg P_n \\
&\quad \wedge \neg F_s \wedge \neg S \wedge \neg \text{FedCM} \\
\texttt{latent\_support} &\Leftrightarrow \text{FedCM} \vee J \vee S \vee F_w \\
\texttt{latent\_usage} &\Leftrightarrow \neg P \wedge \neg P_n \wedge \neg F_s \\
&\quad \wedge (F_w \vee S) \wedge \neg \text{FedCM} \\
\texttt{storage} &\Leftrightarrow S \wedge \neg P \wedge \neg P_n \\
&\quad \wedge \neg F_s \wedge \neg F_w \wedge \neg \text{FedCM} \\
\texttt{webauthn} &\Leftrightarrow F_s \wedge \neg P \wedge \neg P_n \\
&\quad \wedge \neg F_w \wedge \neg S \wedge \neg \text{FedCM} \\
\texttt{full\_fido2} &\Leftrightarrow F_s \wedge (F_w \vee S \vee \text{FedCM})
\end{aligned}
\tag{4.5}
$$

The `Unknown` category covers uncertain or ambiguous cases. It corresponds to mixed or contradictory signals that do not allow for clear classification. It may require further analysis or context-specific interpretation, but it is not expected to see this type of case often. This leads to a single class, `unknown`, defined in the rule 4.6.

$$
\begin{aligned}
\texttt{unknown} \Leftrightarrow\ & \neg\texttt{error} \wedge \neg\texttt{none} \\
& \wedge \neg\texttt{auth\_surface\_unobservable} \\
& \wedge \neg\texttt{password\_only} \\
& \wedge \neg\texttt{password+fido} \wedge \neg\texttt{password+otp} \\
& \wedge \neg\texttt{fido\_only\_ui} \wedge \neg\texttt{latent\_support} \\
& \wedge \neg\texttt{latent\_usage} \wedge \neg\texttt{storage} \\
& \wedge \neg\texttt{webauthn} \wedge \neg\texttt{full\_fido2}
\end{aligned}
\tag{4.6}
$$

The Table 4.4 shows the various inferred classes, as well as their ID, the category they belong to and the dominant conditions. Table 4.5 summarizes the inference rules.

To conclude this section, a remark concerning the accuracy of inference is made. An inference is considered correct as long as the inferred class belongs to the correct category, meaning the exact class is not required. For example, if a website is classified as `webauthn` and the actual usage is `full_fido2`, the categorisation is considered successful. This remark will be useful in the section 5.1, when checking the precision and correctness.

4.4.5 Confidence score

The confidence score represents the reliability estimated by FIDOLOGY in classifying an authentication mechanism. The score is expressed as a numerical value between 0.0 and 1.0, with the lowest value representing zero confidence or insufficient evidence, and the highest value representing high confidence in the use of the inferred authentication. The score is calculated heuristically by aggregating the signals detected during the scraping process.

ID	Class	Dominant Conditions	Category	Weight
NE_1	`none`	No authentication-related signals detected (not even a password field).	`None/Error`	[0, 0.05]
NE_2	`error`	Scraping failure (timeout, crash, or navigation error).	`None/Error`	[0, 0]
PB_1	`password_only`	Password field detected without additional signals (no FIDO2 nor OTP).	`Password-Based`	[0.05, 0.2]
PE_1	`password+otp`	Password field with OTP indicators present.	`Password-Extended`	[0.25, 0.45]
PE_2	`password+fido`	Password field and FIDO2/WebAuthn signals present.	`Password-Extended`	[0.45, 0.7]
F_1	`fido_only_ui`	Only textual/UI WebAuthn indicators detected.	`Fido2-Native`	[0.25, 0.35]
F_2	`latent_support`	Indirect evidence (e.g., FedCM or UI hints) without explicit WebAuthn activity.	`Fido2-Native`	[0.25, 0.4]
F_3	`latent_usage`	Evidence of potential or deferred FIDO2 support (e.g., configuration pages).	`Fido2-Native`	[0.35, 0.5]
F_4	`storage`	Only storage-related FIDO indicators detected.	`Fido2-Native`	[0.4, 0.55]
F_5	`webauthn`	Explicit WebAuthn API usage without password or storage involvement.	`Fido2-Native`	[0.6, 0.8]
F_6	`full_fido2`	WebAuthn API combined with storage and UI indicators.	`Fido2-Native`	[0.8, 1.0]
U_1	`unknown`	Contradictory or insufficient signals to determine classification.	`Unknown`	[0.2, 0.4]

Table 4.4: The 12 classes for FIDO2/authentication usage along with ID, dominant conditions, category and weight.

Signal \ Class	NE_1	NE_2	PB_1	PE_1	PE_2	F_1	F_2	F_3	F_4	F_5	F_6	U_1
ST_1	-	C	S	R	R	C	-	C	C	C	-	C
ST_2	-	C	S	R	R	C	-	C	C	C	-	C
ST_3	-	C	C	C	S	C	-	C	C	R	R	C
ST_4	-	C	C	C	S	C	-	C	C	R	R	C
ST_5	-	C	C	C	S	C	-	C	C	R	R	C
ST_6	-	C	S	-	-	C	-	C	C	C	-	C
ST_7	-	-	C	-	-	-	-	-	-	-	-	C
ST_8	-	C	C	-	S	C	S	C	C	C	-	C
ST_9	-	C	C	-	S	C	S	C	C	C	-	C
SH_1	-	C	C	-	-	C	S	S	R	C	S	C
SH_2	-	C	C	-	-	C	S	S	R	C	S	C
SH_3	-	C	C	-	-	C	S	S	R	C	S	C
SH_4	-	C	C	-	-	R	S	S	C	C	S	C
SH_5	-	C	C	-	-	R	S	S	C	C	S	C
SH_6	-	C	C	R	-	-	-	-	-	-	-	C
SH_7	-	C	C	-	C	C	S	C	C	C	-	C
SH_8	-	-	C	-	-	-	S	-	-	-	-	C
SS_1	-	C	C	-	C	C	S	C	C	C	-	C

Table 4.5: Authentication classes inferred by active signals, C meaning contradictory, S supplementary and R required.

Each signal is associated with a weight reflecting its evidential strength and reliability. The weights are defined in order to ensure stronger, more deterministic technical evidence, like API calls, to dominate the scoring process. While weaker, contextual signals, like keywords presence in UI, have limited influence. There are three ranges:

1. ≥ 0.25 (strong): direct, technical proofs of FIDO2/WebAuthn usage
2. 0.10 - 0.25 (moderate): indirect but reliable clues
3. < 0.10 (weak): heuristic or contextual hints

The weights ensure that stronger, more deterministic technical evidence, like API calls, dominates the scoring process, while weaker, contextual clues have a limited influence. The Table 4.2 shows the score assigned to each signal.

The scoring process consists of four steps. The first step is to detect signals. Each signal is evaluated during the analysis of a website. Depending on the signal, it can be a Boolean or numerical value. The second step is a weighted aggregation. The score starts at 0.0 and increases based on detected signals:

$$score = \sum (signal\_value \times signal\_weight) \tag{4.7}$$

The third step is a contextual adjustment. The raw score is refined according to the inferred authentication usage. Certain combinations of signals can increase the score,

e.g., the score is higher if both WebAuthn API and network activity are confirmed. The score will be lower if only FIDO2 keywords are detected in the UI.

The last step is a diagnostic message. It adds a textual explanation describing the reasoning. It helps understand and interpreting the results. For example, it can be 'Full FIDO2 flow detected' for the `Fido2-Native` category or 'No authentication clues detected' for the `None/Error`. The scoring process places each class within an expected score range. Those ranges are provided in Table 4.4.

4.5 FIDOLOGY in practice

An example of how FIDOLOGY classifies the websites is provided in this section. The website Afas is used for the whole instance. First, on the authentication page in Figure 4.6, it can be easily seen that the passkey option is enabled right away. No user account has been created nor any parameters modified. When FIDOLOGY analysis this page, it will raise to true the `ui_webauthn_keywords_present` signal, which will allow the group F_w (from 4.1) to be true.

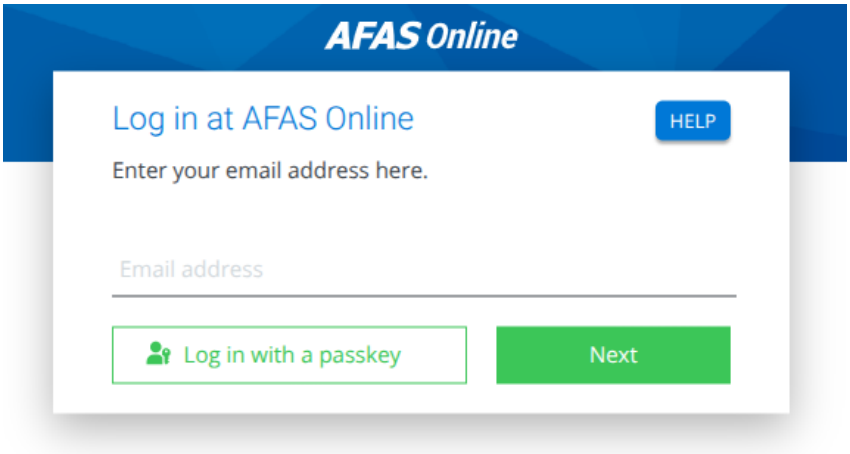


Figure 4.6: Afas login page exposing explicitly the passkey login option.

Figure 4.7 exposes the call to the `navigator.credentials.get()` that FIDOLOGY will detect. It will put the `credentials_api_used` signal to true, which in turn is valued in group F_s. Following inference rules in 4.5, these two groups, F_s and F_w, are required for classifying a website in the `full_fido2` class, therefore it will be in the `Fido2-Native` category. In addition, since this authentication surface exposes the passkey option and is in the `Fido2-Native` category, FIDOLOGY is going to validate the passwordless usage.

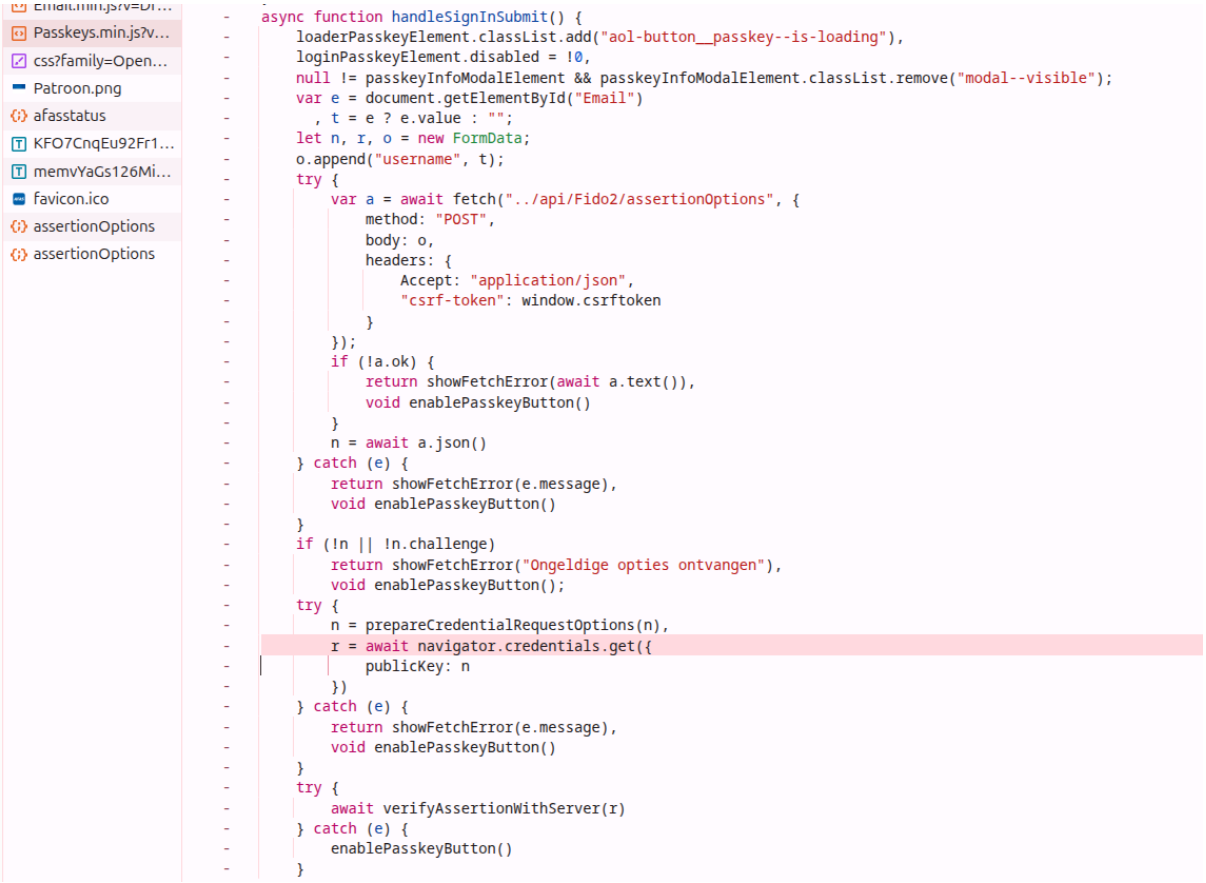


Figure 4.7: Afas network window exposing the `navigator.credentials.get()` call.

CHAPTER 5
GROUNDTRUTH

This chapter presents the manually established groundtruth and explains the methodology 5.1 used to achieve this. It then describes the groundtruth relating to countries, IAB categories and the authentication mechanisms in place 5.2. Following that, it provides an analysis of the raw results (Section 5.3), the filter step (Section 5.4) and the filtered results (Section 5.5), including validation. Finally, in Section 5.6, it assesses the determinism of the tool.

5.1 Methodology

A reliable groundtruth was constructed in order to evaluate the accuracy of FIDOLOGY. As the aim of this research is to study FIDO2, several publicly available lists of websites claiming to support FIDO and passkeys were collected and analysed. However, not all of these lists were reliable; it was therefore necessary to analyse each site manually to determine exactly which authentication mechanisms is used. To get a complete groundtruth database, IAB categories and countries were included. For IAB categories, they were either derived from the lists, or manually assigned. For countries, the mapping tool from Section 4.2.2 is used. In total, four lists were used, resulting in a manually crafted groundtruth of 1,000 lines.

The first list is the Passkey Directory of the Fido Alliance website [3]. Websites were taken on October 16th, 2025. The IAB categories were not provided, and thus were manually assigned to each website.

The FIDO2 section of Hideez [42] constitutes the second list. Websites referenced as using FIDO2, OTP, and access keys were taken into account. The list was analysed on October 17th, 2025. Table B.1 of Appendix B illustrates a mapping between the original categories and the required IAB categories.

The passkey directory provided by 1Password [1] is used as the third list. It is a community-driven index of websites, apps, and services that offer signing in with passkeys. The websites were taken on October 20th, 2025. In Table B.2 of Appendix B is the category mapping between the provided categories and the required IAB categories.

The last list is from the dongle-auth section of BuyBitCoinWorldWide [102] and was analysed between October 22nd and October 27th, 2025. The category mapping is provided in Table B.3 of Appendix B.

Since manual analysis is limited, only 8 classes were attributed during the inspection of the websites. The simplest one is the `none` class and is given to websites that don't have any authentication mechanism. For example, websites like Dsxinc or Dutch-Bros don't allow users to have an account. Websites that use a traditional password system are assigned the `password_only` class. The `password+otp` and `password+fido` classes are assigned when a second security factor is present in addition to a password. The class `webauthn` is used when the API calls `navigator.credentials.get()` or `navigator.credentials.create()` are detected during network inspection. For the websites exposing directly options to use passkeys, i.e., a FIDO2-based method, the class `full_fido2` is associated. Some URLs also promise a FIDO2 option, but an account or an activation in the parameters is needed. These are classified under `latent_support`.

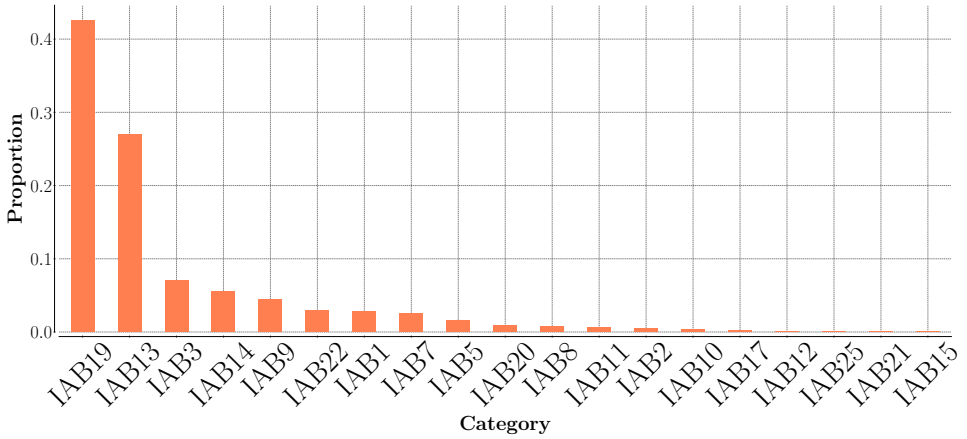
5.2 Groundtruth description

5.2.1 Countries and IAB categories

In Figure 5.1a can be seen a repartition of the countries of the groundtruth. It represents the top 30 countries. As expected, since the lists were referenced from American websites, the United States of America and Canada is in the top 3. Germany is in the second position, as it is in the centre of Europe and has recently published some guidelines and standards in order to take a step forwards joining the passwordless authentication train [58, 89]. The rest of the countries have a limited presence and the variations between them compared to the United States of America are very small. In total, 45 countries are represented in the groundtruth.

Figure 5.1b shows the distribution of IAB categories in the groundtruth. The official mapping for IAB label and IAB code is present in Table B.1. As websites working in Technology & Computing fields are more subject to using FIDO2, the first category is obviously IAB 19. The second place, accorded to Personal Finance (IAB 13), is also not surprising. Finance websites need to have robust security measures in order to protect the financial data and assets [2]. Any failure to comply with this obligation may result in significant financial losses, damage to reputation and legal liability [96]. Fields like Home & Garden and Sports are less impacted by their level of security, and thus appear later on the repartition.

(a) Repartition of the top 30 countries of the groundtruth ($N = 977$).



(b) Repartition of the IAB categories of the groundtruth ($N = 993$).

Figure 5.1: Countries and IAB categories distribution of the groundtruth.

5.2.2 Authentication categories

In Figure 5.2 can be seen the frequency distribution of the authentication categories of the groundtruth. The most commonly seen category is `Fido2-Native` (33.5%). As it is FIDO2-oriented, this set is not neutral towards the authentication mechanisms. That is why collecting data on 1 million URLs will enable to properly assess the actual level of adoption of FIDO2 worldwide.

The second most common category is the `Password-Based` one (32.5%), followed by the `Password-Extended` (18.4%) and the `None/Error` category (15.2%). In almost one third of the `None/Error` websites, they have a login interface, but an interaction with it is needed in order to expose more than just the email/username input. For example, on Daylite, which has its initial login page represented in Figure 5.3a, the password is not clearly exposed. The email needs to be entered and the 'Continue' button pressed in order to reveal the password input field (see Figure 5.3b). Since the intrusion cannot be made by FIDOLOGY, no authentication mechanism can be deducted from the initial login page. The rest of websites do not provide any login option.

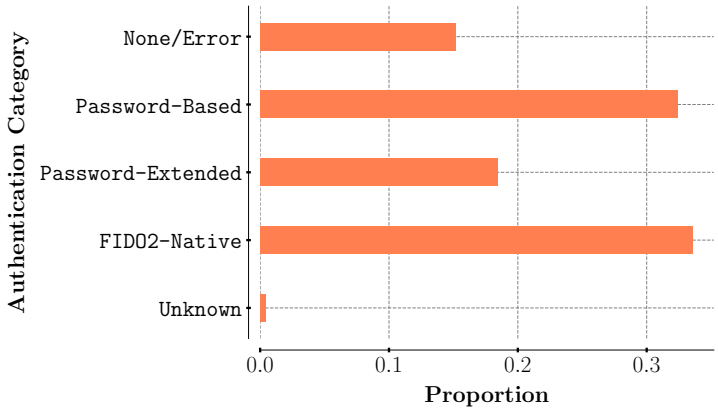
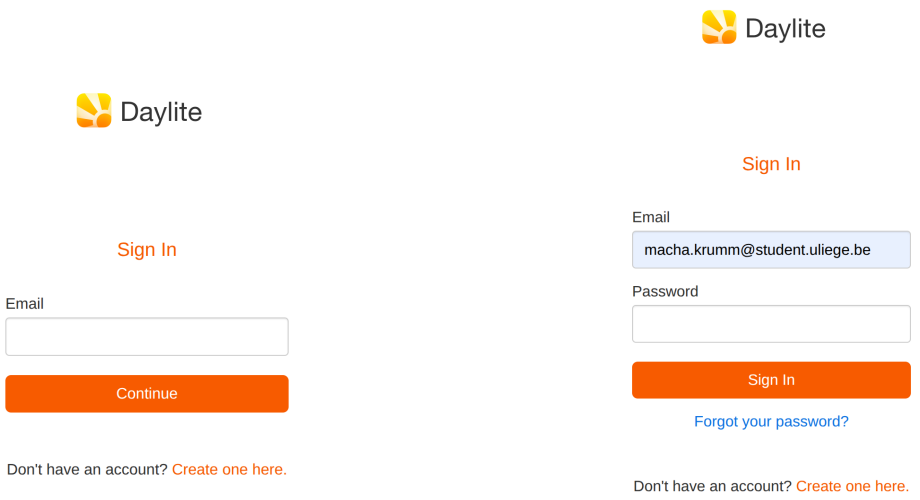


Figure 5.2: Frequency of the 5 authentication categories of the groundtruth ($N = 993$).



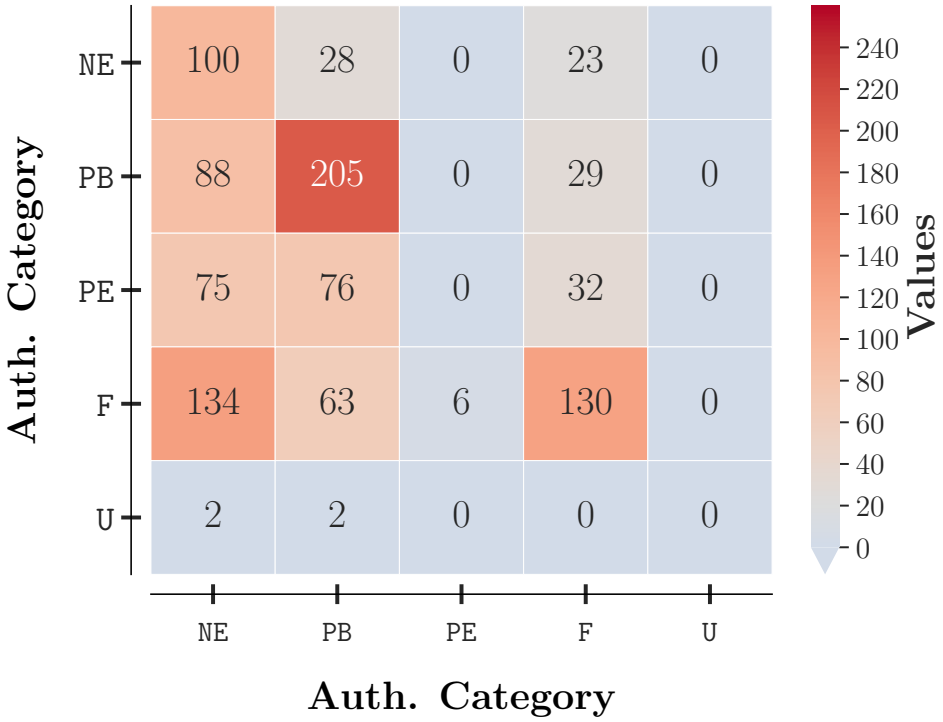
(a) Screenshot of Daylite's login page showing that the password field is not directly presented to user.

(b) Screenshot of Daylite's login page showing that the password field is presented after email and button interaction.

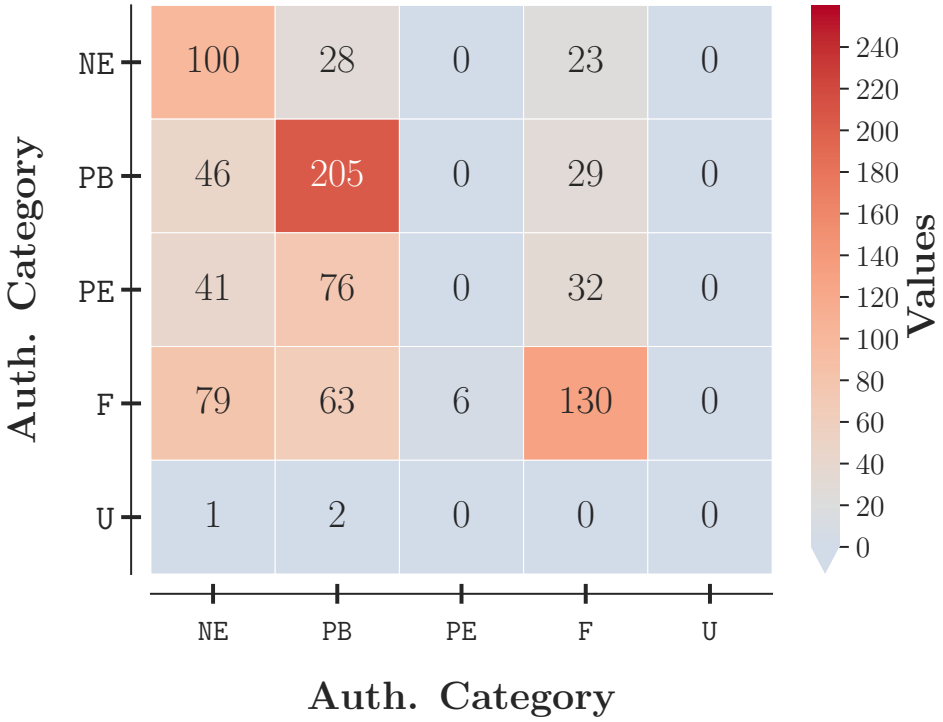
Figure 5.3: Daylite's login page.

