Digital Forensic Artifacts of FIDO2 Passkeys in Windows 11

Patricio Domingues*
School of Technology and Management, Polytechnic Institute of Leiria - Instituto de Telecomunicações
Leiria, Portugal
patricio.domingues@ipleiria.pt

Miguel Frade*
School of Technology and Management, Polytechnic Institute of Leiria - Computer Science and Communication Research Centre
Leiria, Portugal
miguel.frade@ipleiria.pt

Miguel Negrão
School of Technology and Management, Polytechnic Institute of Leiria - Computer Science and Communication Research Centre
Leiria, Portugal
miguel.negrao@ipleiria.pt

ABSTRACT

FIDO2's passkey aims to provide a passwordless authentication solution. It relies on two main protocols – WebAuthn and CTAP2 – for authentication in computer systems, relieving users from the burden of using and managing passwords. FIDO2's passkey leverages asymmetric cryptography to create a unique public/private key pair for website authentication. While the public key is kept at the website/application, the private key is created and stored on the authentication device designated as the authenticator. The authenticator can be the computer itself – same-device signing –, or another device – cross-device signing –, such as an Android smartphone that connects to the computer through a short-range communication method (NFC, Bluetooth). Authentication is performed by the user unlocking the authenticator device. In this paper, we report on the digital forensic artifacts left on Windows 11 systems by registering and using passkeys to authenticate on websites. We show that digital artifacts are created in Windows Registry and Windows Event Log. These artifacts enable the precise dating and timing of passkey registration, as well as the usage and identification of the websites on which they have been activated and utilized. We also identify digital artifacts created when Android smartphones are registered and used as authenticators in a Windows system. This can prove useful in detecting the existence of smartphones linked to a given individual.

CCS CONCEPTS

• **Security and privacy → Systems security**; • **Applied computing → Evidence collection, storage and analysis**.

KEYWORDS

Digital Forensics, Passkeys, FIDO2, Windows 11, Windows Registry, Windows Event Log

ACM Reference Format:
Patricio Domingues, Miguel Frade, and Miguel Negrão. 2024. Digital Forensic Artifacts of FIDO2 Passkeys in Windows 11. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30-August 2, 2024, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3664476.3664496

*All authors contributed equally to this research.

ARES 2024, July 30-August 2, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07
https://doi.org/10.1145/3664476.3664496

1 INTRODUCTION

The username/password pair has been an authentication method since the need arose to identify users to access computer systems. While simple to implement, this method suffers from numerous security problems on both the user side (phishing, weak passwords, reuse, etc.) and the server side (poor hashing, data breaches, etc.) [22]. Methods such as multiple factors authentication (MFA) do exist to strengthen password-based authentication, but they trade usability for security, and thus, a vast percentage of users avoid them [25]. Moreover, malicious agents develop social engineering and other attack techniques [17], such as the so-called *TOTP/push bombing* to circumvent these defenses [31].

The *Fast IDentity Online* (FIDO) [6] alliance comprises a large number of technological companies and has been establishing standards for authentication. One of these standards is FIDO2, often referred to as *FIDO2's passkeys*, or *passkeys*. The FIDO2 passkeys standard aims to provide passwordless solutions to support both device sign-in and cross-device sign-in, transitioning authentication from something the user knows to something the user knows and possesses, such as a smartphone and/or biometrics (fingerprints, face recognition, swipe pattern, PIN code, etc.) to unlock the so-called *authenticator device*. The authenticator can be a computer, a mobile device, a USB security key, or a password manager. Note that FIDO2 passkeys require that the authenticator device – laptop/desktop computer, iOS or Android smartphone – has the screen lock function enabled. FIDO2's passkey-based user authentication relies on asymmetric cryptography.

FIDO2's passkey is based on two protocols: *i*) *Web Authentication*, and *ii*) *CTAP2*. Web Authentication (henceforth *webAuthn*) is a W3C's web authentication protocol [1], while CTAP2 is the 2^{nd} version of the Client-to-Authenticator protocol [18]. The main goal of FIDO2's passkeys is to deliver a passwordless standard that provides secure users' authentication, without the need for users to remember, save, or type passwords. Instead, access to a given site or application by the user requires an authenticator device, which needs to be registered within the remote site/application, following FIDO2's protocol. The authenticator device can be either the device where access to the remote site is sought, such as a personal computer or mobile device, or an external device like a security key, smartphone, or tablet. The former case is referred to as *same-device* authentication, while the latter, which relies on an external device, is termed *cross-device* authentication. Note that a smartphone/tablet

can serve as a same-device authenticator when the user initiates a session directly with the remote site using this mobile device and can function as a so-called *roaming authenticator* when the session is initiated on another device (e.g., desktop or laptop), with the mobile device solely serving as the authenticator [23]. In all cases, a roaming authenticator serves as a proximity verification device, as only short-range communication protocols such as Bluetooth and Near Field Communication (NFC) are permitted between the device running the browser requesting access to the remote site and the authenticator device.

FIDO2's passkeys have only recently been integrated into some major OS platforms, namely Android 9+ [15], Apple's iOS 16+ [16], and Microsoft Windows, the latter through Windows Hello [29]. Likewise, some of the main browsers support the FIDO2's passkeys standard – Google Chrome and Chromium derivatives such as Brave and Edge, besides other popular browsers such as Firefox and Safari. Google has recently made passkeys a default option across personal Google Accounts and has supported passkey upgrades in the Google Pixel smartphone [19]. Google reported that as of April 2024, passkeys have been used to authenticate users more than 1 billion times across over 400 million Google Accounts [9]. Password managers such as Bitwarden [12] and 1password [7] have also added support for passkeys. Others observe that although passkeys usage brings some benefits, it is not without hurdles [13], with some reporting on possible security attacks [10, 34] and the obstacles hindering the deployment and usage of passkey-based authentication systems in a wider scale [24]. Nonetheless, the support for passkeys has been growing, with major companies allowing access to their services through passkeys. Examples include Adobe, Amazon, Apple, Microsoft, Nintendo, NVidia, PayPal, and TikTok, to name a few [8].