5.3 Raw results

The confusion matrices derived from the groundtruth scraping made on March 23rd, 2026 are represented in Figure 5.4. Since seven websites were removed because of service interruption, the groundtruth has a total of 993 lines. The first matrix seen in Figure 5.4a is the heatmap computed with the raw results with no post-processing filter step. This gives FIDOLOGY an accuracy rate of 43.8%.



(a) Matrix before filtering ($N = 993$).



(b) Matrix after filtering ($N = 828$).

Figure 5.4: Confusion matrices of the groundtruth scraping from March 23rd, 2026. The X-coordinates being the categories predicted by FIDOLOGY and the Y-coordinates the groundtruth categories.

What stands out most is the number of websites in the `None/Error` column. As mentioned above, when a website is classified in this category, it may be because it has no authentication mechanism or because no signal could be observed by FIDOLOGY. However, in some cases, it is simply the website that is not responding correctly to the scraper. It can happen that the navigation timeout is exceeded or an 'error empty response' occurs. The error usually occurs when there is no data response from the requested server, which can be due to various reasons, such as server issues (including issues with a dedicated server), network issues, incorrect URLs, website unavailability, or browser cache issues [35]. Analysis errors like that happened 53 times during the scraping. Therefore, in order to increase the accuracy and to remove external factors that makes it lower, a filtering step is applied on the results.

5.4 Filter step

This step removes websites that, despite best efforts, are still classified under the `None/Error` category, meaning they could not be correctly categorized. Seven distinct barriers have been identified. The statistics about this filtering step on the groundtruth can be seen in Table 5.1. The seven limitations are explained in more detail hereafter.

Barrier	Raw value	Percentage
Anti-bot challenge blocking	28	2.82%
Closed shadow DOM	33	3.32%
Headless browser blocking	39	3.93%
Iframe opaque	41	4.03%
JS orchestrated auth ui	25	2.52%
Password without observability	0	0.00%
Stalled auth ui	0	0.00%
Total	165	16.62%

Table 5.1: Results of the filtering step applied to the groundtruth scraping from March 23rd, 2026 ($N = 993$).

The first identified obstacle is an anti-bot challenge. This was encountered 28 times during the groundtruth scraping. These challenges can happen in the form of a Cloudflare Turnstile. It helps confirm that web visitors are real humans and blocks unwanted bots without slowing down web experiences for users [19]. It adapts the type and the difficulty of the challenge to each user and browser interaction by gathering information about the user and the environment [18].

Cloudflare also has a Web Application Firewall (WAF) option. It allows admins to define rules that will help check and filter incoming wed and API requests. Website can thus

easily protect specified paths, like '/login' or '/account', or more generally all different login path of websites [20]. Therefore, the scraper can't access the login UI, despite having the correct login path. Admins can as well add rate limiting rules [21]. It should be less triggered in this analysis. But sometimes, in the testing phase of FIDOLOGY, some websites were more requested, and a rule could have been triggered, leaving further analysis impossible. Other anti-bot solutions exist, but as this is not the main focus of the thesis, only generic detection is used to identify a website as having an anti-bot solution [30].

Closed shadow DOM (see Section 3.1) forms the second barrier. This browser mechanism is identified for 33 websites. When a website is suspected to use this feature, a validation step is applied to confirm the practice. The shadow DOM validation process takes place in three stages. The first step is to identify the web components or elements whose content is displayed but which do not contain any child elements that can be queried. The second is checking if the password inputs are inaccessible via the `querySelector()` despite finding a visible authentication interface. And the last step is comparing these findings to assign a confidence level that indicates that the login form is encapsulated in an inaccessible shadow DOM. A high confidence is given when a suspected closed shadow root exists and when the password input is unreachable. 96% of the closed shadow DOM cases are validated with high confidence. The remaining 4% is due to linguistic issues related to the keywords taken into account (for example, Japanese keywords are not covered).

The third encountered obstacle is headless browser blocking. This represents 3.93% of the groundtruth. The main difference between the manually collected groundtruth results and those collected by FIDOLOGY is that the scraper uses a browser without a graphical user interface. Therefore, as said in Section 4.4.4, some websites forbid directly access when using a headless browser. A strategy to bypass that is to raise some flags during headless browser creation. The flag `-disable-blink-features=AutomationControlled` is used in this project [69]. Another interesting setting to bypass anti-bot detection is to use different user agents and rotate between them [76]. Initially, this strategy was implemented. However, it undermined the deterministic nature of the measurement. That is why a single user agent is now used.

An opaque authentication surface can also become a barrier. It happened for 41 scraped websites of the groundtruth. This occurs when FIDOLOGY can navigate to the login UI but can not find any clues there. These clues can be found manually in the settings and the rest of the website, but for scraping, this is practically impossible. It would require a considerable effort for a small result. The authentication UI can also be orchestrated by JavaScript and thus inaccessible in DOM. This is the case for 2.52%. The last two limitations are password input without observability and stalled authentication UI. These cases were taken into account to be sure not to miss them. However, FIDOLOGY did not came across them.

5.5 Filtered results

5.5.1 Classification

The resulting confusion matrix obtained after applying the filter step to the raw results is illustrated in Figure 5.4b. Concerning the accuracy, it is now at 50%. Figure 5.5 shows more clearly the frequency of authentication categories. The most seen category is still the `None/Error` one, closely followed by the `Password-Based` category. `Fido2-Native` websites are predicted in almost 22% of the cases. `Password-Extended` category is given to only 6 websites. It appears that no websites are classified under the `Unknown` label. That is a sign that the decision tree is quite correctly constructed and that it has an obvious outcome for all URLs.

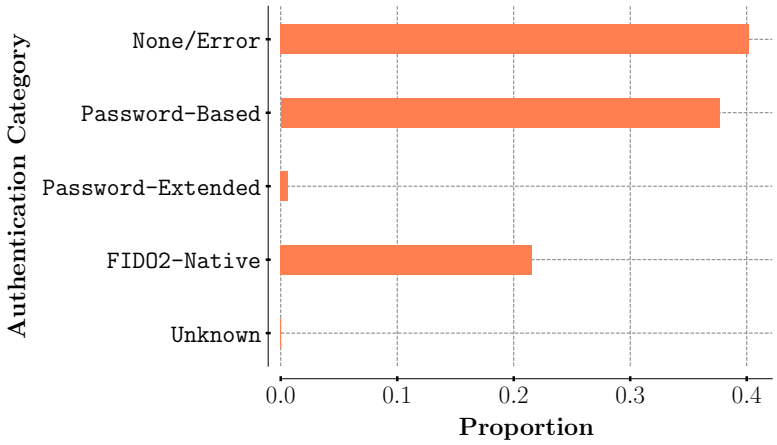


Figure 5.5: Frequency of 5 authentication categories of the groundtruth scraping from March 23rd, 2026 ($N = 828$).

The number of under-classified websites in the `None/Error` accounts now for only 16.82%. From Table 5.2 can be said that 56 of these pages were analysed, while in fact not being an authentication page. As explained in the Section 2.2, finding the login page can be really tricky. Some pages can be misinterpreted and considered as an authentication surface. This implies that a significant part of the apparent errors are in fact due to page interpretation errors rather than incorrect authentication mechanism inferences. There is therefore room for improvement regarding the procedure for accessing the login interface.

For only 3 websites, the analysed page is not interactive. Finally, in the large majority of the cases (64%), the under-classified websites fall into the unexplained residual category. This suggests that some patterns or limitations are still not identified and bypassed during the signal extraction automated process. Nevertheless, it shows that the filter step is quite effective as none of the residual errors are associated with known obfuscation mechanisms.

Metric	Raw value	Percentage
No login surface	0	0.00%
3rd party redirect	0	0.00%
Not an authentication page	56	5.64%
Silent anti-bot block	0	0.00%
Non-interactive UI	3	0.30%
Unexplained residual	108	10.88%
Total	167	16.82%

Table 5.2: Repartition of the websites under-classified in the `None/Error` category. The 167 websites are divided between 6 suspected causes.

About the websites categorized as `Password-Based`, 76 are under-classified and should instead fall under `Password-Extended`. This misclassification happens because of the heuristic used to detect OTP mechanisms, which is intentionally limited in order to remain fast as it is not the primary focus. Moreover, reliably detecting the use of a second authentication factor would require intrusive interaction with the website. In practice, the OTP is typically revealed only after a correct password has been entered (see 5.2.2 for the Daylite example). The remaining 63 websites should have been classified as `Fido2-Native`. This under-classification also arises because FIDO2 is often made available only after a password has been entered or through account settings following an initial traditional authentication.

Still in the `Password-Based`, there are 28 over-classified websites. This accounts for only 7.49% of password schemes categorized URLs during the scraping of the groundtruth. It can overall be explained by the presence of password input in the DOM or shadow DOM that were not discovered during the manual analysis.

Still 84 websites are over-classified in `Fido2-Native`. Figure 5.6 represents each of the 6 FIDO classes together with their confidence score. The `latent_support` class represents 75% of those over-classified websites and the `storage` class accounts for less than 4%. That actually reveals several levels of ecosystem readiness, ranging from technical pre-deployment and identity outsourcing (e.g., FIDOLOGY is redirected to an IdP that could be compatible with FIDO2) to marketing discourse (e.g., claiming to be compatible with FIDO2 on other websites, when this is not the case in practice).

About 18% of them have FIDO2 hints in the user interface, i.e., a raised `fido_only_ui` signal, and these have a low level of confidence associated with them. Websites that propose security solutions, e.g., PrivacyIdea or DoubleOctopus, have no authentication interface but have loads of FIDO2 related keywords on the UI. Only one `full_fido2` and one `webauthn` have been over-classified. This allows to say that 89.5% cases of these classes are correctly classified.

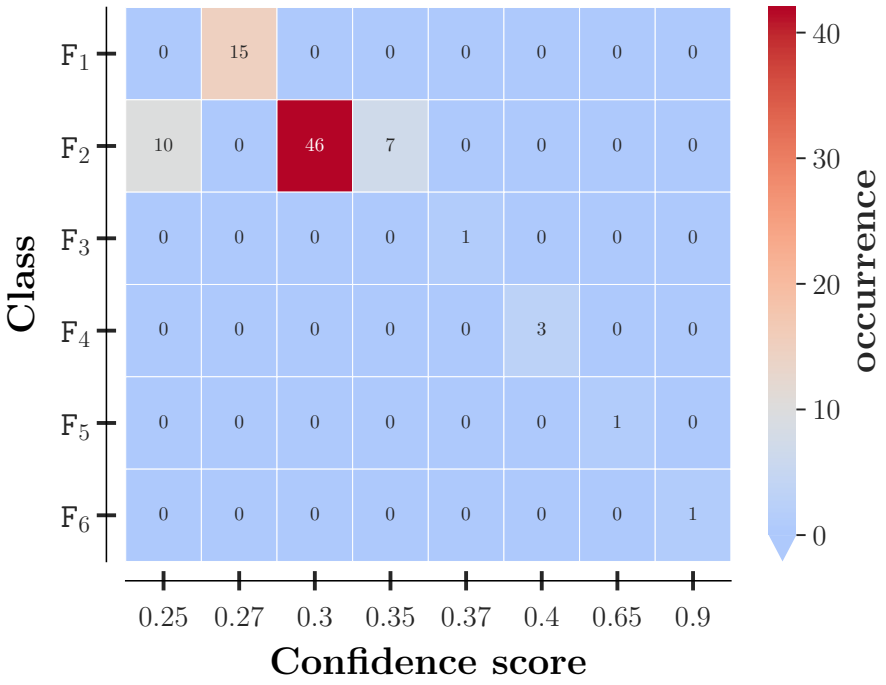


Figure 5.6: Fido confidence score of over-classified websites in the `Fido2-Native` category ($N = 84$).

5.5.2 Signals

The frequency of the most seen signals is represented in Figure 5.7. For the technical signals, the seen elements are a password input ($\texttt{ST}_1$) and a usage of credentials API ($\texttt{ST}_3$). Heuristic signals are observed in the storage, with local storage ($\texttt{SH}_1$), session storage ($\texttt{SH}_2$) and cookies ($\texttt{SH}_3$) containing FIDO clues, and in the UI ($\texttt{SH}_5$). The signal `auth_js_supports_passkey` ($\texttt{SH}_8$), i.e., the fact that the frontend plans on using passkeys, is seen in 15% of the websites.

For websites correctly classified in the `Fido2-Native` category, a forest plot has been generated and is shown in Figure 5.8. It illustrates the odds ratios and p-values in order to provide a better understanding of the contribution of each signal to the classification process. The confidence interval is computed with a 95% confidence. The class `latent_usage` is not provided since there were none in the groundtruth scraping results that are correctly classified.

For the `full_fido2` class, the strongest predictor is the usage of Credentials API (`credentials_api_used`). It has a high odds ratio (OR), equal to 305.96 and a very small p-value ($p < 10^{-12}$). It allows to say that an interaction with the WebAuthn API is a reliable indicator of an active passwordless authentication mechanism. The presence of FIDO related keywords in the user interface reinforce the classification. The signal has an odds

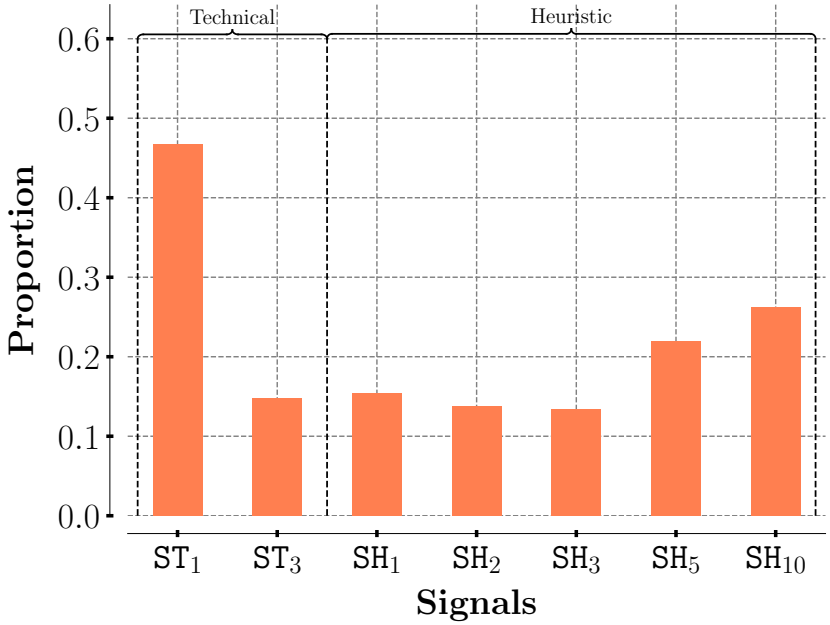


Figure 5.7: Frequency of the most seen signals of the groundtruth scraping from March 23rd, 2026 ($N = 828$).

ratio equal to 26.41, a confidence range that does not cross 1 and a statically significant p-value smaller than 0.0005. Signals having a p-value to 1 are statically non-significant. They have no particular effect on the classification [87]. Web storage features like `local_storage_contains_fido` or `session_storage_contains_fido` reveal no association, with an OR smaller than 1. It implies that the odds of these signals are higher in the other classes than in this one [93].

The importance of detecting Credentials Api usage is reinforced with the `webauthn` class. The only signal with a reliable association is `credentials_api_used`. It has an OR of 23.16, with a confidence range not crossing 1. Even if the sample size is small ($N = 12$), it shows a good statistical significance ($p < 0.05$). The `storage` class is determined by the web storage signals. The most important feature is `session_storage_contains_fido` with a high OR (equal to 40.67) and a statically notable p-value ($p < 0.001$). It is closely followed by `local_storage_contains_fido` ($OR = 9.08$, $p < 0.05$).

The `latent_support` class is driven by the JavaScript authentication support for passkeys (`auth_js_supports_passkey`). This signal is strongly associated with this class ($OR = 4.47$) and has an statistically significant effect with a p-value smaller than 0.001. UI-level indicators are also seen in the `fido_only_ui` class. The associated signal `ui_webauthn_keywords_present` has an OR of 18.50 and a p-value smaller than 0.005. The presence of this signal indicates that visible passwordless keywords are valuable but not satisfactory enough to confirm the usage of passkeys.

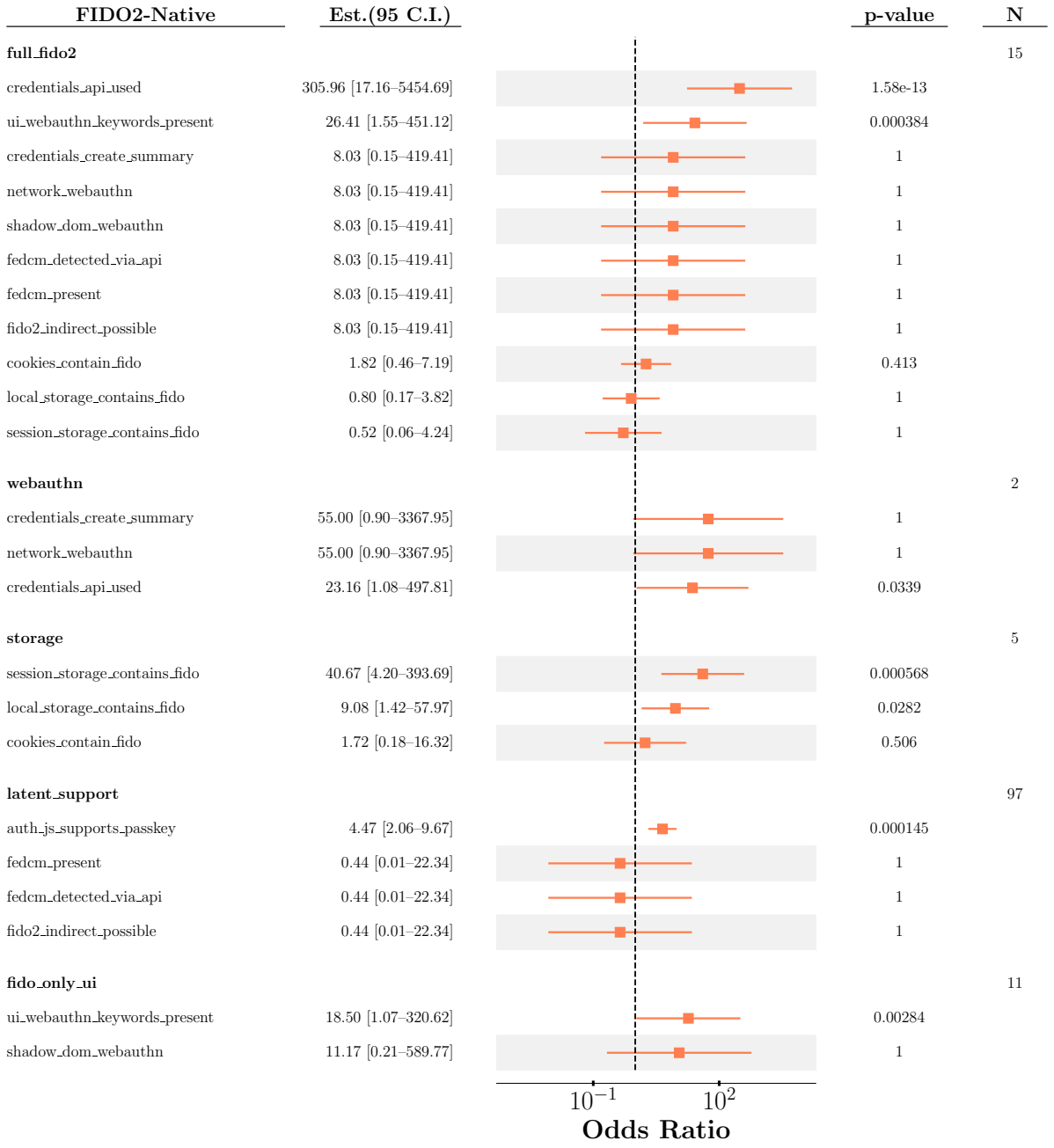


Figure 5.8: Odds ratios, confidence intervals and p-values forest plot for websites that are correctly classified as `Fido2-Native`. Only required or supplementary signal for each class are taken into account ($N = 130$).

To summarize, it can be said that observing usage of WebAuthn Credentials API usage is the most reliable indicator to prove the utilization of passwordless authentication with FIDO2. Storage and UI-level FIDO clues remains important to reinforce the results, but alone are not sufficient to confirm passkey usage implementation.

5.5.3 Validation

As explained in Section 4.4.2, a validation step is applied to `Fido2-Native` websites. The Table 5.3 illustrates the results. It is important to mention that the validation step does not change the initial classification. Classes are assigned based on the presence of observable signals during the scraping phase. In contrast, validation relies on interactive testing (e.g., attempting to click a passkey login option). The COSE algorithms are tried to be extracted. However, this didn't bring any results.

`Fido2-Native`	**Total**	**Validated**	**Validation ratio**
`fido_only_ui`	26	17	65.38%
`latent_support`	160	66	41.25%
`latent_usage`	1	0	0.00%
`storage`	8	1	12.50%
`webauthn`	3	1	33.33%
`full_fido2`	16	12	75.00%
Total	214	97	45.32%

Table 5.3: Total number of websites in the `Fido2-Native` category together with the validation ratios ($N = 214$).

Results are quite aligned with the expectations. Three quarters of the websites classified as `full_fido2` have a validated passwordless usage. Still, the validation step could fail in cases where the passkey button cannot be detected. This can occur when the clickable element is not be easily identifiable through patterns like 'button[aria-label*='Continue with Passkey']', when it is hidden behind a drop-down menu or exposed only to user who already have an account.

For the `storage` class, only 1 out of 8 websites is validated, which is expected since the class concentrates on those were only storage/cookies indicators have been found. Websites identified as having a latent support are validated in 41.25% of the cases. Among the 26 `fido_only_ui` URLs, 17 are validated. As a result, some websites classified as `latent_support` or `fido_only_ui` can be successfully validated, even though they lack sufficiently strong or explicit signals to be classified as `full_fido2`.

An interesting aspect is the comparison between correctly categorized websites and validated ones, as shown in Table 5.4. The results indicates that validated `Fido2-Native` websites are more frequently correctly categorized (72%) compared to non-validated ones (51%). It occurs that 68% of the incorrectly categorized cases are non-validated. This is a non-negligible proportion and confirms the reinforcement brought by this step.

However, in 27 cases, there is a validation while being in the wrong category. This means that these websites should have been in a less secure category, e.g., `Password-Based`.

A manual inspection of these reveals that 17 of them (63%) belong to domains related to security, authentication or access management. This may explain why the heuristic used to detect the passkey option can occasionally fail, as such websites display a lot of passwordless related elements.

Overall, this analyse highlights that the validation step should not be used alone but rather in addition to the original classification in order to improve the robustness of categorization.

	Correct category	**Incorrect category**	**Total**
Validated	70	27	97
Non-validated	60	57	117
Total	130	84	214

Table 5.4: Correctly categorized vs Validated `Fido2-Native` websites ($N = 214$).

5.6 Determinism

To evaluate the determinism of FIDOLOGY, three different scrapings of the groundtruth were realised. The first is the one analysed previously, i.e., the scraping from March 23$^{\text{rd}}$, 2026 (file_1). The second is from March 24$^{\text{th}}$, 2026 (file_2) and the last from March 30$^{\text{th}}$, 2026 (file_3).

The aspect that will affect most determinism is the reaction of the website. Depending on that, the results from the filter step will always variate. Table 5.5 puts to light the differences between the filter step results for the 3 tests. It appears that the number of filtered websites may vary from 107 (10.77%) to 192 (19.33%). The quantity for each type of barrier fluctuates strongly, in particular for closed shadow DOM cases and opaque iframes. As introduced in Section 5.4, those variations depend on several factors. The site may be going through a period where it is particularly vigilant, and therefore flag more easily the scraper as a bot and block it with an anti-bot challenge.

Figure 5.9 represents the frequency of each authentication category for the three files. It is computed over the 529 websites that are in common disregarding the ones filtered out. The graph shows that there is less than 5% variations in each category, and the `Fido2-Native` one only fluctuates by one website. Without taking into account the reaction of the website, other factors may introduce uncertainty during the scraping. For example, the load time of the page and the DOM content can vary based on the state of the network (varying network bandwidth and delay). Therefore, the timeout applied can modify the capture of the signals. During inference, no random decisions are taken and everything is based on heuristic and signal correlation. Overall, disregarding all the nondeterministic behaviour introduces by the web [41], this shows that FIDOLOGY is deterministic.

Barrier	Raw value file$_1$	Raw value file$_2$	Raw value file$_3$
Anti-bot challenge blocking	28	26	20
Closed shadow DOM	33	0	28
Headless browser blocking	39	52	78
Iframe opaque	41	0	46
JS orchestrated auth ui	25	29	20
Password without observability	0	0	0
Stalled auth ui	0	0	0
Total	165	107	192

Table 5.5: Differences in the filter step results for groundtruth scraping on March 23$^{\text{rd}}$, 2026 (file$_1$, $N = 529$), March 24$^{\text{th}}$, 2026 (file$_2$, $N = 529$) and March 30$^{\text{th}}$, 2026 (file$_3$, $N = 529$).

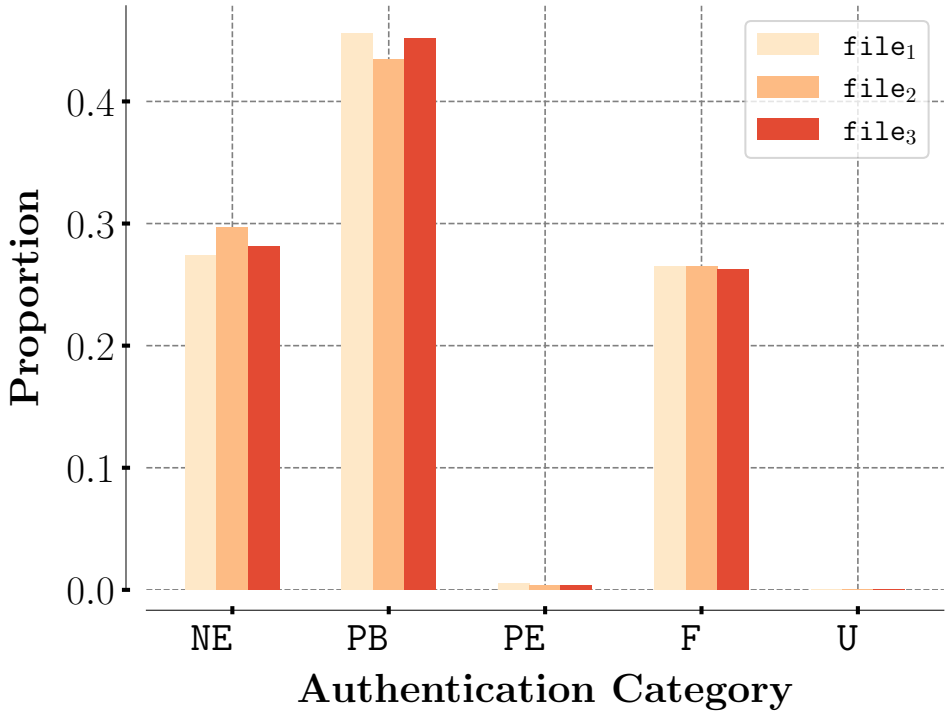


Figure 5.9: Authentication category frequency comparison for groundtruth scraping results on March 23$^{\text{rd}}$, 2026 (file$_1$, $N = 529$), March 24$^{\text{th}}$, 2026 (file$_2$, $N = 529$) and March 30$^{\text{th}}$, 2026 (file$_3$, $N = 529$).

CHAPTER 6

AUTHENTICATION IN THE WILD

This chapter describes the dataset of one million rows used to apply FIDOLOGY to the Web (Section 6.1). It covers the country and category mapping phases. Afterwards, the raw and filtered results of the scraping are exposed and analysed in Section 6.2. An illustration of the authentication mechanisms across countries and categories (Sections 6.3 and 6.4), along with an ethical review of FIDOLOGY in Section 6.5, is hold at the end.

6.1 Methodology

In order to evaluate globally the authentication mechanisms, a large dataset was constructed using three public web ranking lists: Tranco [73], CrUX [15] and Umbrella [17]. All had 1 million URLs at start and were downloaded on October 1^{st}, 2025. Since FIDOLOGY aims to extract data from one million websites, a dataset called the 1M dataset is created and used. This section explains how the countries and IAB categories are determined to form this dataset.

6.1.1 Country mapping

The country mapping step explained in Section 4.2.2 is separately applied on the three lists. Table 6.1 presents the statistics regarding this step. After filtering out websites appearing in multiple countries (*multi-country* label), each list contained on average 706,681 URLs. The method that yielded the best results was geolocation. This was followed by country identification using the TLD code.

After processing the three lists separately, they had to be combined. In total, there were 2,120,043 URLs. As the sets had several websites in common, e.g., Amazon obviously appears in each list, all duplicate domains had to be removed. URLs for which no country is attributed, even after merging, were deleted. In the end, 1,436,260 websites had an associated country.

Metric	CrUX	Umbrella	Tranco
Total length	1,000,000	1,000,000	1,000,000
Multi-countries (with duplicates)	367,077	581,032	183,733
Multi-countries (without duplicates)	69,988	119,966	61,931
Single-countries	632,923	418,968	816,267
Whois countries	73,413	77,842	137,526
Geolocation countries	343,931	334,560	366,778
TLD countries	250,859	59,368	305,182
Found countries	668,203	471,770	809,486
Not found countries	34,708	67,164	68,712
Time taken (seconds)	114,178.34	71,592.54	105,623.78

Table 6.1: Statistics about country distribution and processing time for CrUX, Umbrella, and Tranco URL lists.

To obtain the final 1M dataset, 1 million URLs were randomly selected from the previous combined list. Table 6.2 represents the final statistics about the dataset. It is clear that the URLs appearing in only one country, the *single-country* ones, predominate. It is also visible for the geolocation method, more than half of the countries were identified using that approach.

Metric	Merged list
Total length	1,000,000
Multi-countries	83,153
Single-countries	916,847
Whois countries	145,903
Geolocation countries	528,267
TLD countries	325,830

Table 6.2: Final statistics about country distribution for the 1M dataset.

Figure 6.1 presents the top 30 countries distribution of the new 1M dataset. As for the groundtruth countries, the highest peak corresponds to the United States of America. It accounts here for 42 % of the dataset (versus 60% in the groundtruth). Canada, Germany, France and Japan were among the top 8 in the groundtruth, while here they still rank among the top 7, give or take a few places. All other countries fall below the 5% mark.

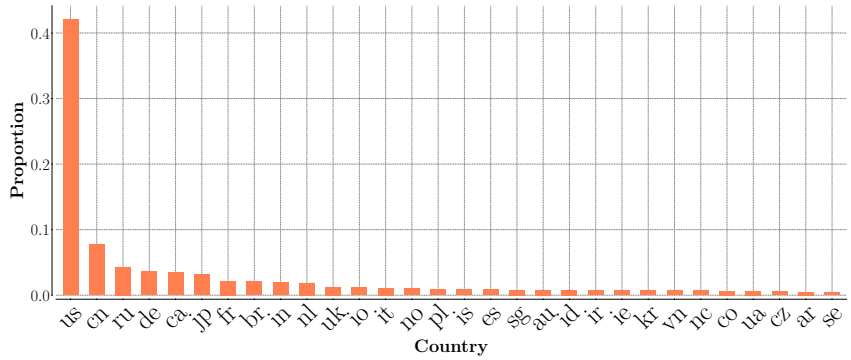


Figure 6.1: Distribution of the top 30 countries of the 1M dataset ($N = 882,385$).

6.1.2 Category mapping

By means of the LLM categorization method, 10,000 categories were derived and the statistics about the sources can be observed in Table 6.3. The websites were taken in the set of filtered results from the 1M dataset scraping (see more in Section 6.2.1), and only from the categories `Fido2-Native` and `Password-Based`. The categories were derived from November 26th to December 12th, 2025 and from March 23rd to April 20th, 2026. Since this category is still important in order to provide further analysis of the FIDOLOGY results, a manual investigation was realised on 100 websites to assess the accuracy. Out of these 100 categories, 23 were incorrect. While it is only an evaluation of a small sample, the results suggest that more than 75% of the found categories by the LLMs are correct.

Metric	Merged list
gemini-2.5-flash-lite	310
mistral-medium	440
mistral-medium-2508	4,167
mistral-small-2503	997
mistral-small-2603	3,979
open-mistral-7b	107
Total length	10,000

Table 6.3: Statistics about IAB categories distribution for 10,000 websites.

Figure 6.2 illustrates the distribution of IAB categories of the 10,000 websites. What stands out in this is the dominance of IAB 19 category, corresponding to 'Technology & Computing'. The same trend for this category was observed in the groundtruth. The next two most common categories are 'Shopping' (IAB 22) and 'Education' (IAB 5). There is then a slight decline until the final category.

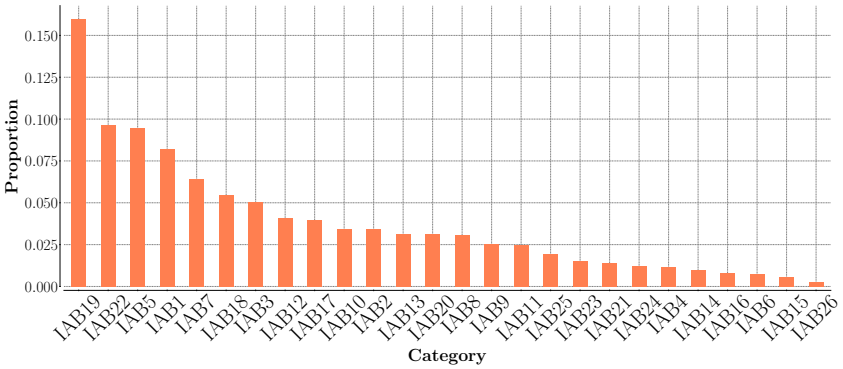


Figure 6.2: Distribution of the categories for 10,000 websites.

6.1.3 Run environment

To scale the FIDOLOGY scraping of the 1M dataset, it was divided into four parts. Each part was run by a machine having the following hardware configuration: dual Intel(R) Xeon(R) Gold 5318Y CPUs at 2.10 GHz, with 96 logical CPUs, and 251 GiB of RAM. Considering these features and the parallelization described in Section 4.1.2, scraping the 1M dataset took three days, from March 24$^{\text{th}}$ to March 27$^{\text{th}}$, 2026.

6.2 Results

6.2.1 Filtering and classification

A pre-scraping filter step is applied on the 1M dataset. It uses heuristics to detect URLs that look like CDN infrastructure or suspicious domains. For example, it removes URLs with TLDs like 'xyz', 'top' or 'icu'. It removed roughly 2% of the URLs. This returns 981,201 websites in the 1M dataset and this is given as input to FIDOLOGY.

As for the groundtruth, the filter step is applied to the raw results and details are provided in Table 6.4. It can be seen that the dominant limitation is the headless browser blocking. Anti-bot challenges are seen in 2.54% of cases. The post-processing filter step removes 56.99% of the 1M dataset, resulting in 422,037 websites.

Figure 6.3 represents the authentication categories frequency for the 1M dataset results. These numbers are based on the 422,037 rows obtained after the filtering step. It can be observed that the `None/Error` category is the predominant one ($\pm$ 55%). `Password-Based` authentication schemes still accounts for 40%. The `Fido2-Native` category is mostly found for the rest of the websites ($\pm$ 5%).

Disregarding the `None/Error` category, 85% of the URLs are categorized as using a password-based scheme. As the `Password-Extended` category is negligible, it can be said that roughly 15% of websites contain FIDO clues ($N = 20,371$). Specially, Figure 6.4

Barrier	Raw value	Percentage
Anti-bot challenge blocking	24,959	2.54%
Closed shadow DOM	0	0.00%
Headless browser blocking	529,429	53.96%
Iframe opaque	0	0.00%
JS orchestrated auth ui	4,776	0.49%
Password without observability	0	0.00%
Stalled auth ui	0	0.00%
Total	559,164	56.99%

Table 6.4: Results of the filtering step applied to the results from the 1M dataset scraping realized from March 24$^{\text{th}}$ to March 27$^{\text{th}}$, 2026 ($N = 981,201$).

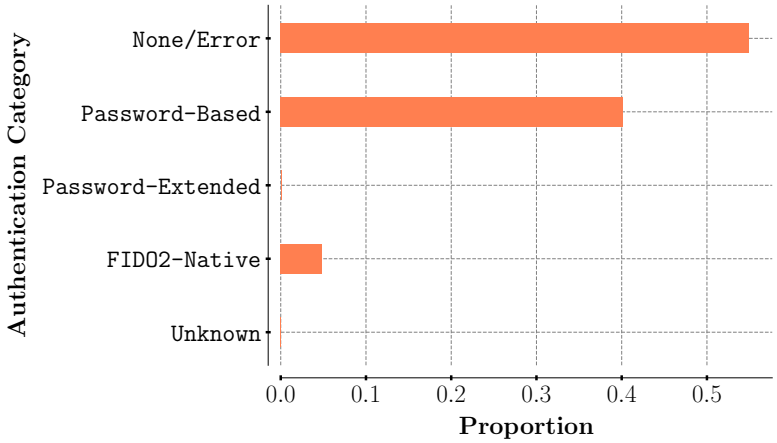


Figure 6.3: Authentication category distribution for the 1M dataset filtered scraping ($N = 422,037$).

displays the detailed distribution for the `Fido2-Native` category. As for the groundtruth, the `latent_support` class (F_1) is the most present one. It indicates that a significant amount of websites of the 1M dataset have integrated FIDO2 infrastructure but without exposing it as a direct authentication option on their login page. Following this is the `storage` class (F_4). The last three, `fido_only_ui` (F_1), `webauthn` (F_5) and `full_fido2` (F_6), are of the same order of magnitude compared to 1 million (400, 286, 361 websites each).

This analysis allows to say that password-based authentication schemes are the dominant visible login mechanism. But, it does not imply that those websites don't implement FIDO at all; it just indicates that a passkey login option is not directly exposed as a login method. Websites that require previous interaction and enrolment to expose FIDO2 authentication are sorted in other categories. Concerning the FIDO2 deployment state in the wild, it should be interpreted as a lower bound, and the real adoption state is in fact even more higher.

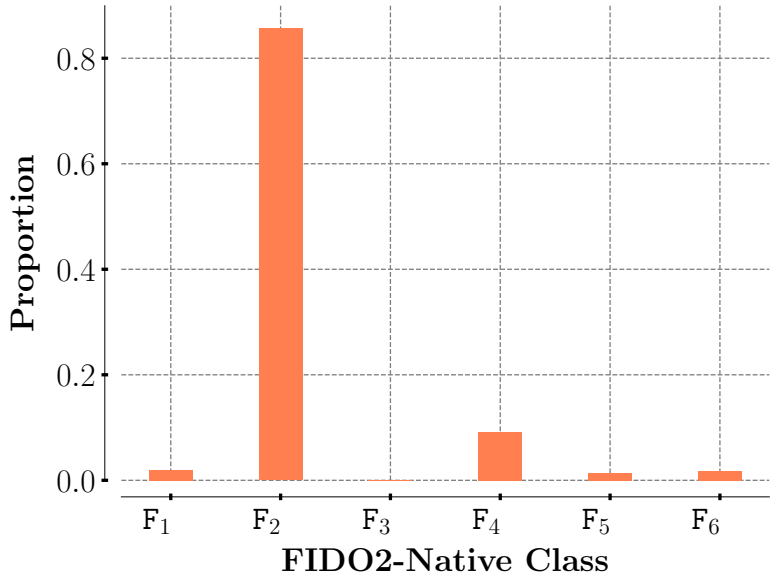


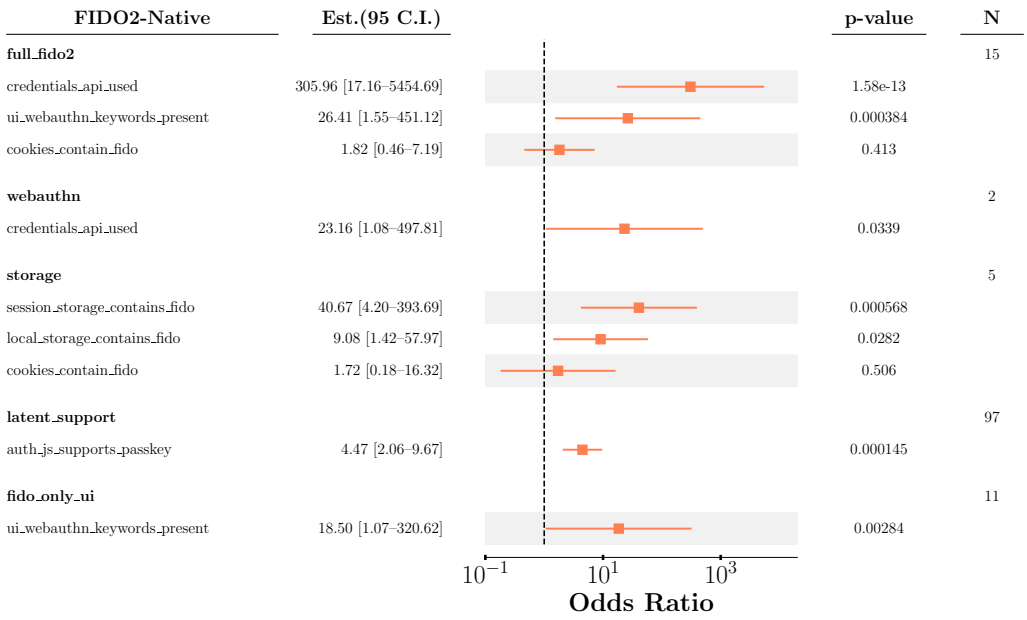
Figure 6.4: Distribution of the `Fido2-Native` classes for the 1M dataset filtered scraping ($N = 20,371$).

6.2.2 Signals

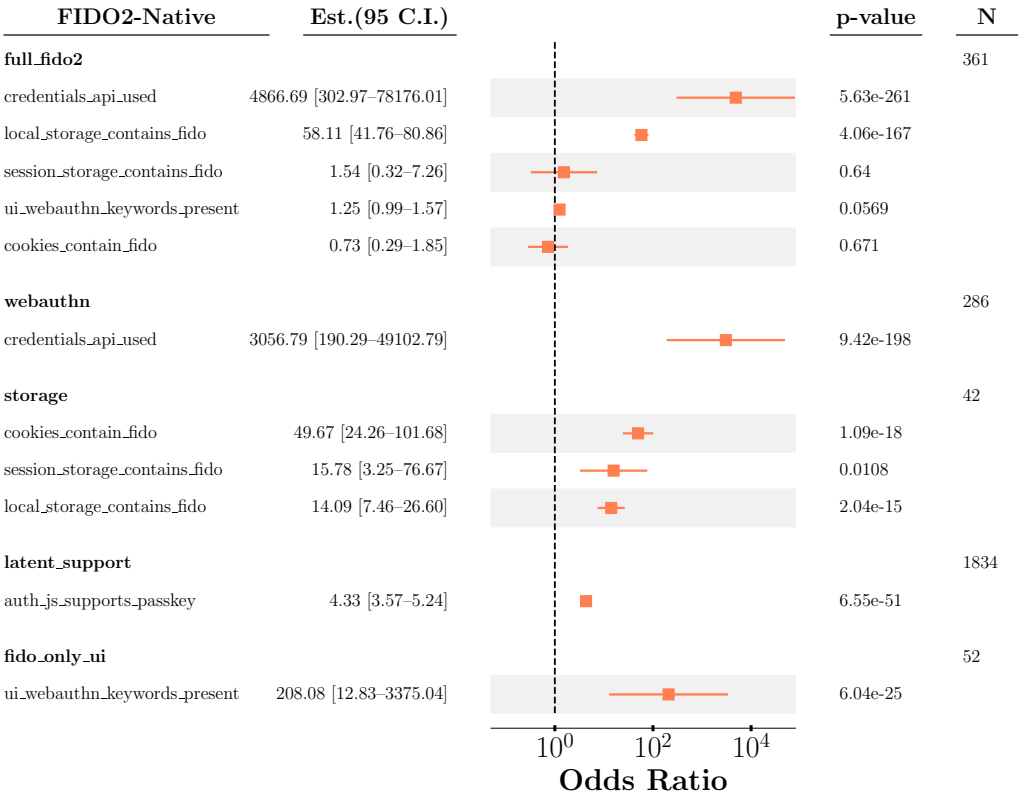
Table 6.5 illustrates the signal frequency for the 1M dataset scraping. For clarity, the percentages are computed over the number of websites in the `Password-Based`, `Password-Extended` and `Fido2-Native` categories (N = 190,270), i.e., disregarding the `None/Error` category. For the technical signals, as expected, the `password_input_present` signal ($\texttt{ST}_5$) is clearly dominant. Also noticed in the groundtruth scraping, the usage of WebAuthn API ($\texttt{ST}_3$) is seen here, although on a smaller scale. Network webauthn ($\texttt{ST}_5$) and password ($\texttt{ST}_6$) indices are only observed a few times.

Concerning the heuristic signals, the same tendencies appear. The storages signals together with UI webauthn clues ($\texttt{SH}_1$, $\texttt{SH}_2$, $\texttt{SH}_3$, $\texttt{SH}_5$) are seen, on average, in 1.12% of the cases. Whereas for `auth_js_supports_passkey` is present for 8.02% of websites, which makes it the second most viewed signal. The synthetic signal is not detected.

In order to compare the contributions of each signal with those found with the groundtruth, two forest plot are represented in Figure 6.5. These are summarized by withdrawing the signals who had a p-value equal to 1, and thus do not have a real impact. To create Figure 6.5b, not all `Fido2-Native` websites were taken into account. As shown with Figure 5.6 in Section 5.5.1, the `webauthn` and `full_fido2` classes have the smallest over classification rate and therefore are pretty trustworthy. All websites in these two classes are directly taken into account. For the four other classes, `fido_only_ui`, `latent_support`, `latent_usage` and `storage`, only websites with a validated FIDO usage are considered. In the end, 2,575 websites are used to compute the forest plot.



(a) Groundtruth summarized forest plot.



(b) 1M dataset summarized forest plot.

Figure 6.5: Forest plot summary comparison between groundtruth ($N = 130$) and 1M dataset scrapings ($N = 2,575$). For the groundtruth, `Fido2-Native` correctly classified websites are taken, while for the 1M dataset, `full_fido2`, `webauthn` websites and other `Fido2-Native` validated ones are taken into account.

Signal	Raw value	Percentage
`password_input_present` (ST_1)	184,881	97.17%
`credentials_api_used` (ST_3)	1,062	0.56%
`network_webauthn` (ST_5)	6	0.00%
`network_password` (ST_6)	293	0.15%
`local_storage_contains_fido` (SH_1)	4,135	2.17%
`session_storage_contains_fido` (SH_2)	823	0.43%
`cookies_contain_fido` (SH_3)	1,132	0.59%
`ui_webauthn_keywords_present` (SH_5)	2,408	1.27%
`auth_js_supports_passkey` (SH_8)	15,261	8.02%

Table 6.5: Signals distribution for the 1M dataset filtered scraping ($N = 422,037$).

Besides the presence, with a p-value different from one, of local and session storage FIDO clues in the `full_fido2` class, the same signal patterns remain. However, sample sizes affect confidence intervals and p-values. Signals that were statistically significant in the groundtruth become here even more important. For example, in the `fido_only_ui` class, the `ui_webauthn_keywords_present` signal had a p-value of 0.003 and reaches now the order of 10^{-25}. For the FIDO2 class, while strong association with the WebAuthn API is preserved, some weaker signals (e.g., `ui_webauthn_keywords_present` and `cookies_contain_fido`) become less consistent. Instead, the Web Storage signals contribute now much more to the class attribution.

To conclude this section about signals distribution, it can be said that this comparison suggests that the large-scale campaign reinforces the main trends observed in the groundtruth, while providing more precise and stable statistics. The most relevant signals are still the usage of WebAuthn API, FIDO2 clues storage and cookies, the presence of keywords in the UI and passkey support option in the JavaScript of the websites. Since most of the websites reveal here only latent support for FIDO2, validation adds confidence in only a small part of `Fido2-Native` websites (10.09%).

6.3 Authentication usage across countries

Figure 6.6 illustrates the proportions of `Password-Based` and `Fido2-Native` authentication mechanisms across countries. To mitigate biases due to small sample size, the statistics are derived using the 15 most represented countries in the 1M scraping filtered results. Password-based authentication obviously dominates in all countries, consistently accounting for more than 85% of the observed login system.

These results suggest that FIDO2 is still in the early stages of adoption. Even in the best case, i.e., in Australia, it accounts for 15% of login systems. While countries such as Australia, the United States of America and Norway have a slightly higher proportion of

`Fido2-Native` websites, the differences between countries remain limited. For instance, Australia shows a proportion of FIDO2 usage roughly three times higher than Japan. However, in absolute terms, adoption remains marginal compared to password-based authentication methods.

At the continental level, this shows that Asia is lagging slightly behind in terms of FIDO adoption. South America and Europe show roughly the same percentage (8.9%), whilst North America and Oceania top the rankings with 13.6% and 15.4% respectively (even if the size of Oceania can add some bias). Nevertheless, variances remain small and suggest that geographical differences still have a minor impact compared to the predominance of password-based authentication systems.