In this paper, we report on the digital forensic artifacts left in Windows 11 when one of the following browsers – Google Chrome, Mozilla Firefox, and Microsoft Edge – is used as a client for registration/authentication through FIDO2's passkeys. Our main goal is to provide digital forensics practitioners with the knowledge to help exploit the digital fingerprinting of FIDO2's passkeys under Windows 11. We believe that the main contributions of this paper are:

- Description and analysis of the forensic artifacts of FIDO2's passkeys in a Windows 11 system.
- Highlighting the examination of certain Windows Registry keys to uncover roaming authenticator devices configured for passkey registration/authentication, such as smartphones.
- Identification and report of a privacy glitch in Mozilla Firefox's logging behavior to Windows Log Event when using passkeys within a private browsing session.

To our knowledge, this is the first paper to document and analyze the digital forensic artifacts related to FIDO2's passkey registration and usage for both same-device and cross-device user sign-in within a Windows environment.

The remainder of this paper is organized as follows. Section 2 summarily describes FIDO2's passkeys and its main operations, while section 3 details Windows 11's digital forensic artifacts linked to passkey usage. Section 4 discusses the main finding of the paper. Lastly, Section 5 concludes the paper and suggests directions for future research.

2 BACKGROUND

We provide background on the FIDO2 standard, focusing on its core protocols and the Windows OS passkeys interface.

2.1 FIDO2 Protocols

As stated earlier, FIDO2's standard relies on two protocols: *Web Authentication* and *CTAP2*. As these protocols are essential to FIDO2, we briefly examine each one.

2.1.1 Web Authentication. Web Authentication is a W3C specification [1]. As the *web* prefix suggests, WebAuthn is a browser-based API that implements passwordless access. The API is instrumental for both the registration and authentication of FIDO2. The protocol involves three parties:

(1) The *user* who is the individual that is trying to authenticate, resorting to a FIDO client, that is, a browser and the underlying OS.
(2) The *relying party* (RP), which is the FIDO2 designation to identify the website the user is trying to authenticate to access resources and services.
(3) The *authenticator*, which can be a piece of software or a physical device such as a smartphone, tablet, or security key, manages and stores the user's cryptographic keys essential for authentication

Note that live analysis of the WebAuthn protocol can be performed with the *WebDevAuthn* tool, a FIDO2/WebAuthn requests and responses analyser web tool [33].

2.1.2 Client To Authenticator Protocol. The *Client To Authenticator Protocol*, version 2 (CTAP2) standardizes the communication between the client, that is the web browser, and the authenticator, which needs to be done over a secure channel. This standardization ensures interoperability across platforms and devices. In addition to managing FIDO2 registration and authentication, CTAP2 also defines guidelines for handling the cryptographic keys used by FIDO2 credentials. This encompasses the processes for generating the cryptographic keys, securely storing them, and implementing robust security measures to protect them.

From the point of view of a user, there are two main stages in FIDO2: *i*) a one-time registration and *ii*) authentication. Authentication occurs whenever the user needs to log in through FIDO2.

2.2 Registration

Registration is the one-time preliminary step that a user needs to perform to activate FIDO2's passkey with a given website, registering an account with a device – the authenticator – to use for future authentication. Registration starts with the user visiting the website that supports FIDO2 authentication and registering by supplying a username. The website creates a unique cryptographic challenge for the user/device pair. It dispatches it to the user's browser, which forwards it to the authenticator device, which can be the system where the browser is running – platform authenticator, such as *Windows Hello* and Android WebAuthn API [3] – or an external device – roaming authenticator – such as a smartphone, a tablet,

or a USB security key. In the authenticator device, the user selects the authentication method – biometrics, pin code, swipe pattern, security key, etc. – and then uses it to authenticate. This allows the authenticator to generate a registration response, which includes a pair of public/private keys, and to send the public key to the user's browser along with an attestation, which forwards these data to the website. The website then registers the response with the public key within the FIDO2 server. This finalizes the registration stage. A simplified overview of the WebAuthn protocol follows: 1) The Relying Party (RP) sends a request to the user's browser to register a new credential; 2) The browser prompts the user to enrol the authenticator; 3) The authenticator generates a new cryptographic key pair, registers the public key along with some additional information, and then sends it to the browser; 4) The browser transmits the registered credential to the RP.

2.3 Authentication

Logging into a website or application using FIDO2's passkeys involves the following sequence of steps:

(1) The user visits the website and initiates the FIDO2's authentication sequence, the relying party sends a cryptographic authentication challenge to the user's browser;
(2) The browser contacts the registered authenticator forwarding the challenge;
(3) The authenticator prompts the user to authenticate him/herself relying on the registered method – PIN, biometrics, security key, etc.;
(4) The authenticator generates a response to the challenge signed with the private key;
(5) The authenticator sends the response to the browser;
(6) Finally, The browser submits the response to the remote server, which validates it by verifying the signature. If the response is accepted as valid, the server authenticates the request, granting access to the requester.

2.4 FIDO2's passkeys in Windows

As stated, Windows 11 provides FIDO2's passkeys, supporting the WebAuthn and CTAP2 protocols. For this, Windows provides two main graphic interfaces: *i*) an interface for selecting the authenticator that can be used for both registration and authentication operations, shown in Figure 1; *ii*) an interface named "Passkey settings" to list and delete the saved passkeys in Windows 11/22H2 and onward, displayed in Figure 2.

3 DIGITAL FORENSIC ARTIFACTS

We now present the main digital forensic artifacts left in Windows 11 computers when FIDO2's passkey is used for registration/authentication at a remote site. First, we outline our study's research methodology and tools. Then, we report on the main digital forensic artifacts.