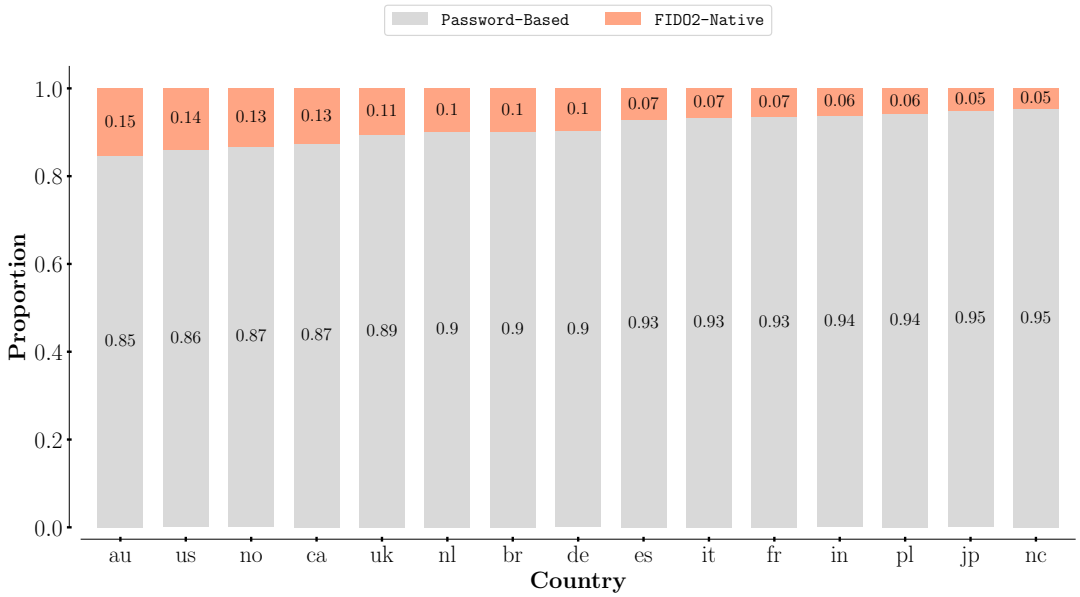


Figure 6.6: Distribution between `Password-Based` and `Fido2-Native` websites for the 15 most influent countries of the 1M scraping filtered results and ranked by FIDO2 adoption rate ($N = 142,056$).

6.4 Authentication usage across IAB categories

Figure 6.7 represents the distribution of IAB categories between the 15 most present ones in the 1M scraping filtered results. In terms of numbers, the same trends can be observed as for countries. The categories IAB 19 ('Technology and Computing') and IAB 13 ('Personal Finance') are in the Top 3. As indicated in section 5.2.1, these sectors have a strong interest in having secure authentication mechanisms. There is also a small decline in FIDO2 adoption along the graph, suggesting that the adoption rate is uniformly low, with no single category standing out significantly.

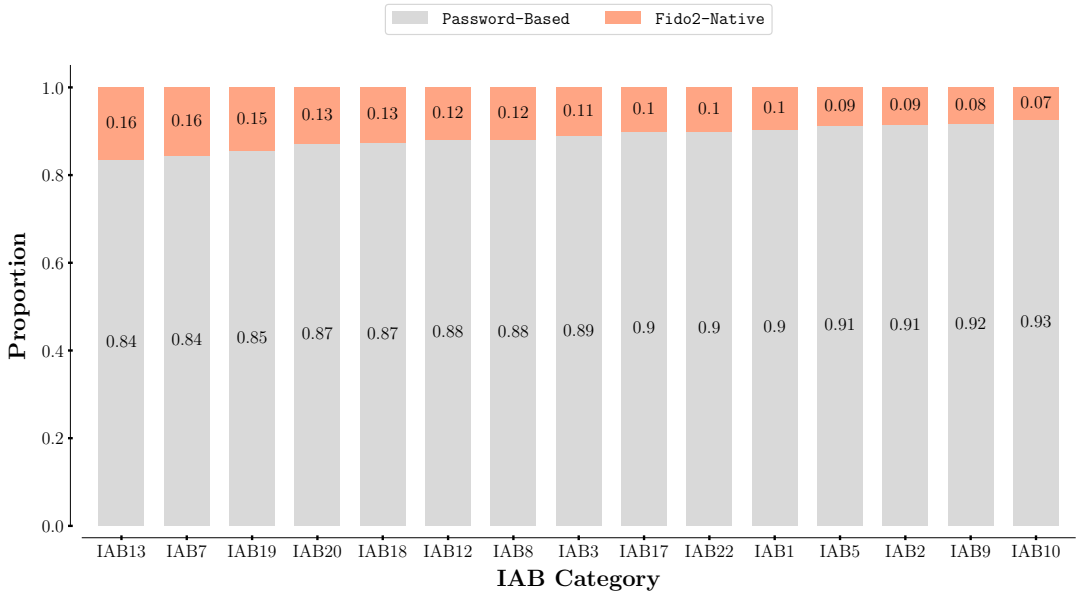
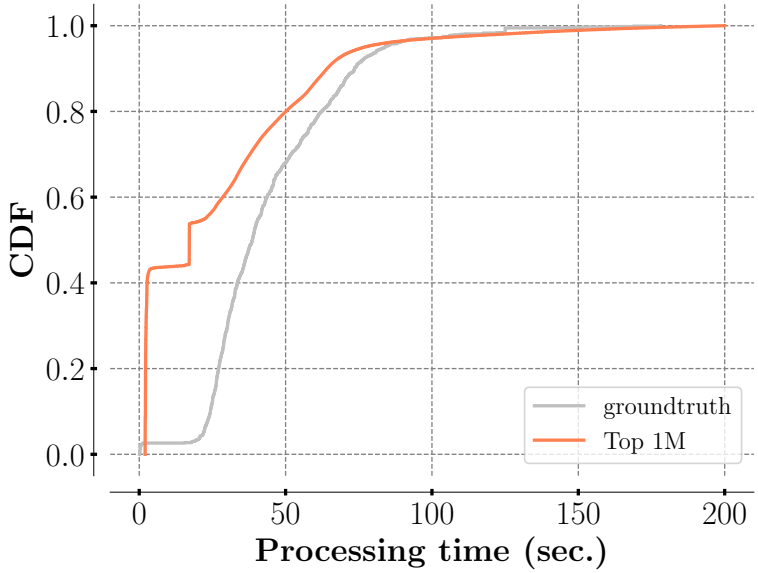


Figure 6.7: Distribution between `Password-Based` and `Fido2-Native` websites for the 15 most influent IAB categories of the 1M scraping filtered results and ranked by FIDO2 adoption rate ($N = 10,000$).

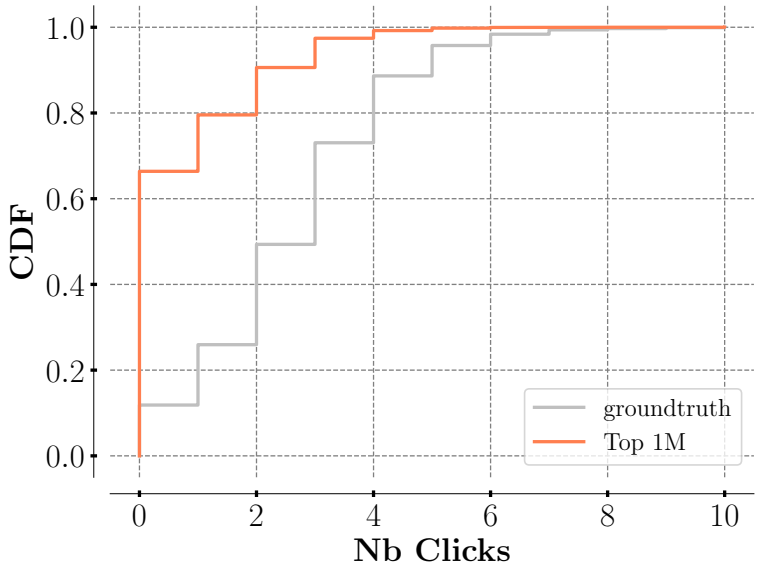
6.5 Ethical considerations

Figure 6.8a represents the two CDFs of the processing time in seconds for the raw results of the groundtruth scraping of March 23rd and of the 1M dataset scraping. The mean website scraping time is 44.9 seconds for the groundtruth and 29.4 seconds for the 1M dataset. This respects the ethical considerations of FIDOLOGY described in Section 2.2.7. All stages of signal collection are protected by a timeout mechanism to prevent deadlocks and ensure that the system does not remain stuck indefinitely at any given stage. When the website supports FIDO2 functionality, the steps are not interrupted by time-outs, as the signals are effectively detected. For the websites were it took less that 3 seconds, it is due to an HTTP2 protocol error, which can be raised by anti-bot detection solutions (see Section 4.4.4).

Figure 6.8b shows the cumulative distribution functions of the number of clicks recorded during the scraping of the groundtruth of March 23rd and the 1M dataset scraping. The mean number is 2.6 clicks for the groundtruth and 0.67 for the 1M dataset. The number of clicks is gradually increasing. It appears that in the high majority of cases in which there is no clicks is when no clues are detected (i.e., the `None/Error` category). This happens when a navigation error occurs at the beginning of the analysis or when no clickable elements are found in the user interface, preventing the authentication pages from being found.



(a) CDF of processing time in seconds for scraping results of the groundtruth and the 1M dataset.



(b) CDF of number of clicks for scraping results of the groundtruth and the 1M dataset.

Figure 6.8: Ethics analysis for the scraping raw results of the groundtruth ($N = 993$) and the 1M dataset ($N = 981,201$).

CHAPTER 7

FIDO2 SECURITY ANALYSIS

This chapter has for objective to assess the security level of implementations of FIDO2/WebAuthn in the real world. Even if there exists clear specifications and requirements, these are not always correctly taken into account, or on the contrary, they can be implemented with care. To accomplish this task, data returned by servers during WebAuthn flow need to be captured. This chapter outlines the methodology used to collect this information (Section 7.1) and analyse the results in Section 7.2, addressing the challenges, policies and ethical issues involved.

7.1 Methodology

To carry out this analysis, it was obviously necessary to use websites from the `Fido2-Native` category. However, in order to optimise the results of this step, only the websites providing the most reliable evidence of passwordless authentication were taken into account. This resulted in the same set of 2,575 websites as the one used previously in Section 6.2.2.

A new non-intrusive probing step is implemented in FIDOLOGY to perform the analysis on that set. FIDOLOGY first tries to navigate to the authentication interface and to click a visible passkey or WebAuthn button when it is available. The objective is to trigger a WebAuthn ceremony. In successful cases, the `PublicKeyCredentialRequestOptions` dictionary returned by the server is captured. From that dictionary can be extracted the challenge, and other elements like the `userVerification` flag.

A common mistake is to treat the challenge as an identifier between sessions rather than a one-time randomly generated nonce. In order to simplify the process and API implementations, some developers use static or predictable challenges, but that method allows replay attacks [98]. To inspect that, the challenge capture step is repeated five times for each website. The process was launched with one machine with the same hardware configuration described in Section 6.1.3 and results were collected on April 1^{st}, 2026.

After capturing the dictionary containing the challenge, the results are extracted and used to assess the quality of the FIDO2 implementations. To accomplish this, an overall score (between 0 and 100) and a security level is attributed to each website. Four classes are distinguished for the security level: `very weak`, `weak`, `moderate`, `strong`. These are derived based on four metrics from which an intermediate score is computed: `userVerification` (`uv_score`), attestation mode (`att_score`), challenge entropy (`entropy_score`) and uniqueness (`challenge_score`). For the `userVerification` flag and attestation mode, the following dictionaries are used:

```
{'required': 1.0, 'preferred': 0.5, 'discouraged': 0.0}
{'direct': 1.0, 'indirect': 0.7, 'none': 0.0}
```

For the challenge uniqueness, the `challenge_score` is either 1 or 0 based on its boolean value. Just as a reminder, challenge uniqueness is True if a different challenge was sent each time during the capture step, and False if challenge reuse was detected. The challenge entropy is divided by 256, and the greatest value between that and one is taken into account. The overall score is then computed with the formula 7.1. Finally, the security level corresponds to one of the four classes based on the overall score computed. The score ranges are presented in Table 7.1.

$$\begin{aligned} \text{overall\_score} &= \text{entropy\_score} \cdot 40 \\ &+ \text{uv\_score} \cdot 25 \\ &+ \text{att\_score} \cdot 15 \\ &+ \text{challenge\_score} \cdot 20 \end{aligned} \tag{7.1}$$

Overall score	Security level	Comment
≥ 80	`strong`	Strong FIDO2 implementation, pushing security beyond requirements
≥ 60	`moderate`	Configuration is in line with the specifications
≥ 40	`weak`	Takes into account the requirements but gray areas remain
< 40	`very weak`	FIDO2 specifications not fulfilled

Table 7.1: Security level mapping based on the value of the overall score of a website computed during the capture step of FIDOLOGY.

7.2 Results

The results are better than expected. Of the 2,575 websites, 1,011 responded correctly. FIDOLOGY was unable to collect the challenges in 61% of cases. In 2.36% of them, this was due to an error occurring while navigating to the website. As a WebAuthn flow is attempted, websites may sometimes expect user interaction after the initial click on a passkey button. This latter situation occurred in 428 cases. In 1.78% of websites, an anti-bot is identified. As for the other analysis errors, they may be due to the same limitations identified previously (see Section 5.4).

7.2.1 Challenge analysis

Figure 7.1 illustrates the security level of the challenges captured during this last FIDOLOGY step. It represents the effective entropy (in bits) against the challenge length (in bytes). The captured challenges represented by dots are colour-coded depending on the `userVerification` option value. Two limits concerning thresholds for effective entropy and challenge length are illustrated on the plot. The minimum effective entropy needed to meet the requirements is of 128 bits and the minimum length is of 16 bytes [98].

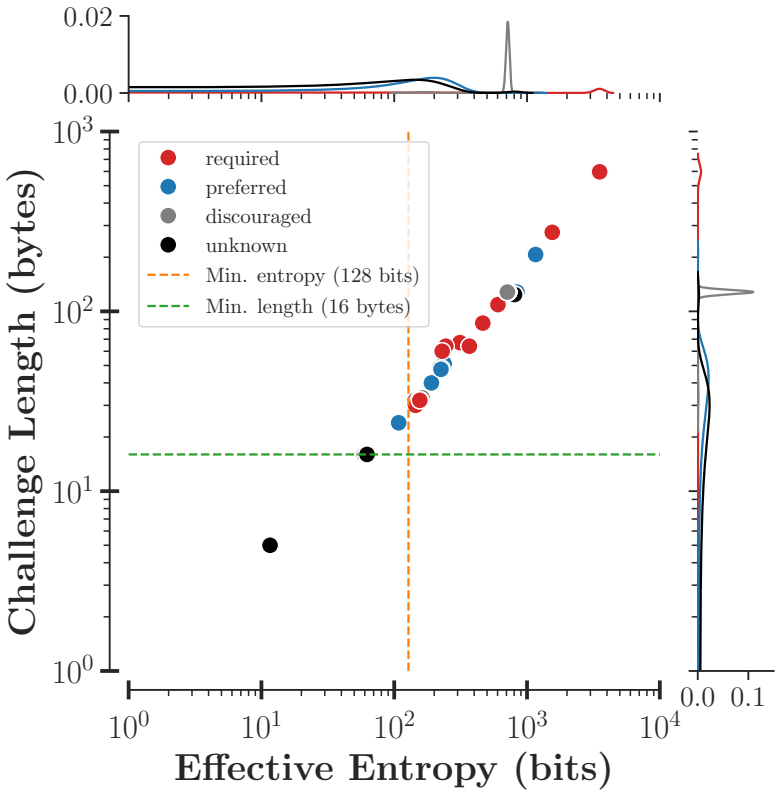


Figure 7.1: Security level of FIDO2 cryptographic challenges sent by the servers ($N = 1,011$). User-verification is colour-coded and requirements limits of entropy and length are illustrated in orange (--) and green (--).

Overall, the statistics are very promising. The average challenge length is 379.14 bytes and the average effective entropy is 2,216.79 bits. This figure reveals that 98% of challenges fall in the upper-right quadrant. This shows that most websites respect the security requirements concerning the FIDO2/WebAuthn challenges. In particular, 577 websites push security well beyond the minimum requirements. These have an effective entropy higher than 3,500 bits and more than 519 bytes for the length of challenges.

The remaining 2% fall below one or both thresholds, indicating that the implementations do not meet the requirements of the FIDO2 specifications. Of these, 19 websites fail to meet the requirements for effective entropy while the length of their challenges is 16 bytes or slightly more. This means that the challenges used are not sufficiently random and they therefore could be guessed more easily. In particular, one website has a challenge length of 5 bytes and an effective entropy value of 11.61. That exhibits very poor implementation of FIDO2.

Table 7.2 presents statistics on the capture step. During the five probing attempts, 52 websites returned only a single challenge, i.e., one attempt was successful and the other four were rejected. An attempt can be rejected by a server if it has some anti-bot protection to block repeated access and requests, like rate limits implemented [21], as seen in Section 5.4 about request filtering. Overall, when only one challenge is captured, it prevents any reuse analysis. Given that, 959 websites returned at least two independent challenges and are used to continue this analysis.

As noted in section 3.3.4, challenges should under no circumstances be reused between different authentication attempts. It is observed that 96.04% of websites do not reuse challenges from one session to the next, meaning that challenge reuse is detected in 3.96% of cases. These reused challenges are exactly the same, i.e., the Hamming distance between at least two independent challenges issued by the same server is 0. This already constitutes a breach of the specifications, as challenges must be randomly generated, single-use nonces and must not be reused between sessions. Fortunately, timestamp patterns are not detected in this dataset. Given that challenges should be generated using a stateless scheme incorporating embedded encrypted information, this practice would have suggested deterministic and predictable challenge generation [29, 47].

Given that some websites reuse challenges, employ some with the same structure, or use completely different challenges, it may be useful to analyse the average Hamming distance. Figure 7.2 shows a histogram of the average Hamming distance between the five consecutive challenges received from the servers when the WebAuthn process is triggered. The average hamming distances clearly fall in two clusters. The first is dominated by an average of 326.57 bits and contains the lowest values corresponding to exact reused of challenge cases highlighted above. On the contrary, the second cluster contains only websites that have unique challenges and shows distance values around 1,150 bits. As expected, all cases of this second cluster exhibits challenge length and effective entropy that are above the thresholds, showing a high quality deployment of FIDO2.

Metric	Raw value	Percentage
Successful captured websites ($N = 1011$)		
Total	1011	100%
Subset: $\geq$ 2 successful attempts ($N = 959$)		
Total	959	100%
Challenge reuse detected	38	3.96%
Timestamp pattern detected	0	0.00%

Table 7.2: Observation statistics about WebAuthn/FIDO2 challenge reuse. Percentages in each section are computed with respect to the total of that section. A successful attempt occurs when a triggered server returns a challenge.

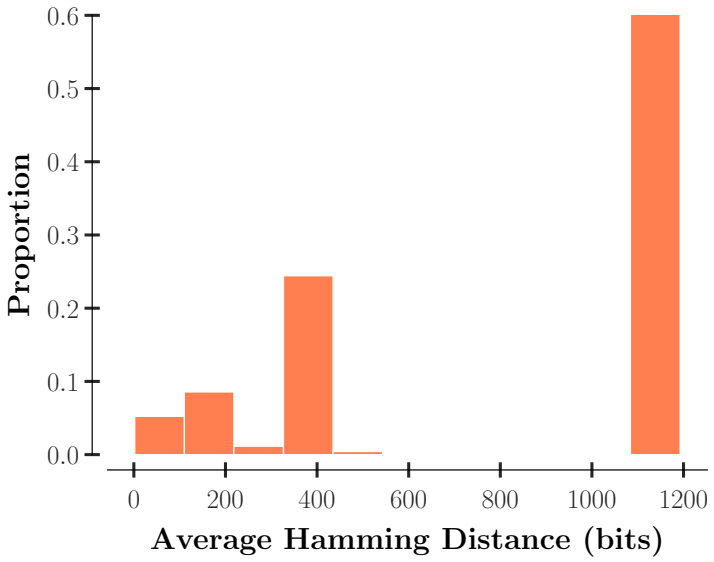


Figure 7.2: Histogram of the average hamming distance expressed in bits between the 5 challenges received by servers during WebAuthn authentication flow trigger ($N = 959$). Websites that did not returned more than one challenge are excluded.

7.2.2 WebAuthn policy analysis

The `userVerification` flag set by each website is extracted by FIDOLOGY during this capture step from the `PublicKeyCredentialRequestOptions` dictionary. As a reminder, the `userVerification` parameter is optional; its default value is *preferred*, and it can also be set to *required* or *discouraged* (see Section 3.3.1). First, the `userVerification` flag is the same across all 5 attempts for a given website. This indicates that servers do not change this option based on the user interaction. For example, a website could decide to change that flag if an authentication flow is attempted and failed several times in a row. If a connection seams malicious and a website have a *preferred* `userVerification` flag, it could pass to *required* in order to increase security.

Second, the Table 7.3 illustrates the distribution of `userVerification` values across the 1,011 websites. The dominant setting is a *required* `userVerification` (66.36%). From a security point of view, this is the best option and it ensures that it is the same user that is consistently performing the authentication ceremonies [27]. As several studies [56, 49] suggested that the `userVerification` can be a barrier to passkey adoption, the *discouraged* value is seen in 23.54% of cases. Using this setting allows to ease the experience for users, whereas it decreases the security level offered by the FIDO2 passwordless scheme. The *preferred* value is seen less often that the other 2, meaning that in only 6.72% of cases, the choice of `userVerification` depends on the user's preferences. As this flag is optional, the value is omitted in 34 websites, leaving the security policy undefined and relying solely on authenticator defaults.

`userVerification` value	Raw value	Percentage
required	671	66.36%
preferred	68	6.72%
discouraged	238	23.54%
unknown	34	3.36%

Table 7.3: Distribution of the value of the `userVerification` flag used across the authentication data returned by the websites captured by FIDOLOGY ($N = 1,011$).

A final analysis of `userVerification` reveals a correlation with the entropy requirements and Hamming distances outlined above. Of the 577 websites with an effective entropy greater than 3,000 bits, none have a `userVerification` field other than *required*, and each of them has a Hamming distance belonging to the second group, i.e., an average Hamming distance greater than 1,000 bits. This means that all websites with a `userVerification` parameter of type *discouraged*, *preferred* or *unknown* have a lower Hamming distance and less robust FIDO2 implementations.

Another field included in the returned dictionary is the relying party identifier (RP ID). As this field is also optional, 31.05% of websites did not provide a trusted party identifier during configuration. One relying party stands out in particular: login.microsoft.com. It appears in 57.17% of cases. It is therefore considered the most commonly used RP for delegating the authentication flow. The second most frequent relying party is GitHub, used by 29 websites. In the end, there are 90 instances where websites use their own implementation or delegate their authentication procedures to a less popular trusted party.

The final field of interest is the timeout parameter, which is also an optional element. Just like the `userVerification`, this field is always identical across the five captured dictionaries, when provided. Table 7.4 shows all the observed timeouts and their proportions. The W3C specifications [98] recommend a range of between 300,000 and 600,000 milliseconds, with a default value of 5 minutes, meaning that 59.03% of websites comply

with this recommendation. The two most common values are 60,000 and 600,000 milliseconds. Consequently, in most cases, a user has between 1 and 10 minutes to complete the authentication process before the relying party stops waiting. The RP Microsoft highlighted above uses the 600,000 milliseconds setting. Thus, it finds itself in the top of the range, which indicates that they try to smooth the user experience by giving them as more time as possible, while being in agreement with the requirements.

Timeout value (in milliseconds)	Raw value	Percentage
None	66	6.53%
6,000 - 60,000	294	29.06%
120,000 - 240,000	19	1.86%
300,000	16	1.57%
600,000	581	57.46%
$>$ 600,000	34	3.35%

Table 7.4: Distribution of the timeout values used across the authentication data returned by the websites captured by FIDOLOGY ($N = 1,011$).

7.2.3 Ethical considerations

The processing time for each website is a bit similar to the one observed in the 1M dataset scraping and is illustrated by a CDF in Figure 7.3. It can be observed that for successful captures, the processing time is, for the majority of the websites, between 100 and 125 seconds. Half of the failed captured did not last more than 50 seconds. For the number of clicks, it is either 0 or 5, depending on the success of the capture step. For the 1,011 successful websites, the number of clicks is equal to 5; for the unsuccessful ones ($= 1,564$), the number is 0.

7.2.4 Overall security level

Figure 7.4 represents the overall scores across the four different security levels for the 1,011 successfully captured websites and listed in descending order of frequency. It can be said that the `moderate` level is the most observed one, closely followed by the `weak` level. These two classes have noticeable outliers, indicating variations from the requirements in the corresponding implementations. For the `very weak` level, the outliers are more appearing above the mean, with one outlier with a really low score. This outlier is the one mentioned earlier in Section 7.2.1, which is characterised by very low entropy and a very short challenge length. For the `strong` security level class, the variance is very low, indicating that implementations have a consistent high quality.

To summarize, this section has revealed the variations in the security levels of the implementations. In most cases, the challenges used meet expectations, although some websites use challenges that are too short or lack sufficient entropy. Robust practices

may be accompanied by a `userVerification` flag set to *required*, but many *discouraged* practices have been identified, which reduces the overall security of the login flow. This suggests that while FIDO2/WebAuthn appears robust in practice, both security quality and deployment remain heterogeneous across websites.

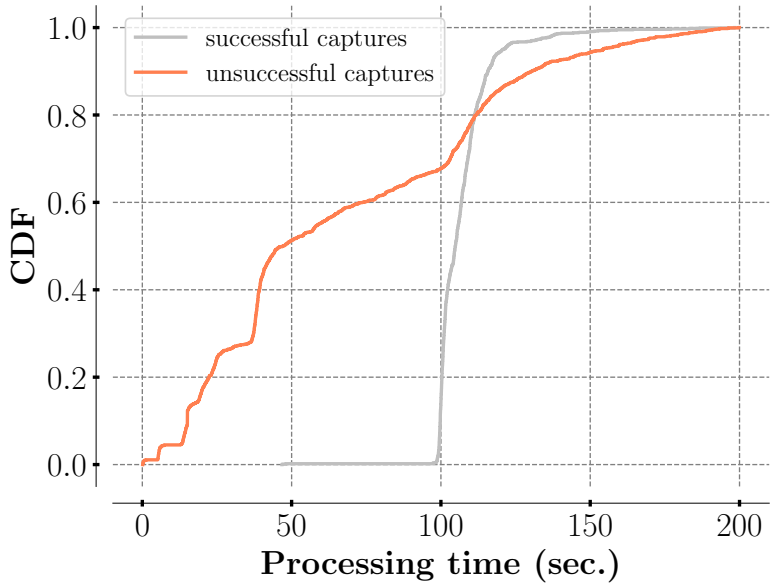


Figure 7.3: CDF of processing time in seconds for capture results of the capture dataset ($N = 2,575$).

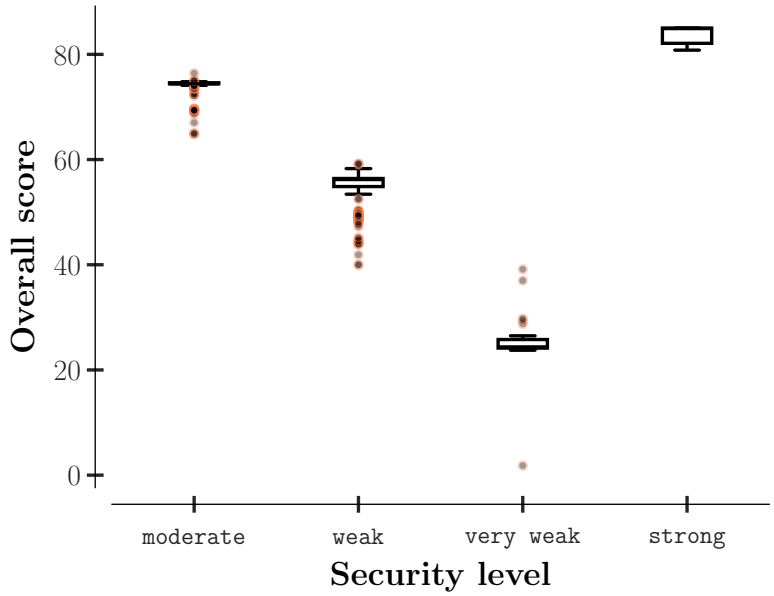


Figure 7.4: Overall score across the four security classes obtained from the FIDOLOGY capture step for the successfully captured websites and listed in descending order of frequency ($N = 1,011$).

CHAPTER 8

CONCLUSION

This work introduced a tool called FIDOLOGY. It is used for identifying and categorizing the various authentication mechanisms implemented by websites. This identification is carried out in the most deterministic manner possible, using non-intrusive methods and combining headless-browser automation with DOM, shadow DOM, storage, network request and runtime API analysis in order to highlight multi-layered signal inference.

FIDOLOGY was initially tested against a manually build groundtruth of 1,000 websites. It revealed that, in most cases, the classification was correct, with a tendency to under-classify websites rather than over-classify them. The `Password-Based` category had the highest rate of correct classification, while `None/Error` had the highest incorrectly categorized rate, highlighting navigation errors or authentication mechanisms that are only partially exposed to users. An important fraction of correctly identified websites in the `Fido2-Native` category can be validated as well using passkey interaction.

Furthermore, this tool has been applied in the wild using a dataset of 1,000,000 websites. This suggested that the `Password-Based` category remains the dominant authentication mechanism. However, multiple signals of FIDO2/WebAuthn implementations have been identified, from being only in latent forms to revealing full FIDO usages. The proportion of websites having a passkey deployment, and exposed here as `full_fido2`, remains, for now, a small part of the web and should be interpreted as a lower bound.

Finally, an analysis of FIDO2/WebAuthn challenges and implementations has been carried out and highlights that the majority of websites satisfy the requirements of the FIDO2 specifications, with some of them clearly exceeding expectations. An issue requiring attention has been discovered and reveals websites that do not respect the requirements, specially with discouraged user verification and reused challenges between sessions, which create an open alley to replay attacks.

8.1 Future works

The confidence scoring mechanism was implemented but did not produce the expected results, and was therefore not exploited as much as wanted. They were used to highlight the score of over-classified websites, but generally speaking, correctly and incorrectly classified websites yield the same confidence scores. They should be rethought, in order for them to better highlight the ambiguous signals and categorization confidence. The weights of the various signals could be tuned with help of machine learning to achieve better results and greater accuracy.

Continuing the implementation of the login page detection tool could greatly benefit the misclassification rate of the `None/Error` category. The amelioration of this tool would facilitate the analysis of the target login page, increase the number of classified websites and, at the same time, improve confidence scores, as it would enable the analysed page to be assessed with a high degree of certainty.

FIDOLOGY can be useful in order to create a relatively large dataset of websites implementing FIDO2/WebAuthn, which could benefit further researches on this passwordless authentication mechanism. As example, this could be used to study the usage rate of passkeys [22] and to extend the FIDO security analysis to a larger number of websites.

This latter analysis could be extended to assess actual security levels more precisely, for example by examining challenge handling across browsers, conducting an in-depth study of the most widely used relying party providers, such as Microsoft, or analysing the challenge exchange process to determine whether it is dependent on client behaviour and whether this could compromise the security of the protocol. The temporal stability of cryptographic practices across deployments could also be investigated.

BIBLIOGRAPHY

[1] 1Password. *Passkeys.directory*. URL: https://passkeys.directory/ [Last Accessed: November 5th, 2025].

[2] A. M. Aburbeian and M. Fernández-Veiga. "Secure Internet Financial Transactions: A Framework Integrating Multi-Factor Authentication and Machine Learning". In: *AI* 5.1 (2024), pp. 177–194. ISSN: 2673-2688. DOI: 10.3390/ai5010010. URL: https://www.mdpi.com/2673-2688/5/1/10.

[3] Fido Alliance. *Passkey Directory*. URL: https://fidoalliance.org/passkeys-directory/ [Last Accessed: October 25th, 2025].

[4] FIDO alliance. *FIDO Functional Certification: Authenticators/Clients*. URL: https://fidoalliance.org/certification/functional-certification/ [Last Accessed: April 9th, 2026].

[5] A. Angelogianni, I. Politis, and C. Xenakis. "How many FIDO protocols are needed? Analysing the technology, security and compliance". In: *ACM Comput. Surv.* 56 (2024). ISSN: 0360-0300. DOI: 10.1145/3654661.

[6] Auth0. *Web Authentication*. URL: https://www.webauthn.me/introduction [Last Accessed: December 4th, 2025].

[7] M. Barbosa et al. "Provable Security Analysis of FIDO2". In: *Proceedings of 4the 1st Annual International Cryptology Conference*. (2021). DOI: 10.1007/978-3-030-84252-9_5.

[8] J. Berrios et al. "Factorizing 2FA: Forensic analysis of two-factor authentication applications". In: *Forensic Science International: Digital Investigation* 45 (2023), p. 301569. ISSN: 2666-2817. DOI: https://doi.org/10.1016/j.fsidi.2023.301569. URL: https://www.sciencedirect.com/science/article/pii/S2666281723000781.

[9] P. Bhardwaj and N. Sastry. "State of Passkey Authentication in the Wild: A Census of the Top 100K Sites". In: *Proceedings of the 27th International Conference on Passive and Active Measurement, PAM 2026*. (2026). DOI: arXiv:2602.15135v1.

[10] N. Bindel, C. Cremers, and M. Zhao. "FIDO2, CTAP 2.1, and WebAuthn 2: Provable Security and Post-Quantum Instantiation". In: *2023 IEEE Symposium on Security and Privacy (SP)* (2023). DOI: 10.1109/SP46215.2023.10179454.

[11] Christiaan Brand. *Client to Authenticator Protocol (CTAP).* URL: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.pdf [Last Accessed: December 6th, 2025].

[12] Samuel Brown. *Why Passwords Are Insecure and What You Can Do about It.* URL: https://www.pingidentity.com/en/resources/blog/post/what-you-can-do-about-insecure-passwords.html [Last Accessed: November 24th, 2025].

[13] Git issues chat. *Should clients enforce challenge length?* URL: https://github.com/w3c/webauthn/issues/1115 [Last Accessed: April 11th, 2026].

[14] StackOverflow chat. *Why time-based nonce should be avoided?* URL: https://stackoverflow.com/questions/42643421/why-time-based-nonce-should-be-avoided [Last Accessed: April 16th, 2026].

[15] Google Chrome. *Chrome (CrUX) Top Million Websites.* URL: https://github.com/zakird/crux-top-lists [Last Accessed: October 1st, 2025].

[16] Vinod Chugani. *Ethical Web Scraping: Principles and Practices.* URL: https://www.datacamp.com/blog/ethical-web-scraping [Last Accessed: February 12th, 2026].

[17] Cisco. *Cisco Popularity List.* URL: https://umbrella-static.s3-us-west-1.amazonaws.com/index.html [Last Accessed: October 1st, 2025].

[18] Cloudflare. *Cloudflare Turnstile.* URL: https://developers.cloudflare.com/turnstile/ [Last Accessed: March 11th, 2026].

[19] Cloudflare. *Cloudflare Turnstile: A verification tool to replace CAPTCHAs.* URL: https://www.cloudflare.com/application-services/products/turnstile/ [Last Accessed: March 11th, 2026].

[20] Cloudflare. *Cloudflare Web Application Firewall.* URL: https://developers.cloudflare.com/waf/ [Last Accessed: March 11th, 2026].

[21] Cloudflare. *Rate limiting rules.* URL: https://developers.cloudflare.com/waf/rate-limiting-rules/ [Last Accessed: March 11th, 2026].

[22] Corbado. *Passkey Usage Rate.* URL: https://www.corbado.com/kpi/passkey-usage-rate [Last Accessed: May 4th, 2026].

[23] C. Daniela. *How many websites are there? Key statistics and growth trends.* URL: https://www.hostinger.com/tutorials/how-many-websites-are-there [Last Accessed: April 23th, 2026].

[24] A. Das et al. "The Tangled Web of Password Reuse". In: *NDSS Symposium 2014* (2014). DOI: 10.14722/ndss.2014.23357.

[25] O. Dastane. "The Effect of Bad Password Habits on Personal Data Breach". In: *International Journal of Emerging Trends in Engineering Research, Volume 8. No. 10* (2020). URL: https://ssrn.com/abstract=3716898.

[26] DeepL. *DeepL*. URL: https://www.deepl.com/fr/translator [Last Accessed: April 1st, 2026].

[27] Vincent Delitz. *What is User Verification in WebAuthn ?* URL: https://www.corbado.com/glossary/user-verification [Last Accessed: April 20th, 2026].

[28] Python Docs. *Multiprocessing*. URL: https://docs.python.org/3/library/multiprocessing.html [Last Accessed: April 3rd, 2026].

[29] Dolda. *How to properly manage WebAuthn challenges?* URL: https://security.stackexchange.com/questions/268308/how-to-properly-manage-webauthn-challenges [Last Accessed: April 16th, 2026].

[30] Kameleo Engineers. *The Best Headless Chrome Browser for Bypassing Anti-Bot Systems*. URL: https://kameleo.io/blog/the-best-headless-chrome-browser-for-bypassing-anti-bot-systems [Last Accessed: March 11th, 2026].

[31] Europol. *The SIM highjackers: how criminals are stealing millions by highjacking phone numbers*. URL: https://www.europol.europa.eu/media-press/newsroom/news/sim-highjackers-how-criminals-are-stealing-millions-highjacking-phone-numbers [Last Accessed: April 9th, 2026].

[32] Crypto Stack Exchange. *Is it safe to use a timestamp to build a nonce?* URL: https://crypto.stackexchange.com/questions/61369/is-it-safe-to-use-a-timestamp-to-build-a-nonce [Last Accessed: April 16th, 2026].

[33] F. M. Farke et al. ""You still use the password after all" – Exploring FIDO2 Security Keys in a Small Company". In: *Proceedings of Sixteenth Symposium on Usable Privacy and Security* (2020).

[34] Frontegg. *FIDO2 vs U2F: 5 Key Differences, Pros/Cons, and How to Choose*. URL: https://frontegg.com/blog/fido2-vs-u2f [Last Accessed: December 7th, 2025].

[35] Rachel Furtado. *How to Fix an ERR_EMPTY_RESPONSE Error?* URL: https://www.vodien.com/learn/how-to-fix-an-err_empty_response-error/ [Last Accessed: March 12th, 2026].

[36] Galileo. *Mistral Medium 2508 Overview*. URL: https://galileo.ai/model-hub/mistral-medium-2508-overview [Last Accessed: April 18th, 2026].

[37] Google. *Gemini 2.5 Flash-Lite*. URL: https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash-lite?hl=fr [Last Accessed: November 4th, 2025].

[38] Filip Grebowski and Daniel Bass. *Cookies vs. Local Storage: What's the Difference? When and Where to Use Each?* URL: https://www.permit.io/blog/cookies-vs-local-storage [Last Accessed: March 13th, 2026].

[39] Jing Gu. *FIDO2 vs. U2F: What's the Difference?* URL: https://www.beyondidentity.com/resource/fido2-vs-u2f-whats-the-difference [Last Accessed: March 14th, 2026].

[40] J. Guan et al. "A Formal Analysis of the FIDO2 Protocols". In: *Proceedings of the 27th European Symposium on Research in Computer Security Copenhagen* (2022). DOI: 10.1007/978-3-031-17143-7_1.

[41] J. Henno, H. Jaakkola, and J. Mäkelä. "Non-determinism in nowadays computing and IT". In: *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. 2020, pp. 794–801. DOI: 10.23919/MIPRO48935.2020.9245377.

[42] Hideez. *Services and Applications Supporting FIDO2/U2F Passwordless Authentication and Passkeys*. URL: https://hideez.com/en-fr/pages/supported-services?srsltid=AfmBOoqZSTv1JRrgpoamjtatowiqFVZsxFMeBT1YsQp_1Lv78a-CMzw [Last Accessed: October 26th, 2025].

[43] E. Huseynov. "User Behaviour in Choosing PIN Codes for FIDO2 Security Keys: An Empirical Study and a Market Research". In: *2024 6th International Communication Engineering and Cloud Computing Conference (CECCC)* (2024), pp. 55–58. DOI: 10.1109/CECCC62598.2024.11063569.

[44] InstaTunnel. *The WebAuthn Loop: Common Logic Flaws in the "Passwordless" Handshake*. URL: https://medium.com/@instatunnel/the-webauthn-loop-common-logic-flaws-in-the-passwordless-handshake-017065517f83 [Last Accessed: April 11th, 2026].

[45] InWord. *Mistral Small 2603*. URL: https://inworld.ai/models/mistral-mistral-small-2603 [Last Accessed: April 18th, 2026].

[46] IONOS. *CTAP : protocole pour plus de sécurité et de confort sur Internet*. URL: https://www.ionos.fr/digitalguide/serveur/securite/client-to-authenticator-protocol-ctap/ [Last Accessed: December 6th, 2025].

[47] WebAuthn Issue. *Clarify the need for truly randomly generated challenges (aka challenge callback issue)*. URL: https://github.com/w3c/webauthn/issues/1856 [Last Accessed: April 16th, 2026].

[48] Emily Johnson. *Ethical Web Scraping: Best Practices and Legal Considerations*. URL: https://www.secureitworld.com/article/ethical-web-scraping-best-practices-and-legal-considerations/ [Last Accessed: February 12th, 2026].

[49] M. Kepkowski et al. "Challenges with Passwordless FIDO2 in an Enterprise Setting: A Usability Study". In: *Proceedings of the 2023 IEEE Secure Development Conference (SecDev)* (2023). DOI: 10.1109/SecDev56634.2023.00017.

[50] Eiji Kitamura. *userVerification deep dive*. URL: https://web.dev/articles/webauthn-user-verification [Last Accessed: April 20th, 2026].

[51] Kavindu Kokila. *Why I Don't Use LocalStorage for Tokens — And What I Do Instead*. URL: https://javascript.plainenglish.io/why-i-dont-use-localstorage-for-tokens-and-what-i-do-instead-0f61b943a87f [Last Accessed: March 13th, 2026].

[52] R. Kondakrindi. "Is FIDO2 Passwordless Authentication a Hype or for Real?: A Position Paper". In: *Proceedings of the 15th International Conference on Information Security and Cryptography (ISCTURKEY)* (2022). DOI: 10.1109/ISCTURKEY56345.2022.9931832.

[53] R. Kondakrindi. "FIDO2: a new era in secure web authentication". In: *International Journal of Computer Engineering and Technology (IJCET)* 15.4 (2024), pp. 841–858. URL: https://openlibindex.com/index.php/IJCET/article/view/IJCET_15_04_074.

[54] D. Kuchhal et al. "Evaluating the Security Posture ofReal-World FIDO2 Deployments". In: *Proceedings of the 2023 ACM SIGSAC Conference on CCS* (2023). DOI: 10.1145/3576915.3623063.

[55] T. Kurt et al. "Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1421–1434. DOI: 10.1145/3133956.3134067.

[56] L. Lassak et al. "Why Aren't We Using Passkeys? Obstacles Companies Face Deploying FIDO2 Passwordless Authentication". In: *33rd USENIX Security Symposium (USENIX Security 24)* (2024), pp. 7231–7248. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/lassak.

[57] Edwin Lim and Isaac Ejeh. *Managing user sessions: localStorage vs sessionStorage vs cookies*. URL: https://stytch.com/blog/localstorage-vs-sessionstorage-vs-cookies/ [Last Accessed: March 13th, 2026].

[58] Ayang Macdonald. *Germany pushes passkey adoption, releases draft technical guidelines*. URL: https://www.biometricupdate.com/202510/germany-pushes-passkey-adoption-releases-draft-technical-guidelines [Last Accessed: March 4th, 2026].

[59] Sandeep Machiraju. *Understanding Shadow DOM*. URL: https://www.linkedin.com/pulse/understanding-shadow-dom-sandeep-machiraju-lzxnc/ [Last Accessed: May 13th, 2026].

[60] S. Madnick. "Why Data Breaches Spiked in 2023". In: *Harvard Business Review* (2024). URL: https://cams.mit.edu/wp-content/uploads/Why-Data-Breaches-Spiked-in-2023.pdf.

[61] A. T. Mahdad, M. Jubur, and N. Saxena. "Breaching Security Keys without Root: FIDO2 Deception Attacksvia Overlays exploiting Limited Display Authenticators". In: *Proceedings of the 2024 ACM SIGSAC Conference on CCS* (2024). DOI: 10.1145/3658644.3690286.

[62] MDN. *Using the Web Storage API*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API [Last Accessed: March 13th, 2026].

[63] Mistral. *Mistral 7B*. URL: https://docs.mistral.ai/models/mistral-7b-0-1 [Last Accessed: November 4th, 2025].

[64] Mistral. *Mistral Medium 3.1*. URL: https://docs.mistral.ai/models/mistral-medium-3-1-25-08 [Last Accessed: November 4th, 2025].

[65] Mistral. *Mistral Small 3.2*. URL: https://docs.mistral.ai/models/mistral-small-3-2-25-06 [Last Accessed: November 4th, 2025].

[66] Dan Moore. *What is FedCM ?* URL: https://fusionauth.io/blog/what-is-fedcm [Last Accessed: January 13th, 2026].

[67] NordPass. *Top 200 Mots de passe les plus courants : Les générations se succèdent, les habitudes de mots de passe restent*. URL: https://nordpass.com/fr/most-common-passwords-list/ [Last Accessed: November 24th, 2025].

[68] Okta. *Principe de fonctionnement de WebAuthn*. URL: https://www.okta.com/sites/default/files/2020-07/How_WebAuthn_Works_FR.pdf [Last Accessed: December 4th, 2025].

[69] Idowu Omisola. *Using –disable-blink-features=AutomationControlled to Reduce Detection in Headless Chrome*. URL: https://www.zenrows.com/blog/disable-blink-features-automationcontrolled#playwright [Last Accessed: March 26th, 2026].

[70] I. Palamà et al. "Attacks and vulnerabilities of Wi-Fi Enterprise networks: User security awareness assessment through credential stealing attack experiments". In: *Computer Communications* 212 (2023), pp. 129–140. DOI: 10.1016/j.comcom.2023.09.031.

[71] R. Penman. *python-whois 0.9.6*. URL: https://pypi.org/project/python-whois/ [Last Accessed: May 13th, 2026].

[72] Peyman. *Building a Reliable Scraper for JavaScript-Heavy Sites with Playwright*. URL: https://medium.com/@peikaar1/building-a-reliable-scraper-for-javascript-heavy-sites-with-playwright-145c1c11a1c1 [Last Accessed: March 3rd, 2026].

[73] Victor Le Pochat and co. *A Research-Oriented Top Sites Ranking Hardened Against Manipulation*. URL: https://tranco-list.eu/ [Last Accessed: October 1st, 2025].

[74] P. Poornachandran et al. *Password Reuse Behavior: How Massive Online Data Breaches Impacts Personal Data in Web*. Springer Singapore, 2016, pp. 199–210. ISBN: 978-981-10-0419-3.

[75] Purushotham.R. *How I Combined WHOIS, DNS, and Shodan in One Powerful Python Script*. URL: https://medium.com/@Purushothamr/how-i-combined-whois-dns-and-shodan-in-one-powerful-python-script-b1031962e262 [Last Accessed: October 15th, 2025].

[76] Raian. *How Headless Browser Detection Works and How to Bypass It*. URL: https://latenode.com/blog/web-automation-scraping/avoiding-bot-detection/how-headless-browser-detection-works-and-how-to-bypass-it [Last Accessed: March 11th, 2026].

[77] Raian. *Python Headless Browser: Best Libraries for Automation*. URL: https://latenode.com/blog/python-headless-browser [Last Accessed: May 15th, 2026].

[78] Suby Raman. *WebAuthn: A better alternative for securing our sensitive information online*. URL: https://webauthn.guide/ [Last Accessed: December 4th, 2025].

[79] N. Ravanbakhsh, M. Mohammadi, and M. Nikooghadam. "Perfect forward secrecy in VoIP networks through design a lightweight and secure authenticated communication scheme". In: *Multimed Tools Appl* 78 (2019), pp. 11129–11153. DOI: 10.1007/s11042-018-6620-2.

[80] K. Reese et al. *A Usability Study of Five Two-Factor Authentication Methods*. USENIX Association, 2019, pp. 357–370. ISBN: 978-1-939133-05-2. URL: https://www.usenix.org/conference/soups2019/presentation/reese.

[81] Andrios Robert. *Unlocking the Future: Federation Passwordless Authentication Explained*. URL: https://hoop.dev/blog/unlocking-the-future-federation-passwordless-authentication-explained/ [Last Accessed: March 14th, 2026].

[82] G. L. Sanam et al. "Is FIDO2 the Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication". In: *2020 IEEE Symposium on Security and Privacy (SP)* (2020), pp. 268–285. DOI: 10.1109/SP40000.2020.00047.

[83] J. Santos et al. "An empirical study of tactical vulnerabilities". In: *Journal of Systems and Software* 149 (2019), pp. 263–284. ISSN: 0164-1212. DOI: 10.1016/j.jss.2018.10.030.

[84] J. Schaad and August Cellars. *RFC 9053 CBOR Object Signing and Encryption (COSE): Initial Algorithms*. URL: https://www.rfc-editor.org/rfc/rfc9053.html [Last Accessed: December 20th, 2025].

[85] Gwen Shapira. *Secure Authentication with Cookies*. URL: https://www.thenile.dev/blog/auth-and-cookies [Last Accessed: March 13th, 2026].

[86] R. Shay et al. "Encountering stronger password requirements: user attitudes and behaviors". In: *Proceedings of the Sixth Symposium on Usable Privacy and Security*. 2010. ISBN: 9781450302647. DOI: 10.1145/1837110.1837113.

[87] J. Shreffler and M. R. Huecker. "Hypothesis Testing, P Values, Confidence Intervals, and Significance". In: *National Library of Medicine* (2023). URL: https://pubmed.ncbi.nlm.nih.gov/32491353.

[88] H. Siadati et al. "Mind your SMSes: Mitigating social engineering in second factor authentication". In: *Computers & Security* 65 (2017), pp. 14–28. ISSN: 0167-4048. DOI: 10.1016/j.cose.2016.09.009. URL: https://www.sciencedirect.com/science/article/pii/S016740481630116X.

[89] Bundesamt für Sicherheit in der Informationstechnik. *BSI TR-03188: Technische Richtlinie Passkey Server*. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03188/BSI-TR-03188.pdf?__blob=publicationFile&v=2 [Last Accessed: March 4th, 2026].

[90] Wayne C Summers and Edward Bosworth. "Password policy: the good, the bad, and the ugly". In: *Proceedings of the winter international synposium on Information and communication technologies*. 2004, pp. 1–6.

[91] T. I. Tanni et al. "Is My Password Strong Enough?: A Study on User Perception in The Developing World". In: *EAI Endorsed Transactions on Creative Technologies* (2018). DOI: 10.4108/EAI.11-2-2022.173452.

[92] Stytch Team. *Stytch's guide to passwordless authentication*. URL: https://stytch.com/blog/stytch-guide-to-passwordless-authentication/ [Last Accessed: May 13th, 2026].

[93] S. Tenny and M. R. Hoffman. "Odds Ratio". In: *National Library of Medicine* (2023). URL: https://pubmed.ncbi.nlm.nih.gov/28613750.

[94] Thales. *Shadow DOM*. URL: https://www.imperva.com/learn/application-security/shadow-dom/ [Last Accessed: February 12th, 2026].

[95] R. Tuli. "Packet Sniffing and Sniffing Detection". In: *International Journal of Innovations in Engineering and Technology (IJIET)* (2020). DOI: 10.21172/ijiet.161.04.