3.1 Methodology

The methodology to study the effects of FIDO2 on Windows 11 machines involved performing FIDO2 registration and authentication

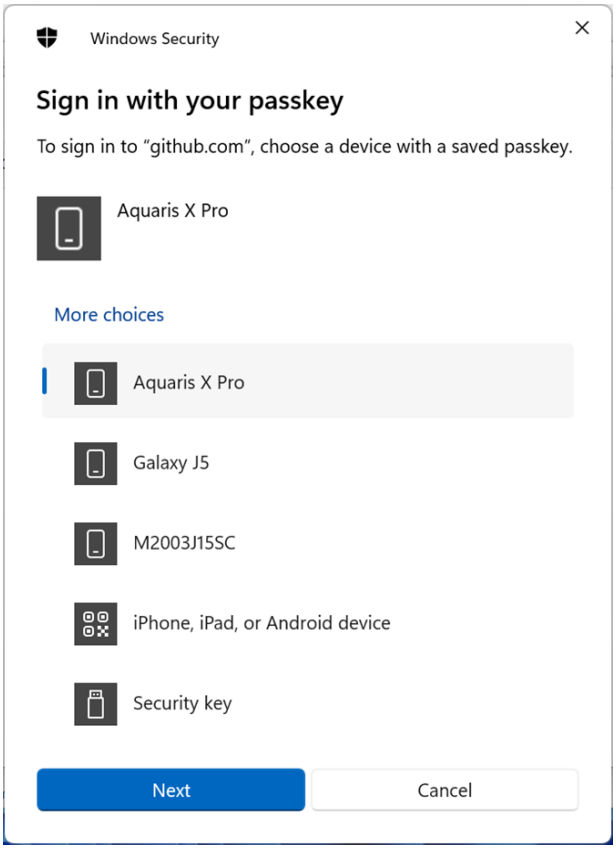

Figure 1: Window displayed by the OS to select the proper passkey device during an authentication operation

operations using both platform authenticators and roaming authenticator devices. To monitor the interactions and effects on the filesystem, network and Windows Registry, we resorted to several tools, namely, two Sysinternals' tools: `System Monitor (sysmon)` [28] and `Process Monitor (procmon)` [27]. The former records several events on a Windows Event Log file according to a given XML configuration file. We used the `sysmonconfig-excludes-only.xml` configuration file [20] The registry analysis was performed with Windows' *Registry Editor*. In addition, we also monitored Windows' event log directory – `C:\Windows\System32\winevt\logs` – which is the directory where Windows Log Events files (extension `.evtx`) are kept, assuming the conventional installation of Windows on the `C:` drive, as is commonly encountered. We also resorted to the online tool `cyberchef.io` [2] to decode hexadecimal content. The experiments were conducted on systems with *Windows 11 22H2 - OS Build 22621.2861* and *Windows 11 23H2 - OS Build 22631.2861*.

No meaningful differences were detected between the two versions of Windows 11. We performed experiments for registration and authentication, employing two types of authenticator: *i*) same-device signing through Windows Hello and *ii*) cross-device signing resorting to an Android Xiaomi M2003J15SG with Android 12. We also tested two other Android smartphones – an Aquaris X Pro/Android 8.1 and a Samsung J5/Android 9. However, in both devices,

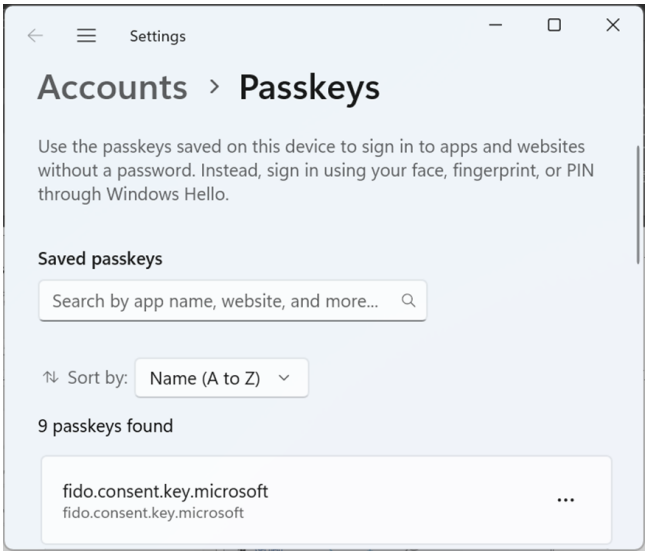

Figure 2: Windows interface to manage passkeys saved in the local machine

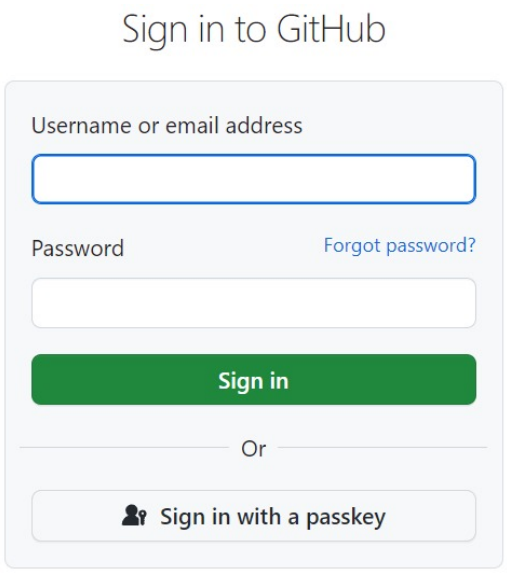

Figure 3: Github sign-in prompt with passkey option.

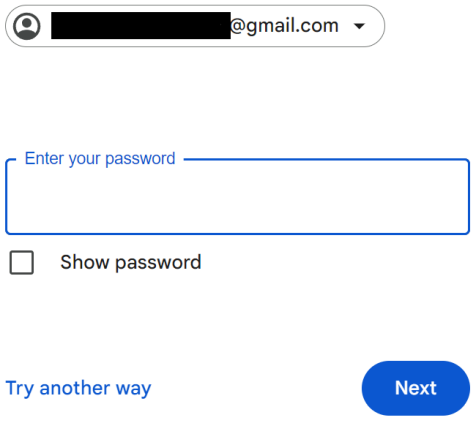

Figure 4: Google sign-in prompt with passkey option ("Try another way").

the respective Android versions do not support the role of FIDO2's roaming authenticator. Nonetheless, as we shall see, even for these non-viable Android devices, some digital forensic artifacts were produced in the Windows Registry.