[96] EC-Council University. *The Importance of Cybersecurity in the Financial Industry*. URL: https://www.eccu.edu/blog/why-is-cyber-security-important-in-the-financial-industry/ [Last Accessed: March 4th, 2026].

[97] Vincent. *FedCM (Federated Credential Management API) Overview*. URL: https://www.corbado.com/blog/fedcm-federated-credential-management-api [Last Accessed: March 14th, 2026].

[98] W3C. *Web Authentication: An API for accessing Public Key Credentials Level 3*. URL: https://w3c.github.io/webauthn/#sctn-intro [Last Accessed: April 21th, 2026].

[99] W3SCHOOLS. *HTML Web Storage API*. URL: https://www.w3schools.com/html/html5_webstorage.asp [Last Accessed: March 11th, 2026].

[100] Michael Waterman. *How FIDO2 works, a technical deep dive*. URL: https://michaelwaterman.nl/2025/04/02/how-fido2-works-a-technical-deep-dive/ [Last Accessed: December 4th, 2025].

[101] Webshrinker. *IAB Categories*. URL: https://docs.webshrinker.com/v3/iab-website-categories.html#tier-1-and-tier-2-categories [Last Accessed: March 3rd, 2026].

[102] Buy Bitcoin World Wide. *Fido2, Webauthn and U2F Supported Sites (Full List)*. URL: https://buybitcoinworldwide.com/dongle-auth/ [Last Accessed: October 25th, 2025].

[103] Wikidata. *Wikidata*. URL: https://www.wikidata.org/wiki/Wikidata:Main_Page [Last Accessed: May 15th, 2026].

[104] Wikipedia. *Document Object Model*. URL: https://en.wikipedia.org/wiki/Document_Object_Model [Last Accessed: March 11th, 2026].

[105] WorldStandards. *Internet country domains list (TLDs)*. URL: https://www.worldstandards.eu/other/tlds/#google_vignette [Last Accessed: October 2nd, 2025].

[106] L. Würsching et al. "FIDO2 the Rescue? Platform vs. Roaming Authentication on Smartphones". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (2023). DOI: 10.1145/3544548.3580993.

[107] T. K. Yadav and K. Seamons. "A Security and Usability Analysis of Local Attacks Against FIDO2". In: *Proceedings of Sixteenth Symposium on Usable Privacy and Security* (2020). DOI: 10.48550/arXiv.2308.0297.

[108] Yubico. *How does CTAP work?* URL: https://www.yubico.com/resources/glossary/ctap/ [Last Accessed: December 6th, 2025].

[109] Yubico. *Platform and Roaming Authenticators.* URL: https://developers.yubico.com/Developer_Program/WebAuthn_Starter_Kit/Platform_and_Roaming_Authenticators.html [Last Accessed: December 7th, 2025].

[110] Yubico. *Web Authentication API.* URL: https://developers.yubico.com/WebAuthn/WebAuthn_Developer_Guide/WebAuthn_Client_Registration.html [Last Accessed: November 29th, 2025].

[111] Yubico. *Web Authentication API.* URL: https://developers.yubico.com/WebAuthn/WebAuthn_Developer_Guide/WebAuthn_Client_Authentication.html [Last Accessed: November 29th, 2025].

[112] Yuboci. *FIDO2 Credentials.* URL: https://docs.yubico.com/yesdk/users-manual/application-fido2/fido2-credentials.html#:~:text=In%20FIDO2%2C%20a%20credential%20is,challenge%20from%20the%20relyng%20party. [Last Accessed: December 4th, 2025].

[113] M. Yusop et al. "Advancing Passwordless Authentication: A Systematic Review of Methods, Challenges, and Future Directions for Secure User Identity". In: *IEEE Access* 13 (2025), pp. 13919–13943. DOI: 10.1109/ACCESS.2025.3528960.

APPENDIX A

SUBMITTED ARTICLE

This work on FIDOLOGY has resulted in writing an article. This has been submitted at the ACM Conference on Computer and Communications Security (CSS) 2026 and is currently under review. The content is essentially a compressed version of this thesis, which explores in less details the findings and results.

Highway to Auth: A Large-Scale Measurement of Web Authentication Signals with Atlas

Macha Krumm
University of Liège
Liège, Belgium
florian.dekinder@uliege.be

Florian Dekinder
University of Liège
Liège, Belgium
florian.dekinder@uliege.be

Benoit Donnet
University of Liege
Liège, Belgium
benoit.donnet@uliege.be

Abstract

Web authentication has evolved significantly over the past decade, yet no large-scale empirical picture exists of which mechanisms are actually deployed in the wild —- a gap that hinders both research and practice. We introduce Atlas, a methodology for automatically identifying web authentication mechanisms at scale through headless-browser automation and multi-layered signal inference, classifying websites into five categories without requiring user interaction or credentials. Evaluated against a manually curated groundtruth, Atlas demonstrates reasonable accuracy across most authentication categories. Applied to a large dataset of websites, it reveals that traditional password-based authentication remains the dominant visible login mechanism, while Fido2/WebAuthn-based flows account for a small but non-negligible fraction of deployments. Beyond detection, we conduct a focused analysis of deployed Fido2/WebAuthn implementations by actively capturing the cryptographic challenges exchanged during live authentication flows. Most captured challenges meet or exceed Fido2 security recommendations, yet a non-negligible minority of deployments exhibit weak configurations — including cases of exact challenge reuse across sessions, which directly expose affected websites to replay attacks.

CCS Concepts

• **Security and privacy** → *Web application security*; *Browser security*; • **Human-centered computing** → Empirical studies in interaction design.

Keywords

Atlas, Fido2, WebAuthn, signals, authentication classification, Fido2 challenge

1 Introduction

For decades, passwords dominated as the primary authentication mechanism despite well-known security drawbacks (e.g., reuse, weak entropy) and vulnerability to phishing and credential theft [2, 3, 14, 31]. To strengthen account security, service providers introduced *two-factor authentication* (2FA) [34] methods such as one-time passcodes (Otp) via SMS or authenticator apps—layered on top of passwords. These Otp-based 2FA schemes offered only an incremental improvement, mitigating some risks but remaining vulnerable to social engineering and man-in-the-middle attacks [18, 22, 43]. A response to these drawbacks was an industry push toward phishing-resistant authentication. The latter motivated the formation of the Fast IDentity Online 2 (Fido2) Alliance in 2012, uniting major companies to develop open standards that reduce reliance on passwords. These efforts led to the Fido2 standards [42] – comprising the W3C Web Authentication (WebAuthn) API [21] and the CTAP protocols [15] – which were finalized in 2018–2019 as a robust, public-key-based alternative to passwords. By design, Fido2 authenticators use asymmetric cryptography and local user verification (e.g., biometrics), making phishing significantly more difficult compared to passwords or Otp codes [21, 43].

Despite the importance of authentication in web security, we still lack a clear large-scale view of which authentication mechanisms are actually deployed by contemporary websites. Existing mechanisms differ widely in how they are exposed to users and browsers, and their presence is often difficult to identify automatically because authentication interfaces may be dynamic, delegated, or only partially observable.

Our contributions to bridge that gap are as follows. First, we introduce **A**uthentication **T**echnoLogy **A**nalysi**S** (Atlas), a tool for identifying web authentication mechanisms at scale through automated web scraping and signal inference. Atlas systematically crawls websites and detects authentication schemes by combining headless-browser automation with a multi-layered signal analysis engine, making it deterministic and non-intrusive. Atlas classifies the authentication scheme into one of the five categories: None/Error (no observable authentication features have been observed), Password-Based (traditional login relying solely on passwords), Password-Extended (transitional forms of authentication combining passwords with another factor), Fido2-Native (modern authentication flows primarily relying on Fido2/WebAuthn), or Unknown (mixed or contradictory signals that do not allow clear classification).

Second, we evaluate Atlas against a manually crafted groundtruth of 993 websites, allowing us to assess its accuracy against authentication mechanisms identified with high confidence through manual inspection. Third, we apply Atlas in the wild to a large-scale dataset of 981, 201 websites, providing a broader analysis of authentication patterns. We find that the majority of reachable websites still expose traditional password-based authentication as their primary login mechanism.

Finally, because Fido2/WebAuthn represents a distinctive class of modern authentication, we devote a focused analysis to it. In

CCS '26, The Hague, The Netherlands.

ACM ISBN XXXX
https://doi.org/XXXX

particular, beyond identifying its presence, we go beyond mere detection and characterize FIDO2/WebAuthn implementations by evaluating the cryptographic challenges exposed during authentication flows. We observed that most websites offer strong cryptographic challenges (i.e., high entropy values; ≈ 3,500 bits; with long challenges; greater than 600 bytes), while a few of them poorly implement FIDO2/WebAuthn standards. More critically, and perhaps most alarmingly, we identify cases of exact cryptographic challenge reuse across sessions, directly exposing affected websites to replay attacks.

The remainder of this paper is organized as follows: Sec. 2 introduces the technical background on Web authentication mechanisms; Sec. 3 discusses the various technical challenges and how ATLAS tackles them; Sec. 4 presents ATLAS by detailing its overall design, signal extraction process, and authentication methods classification; Sec. 5 confronts ATLAS with a groundtruth manually built; Sec. 6 analyzes the results of our large-scale study and discusses observed authentication patterns; Sec. 7 provides a preliminary analysis of FIDO2/WebAuthn challenge parameters; Sec. 8 positions our work with respect to the state of the art; finally, Sec. 9 concludes the paper and outlines future research directions.

2 Background

In this section, we introduce the required technical background for the rest of this paper. In particular, Sec. 2.1 recalls the web concepts needed to understand the authentication signals introduced later; Sec. 2.2 summarizes the evolution of common web authentication mechanisms, while Sec. 2.3 focuses on passkey-based authentication.

2.1 Technical Prerequisites

A fundamental concept is the Document Object Model (DOM) [47]. It represents the webpage structure as an object tree that can be accessed and manipulated via scripts. The DOM is essential for rendering the user interface in browsers, allowing authentication forms (e.g., login fields and buttons) to be dynamically created or modified at runtime. In addition, websites also introduce the concept of a Shadow DOM [47], an encapsulated sub-tree of the DOM associated with Web components. The Shadow DOM hides its internal structure and styling from the main document context, but it must be inspected for a complete user interface (UI) level analysis. Because most authentication mechanisms are ultimately rendered as HTML elements (e.g., input fields, buttons, dialogs), inspecting both the DOM and the Shadow DOM could provide textual clues about the supported authentication mechanisms (e.g., "Use passkey", "Continue with Google", or "Enter the code") [40].

Web applications also maintain client-side state using web storage mechanisms. The Web Storage API provides two facilities: *local storage* and *session storage* [48]. Data in local storage persists across browser sessions (until explicitly cleared), whereas session storage is limited to the duration of the page session and is cleared once the tab is closed. Web developers may use local storage or session storage to cache information relevant to authentication, and inspecting them is thus relevant in the context of our study.

Yet another client-side storage mechanism is the cookie [7], a small piece of data that the browser stores and automatically includes with subsequent HTTP requests to the same server. HTTP being a stateless protocol, cookies are often seen as the backbone of session management on the web. Indeed, cookies are frequently used to authenticate users (e.g., by carrying a session identifier or other authentication-related token). Therefore, inspecting cookie contents can also provide useful clues about which authentication mechanisms a website relies on.

2.2 Evolution of Authentication Mechanisms

Historically, password-based authentication has been the dominant authentication mechanism on the Web, with most websites relying on username/password credentials for user authentication [33, 41]. Despite their simplicity, passwords suffer from well-known weaknesses, including reuse across services and vulnerability to phishing and credential theft [10, 14]. To improve resilience against these weaknesses, service providers widely deployed *two-factor authentication* (2FA), most commonly through One-Time Password (OTP) delivered by SMS/email or generated by authenticator applications [34]. While OTP-based 2FA is more secure than password-alone authentication, it remains vulnerable to real-time phishing and some social-engineering strategies.

In parallel, the web ecosystem increasingly adopted federated authentication (single sign-on, SSO) [5], where a relying party delegates user authentication to an Identity Provider (IDP) through standardized protocols such as OAuth [19] and OpenID Connect [16]. Recently, browsers proposed the *Federated Credential Management (FedCM)* API [44] to make federated login more privacy-preserving and less dependent on third-party cookies by allowing the browser to mediate the IDP selection and credential exchange.

Finally, an industry-wide push for phishing-resistant passkey-based authentication led to the Fast IDentity Online 2 (FIDO2) standards, which combine the W3C Web Authentication (WebAuthn) API with the CTAP protocols [15, 21, 42]. In contrast to passwords and OTPs, WebAuthn relies on asymmetric cryptography, enabling passkey-based and passwordless authentication designed to be resistant to credential stealing and phishing attacks [43].

2.3 Passkey-Based Authentication

Fig. 1 illustrates the FIDO2 passwordless authentication architecture [42], which combines the W3C Web Authentication API (WebAuthn) [21] with the Fast IDentity Online (FIDO) Alliance's Client-to-Authenticator Protocol (CTAP) [15]. In this model, there are three key entities: the *Relying Party* (RP) server, the client (the user's web browser acting as the WebAuthn client), and the authenticator device. The WebAuthn API defines the interface between the web application (RP) and the client, while the CTAP standard handles communication between the client platform and an external authenticator.

Authenticators can either be internal or external. An internal authenticator is integrated into the user's device (e.g., MacOS's TouchID), while an external authenticator is a separate hardware entity (e.g., a USB security key) that can be used across multiple clients.

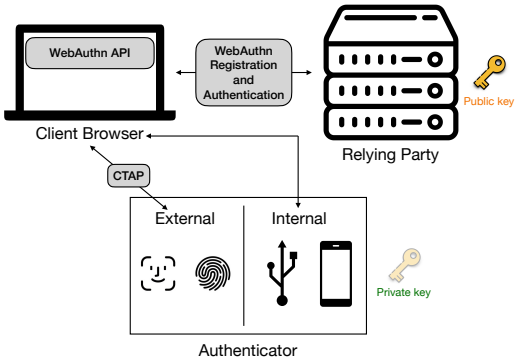


Figure 1: Global overview of Fido2 authentication scheme.

With Fido2, authentication relies on asymmetric cryptography: for each RP, the authenticator maintains a private key, and the RP stores the corresponding public key. The core function of any authenticator is to securely store the private key and perform cryptographic operations (like digital signatures) when requested. To perform such operations, a user action will be requested (e.g. a biometric test or a PIN) before completing *a registration or a sign-in ceremony*.

Registration ceremony. During registration, the RP generates a fresh cryptographic challenge and sends it to the browser together with credential creation parameters (RP identifier, user handle, supported algorithms, etc.). The browser then invokes `navigator.credentials.create()` to request credential creation via WebAuthn. The request is then passed to an authenticator which, after user verification, generates a new public-private key pair bound to the RP. The private key remains inside the authenticator, while the public key is returned to the RP and stored server-side for future logins.

Authentication ceremony. During authentication (sign-in), the RP sends a cryptographic challenge to the client (minimum 16 bytes long with a minimum entropy of 128 bits [46]) and indicates which registered credentials are acceptable for this account. The browser then invokes `navigator.credentials.get()` to ask the authenticator to produce a signed response. After user verification, the authenticator signs the challenge with the private key corresponding to the selected credential. The browser forwards the signed challenge to the RP, which verifies the signature using the stored public key. If both signatures match, the user is authenticated.

3 Challenges, Threats to Validity, and Ethics

In this section, we discuss the main challenges in building an automated tool for identifying authentication mechanisms on websites, as well as the associated threats to validity and ethical constraints.

Structural challenges: Nowadays, there are multiple standard authentication mechanisms: classic password, one-time password (Otp), Fido2, FedCM, single-sign on (SSO), any combination of the previous possibilities, and proprietary implementations co-exist in the modern web ecosystem [10]. Further, a website may expose different mechanisms according to the context (e.g., authenticating through a web browser might be done through passwords, while through passkeys if the device is a smartphone). This heterogeneity makes the authentication mechanism identification difficult. With Atlas, we try to cover a large spectrum in website through independent signals (DOM, network, storage, APIs), allowing us to identify authentication categories rather than particular implementations. However, Atlas cannot guarantee the exhaustivity of authentication mechanisms it is able to identify.

In modern websites, authentication surface might be hidden behind modals, popups, iframes, or are exposed only after some interactions with the user. Additionally, authentication page might be hidden behind some *closed Shadow DOM* [45], i.e., an encapsulation mode preventing external JavaScript from accessing the component's internal DOM tree via the shadowRoot. This makes any static analysis challenging, even sometimes impossible. Atlas relies on active and non-intrusive heuristics, e.g., navigation to login, forcing to expose UI, or the authentication surface stabilization. However, despite this, some interfaces might stay inaccessible as they explicitly require human interaction or a particular applicative state. Therefore, the results reported in this paper correspond to a lower bound.

Websites may delegate authentication to iframes, possibly cross-origin, preventing browsers from accessing them [8]. Websites apply this delegation for security reasons. Atlas is conservative and careful in the way it classifies authentication mechanisms. Indeed, Atlas is able to explicitly detect such situations and categorizes them as such (e.g., `unknown_cross_origin` or `none`), avoiding false negatives. Consequently, it may be impossible for Atlas to observe signals and, thus, categorize authentication mechanisms of websites.

Interpretation challenges: A given signal (e.g., the keyword "passkey") may indicate an active implementation or an inactive functionality. To face such a challenge, Atlas prioritizes inference traceability instead of a binary classification, potentially erroneous.

Analyses based on dynamic rendering, timing, or behavioral heuristics are often non-deterministic and difficult to compare scientifically. Atlas has been designed to be deterministic within a constant scope (same site, same browser, same rules). Sources of non-determinism are identified (network, blocking, dynamic content) and explicitly displayed in the results.

Anti-automation challenges: Modern websites rely on CAPTCHA [17], Turnstile [13], fingerprinting or conditional loading to avoid automated analysis. This may block or distort the observation. Atlas detects and documents explicitly those situations, instead of producing erroneous results. Nevertheless, when a blocking is detected, the analysis is interrupted.

4 Atlas

In a nutshell, the identification of authentication mechanisms relies on a series of *signals* (Sec. 4.2) captured during an automated scraping of each website. Those signals are next combined to produce a *classification of authentication mechanism* (Sec. 4.3). We have implemented that process in a tool we call Atlas.

Table 1: List of signals extracted from a website by Atlas.

ID	Signal	Description	Type
ST_1	`password_input_present`	Presence of a visible password field in the main DOM	Technical (DOM)
ST_2	`password_input_in_shadow_dom`	Password field hidden inside a Shadow DOM	Technical (DOM)
ST_3	`credentials_api_used`	Explicit use of `navigator.credentials.*` (JavaScript API)	Technical (API)
ST_4	`credentials_create_summary`	Details of parameters passed to `navigator.credentials.create()`	Technical (API)
ST_5	`network_webauthn`	Network requests targeting WebAuthn endpoints (`/webauthn`, `/assertion`, etc.)	Technical (Network)
ST_6	`network_password`	Network request explicitly transmitting a user password (e.g., form POST, JSON payload)	Technical (Network)
ST_7	`passkey_setup_endpoint_present`	Backend exposes passkey	Technical (Network)
ST_8	`fedcm_present`	FedCM detected through console logs	Technical (Federated API)
ST_9	`fedcm_detected_via_api`	Direct detection of FedCM activity via intercepted `navigator.credentials.get()` API calls	Technical (Federated APi)
SH_1	`local_storage_contains_fido`	Fido2/WebAuthn keywords found in `localStorage`	Heuristic (storage)
SH_2	`session_storage_contains_fido`	Same Fido2/WebAuthn keywords found in `sessionStorage`	Heuristic (storage)
SH_3	`cookies_contain_fido`	Cookies containing Fido2/WebAuthn-related strings	Heuristic (storage)
SH_4	`shadow_dom_webauthn`	WebAuthn-related keywords detected inside Shadow DOMs	Heuristic (UI/Structure)
SH_5	`ui_webauthn_keywords_present`	Visible UI keywords such as "security key", "passkey", etc.	Heuristic (UI)
SH_6	`otp_indicators_present`	Evidence of an Otp flow – SMS/email/TOTP code, etc.	Heuristic (UI/Text)
SH_7	`fedcm_provider`	Detected FedCM identity provider (e.g., Google, Apple, etc.)	Heuristic (Provider)
SH_8	`fedcm_fido2_hint`	FedCM signal with known Fido2 2-capable IdP (e.g., Google, Apple)	Heuristic (Derived)
SH_9	`multistep_login`	Presence of multi-step login flow (e.g., redirects, stepper UI)	Heuristic (Flow)
SH_{10}	`auth_js_supports_passkey`	Frontend plans passkey	Heuristic (JS Bundle)
SS_1	`fido2_indirect_possible`	Inference that FedCM could trigger Fido2 (based on known IdP)	Synthetic (Inductive)

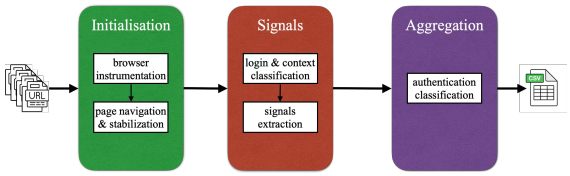


Figure 2: Global overview of Atlas scraping.

4.1 Global Overview

Fig. 2 illustrates the global overview of Atlas scraping process. It receives as input URLs and, for each of them, performs a sequential process. It first starts an *Initialization* process by checking the URL validity[1] and next, creates the context for the playwright headless browser [30]. The browser is instrumented with JavaScript hooks to intercept authentication-related API calls at runtime. This layer captures signals that are not observable via DOM inspection or network traffic alone, such as calls to `navigator.credentials`, WebAuthn, or FedCM. It provides evidence of client-side authentication intent and execution. Next,Atlas navigates to the target site, handles consent banners (including those potentially hidden in iFrames), and stabilizes the page state. This step ensures that authentication elements and scripts are fully loaded and executable. In addition, it normalizes the user interface (UI) by dismissing pre-authentication barriers (e.g., language or locale popups) to expose authentication-related elements.

The next step concerns the signal capture (*Signals* box in Fig. 2). Atlas attempts to locate and reach authentication entry points (login pages or dialogs), possibly through multiple steps (e.g., email first, next passwords). This is achieved heuristically, i.e., it relies on keywords (in several languages) typically used for processing login on a web page, standard buttons, and common login paths. Pages are classified to detect blocking, errors, or atypical contexts (e.g., bot protection). This determines whether further authentication analysis can proceed meaningfully. Multiple detectors then collect signals (see Sec. 4.2 for details) from the DOM, Shadow DOM, storage, network requests, and runtime APIs. Signals include password fields, WebAuthn usage, Fido2 artifacts in storage, UI keywords, Otp indicators, and FedCM activity. The result is a structured, multi-dimensional representation of the site's authentication behavior.

The last step concerns the potential authentication mechanism inference (*Aggregation* box in Fig. 2). Extracted signals are combined through a deterministic decision model to infer the authentication mechanism category. The model distinguishes between password-only, hybrid, Fido2-native, latent, and ambiguous scenarios (see Sec. 4.3).

Prior to saving results in a CSV file, for Fido2-compatible classifications, Atlas optionally attempts a controlled interaction with passkey or WebAuthn flows. This may extract cryptographic parameters (e.g., cryptographic challenge or Cose algorithms [37–39])[2] or confirm activation paths. The step provides empirical validation without requiring user credentials.

Finally, it is worth noticing that authentication mechanisms are evaluated within a bounded functional scope. Redirections to secondary domains are only considered authoritative if they are explicitly linked through standardized identity protocols (e.g., OAuth, FedCM). Otherwise, authentication signals are attributed to a distinct service and do not override mechanisms detected on the primary site.

4.2 Signals Extraction

Atlas, when analyzing a website, is able to capture 23 signals, with the lowest possible interaction. Those signals are distributed among three different types: *technical*, *heuristic*, and *synthetic*.

[1]We skip URLs like `https://www.rr5---sn-2ugxh5a5-jb3z.gvt1.com` or `https://www.rq6zxa4dsfdaj1g.xyz`, as well as CDN-like URLs (e.g., `drive.google.com`) and suspicious top-level domains (e.g., `.online` or `.click`).

[2]Note that we do not expect to be able to extract Cose algorithms, as no registration process has been run with any website. We added this for consistency reasons and double-check any implementation issues on the web server side.

Technical signals are a direct factual observation, i.e., an objective proof of mechanism usage. In other words, these are direct, verifiable pieces of evidence of authentication mechanisms observed through browser-level inspection (e.g., DOM elements, API calls, or network requests). They originate from explicit implementation traces of Fido2/WebAuthn, or classical login flows. Examples include the presence of a password input field in the DOM, calls to `navigator.credentials.create()` or `navigator.credentials.get()`, network traffic referencing `/webauthn` endpoints, or FedCM events detected in console logs. We consider nine technical signals, listed and described in Table 1.

Heuristic signals are indirect contextual clues, i.e., any interpretation based on user interface (UI) or storage data. In other words, these are indirect clues suggesting the use of a given authentication mechanism. They rely on observed patterns or keywords in UI text[3], stored data, or structural hints. While they are not conclusive on their own, they help strengthen inferences when combined with technical signals. Examples include Fido2-related terms in local storage or session storage, UI text mentioning "security key" or "passkey", cookies containing Fido2/WebAuthn references or One-Time Password (Otp) related messages in the UI. Atlas is able to identify ten heuristic signals, all described in Table 1.

Finally, synthetic signals refer to logical inference, deduced from correlations between signals. In other words, there are derived or inferred signals, built by correlating multiple technical and heuristic observations. They do not come directly from the page but result from reasoning logic implemented in Atlas. They represent aggregated evidence or probabilistic deductions. Examples include FedCM provider likely supporting Fido2 or cross-layer inference combining WebAuthn API and UI hints. A single synthetic signal is supported, as described in Table 1.

4.3 Authentication Classification

Once the signals have been extracted, they are combined to infer an authentication mechanism, e.g., Fido2 or password-based. We consider 12 different classes aggregated in five categories (see Table 2).

To simplify the inference process, we aggregate signals (see Table 1) into a set of higher-level boolean variables capturing different aspects of authentication mechanisms. In particular, we distinguish between strong Fido2 signals (e.g., API or network-level evidence), weaker UI-based indicators, and persistent traces observed in browser storage. These aggregated signals are designed to be mutually informative but not mutually exclusive. It is formalized as follows:

Password-related signals:
P = `password_input_present` ∨ `password_input_in_shadow_dom`

P_n = `network_password` ∧ ¬P

Multi-factor / OTP:
O = `otp_indicators_present`

Strong FIDO2 signals (API / network-level):
F_s = `credentials_api_used` ∨ `credentials_create_summary` ∨ `network_webauthn`

Weak FIDO2 signals (UI / DOM hints):
F_w = `ui_webauthn_keywords_present` ∨ `shadow_dom_webauthn`

[3]Keywords have been translated into several languages.

Persistent FIDO-related traces:
S = `cookies_contain_fido` ∨ `local_storage_contains_fido` ∨ `session_storage_contains_fido`

Federated / indirect signals:
$FedCM$ = `fedcm_present` ∨ `fedcm_detected_via_api` ∨ `fido2_indirect_possible`

Additional contextual signals:
J = `auth_js_supports_passkey`

U = `auth_surface_unobservable`

A = `auth_form_appeared`

E = (analysis error)

The *None/Error* category captures cases where Atlas is unable to determine the authentication mechanism due to a lack of observable signals. In such situations, no authentication-related features could be identified during automated navigation. Importantly, `none` does not imply the absence of authentication, but rather a limitation in observability (e.g., no login interaction triggered, no authentication UI exposed, or authentication gated behind user interaction). In contrast, some websites (e.g., `adobe.com`) actively block automated browsers, resulting in network-level failures (e.g., HTTP/2 protocol errors). These cases reflect failures of the measurement process itself and are therefore classified as analysis errors (`error`), rather than the absence of authentication signals. We thus distinguish two classes within the None/Error category, defined as follows:

$$\texttt{none}\ (\mathrm{NE}_1) \Leftrightarrow \neg P \wedge \neg P_n \wedge \neg O \wedge \neg F_s \wedge \neg F_w \wedge \neg S \wedge \neg \mathrm{FedCM}$$

$$\texttt{error}\ (\mathrm{NE}_2) \Leftrightarrow E$$

The *Password-Based* category refers to classic authentication schemes. It corresponds to login relying solely on passwords. No additional protection or Fido2 involvement has been detected. We consider a single class, `password_only`, through the following inference rule:

$$\texttt{password\_only}\ (\mathrm{PB}_1) \Leftrightarrow (P \vee P_n) \wedge \neg O \wedge \neg F_s \wedge \neg F_w \wedge \neg S \wedge \neg \mathrm{FedCM}$$

We require the absence of both strong and weak Fido2 signals to avoid misclassifying websites that expose residual or inactive Fido2-related artifacts.

The *Password-Extended* category gathers two-factor (2FA) and transitional mechanisms. It refers to transitional forms of authentication combining classic passwords with another factor – either Otp (One-Time Password such as Time-based One-Time Password, SMS, email code) or Fido2/WebAuthn. It represents incremental adoption of stronger security. Atlas considers two classes, `password+otp` and `password+fido`, through the following inference rules:

$$\texttt{password+otp}\ (\mathrm{PE}_1) \Leftrightarrow P \wedge O \wedge \neg F_s$$

$$\texttt{password+fido}\ (\mathrm{PE}_2) \Leftrightarrow P \wedge (F_s \vee \mathrm{FedCM})$$

In cases where both Otp and Fido2 signals are present, Fido2 takes precedence in Atlas inference as it represents a stronger authentication mechanism.

Table 2: List of authentication mecanisms classified by Atlas according to signals captured.

ID	Class	Dominant Conditions	Category
NE_1	none	No authentication-related signals detected (not even a password field)	None/Error
NE_2	error	Analysis failure (timeout, crash, or navigation error)	
PB_1	password_only	Password field detected without any additional signal (no Fido2 nor Otp)	Password-Based
PE_1	password+otp	Presence of a password field and Otp indicators	Password-Extended
PE_2	password+fido	Password field detected and Fido2/WebAuthn signals present	
F_1	webauthn	Clear usage of the WebAuthn API (without password or storage involvement)	Fido2-Native
F_2	storage	Only storage-related signals in localStorage, sessionStorage, or cookies)	
F_3	fido_only_ui	Only textual/UI indicators (e.g., "Use your security key")	
F_4	latent_support	Weak or indirect evidence (FedCM or UI hints) without explicit WebAuthn activity	
F_5	latent_usage	Evidence of potential or inactive Fido2 support (e.g., configuration pages or deferred setup)	
F_6	full_fido2	Complete combination: WebAuthn API + storage + UI	
U_1	unknown	Contradictory or insufficient signals to decide	Unknown

The *Fido2-Native* category concerns passwordless or Fido2-centric mechanisms. It refers to modern authentication flows primarily relying on Fido2/WebAuthn. It includes both direct usage (WebAuthn API) and latent/indirect indicators (e.g., FedCM, UI hints, or stored credentials). We have six classes (webauthn, storage, fido_only_ui, latent_support, latent_usage, an full_fido2) according to the following inferrence rules:

$$\begin{aligned}
\texttt{webauthn}\ (F_1) &\Leftrightarrow F_s \wedge \neg P \wedge \neg P_n \\
&\quad \wedge \neg F_w \wedge \neg S \wedge \neg \text{FedCM} \\
\texttt{storage}\ (F_2) &\Leftrightarrow S \wedge \neg P \wedge \neg P_n \\
&\quad \wedge \neg F_s \wedge \neg F_w \wedge \neg \text{FedCM} \\
\texttt{fido_only_ui}\ (F_3) &\Leftrightarrow F_w \wedge \neg P \wedge \neg P_n \\
&\quad \wedge \neg F_s \wedge \neg S \wedge \neg \text{FedCM} \\
\texttt{latent_support}\ (F_4) &\Leftrightarrow FedCM \vee J \\
\texttt{latent_usage}\ (F_5) &\Leftrightarrow \neg P \wedge \neg P_n \wedge \neg F_s \\
&\quad \wedge (F_w \vee S) \wedge \neg \text{FedCM} \\
\texttt{full_fido2}\ (F_6) &\Leftrightarrow F_s \wedge (F_w \vee S \vee \text{FedCM})
\end{aligned}$$

For all intermediate Fido2-related classes ($F1 \rightarrow F_5$), we explicitly require the absence of password-related signals in order to isolate and characterize individual Fido2 artifacts.

However, this constraint is intentionally relaxed for the full_fido2 class (F_6). In practice, a website may simultaneously expose password-based login forms and passkey-based authentication options within the same interface, although these mechanisms are mutually exclusive at runtime. To account for this design pattern, we define full_fido2 authentication based on the presence of a complete WebAuthn flow, regardless of whether password inputs are also present in the DOM.

Finally, the *Unknown* category encompasses uncertain or ambiguous cases. It corresponds to mixed or contradictory signals that do not allow clear classification. Any contradictory or overlapping combination not satisfying the above rules will lead to unknown class. It also includes cases where the authentication surface is cross-origin. Atlas considers a single class, unknown, defined as follows:

$$\begin{aligned}
\texttt{unknown}\ (U_1) \Leftrightarrow\ & \neg\texttt{error} \wedge \neg\texttt{none} \\
& \wedge \neg\texttt{auth_surface_unobservable} \\
& \wedge \neg\texttt{unknown_cross_origin} \wedge \neg\texttt{password_only} \\
& \wedge \neg\texttt{password+otp} \wedge \neg\texttt{password+fido} \\
& \wedge \neg\texttt{webauthn} \wedge \neg\texttt{fido_only_ui} \wedge \neg\texttt{storage} \\
& \wedge \neg\texttt{latent_support} \wedge \neg\texttt{latent_usage} \\
& \wedge \neg\texttt{full_fido2}
\end{aligned}$$

5 Groundtruth

This section aims at confronting Atlas with a groundtruth. Sec. 5.1 describes how the groundtruth has been built and how the Atlas data has been collected; Sec. 5.2 discusses the results.

5.1 Methodology

We manually built a groundtruth dataset between October 16th and October 27th, 2025. The objective of this dataset was to evaluate Atlas on websites whose authentication mechanisms could be identified with high confidence through manual inspection.

To build this dataset, we relied on five public sources. The first source was the Passkey Directory of the Fido2 Alliance website [4]. The second source was the Hideez directory [20], from which we considered websites listed in the Fido2-related sections, including entries referring to Fido2, Otp, and passkeys. The third source was the passkey directory provided by 1Password [1],which is basically an index of services supporting sign-in with passkeys. The fourth source was the Yubico public database [50]. The last source was the dongle-auth section of BuyBitcoinWorldwide [11], which was processed similarly.

As a result, the final groundtruth list contains 993 websites. We then provided this list as input to Atlas and ran the tool from a single measurement point located on our campus on March 23rd, 2026. Atlas analyzed each target using the same automated pipeline described in Sec. 4.

Table 3: Results of the Atlas filtering postprocessing for both the groundtruth (N = 993) and Top 1M (N = 981, 201). The total row is computed over the groundtruth and Top 1M.

Metric	Groundtruth		Top 1M Websites	
	RawValue	Percentage	Raw Value	Percentage
headless browser blocking	39	3.93%	529.429	53.96%
anti-bot challenge	28	2.82%	24.959	2.54%
closed Shadow DOM	33	3.32%	0	0.00%
JS orchestrated auth. UI	25	2.52%	4.776	0.49%
opaque iframe	40	4.03%	0	0.00%
stalled authentication UI	0	0.00%	0	0.00%
pwd without observablity	0	0.00%	0	0.00%
Total	165	16.62%	559.164	56.99%

5.2 Results

We first quantify the proportion of websites for which authentication analysis is fundamentally hindered by observability limitations. Overall, 16.62% of the scrapped websites fall into such cases, as illustrated by Table 3.

We identify two main categories of limitations. (*i*) Active blocking mechanisms. A total of 3.93% of websites explicitly block headless browsers, while an additional 2.82% deploy anti-bot challenges (e.g., Cloudflare), preventing any meaningful interaction. These defenses directly impact automated analysis and represent an inherent limitation of large-scale measurement approaches. (*ii*) Technical obfuscation of the authentication surface. Several websites deliberately or implicitly hide authentication elements. Closed Shadow DOM structures account for 3.32% of cases, preventing DOM-level inspection despite visible UI elements. We validate these cases through a combination of heuristics, including inaccessible password fields and rendered but non-queryable components, achieving high-confidence detection in 100% of cases. Additionally, 4.03% of websites expose an authentication surface that remains opaque despite successful navigation (e.g., dynamically injected or gated content), while 2.52% rely on JavaScript-orchestrated interfaces that do not materialize as stable DOM structures. Interestingly, we observe no instances of stalled authentication flows or password-based authentication without observability, suggesting that most failures can be attributed to explicit blocking or structural opacity rather than partial execution issues. Overall, these results highlight that a non-negligible fraction of authentication surfaces are inherently resistant to automated analysis, not due to shortcomings of Atlas, but because of deliberate design choices or modern web encapsulation mechanisms.

Once the collected data has been filtered, we compute the confusion matrix shown in Fig. 3, comparing Atlas's predictions with the groundtruth. Overall, we observe a relatively high proportion of misclassifications (precision of 50%), largely driven by the None/Error category. A significant number of websites belonging to other classes (e.g., Fido2-Native and Password-Based) are incorrectly classified as none. This confirms that none does not necessarily indicate the absence of authentication, but rather a lack of observable signals during automated analysis.

For the Password-Based category, a large majority of password-based websites are correctly identified. In contrast, more complex or modern authentication mechanisms introduce ambiguity. In particular, we observe substantial confusion between Fido2-Native

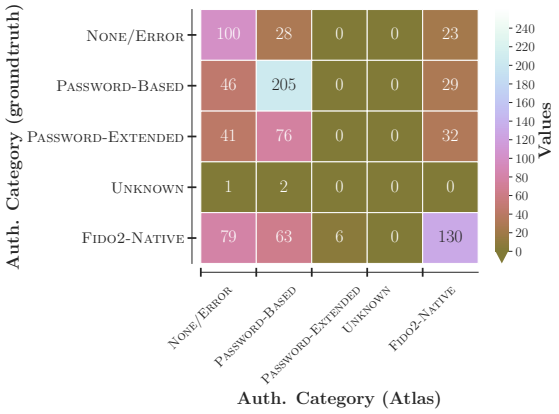


Figure 3: Confusion matrix for the groundtruth.

Table 4: Analysis of websites erroneously classified in the None/Error category. The total is computed over the groundtruth size (N = 993).

Metric	Raw Value	Percentage
No login surface	0	0.00%
3rd party redirect	0	0.00%
Not authentication page	56	5.64%
Silent anti-bot block	0	0.00%
Non-interactive UI	3	0.30%
Unexplained residual	108	10.88%
Total	167	16.82%

and Password-Based categories, reflecting the coexistence of multiple authentication options (e.g., password and passkey) within the same interface. Additionally, the Password-Extended category is never predicted, suggesting that either the corresponding signals are insufficiently discriminative or that this category does not clearly manifest in observable web features. Conversely, the unkn own category remains marginal, indicating that the classification framework effectively covers most observable cases.

It is worth reminding that when Atlas categorizes a website in Fido2-Native, it immediately tries to validate that category by clicking on a passkey login button (if any). Among the websites correctly classified as Fido2-Native, this occurred in 50% of the cases in our groundtruth.

We further analyze the website misclassified in the None/Error category (see Table 4). The results show that 16.82% of the dataset falls into this category after filtering. Among these cases, approximately 33% correspond to pages that are not actual authentication interfaces (*Not authentication page*). This suggests that a non-negligible portion of the apparent errors is in fact due to navigation or ground truth mismatches, rather than incorrect inference. More importantly, the majority of cases (64.7%) fall into an *Unexplained residual* category, meaning they cannot be attributed to any known failure mode such as anti-bot blocking, Shadow DOM encapsulation, or cross-origin redirection. This indicates the presence

of previously unidentified patterns or limitations in the current signal extraction pipeline. Interestingly, none of the residual errors are associated with known obfuscation mechanisms (e.g., closed Shadow DOM or non-interactive UI), confirming the effectiveness of the filtering step in isolating these cases. Overall, these findings reinforce that while some misclassifications stem from dataset or navigation artifacts, a significant portion highlights intrinsic blind spots in current web authentication observability, opening avenues for future improvements.

The remaining misclassifications in Fig. 3 for the FIDO2-NATIVE category can be explained by two main phenomena. First, some websites are effectively under-classified, i.e., labeled as PASSWORD-BASED while actually supporting FIDO2-based authentication. In many real-world deployments, FIDO2 authentication is only exposed after an initial password-based login or through account configuration settings. For instance, e-commerce platforms often require users to first register with an email and password before possibly enabling passkeys. Such cases remain inherently difficult to capture for ATLAS, as detecting them would require deeper interaction with the service, raising both ethical and practical limitations. Second, some websites are over-classified, i.e., labeled as FIDO2-NATIVE while in reality relying on PASSWORD-BASED mechanisms. These cases reveal multiple layers of ecosystem readiness, ranging from technical pre-deployment (e.g., dormant WebAuthn endpoints or unused APIs) and identity outsourcing (e.g., redirection to a potentially FIDO2-capable IdP), to purely declarative or marketing-driven signals (e.g., mentions of passkeys in UI without actual deployment). Interestingly, a large proportion of these over-classified cases (up to 73%) fall into the `latent_support` category, highlighting that most of those classification errors correspond to latent or indirect signals rather than fully deployed FIDO2 authentication.

To better understand the contribution of individual signals to the detection of full FIDO2 deployments, we compute odds ratios (OR) on our groundtruth dataset for websites correctly classified as `full_fido2`. Those results are illustrated by the forest plot in Fig. 4. A forest plot is a graphical representation that displays, for each signal, its odds ratio (i.e., how strongly it is associated with a given authentication category) along with its confidence interval, shown as a horizontal line. Signals whose confidence interval does not cross the vertical reference line at OR = 1 are statistically significant, meaning they genuinely discriminate the category from the rest of the dataset.

The results show that the use of the Credentials API (`credentials_api_used`) is by far the strongest predictor of `full_fido2` authentication, with a very high odds ratio (OR = 305.96, $p < 10^{-12}$). This confirms that explicit interaction with the WebAuthn API constitutes a highly reliable indicator of active passkey-based authentication flows. UI-level signals (`ui_webauthn_keywords_present`) also exhibit a strong and statistically significant association (OR = 26.41, $p < 0.001$), indicating that visible passkey-related elements provide meaningful complementary evidence of passwordless authentication. Other signals, such as network-level WebAuthn traces, Shadow DOM indicators, and FedCM-related features, show elevated odds ratios (OR $\approx$ 8) but remain statistically non-significant due to limited sample size ($N = 15$). These should therefore be interpreted as suggestive rather than conclusive. In contrast, storage-based signals (e.g., `local_storage_contains_fido` and `session_storage_contains_fido`) exhibit no significant association (OR $\leq$ 1), confirming that such artifacts primarily reflect residual or historical traces rather than active authentication mechanisms.

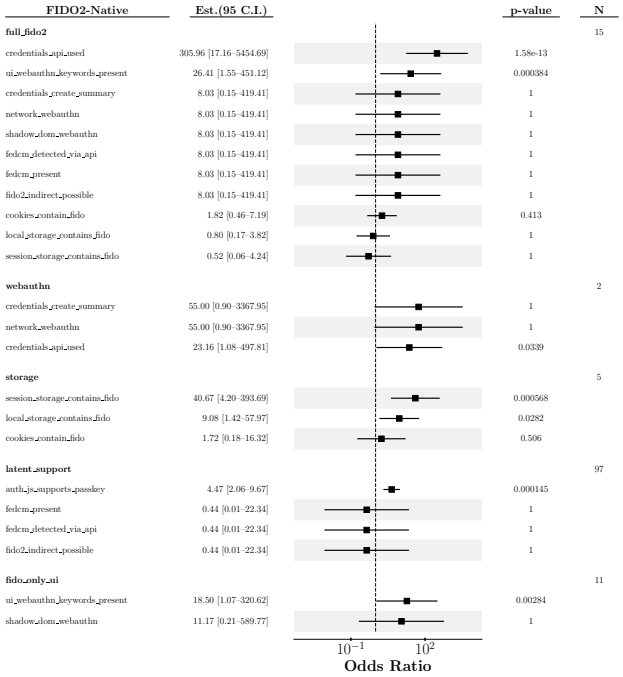


Figure 4: Forest plot for the odds ratios illustrating the signals importance for FIDO2-NATIVE correctly classified ($N = 130$).

For the other categories, distinct signal patterns emerge. The `webauthn` class is significantly associated with `credentials_api_used` (OR = 23.16, $p < 0.05$), reinforcing the importance of low-level API observation for detecting explicit WebAuthn usage. The `storage` class is driven by persistence-related signals, with both `local_storage_contains_fido` (OR = 9.08, $p < 0.05$) and `session_storage_contains_fido` (OR = 40.67, $p < 0.001$) showing strong and significant associations.

The `fido_only_ui` class is primarily characterized by UI-level indicators (`ui_webauthn_keywords_present`, OR = 18.50, $p < 0.01$), suggesting that visible passkey prompts are informative but not sufficient to confirm protocol execution. Finally, the `latent_support` class is mainly explained by JavaScript-level support (`auth_js_supports_passkey`), which shows a moderate but significant association (OR = 4.47, $p < 0.001$), while FedCM-related signals remain non-discriminative. It is worth noticing that, in the groundtruth, we did not find any website falling in the `latent_usage` class.

Overall, this analysis shows that ATLAS is effective at identifying major authentication patterns when the authentication surface is observable, but also that observability remains the main bottleneck. In particular, reliable detection of FIDO2 authentication is based on observing explicit WebAuthn API activity, while UI and storage signals, although useful, remain insufficient to establish the presence of active passwordless authentication on their own. The latter

explains why most misclassifications are driven by websites classified as None/Error, which often correspond either to navigation mismatches or to authentication mechanisms that are only partially exposed to automated analysis. At the same time, Password-Based websites are mostly well classified, and a large fraction of correctly classified Fido2-Native websites can be further validated through passkey interaction, confirming that Atlas captures meaningful Fido2/WebAuthn evidence when such evidence is directly exposed.

6 Authentication in the Wild

This section evaluates Atlas at scale on a large and diverse set of websites. Sec. 6.1 describes how the dataset was built and how the measurements were collected; Sec. 6.2 presents the resulting observations.

6.1 Methodology

To study authentication mechanisms at scale, we built a large and diverse dataset from three popular public web rankings: Tranco [28], CrUX [35, 36], and Umbrella [12]. The three lists were downloaded on May 21st, 2025.

The three separate URL lists were merged into a single consolidated dataset. Duplicate entries – defined as identical URLs appearing more than once – were removed, retaining only one instance of each unique URL. Subsequently, URLs belonging to the same corporate entity but registered under different top-level domains (e.g., amazon.com and amazon.co.uk) were deduplicated, keeping a single representative URL per organization. From this cleaned and normalized dataset, a random sample of 1,000,000 URLs was drawn for analysis.

The measurement campaign ran from March 24th, 2026 to March 27th, 2026. To scale the analysis, the list of 1,000,000 websites was split across four measurement machines, all provisioned on our campus. Each machine executed 40 independent Atlas processes in parallel. The machines shared the same hardware configuration: dual Intel(R) Xeon(R) Gold 5318Y CPUs at 2.10 GHz, with 96 logical CPUs, and 251 GiB of RAM per machine.

6.2 Results

As mentioned in Sec. 4.1, before trying to connect to a website, Atlas checks the URL validity. Roughly, 2% of the URLs were considered as invalid, leading to a dataset containing 981,201 probed URLs.

As for the groundtruth, we first start by filtering the collected data to remove websites hindered by observability limitations. This is provided in Table 3. The same categories of limitations as for the groundtruth are observed. However, we see that more than half of the websites explicitly block headless browsers. In the end, this filtering step removes roughly 60% of the dataset.

Table 5 summarises the authentication mechanisms observed by Atlas across the filtered dataset of 422,037 websites. The None/Error category dominates, accounting for 54.92% of the dataset (N = 231,767), with most cases classified as `none` (53.06%) and a smaller fraction as analysis errors (2.39%). This confirms that, even after filtering, a large portion of the web remains difficult to characterize in terms of authentication, either due to missing interaction triggers or limited observability of login mechanisms. Password-based authentication (`password_only`) is the dominant *visible* login mechanism, observed on 40.15% of websites. It is worth noting that this does not preclude the presence of Fido2 support on these websites. Rather, it indicates that Fido2 is not exposed as a direct login method, and may require prior account configuration by the user, as is the case for several major platforms.

Table 5: Authentication mechanism as observed by Atlas for the top 1M website. Proportions are computed after the filtering step (N = 422,037).

Category	Class	Classification		Total	
		Raw	(%)	Raw	(%)
None/Error	`none`	223 917	53.06%	231 767	54.92%
	`error`	7 850	2.39%		
Password-Based	`password_only`	169 461	40.15%	169 461	40.15%
Password-Extended	`password+otp`	0	0.00%	421	0.10%
	`password+fido`	421	0.10%		
Fido2-Native	`webauthn`	286	0.07%	20 371	4.83%
	`storage`	1 863	0.44%		
	`fido_only_ui`	400	0.09%		
	`latent_support`	17 452	4.14%		
	`latent_usage`	9	0.00%		
	`full_fido2`	361	0.09%		
Unknown	`unknown`	17	0.00%	17	0.00%

The Password-Extended category, which combines a password-based primary login with an additional Fido2 factor, represents 0.10% of the dataset, entirely driven by `password+fido`. No instance of `password+otp` was detected, which may reflect the limitations of automated detection for Otp-based flows rather than their actual absence. The Fido2-Native category accounts for 4.83% of the dataset (20,371 websites), with `latent_support` being its dominant subclass (4.14%), indicating that a significant share of websites have integrated Fido2 infrastructure —- detectable through API calls or JavaScript artifacts -— without exposing it as a direct authentication option on their login page.

Finally, `full_fido2`– websites offering Fido2 as a standalone, direct login method – represents 0.09% of the dataset (361 websites). This figure should be interpreted as a lower bound on actual Fido2 deployment, since websites requiring prior password-based enrollment before passkey activation, such as Amazon, are captured in other categories. The `fido_only_ui` (0.09%) and `webauthn` (0.07%) subclasses further suggest that a non-trivial number of websites expose Fido2-related UI elements or API calls without yet offering a fully autonomous passwordless experience.

In the wild, passwords still dominate what is directly observable on login pages, whereas Fido2/WebAuthn is usually present in latent form rather than as a fully exposed passwordless form. Due to observability limitations and headless-browser blocking, visible Fido2 adoption remains rare and should be interpreted as a lower bound on real deployment.

7 Preliminary Fido2 Challenge Analysis

This section provides a first look at the security level of websites falling in the Fido2-Native category, with a focus on the quality of Fido2 challenges sent by the server to the browser. Sec. 7.1 describes the measurement methodology, while Sec. 7.2 discusses the results.