In total, we performed four main experiments, two for registration and another two for authentication. Furthermore, as web browsers used for FIDO2's passkeys registration/authentication influence some of the artifacts, we resorted to the three main web browsers: *i*) Google Chrome; *ii*) Microsoft Edge; and *iii*) Mozilla Firefox. Table 1 lists the software tools used in the experiments. As FIDO2's passkey-ready site, we used two sites whose focus is passkey testing – `webauthn.io` and `passkeys.io`. We also resorted to two well-known and widely used services: `GitHub` and `Google`. Note that these services have different approaches in their FIDO2's web authentication. Google has a two-step authentication, with the user filling in his/her account username in the first step, and then clicking the "next" button to access the second step, where the password is sought. If the account is configured for passkey access, the option "Try another way" is activated and clicking in it allows for passkey authentication, as shown in Figure 4. This means that a FIDO2's device authenticator can be configured for accessing more than one Google account. Conversely, passkey-based authentication in `github.com` is single-step, as the selection of "Sign in with a passkey", in the web authentication interface (Figure 3) goes directly to the Windows Security popup. There, the user has to select the passkey authenticator device, and thus the login is performed with the username that was registered with the passkey. This way, GitHub's passkey effectively mandates that only a single GitHub account can be associated to an authenticator device. Interestingly, Google's "Try another way' option in the login screen means that one can assess whether a given Google account is configured for passwordless access or not.

3.2 Windows Event Log Forensic Artifacts

Windows Event Log is Windows's system log and encompasses, in Windows 11, over 200 EVTX files located at the `%SystemRoot%\System32\Winevt\Logs\` directory [21]. Each EVTX file corresponds to a log file that records log entries designated as log events. An important metadata of a log entry is the `Event ID`, a 16-bit unsigned integer associated with the nature of a logged event. Our analysis refers to the events through their respective Event ID.

3.2.1 Microsoft-Windows-WebAuthN%4Operational.evtx log. In Windows, the `Microsoft-Windows-WebAuthN%4Operational.evtx` file (henceforth, *webAuthN EVTX*) records Web Authentication related events and thus registers a significant amount of events when

Table 1: Software tools used in the study of passkeys digital forensic artifacts in Windows 11

Name	Version
Registry Editor	Windows 11
Windows Event Viewer	Windows 11
wevutil	Windows 11
System Monitor	15.11
Process Monitor	3.96
Cyberchef.io	9.37.3
Google Chrome	120.0.6099.217
Microsoft Edge	120.0.2210.121
Mozilla Firefox	123.0

FIDO2's operations are performed. We analyze the most meaningful of these recorded events for the registration and authentication operations.

FIDO2's operations are verbosely logged in Windows, with many messages recorded in the *webAuthN EVTX* log file. There are two main sources for the log messages: *i*) services of the OS and *ii*) the web browser used in FIDO2's registration/authentication process. Table 2 shows the log's recorded messages per operation/web browser.

Log verbosity depends on three factors: *i*) web browser; *ii*) whether it is a same-device or a cross-device operation; *iii*) whether browser private mode is used or not. While variations across web browsers are not meaningful, cross-device operations generate more log messages due to the additional CTAP2 interaction between the client device and the roaming authenticator.

Identifying data. A significant number of messages saved in *webAuthN EVTX* hold the user's Windows *Security Identifier* (SID), thus allowing to identify the Windows' account used to perform the logged operation, be it a registration or an authentication (line 22 of Listing 1). The SID is present in all browsers' log messages, even in private mode. Furthermore, FIDO2's operations have a `TransactionId`, which is a globally unique identifier (GUID), such as `069d40bb-8daa-456f-9b5d-1c2be41c12da` (line 8 of Listing 1). This GUID is unique to a specific registration or authentication session, but not all events in the log will have it. Nonetheless, this GUID allows to track a particular FIDO2 operation within the event log file.

Private Mode. Some of the logged messages carry data that can be considered private, such as the *Relying Party*, the email address used as a login, and the model of the roaming authenticator (e.g., "Galaxy J5" for a Samsung Galaxy J5 smartphone) in cross-device operations. This explains why, in private mode, browser logs have a much more reduced presence in *webAuthN EVTX*, with the few logged messages holding no personal data apart from the user's Windows security UserID. The exception is Mozilla's Firefox, which produces the same log output despite being in private mode, as shown in Table 2. We believe this is an oversight and have submitted a bug report to Bugzilla.

Listing 1: A XML record documenting the passkey authentication event ID 1103 for the `github.com` website (edited for conciseness and readability)

```
1  <?xml version="1.0" encoding="utf-8" standalone="yes"?>
2  <Events>
3    <Event
4      xmlns='http://schemas.microsoft.com/win/2004/08/
5  events/event'>
6      <System>
7        <Provider Name='Microsoft-Windows-WebAuthN'
8        Guid='{069d40bb-8daa-456f-9b5d-1c2be41c12da}'/>
9        <EventID>1103</EventID>
10       <Version>0</Version>
11       <Level>4</Level>
12       <Task>103</Task>
13       <Opcode>12</Opcode>
14       <Keywords>0x8000000000000007</Keywords>
15       <TimeCreated SystemTime='2024-02-21T16:31:19.7597Z'/>
16       <EventRecordID>6469</EventRecordID>
17       <Correlation/>
18       <Execution ProcessID='12476' ThreadID='17984'/>
19       <Channel>Microsoft-Windows-WebAuthN/Operational
20 </Channel>
21       <Computer>PatinhasAMD</Computer>
22       <Security UserID='S-1-5-21-2933784364-(...)'/>
23     </System>
24     <EventData>
25       <Data Name='TransactionId'>{069d40bb-8daa-456f
26 -9b5d-1c2be41c12da}</Data>
27       <Data Name='RpId'>github.com</Data>
28       <Data Name='ClientDataHashAlgId'>SHA-256</Data>
29       <Data Name='ClientDataLength'>131</Data>
30       <Data Name='ClientDataHashLength'>32</Data>
31       <Data Name='ClientDataHash'>75DAD7(...)/Data>
32       <Data Name='CredentialCount'>0</Data>
33       <Data Name='RequestLength'>55</Data>
34       <Data Name='Request'>02A3016A(...)</Data>
35     </EventData>
36     <RenderingInfo Culture='en-150'>
37       ...
38     </RenderingInfo>
39   </Event>
40 </Events>
```

Even though the number of event log messages varies across browsers, they all resort to the same `EventIDs` for the messages deemed pertinent regarding forensic significance. Next, for each of the four analysed experiments, we describe the most relevant messages from the digital forensic point of view and enumerate the most meaningful `EventIDs` for FIDO2's registration and authentication operations.

Registration Operation. Table 3 lists the Event IDs of relevant forensic messages logged by Windows in the webAuthN EVTX file during a registration operation. The messages are shown in chronological order of appearances. The leftmost column identifies the Event ID, the center column refers to registrations with *same device* authenticators, and the rightmost column documents registrations with *roaming* authenticators. Note that some event log messages have a payload kept in hexadecimal format. Although the payload is binary, when decoded to UTF-8, one can observe the values of some relevant parameters that appear among binary chunks. For example, the decoded payload of messages with Event ID 1101 contains the username/email address whose FIDO2's passkey is sought at the remote site, as well as the remote site (e.g., `www.passkeys.io`) and the

Table 2: Number of logged events per operation of Windows's FIDO2 per browser and browser mode

# of events per operation	Registration same-device authenticator	Registration roaming authenticator	Authentication same-device authenticator	Authentication roaming authenticator
Chrome	19	35	17	42
Chrome (private)	1	1	1	6
Edge	18	35	16	41
Edge (private)	1	2	1	12
Firefox	18	35	16	46
Firefox (private)	18	35	16	46

domain of the remote site (`passkeys.io`). Figure 5 gives an example. Additionally, for Google accounts, the *Google account ID* is also saved in the payload of event 1001, labelled as *GOOGLE_ACCOUNT*, as it is visible in Figure 5. This event also has the explicit field`RpId`, which identifies the *Relying Party*, that is, the remote site.

The AAGUID – *Authenticator Attestation Global Unique Identifier* – is a field in event 1102. An AAGUID is a GUID identifier assigned to a FIDO2 authenticator by its manufacturer. For instance, the AAGUID shown in Table 3 – ea9b8d66-4d01-1d21-3ce4-b6b48cb575d4 – is reported as identifying the `GooglePasswordManager`, in the AAGUIDs site, although as *unofficial passkey blob* [4, 30]. Note that `GooglePasswordManager` intervenes in saving Android's passkeys to Google Clouds. This allows users with a given Google account to synchronize their passkeys to other Android devices, provided the same Google account is used on the devices [11].

Authentication Operation. Table 4 lists the main events recorded in the *webAuthN EVTX* file when a FIDO2-based authentication occurs. The *Same device authentication* column shows the main data for a same-device authentication (e.g., Windows PIN), while the rightmost column displays the data for a cross-device authentication. Similar to the registration procedure, a same-device FIDO2 authentication generates no more than 20 event log entries. In contrast, depending on the browser, cross-device authentication ranges from 41 to 46 new messages in the webAuthN EVTX log file, as shown in Table 2. Through event 1103 it is possible to identify the remote site name, via the RpId field. Listing 1 shows the XML for event 1103, where the RpId field is visible in line 23. Contrary to the registration operation, the account's username is not recorded in the log entry when an authentication operation is performed. However, for Google accounts, similar to registration, the *Google Account ID* can be found within the hexadecimal payload of event 1104, labelled `GOOGLE_ACCOUNT`, but without the associated username.

3.3 Windows Registry Forensic Artifacts

The Windows Registry hosts a plethora of valuable data for digital forensics [14, 32]. This is also the case for FIDO2's authentication. Data about roaming authenticators are kept in the registry under the `HKEY\USERS` hive. Specifically, for each smartphone used as authenticator, a registry entry exists under the path `HKEY\USERS\S-1-5-20\Software\Microsoft\Cryptography\FIDO\SID\LinkedDevices\`, where SID corresponds to the user SID (e.g., `S-1-5-21-10digits-10digits-10digits-1001`). In this entry, there is a subentry for each external device attempted to be set as the authenticator. Note that even devices that cannot act as authenticators, for instance, a smartphone that runs an Android version that does not support FIDO2's passkey but that a user has attempted to configure as a roaming authenticator have a dedicated entry in the registry. The subentry has two keys, called `name` and `data`. The first one holds the name identifying the brand/model of the device, such as *Aquaris X Pro* or *Galaxy J5*. The `data` key contains a blob of around 1 000 bytes of binary data that we could not decode, although we observed that the first 24 bytes are common to all the entries studied. Moreover, this byte pattern – `01000000D08C9DDF0115D1118C7A00C04FC297EB01000000` – seems to indicate that the whole blob corresponds to an encrypted `SecureString`. A `SecureString` is a data type available under the .Net framework, which cannot be displayed, hence the `SecureString` designation [26]. Under PowerShell, a `SecureString` with content *TEST* can be created with the cmdlet `ConvertTo` fitted with the command line options `-SecureString`, `-String`, `-AsPlainText`, and `-Force`. An example is given in line 1 of Listing 2, with the string "TEST" converted to a `SecureString` and kept in the `secS` PowerShell variable. The `SecureString secS` is then encrypted through the cmdlet `ConvertFrom-SecureString`, as shown in line 3 of 2. The result, kept in the `encryptedS` variable, has the same 24 initial bytes observed in the `data` field of the registry keys. Note that the `ConvertFrom-SecureString` command supports an additional `-key<value>` parameter. However, if this parameter is specified, the common set of 24 bytes no longer occurs.