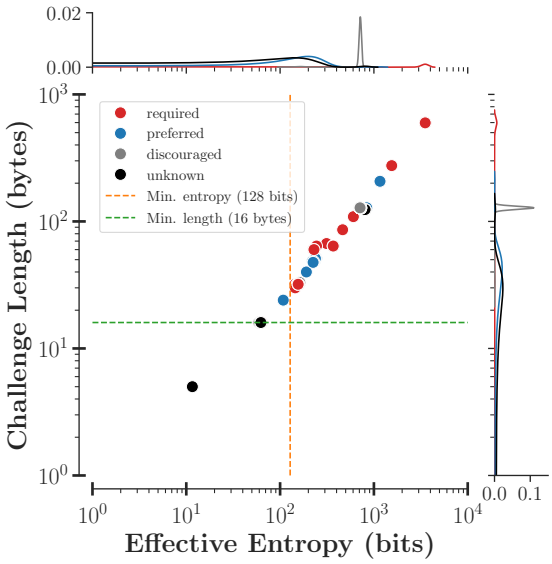


Figure 5: Security level of Fido2 cryptographic challenges sent by the server ($N = 1,011$).

7.1 Methodology

To perform a preliminary analysis of Fido2/WebAuthn challenges exposed by websites in the wild, we started from the Atlas results obtained in Sec. 6 and restricted the analysis to websites for which Atlas had already produced the most reliable evidence of explicit WebAuthn usage. Concretely, we selected websites classified in the `full_fido2` and `webauthn` classes. We also added websites falling in the Fido2-Native category that were validated during the Atlas scraping process. Finally, it corresponds to a list of 2,575 websites.

For this subset of websites, we extended Atlas with an additional non-intrusive probing step. After reaching the authentication interface, Atlas attempted to click the visible passkey or WebAuthn login button, when such a button was available and interactable. If this interaction successfully triggered a WebAuthn flow, Atlas then attempted to capture the cryptographic challenge returned by the server. Because challenge generation may vary across repeated executions on the same website, we repeated this interaction procedure five times for each selected website.

The data for this experiment was collected on April 1st, 2026 from the same measurement points described in Sec. 5.

7.2 Results

Over the 2,575 probed websites, we were able to collect 1,011. Atlas failed collecting cryptographic challenges in ≈ 40% of the cases due to anti-bot systems or multistep login (e.g., click on the passkey button and then select your geographic region).

Fig. 5 plots the effective entropy against the challenge length for all captured challenges, color-coded by the `userVerification` setting. The vast majority of challenges cluster in the upper-right quadrant, i.e., above both the minimum entropy threshold of 128 bits and the minimum length of 16 bytes recommended by the Fido2 specification [46], which suggests that most `full_fido2` deployments meet the baseline cryptographic requirements. In particular, a few websites exhibit particularly strong challenges, with entropy values reaching approximately 3,500 bits and challenge lengths exceeding 600 bytes, demonstrating that some deployments push security well beyond the minimum requirements. However, a non-negligible fraction of challenges – predominantly those with `unknown` user verification – fall below one or both thresholds, indicating incomplete or careless implementations. Notably, two outliers exhibit extremely low entropy and short length simultaneously, representing a clear violation of the Fido2 specification and a potential replay attack vector.

Table 6: `userVerification` option distribution in challenges captured by Atlas ($N = 1,011$).

User Verification	Raw	Percentage
`required`	671	66.37%
`preferred`	68	6.73%
`discouraged`	260	25.72%
`unknown`	12	1.19%

Table 7: Cryptographic challenges reuse as observed by Atlas ($N = 959$).

Observation	Raw	Percentage
Challenge reused detected	38	3.96%
Exact reuse (Hamming distance = 0)	38	3.96%
Timestamp pattern detected	0	0.00%

Regarding the `userVerification` policy, Table 6 reveals that `required` is by far the dominant setting (66.37%), which is encouraging from a security standpoint as it mandates explicit user presence verification (e.g., biometrics or PIN). Conversely, `discouraged` accounts for a surprisingly large share (25.72%), meaning that roughly one in four challenges explicitly waives user verification, potentially weakening the authentication guarantee. The small fraction of `unknown` values (1.19%) confirms our earlier observation that some deployments omit this parameter entirely, leaving the security policy undefined and relying solely on authenticator defaults.

Table 7 reports the observed challenge reuse across the 959 successfully captured websites for which at least two independent challenges could be collected. Among these, 3.96% of websites exhibit exact challenge reuse across probing sessions —- i.e., a Hamming distance of zero between at least two challenges issued by the same server — which constitutes a direct replay attack vector. No timestamp-based pattern was detected in this population. It is worth noting that 52 websites (5.14% of successfully captured sites) returned only a single challenge across all five probing attempts. This behaviour is consistent with server-side rate limiting or anti-bot protections that block repeated passkey flows from the same client, and prevents any reuse analysis for these sites

Fig. 6 displays the distribution of the average Hamming distance across captured websites. The distribution is clearly bimodal: a first cluster of sites concentrates near zero, corresponding to the reuse

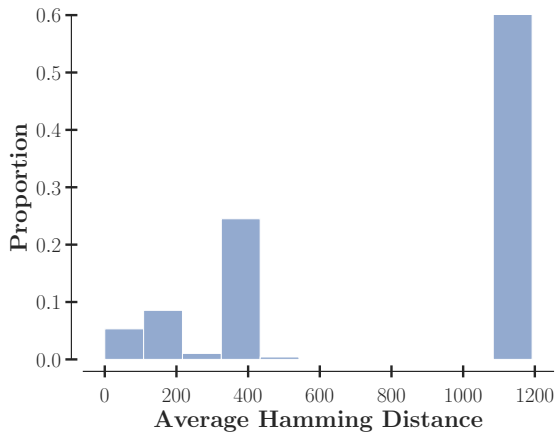


Figure 6: Histogram of the average Hamming distance between the five consecutive cryptographic challenges received for each web site (N = 959).

cases identified above, while the vast majority of websites exhibit high Hamming distances, well above 400 bits, indicating strong inter-session challenge diversity. This bimodal structure highlights a clear dichotomy in deployment quality -— most `full_fido2` implementations generate cryptographically sound challenges, while a minority of sites rely on weak or deterministic generation schemes that undermine the replay-resistance guarantees of the Fido2 specification.

In summary, most websites appear to implement cryptographic challenges that satisfy the baseline Fido2 requirements, often by a large margin. However, there is a non-negligible minority of weaker deployments, including cases with low-entropy or undersized challenges, configurations that discourage user verification, and — most critically — exact challenge reuse across sessions, which directly exposes affected websites to replay attacks. This suggests that while Fido2/WebAuthn appears robust in practice, both security quality and deployment remain heterogeneous across websites.

8 Related Work

A first line of work studies authentication practices at scale on the Web. Al Roomi and Li [2] performed a large-scale measurement of website login policies, focusing on password-based workflows and implementation choices such as HTTPS usage, credential-entry restrictions, password transmission, and indicators of insecure password handling. Shepherd [23] proposed an automated framework for robustly analyzing website login mechanisms at scale. Ardi et al. [5] also studied the prevalence of single sign-on (SSO) on the Web, showing that SSO can also be measured automatically at scale.

Regarding Fido2 deployment, Lassak et al. [27] analyzed the obstacles faced by companies deploying passkeys, highlighting ecosystem dependencies, rollout complexity, and other factors that can slow down the transition to passwordless authentication. The latter helps explain why modern authentication mechanisms such as Fido2 may remain difficult to observe or unevenly deployed across websites. More recently, Bhardwaj et al. [9] presented `Fidentikit`, a large-scale crawler implementing 43 heuristics to detect passkey support across the Tranco Top 100K websites [28]. While their work provides valuable insight into deployment prevalence and the role of external identity providers, it focuses exclusively on passkeys and does not consider the broader authentication landscape —- notably password-based or hybrid mechanisms. Furthermore, their analysis remains at the deployment level, without actively capturing or evaluating the cryptographic material exchanged during authentication, such as challenge entropy, length, or reuse patterns, as we did in this paper.

A parallel line of work focuses on the usability and user perception of Fido2 authentication. Lyastani et al. [29] conducted the first large-scale lab study of Fido2 single-factor authentication, showing that users are generally willing to accept passwordless authentication with security keys but identifying adoption barriers such as account recovery and technology mistrust. Würsching et al. [49] extended this analysis to smartphones, comparing platform and roaming authentication, while Owens et al. [32] specifically studied smartphones as Fido2 roaming authenticators. Lassak et al. [26] further investigated user misconceptions about Fido2 biometric authentication, and Kepkowski et al. [24] studied the practical challenges faced by IT professionals when deploying Fido2 in enterprise settings. On the formal security side, Barbosa et al. [6] provided the first provable security analysis of the Fido2 protocols, confirming the authentication security of WebAuthn while identifying weaknesses in CTAP. Collectively, these works establish Fido2 as both a usable and formally sound authentication standard, but none of them addresses the server-side quality of the cryptographic material issued during authentication flows in the wild.

These works show that authentication can be studied automatically on large numbers of websites, but they focus on specific facets of the login ecosystem, such as password-based workflows or login-page analysis. In contrast, our work targets the broader problem of identifying authentication mechanisms used by websites through a deterministic and non-intrusive inference pipeline. Rather than focusing exclusively on passwords, SSO, or Fido2, Atlas is designed to capture multiple classes of authentication schemes from observable web signals.

Finally, while Kuchhal et al. [25] evaluate the security posture of Fido2 deployments from the perspective of compromised clients — focusing on authenticator configurations and malware threats — our work takes a complementary, server-centric approach by actively capturing and analysing the cryptographic challenges issued by relying parties, enabling the assessment of server-side generation quality, challenge reuse, and replay vulnerability at scale.

9 Conclusion

This paper introduced **A**uthentication **T**echnoLogy **A**nalysi**S** (Atlas), a tool for identifying website authentication mechanisms in a deterministic and non-intrusive way, combining headless-browser automation with a multi-layered signal inference engine spanning the DOM, Shadow DOM, storage, network requests, and runtime APIs.

Evaluated against a manually crafted groundtruth, Atlas proves effective at identifying the main authentication patterns when the authentication surface is observable. In particular, Password-Based

websites are mostly well classified, and a large fraction of correctly classified FIDO2-NATIVE websites can be further validated through passkey interaction. At the same time, the groundtruth highlights that observability remains the main bottleneck: most misclassifications stem from websites classified as NONE/ERROR, corresponding either to navigation mismatches or to authentication mechanisms that are only partially exposed to automated analysis.

Applied in the wild to a large dataset of 1,000,000 websites, traditional PASSWORD-BASED authentication remains by far the dominant directly observable mechanism. FIDO2/WebAuthn-related signals appear in multiple forms, from latent support to fully exposed standalone passwordless login. Directly exposed `full_fido2` websites represent only a small fraction of the web and should be interpreted as a conservative lower bound on actual passkey deployment. Finally, a focused analysis of FIDO2/WebAuthn cryptographic challenge quality reveals that most deployments satisfy the baseline requirements of the FIDO2 specification, and some even largely exceed them. However, weaker implementations also exist, including configurations that discourage user verification and, more critically, cases of exact cryptographic challenge reuse across sessions —- a direct replay attack vector that existing passive measurement approaches cannot detect.

This last finding illustrates that ATLAS is not only useful for detecting authentication mechanisms, but can also support an initial security characterization of deployed FIDO2/WebAuthn implementations — an angle that, to the best of our knowledge, no prior large-scale study has addressed. Future work will investigate the addition of a confidence score to ATLAS classifications, to better account for partial observability and ambiguous signal combinations. We also plan to deepen the FIDO2 security analysis, notably by extending challenge capture to a larger and more diverse set of websites and by studying the temporal stability of cryptographic practices across deployments.

References

[1] 1Password. [n. d.]. Passkeys.directory. https://passkeys.directory [Last Accessed: March 26th, 2026].

[2] S. Al Roomi and F. Li. 2023. A Large-Scale Measurement of Webiste Login Policies. In *Proc. USENIX Security Symposium*.

[3] S. Al Roomi and F. Li. 2023. Measuring Website Password Creation Policies at Scale. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[4] FIDO2 Alliance. 2026. Passkey Directory. https://fidoalliance.org/passkeys-directory/ [Last Accessed: March 26th, 2026].

[5] C. Ardi and M. Calder. 2023. The Prevalence of Single Sign-On on the Web: Towards the Next Generation of Web Content Measurement. In *Proc. ACM Internet Measurement Conference (IMC)*.

[6] M. Barbosa, A. Boldyreva, S. Chen, and B. Warinschi. 2021. Provable Security Analysis of FIDO2. In *Proc. Annual International Cryptology Conference (CRYPTO)*.

[7] A. Barth. 2011. *HTTP State Management Mechanism.* RFC 6265. Internet Engineering Task Force.

[8] A. Barth. 2011. *The Web Origin Concept.* RFC 6454. Internet Engineering Task Force.

[9] P. Bhardwaj and N. Sastry. 2026. State of Passkey Authentication in the Wild: A Census of the Top 100K sites. In *Proc. Passive and Active Measurement Conference (PAM)*.

[10] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. 2012. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *Proc. IEEE Symposium on Security and Privacy (S&P)*.

[11] Buy Bitcoin Worldwide. [n. d.]. FIDO2, WebAuthn and U2F Supported Sites (Full List). https://buybitcoinworldwide.com/dongle-auth/ [Last Accessed: March 26th, 2026].

[12] Cisco. 2026. Umbrella Popularity List. https://umbrella-static.s3-us-west-1.amazonaws.com/index.html

[13] Cloudfare. [n. d.]. Turnstile. https://developers.cloudflare.com/turnstile/get-started/ [Last Accessed: January 21st, 2026].

[14] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. 2014. The Tangled Web of Password Reuse. In *Proc. Annual Network and Distributed System Security Symposium (NDSS)*.

[15] FIDO Alliance. 2019. Client to Authenticator Protocol (CTAP), Version 2.0. https://fidoalliance.org/specs/fido-v2.0-ps-20190130/ [Last Accessed: January 15th, 2026].

[16] OpenID Foundation. 2023. OpenID Connect Core 1.0. https://openid.net/specs/openid-connect-core-1_0.html [Last Accessed: February 3rd, 2026].

[17] Google. 2026. reCAPTCHA. https://docs.cloud.google.com/recaptcha/docs?hl=en [Last Accessed: January 21st, 2026].

[18] P. A. Grassi, J. L. Fenton, E. M. Newton, R. A. Perlner, A. R. Regenscheid, W. E. Burr, J. P. Richer, N. B. Lefkovitz, J. M. Danker, Y.-Y. Choong, K. K. Green, and M. F. Theofanos. 2017. *Digital Identity Guidelines – Authentication and Lifecycle Management.* NIST Special Publication 800-63B. National Institute of Standards and Technology.

[19] D. Hardt. 2012. *The OAuth 2.0 Authorization Framework.* RFC 6749. Internet Engineering Task Force.

[20] HIDEEZ. 2026. Services and Applications Supporting FIDO2/U2F Passwordless Authentication and Passkeys. https://hideez.com/pages/supported-services?srsltid=AfmBOoqZSTv1JRrgpoamjtatowiqFVZsxFMeBT1YsQp_1Lv78a-CMzwO [Last Accessed: March 26th, 2026].

[21] J. Hodges, J.C. Jones, M. B. Jones, A. Kumar, E. Lundberg, J. Bradley, C. Brand, A. Langley, G. Mandyam, N. Satragno, N. Steele, J. Tan, S. Weeden, M. West, and J. Yasskin. 2021. *Web Authentication: An API for accessing Public Key Credentials Level 2.* Recommendation 2-20210408. W3C.

[22] M. Jakobsson. 2020. Social Engineering Resistant 2FA. In *Security, Privacy and User Interaction*, M. Jakobsson (Ed.). Springer, 113 – 121.

[23] H. Jonker, S. Karsch, B. Krummow, and M. Sleegers. 2020. Shepherd: a Generic Approach to Automating Website Login. In *Proc. Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*.

[24] M. Kepkowski, M. Machulak, I. Wood, and D. Kaafar. 2023. Challenges with Passwordless FIDO2 in an Enterprise Setting: A Usability Study. In *Proc. IEEE Secure Development Conference (SecDev)*.

[25] D. Kuchhal, M. Saa, A. Oest, and F. Li. 2023. Evaluating the Security Posture of Real-World FIDO2 Deployments. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

[26] L. Lassak, A. Hildebrandt, M. Golla, and B. Ur. 2021. It's Stored, Hopefully, on an Encrypted Server: Mitigating Users' Misconceptions About FIDO2 Biometric WebAuthn. In *Proc. USENIX Security Symposium*. 91–108.

[27] L. Lassak, E. Pan, B. Ur, and M. Golla. 2024. Why Aren't We Using Passkeys? Obstacles Companies Face Deploying FIDO2 Passwordless Authentication. In *Proc. USENIX Security Symposium*.

[28] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proc. Annual Network and Distributed System Security Symposium (NDSS)*.

[29] S. G. Lyastani, M. Schilling, M. Neymayr, M. Backes, and S. Bugiel. 2020. Is FIDO2 the Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. In *Proc. IEEE Symposium on Security and Privacy (S&P)*.

[30] Microsoft. 2026. Python Playwright. https://playwright.dev/python/ [Last Accessed: January 14th, 2026].

[31] E. Naprys. 2025. 19 Billion Leaked Passwords Reveal Deepening Crisis: Lazy, Reused, and Stolen. https://cybernews.com/security/password-leak-study-unveils-2025-trends-reused-and-lazy/ [Last Accessed: January, 15th, 2026].

[32] K. Owens, O. Anise, A. Krauss, and B. Ur. 2021. User Perceptions of the Usability and Security of Smartphones as FIDO2 Roaming Authenticators. In *Proc. USENIX Symposium on Usable Privacy and Security (SOUPS)*.

[33] S. Raponi and R. Di Pietro. 2020. A Longitudinal Study on Web-Sites Password Management (in)Security: Evidence and Remedies. *IEEE Access* 8 (March 2020), 52075 – 52090.

[34] K. Reese, T. Smith, J. Dutson, J. Armknecht, J. Cameron, and J. Seamons. 2019. A Usability Study of Five Two-Factor Authentication Methods. In *Proc. USENIX Symposium on Usable Privacy and Security (SOUPS)*.

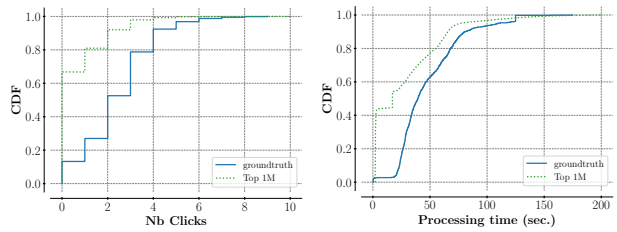
[35] K. Ruth, a. Fass, J. Azose, M. Pearson, E. Thomas, C. Sadowski, and Z. Durumeric. 2022. A World Wide Web View of Browsoing the World Wide Web. In *Proc. ACM Internet Measurement Conference (IMC)*.

[36] K. Ruth, D. Kuma, B. Wang, L. Valenta, and Z. Durumeric. 2022. Toppling Top Lists: Evaluating the Accuracy of Popular Website Lists. In *Proc. ACM Internet Measurement Conference (IMC)*.

[37] J. Schaad. 2022. *CBOR Object Signing and Encryption (COSE): Hash Algorithms.* RFC 9054. Internet Engineering Task Force.

[38] J. Schaad. 2022. *CBOR Object Signing and Encryption (COSE): Initial Algorithms.* RFC 9053. Internet Engineering Task Force.

[39] J. Schaad. 2022. *CBOR Object Signing and Encryption (COSE): Structures and Process.* RFC 9052. Internet Engineering Task Force.



(a) Number of clicks per site. **(b) Processing time per site.**

Figure 7: Interaction cost of Atlas with websites, in terms of clicks and processing time, for both the groundtruth and the large-scale dataset. A click count of 0 indicates that the website could not be meaningfully accessed, e.g., because of anti-bot defenses, headless-browser blocking, or failure to locate a login page. Long processing times are mainly due to conservative timeout values and the exploration of deeply nested shadow DOM structures.

[40] A. Senol, A. Ukani, D. Cutler, and I. Bilogrevic. 2024. The Double Edged Sword: Identifying Authentication Pages and their Fingerprinting Behavior. In *Proc. ACM Web Conference*.
[41] T. G. Tan, P. Szalachowski, and J. Zhou. 2022. Securing Password Authentication for Web-based Applications. In *Proc. IEEE Conference on Dependable and Secure Computing (DSC)*.
[42] Fido Alliance. 2013. Fido2: WebAuthn & CTAP. https://fidoalliance.org/passkeys/ [Last Accessed: January 12th, 2026].
[43] E. Ulqinaku, H. Assal, A. Abdou, S. Chiasson, and S. Capkun. 2021. Is Real-time Phishing Eliminated with Fido? Social Engineering Downgrade Attacks against Fido Protocols. In *Proc. USENIX Security Symposium*.
[44] W3C. 2024. Federated Credential Management (FedCM). https://www.w3.org/TR/fedcm/ [Last Accessed: February 3rd, 2026].
[45] W3C Working Group. 2018. *Shadow DOM*. Note 01. W3C.
[46] D. Waite, J. Bradley, M. Miller, and T. Cappalli. 2024. *Server Requirements (WebAuthn Level 2 and CTAP 2.1)*. Review Draft fido-server-v1.0-rd-20240116. Fido2 Alliance.
[47] Web Hypertext Application Technology Working Group (WHATWG). 2026. DOM Standard. https://dom.spec.whatwg.org/
[48] Web Hypertext Application Technology Working Group (WHATWG). 2026. HTML Standard — Web Storage. https://html.spec.whatwg.org/multipage/webstorage.html
[49] L. Wursching, F. Putz, S. Haesler, and M. Hollick. 2023. Fido2 the Rescue? Platform vs. Roaming Authentication on Smartphones. In *Proc. Conference on Human Factors in Computing Systems (CHI)*.
[50] Yubico. [n. d.]. Works with YubiKey catalog. https://www.yubico.com/works-with-yubikey/catalog/?series=1&sort=popular-for-individuals [Last Accessed: March 26th, 2026].

Open Science

For reproducibility purposes, Atlas source code as well as collected data are made available for the research community at https://anonymous.4open.science/r/atlas-5F72. The anonymous repository contains a `README.md` file explaining how to execute the code and how to reproduce the results discussed in this paper. Data used to compute results discussed in Sec. 7 has been anonymzed for obvious ethical and security reasons.

Ethical Considerations

Ethics strongly constrain what can be measured automatically. Any real authentication attempt, such as form submission, account creation, or Otp triggering, would introduce bias, legal risks, and non-reproducibility. We therefore implemented Atlas with a strict non-intrusive design: it does not perform any binding action, limits the number of clicks to lightweight interactions such as accepting cookie banners or navigating to login pages, and avoids spending too much time on each website. This ensures a safe and ethical analysis, but also implies that some mechanisms, for instance those revealed only after entering an email address or after deeper interaction, cannot be observed directly. These constraints are summarized in Fig. 7.

Finally, the data collected cannot be used for malicious purposes.

APPENDIX B

IAB CATEGORY MAPPINGS

Passkey Directory category	**Mapped IAB category**
Authentication Provider	Technology & Computing
Automotive	Automotive
E-Commerce	Shopping
Education	Education
Entertainment	Arts & Entertainment
Email	Technology & Computing
Finance	Personal Finance
Government	Law, Government & Politics
Health & Wellness	Health & Fitness
Identity Provider	Technology & Computing
Information Technology	Technology & Computing
Lifestyle & Leisure	Hobbies & Interests
Marketing	Business
Productivity	Business
Real Estate	Real Estate
Science & Engineering	Science
Social	Society
Social Media	Society
Social Networking	Society
Telecommunications	Technology & Computing
Travel & Leisure	Travel
Travel & Tourism	Travel

Table B.2: Category mapping between 1Password categories and IAB categories.

Hideez category	**Mapped IAB category**
Security	Technology & Computing
Authentication	Technology & Computing
ID & Access management	Technology & Computing
Password management	Technology & Computing
Hosting & servers	Technology & Computing
Communication	Society
Cloud service	Technology & Computing
Commerce	Shopping
Developer tools	Technology & Computing
Email provider	Technology & Computing
Finance & Cryptography	Personal Finance
Social Media	Society
Desktop access	Technology & Computing
Entertainment	Arts & entertainment
Education	Education
Government	Law, Government & Politics

Table B.1: Category mapping between Hideez categories and IAB categories.

BuyBitCoinWorldWide category	Mapped IAB category
Backup and Sync	Technology & Computing
Banking	Personal Finance
Betting	Hobbies & Interests
Cloud Computing	Technology & Computing
Communication	Technology & Computing
Cryptocurrencies	Personal Finance
Developer	Technology & Computing
domains	Business
Education	Education
Email	Technology & Computing
Entertainment	Arts & Entertainment
Finance	Personal Finance
Food	Food & Drink
Gaming	Hobbies & Interests
Government	Law, Government & Politics
Health	Health & Fitness
Hosting/VPS	Technology & Computing
Identity Management	Technology & Computing
Investing	Personal Finance
IoT	Technology & Computing
Legal	Law, Government & Politics
Other	Uncategorized
Payments	Personal Finance
Remote Access	Technology & Computing
Retail	Shopping
Security	Technology & Computing
Social	Society
Self-hosted Software	Technology & Computing
Task Management	Business
Transport	Travel
Utilities	Technology & Computing
VPN Providers	Technology & Computing

Table B.3: Category mapping between BuyBitCoinWorldWide categories and IAB categories.