Listing 2: PowerShell commands to create and encrypt a .NET SecureString (edited to preserve space)

```
1 $secS = ConvertTo-SecureString -String "TEST"\\
2 -AsPlainText -Force
3 $encryptS = ConvertFrom-SecureString -SecureString $secS
4 echo $encryptS
5 01000000d08c9ddf0115d1118c7a00c04fc297eb01000000|c5f2(...)
```

Figure 6 illustrates the registry entries, highlighting the data for the smartphone *Aquaris X Pro*. Windows uses these registry entries to create the list of authenticators when a user requests a FIDO2's passkey authentication so that one can be selected. Figure 1 exemplifies the list as displayed by Windows.

It is important to note that the identifier *S-1-5-20* that exists in the path of the FIDO registry keys corresponds to the identifier

Table 3: Main Event ID for FIDO2's same-device (left) and cross-device (right) registration

ID	Same-device registration	Cross-device registration
1000	"WebAuthN Ctap MakeCredential started." First event of the registration. Contains Transaction ID.	Idem.
1101	"Cbor encode MakeCredential request." Field RpId=remote site name. Hex payload contains username and remote site name.	Idem.
2106	"Ctap Name: ImageName". It records the full path of the web browser (example: (...)\Firefox64\firefox.exe)	Idem.
2104	"Ctap DeviceInfo." ProviderName: MicrosoftPlatformProvider (identifies as same device authenticator)	"Ctap DeviceInfo." ProviderName: MicrosoftCtapHybridProvider (identifies as a cross-device authenticator) AAGUID is all zeros ({00000000-0000-0000-0000-...}).
1102	"Cbor decode MakeCredential response." PublicKey: holds the public key (hex format). AAGUID is all zeros ({00000000-0000-0000-0000-...})	"Cbor decode MakeCredential response". PublicKey: holds the public key (hex format) AAGUID is filled (example: ea9b8d66-4d01-1d21-3ce4-b6b48cb575d4, which identifies "Google Password Manager").

```
.§.X wÐA.Ác5.xGÜÖx8.Ê.¨ã.<.ÇPñí«!Ø
¸..¢bidjgoogle.comdnamefGoogle.
£bidX$GOOGLE_ACCOUNT:xxxxxxxxxxxxxxxxxxxxxdnamex.xxxxxxxxxxxxxxxxxxx@gmail.comkdisplayNamer
Patricio Domingues..¢calg&dtypejpublic-key¢calg9..dtypejpublic-key..£bidX S©Ãp§.SÑ.©èDÁ.
[1.a.ð..Ð¼33o...0.dtypejpublic-keyjtransports.7£bidX
W®Ñ$Vî..Z.>Z..d6w6.)¶Gw.'s|.>r²ldtypejpublic-keyjtransports.7£bidP.@VÇ-
x¸.vI·Ð...Èdtypejpublic-keyjtransports.7.¡kcredProtect..¡brkõ
```

Figure 5: Hexadecimal payload of an Event ID 1101 decoded as UTF-8 (redacted to preserve privacy)

of the Windows Network Service account. In terms of physical location, this means that the FIDO registry keys are located in the registry file of the Windows network account. This file is named `NTUSER.DAT` and it is located in `c:\Windows\ServiceProfiles\NetworkAccount\`. Therefore, when performing a forensic analysis on a forensic image of a Windows disk, this is the registry file that the forensic practitioner needs to target.

4 DISCUSSION

The usage of FIDO2's passkeys in a Windows 11 system leaves digital forensic artifacts that, properly analyzed by the forensic team, can unlock other sources of data. Indeed, by analyzing the *webAuthNEVTX* log file, the forensic practitioner can find records pointing out that authentication to cloud accounts is enabled through same-device authenticator(s). In scenarios where the system PIN code serves as the same-device unlock method, and if a forensic practitioner is aware of the Windows system's PIN code, it becomes feasible to gain access to resources protected by FIDO2's passkey (such as a cloud-based account) through passkey. Moreover, even if the use of passkeys to access the resources is not viable, for instance, the same-device authenticator is protected by unavailable biometrics, or a legal authorization to access the resources cannot be obtained, knowing the existence of such resources can still be relevant for the investigation.

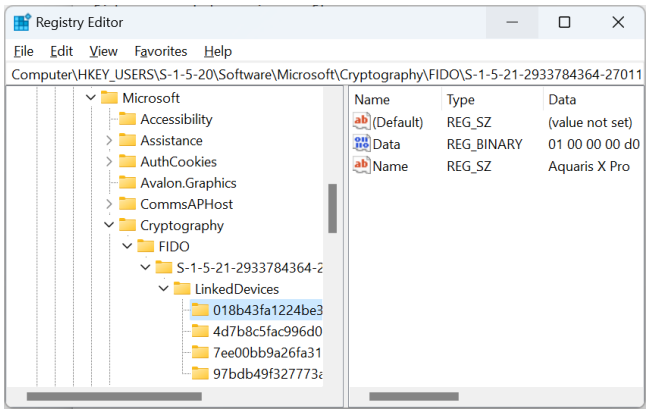

Figure 6: FIDO2 registry keys of devices set as roaming authenticators

Identifying the presence of a registered roaming authenticator is also pertinent in digital forensic analysis. This can be done either

Table 4: Main Event ID for FIDO2's same-device (left) and cross-device (right) authentication

ID	Same-device authentication	Cross device authentication
1003	"Cbor encode MakeCredential request." Contains Transaction ID.	Idem.
1103	"Cbor encode GetAssertion request." RpId: remote site name (also present in the hexadecimal payload)	Idem.
2106	"Ctap Name: ImageName". It records the full path of the web browser (example: (...)\Edge\Application\msedge.exe)	Idem.
2104	"Ctap device info." ProviderName: "MicrosoftPlatformProvider"; TransactionID; DevicePath:(empty); Manufacturer:(empty); Product:(empty); AAGuid:all zeros; U2fProtocol:false.	"Ctap device info."; TransactionID; ProviderName: "MicrosoftCtapHybridProvider" DevicePath: (smartphone model) Manufacturer: (empty); Product: (empty); AAGuid: all zeros; U2fProtocol:false.
1104	"Cbor decode GetAssertion response." CredentialID Hexadecimal payload: - GOOGLE_ACCOUNT: (21-digit-number)	"Cbor decode GetAssertion response". CredentialID; Hexadecimal payload: PublicKey
1004	"WebAuthN Ctap GetAssertion completed." - TransactionID	Idem.

by finding the appropriate entries in the *webAuthN EVTX* log file or by examining the registry for `FIDO\SID\LinkedDevices\` keys as detailed in Section 3.3. The former method is preferred, as it yields the URL of the remote service and the username if a log entry for the registration operation is found. Furthermore, if the roaming authenticator is available, it could be used to access the account for which it is registered. Even if no passkey-related entry in *webAuthN EVTX* log file is exploitable, the presence of `FIDO\SID\LinkedDevices\` registry keys indicates the existence of external authenticator(s) that can be attributed to a given Windows account through the corresponding SID. This can be, for instance, a smartphone that was not yet known by the authority pursuing the forensic analysis. This way, potentially interesting devices can be uncovered. It is important to note that a registry key for a device does not necessarily mean that the device was or can be used as an authenticator. For instance, it is possible to try to register an incompatible Android smartphone as a roaming authenticator in Windows. Despite being unsuitable, Windows Registry will still list the smartphone as a usable device during authentication attempts.

Entries of authentication operations recorded in the *webAuthN EVTX* Windows event log file can help practitioners establish a timeline of login operations to the remote site by the user identified by the Windows' SID. This, too, can prove helpful in an investigation. Given that the RpId field of event 1103 identifies the remote site name, it is possible to distinguish between login operations to different remote sites. For comparison, using username and password authentication in browsers does not generate entries in the Windows event log. However, it is possible to infer a login event from the browser history if the website employs distinct URLs for the login process and subsequent access post-login (e.g., https://accounts.google.com/v3/signin vs. https://www.google.com). This method necessitates a database of login URLs. Additionally, one might retrieve a login timestamp from a session cookie, though this approach demands understanding the cookie's internal structure. In either case, extracting a login timestamp in these ways requires specific knowledge pertinent to each website. On the other hand, obtaining a login timestamp through passkey entries in the Windows event log is a universal method that applies to any website, offering a more generalized approach.

A weak spot in FIDO2's passkeys is its limited availability. The reduced number of passkey authentication providers is mostly due to the solution's novelty. However, more services are expected to adopt the passkey authentication method. Furthermore, more than 110 sites are listed as supporting FIDO2 passkeys at the time of this writing, and this number is growing [8]. Additionally, the support and endorsement of passkeys by major OS and web browsers also contribute to increasing passkey adoption [15, 19, 29].

A limitation of relying on the *webAuthN EVTX* log file for digital forensic artifacts lies in its limited maximum size. Event logs in Windows are managed as circular buffers, with old entries replaced by newer ones when the log maximum file size is reached. In Windows 11 22H2 and 23H2, by default, the maximum size for the Windows' *webAuthN EVTX* log file is 5 MiB. Based on the average size of 850 bytes per message logged in the webAuthN EVTX log that we observed in our testing environment, and considering the most demanding scenario – cross-device authentication that triggers 46 event messages per authentication – we believe that at least 130 authentication sessions can be recorded in the *webAuthN EVTX* log before any rollover starts to overwrite oldest entries. Advanced users might employ anti-forensic measures to control or eliminate the *webAuthN EVTX* log file. For instance, an anti-forensic tool such as the open source `BleachBit` [5] can delete all Windows Event

Log files. A more subtle anti-forensic approach is to decrease the maximum allowed size for the *webAuthN EVTX* log file, resulting in fewer messages being retained in the log file.

5 CONCLUSION

This paper documents the key digital forensic artifacts left by using FIDO2's passkeys in Windows 11. We believe this is the first study dedicated to the forensic analysis of FIDO2 passkey artifacts within the Windows operating system. Both registration and authentication operations with FIDO2's passkeys in Windows 11 leave important trails of log entries in the *webAuthN EVTX* log file.

From a digital forensic perspective, the most valuable entries left by a passkey registration operation in the *webAuthN EVTX* log file are events with ID 1101, 2106, and 2104. These events record data such as the username/email address of the account and the browser used for the operation. During cross-device registration, the model name of the roaming authenticator device is also saved, as well as a meaningful AAGUID. All of this comes with an associated timestamp and a Windows SID account identifier. A FIDO2-based authentication operation triggers 18 to 46 log entries in *webAuthN EVTX* depending on whether it is a same- or cross-device authentication. The most meaningful entries are the ones with ID 1103, 2106, 2104, and 1104. These log entries also carry the date/time, the SID of the Windows account that performed the authentication attempt, and the remote site URL, although they lack the remote username/e-mail address. Still, for Google accounts, the Google numeric account ID is logged.

Another relevant artifact yielded by the usage of FIDO2 lies in the Windows Registry, more precisely in the `FIDO\\SID\LinkedDevices\\` registry keys. This artifact is only created when an external device, such as a smartphone, is registered/used as FIDO2's cross-device authenticator. Although we could not decode the main data kept in these registry keys for smartphones set as cross-device authenticators, the model name of these smartphones is easily accessible under the string field named `model`. This can allow the uncovering of relevant devices still unknown to the investigation. While the webAuthN EVTX log in Windows also contains information on cross-device authenticators, the `FIDO\\SID\\LinkedDevices\\` registry keys become valuable if the webAuthN EVTX file is unavailable due to deletion or disabled logging.

In future work, we plan to study the digital forensic footprint of FIDO2's passkeys when the roaming authenticator is a security key. We also intend to analyze the digital artifacts left in browsers acting as FIDO2's WebAuthN clients in private and non-private modes, and the footprint of passkey authentication in live memory.

ACKNOWLEDGMENTS

This research was partially supported under the UIDB 04524/2020 project by FCT/MCTES and EU funds under the UIDB/EEA 50008/2020 project and the LA/P/0109/2020 project. The authors thank the anonymous reviewers for their insightful comments and suggestions.

REFERENCES

[1] 2021. Web Authentication: An API for accessing Public Key Credentials - Level 2. https://www.w3.org/TR/webauthn-2 [Online; accessed 26. Dec. 2023].

[2] 2022. CyberChef - The Cyber Swiss Army Knife. https://cyberchef.io [Online; accessed 01. Feb. 2024].

[3] 2022. FIDO2 API for Android. https://developers.google.com/identity/fido/android/native-apps [Online; accessed 8. Jan. 2024].

[4] 2024. AAGUIDs. https://aaguid.nicolasuter.ch [Online; accessed 02. Feb. 2024].

[5] 2024. Clean Your System and Free Disk Space | BleachBit. https://www.bleachbit.org [Online; accessed 22. Feb. 2024].

[6] 2024. FIDO Alliance. https://fidoalliance.org [Online; accessed 8. Jan. 2024].

[7] 1Password. 2024. Passkeys in 1Password: The Future of Passwordless Authentication. https://1password.com/product/passkeys [Online; accessed 27. Feb. 2024].

[8] 1Password. 2024. Passkeys.directory. https://passkeys.directory [Online; accessed 21. Feb. 2024].

[9] Heather Adkins. 2024. Passkeys, Cross-Account Protection and new ways we're protecting your accounts. https://blog.google/technology/safety-security/google-passkeys-update-april-2024

[10] Manuel Barbosa, André Cirne, and Luís Esquível. 2023. Rogue key and impersonation attacks on FIDO2: From theory to practice. In *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES 2023)*. ACM. https://doi.org/10.1145/3600160.3600174

[11] Arnar Birgisson. 2022. Security of Passkeys in the Google Password Manager. https://security.googleblog.com/2022/10/SecurityofPasskeysintheGooglePasswordManager.html [Online; accessed 11. Jan. 2024].

[12] Bitwarden. 2024. Innovating in passwordless. https://bitwarden.com/password-less-passkeys/ [Online; accessed 27. Feb. 2024].

[13] Matt Burgess. 2024. I Stopped Using Passwords. It's Great and a Total Mess. *WIRED* (Feb. 2024). https://www.wired.com/story/stopped-using-passwords-passkeys

[14] Harlan Carvey. 2016. (second edition ed.). Syngress, Boston. 1–35 pages. https://doi.org/10.1016/B978-0-12-803291-6.00001-2

[15] Ali Naddaf Ken Buchanan Diego Zavala, Christiaan Brand. 2022. Bringing passkeys to Android & Chrome. https://android-developers.googleblog.com/2022/10/bringing-passkeys-to-android-and-chrome.html [Online; accessed 12. Jan. 2024].

[16] Glenn Fleishman. 2023. *Take control of your Apple ID, 4th edition.* Alt Concepts.

[17] Conor Gilsenan, Fuzail Shakir, Noura Alomar, and Serge Egelman. 2023. Security and Privacy Failures in Popular 2FA Apps. In *32nd USENIX Security Symposium (USENIX Security 23)*.

[18] Jingjing Guan, Hui Li, Haisong Ye, and Ziming Zhao. 2022. A Formal Analysis of the FIDO2 Protocols. In *Computer Security – ESORICS 2022*, Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng (Eds.). Springer Nature Switzerland, Cham, 3–21.

[19] Sherif Hanna. 2024. Effortlessly upgrade to Passkeys on Pixel phones with Google Password Manager. https://security.googleblog.com/2024/01/upgrade-to-passkeys-on-pixel-with-google-password-manager.html [Online; accessed 21. Feb. 2024].

[20] Olaf Hartong. 2023. sysmon-modular. https://github.com/olafhartong/sysmon-modular/blob/master/sysmonconfig-excludes-only.xml [Online; accessed 01. Feb. 2024].

[21] Nurdeen M. Ibrahim, Ameer Al-Nemrat, Hamid Jahankhani, and Rabih Bashroush. 2012. *Sufficiency of Windows Event Log as Evidence in Digital Forensics.* Springer Berlin Heidelberg, 253–262. https://doi.org/10.1007/978-3-642-33448-1_34

[22] Blake Ives, Kenneth R Walsh, and Helmut Schneider. 2004. The domino effect of password reuse. *Commun. ACM* 47, 4 (2004), 75–78.

[23] Dhruv Kuchhal, Muhammad Saad, Adam Oest, and Frank Li. 2023. Evaluating the Security Posture of Real-World FIDO2 Deployments. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM. https://doi.org/10.1145/3576915.3623063

[24] Leona Lassak, Elleen Pan, Blase Ur, and Maximilian Golla. 2024. Why Aren't We Using Passkeys? Obstacles Companies Face Deploying FIDO2 Passwordless Authentication USENIX. In *33rd USENIX Security Symposium (USENIX Security 24). Philadelphia, PA: USENIX Association.* https://www.usenix.org/conference/usenixsecurity24/presentation/lassak

[25] Karola Marky, Kirill Ragozin, George Chernyshov, Andrii Matviienko, Martin Schmitz, Max Mühlhäuser, Chloe Eghtebas, and Kai Kunze. 2022. "Nah, it's just annoying!" A Deep Dive into User Perceptions of Two-Factor Authentication. *ACM Transactions on Computer-Human Interaction* 29, 5 (2022), 1–32.

[26] Microsoft. 2024. SecureString Class (System.Security). https://learn.microsoft.com/en-us/dotnet/api/system.security.securestring?view=net-8.0 [Online; accessed 13. May 2024].

[27] Mark Russinovitch (Microsoft). 2023. Process Monitor - Sysinternals. https://learn.microsoft.com/en-us/sysinternals/downloads/procmon [Online; accessed 01. Feb. 2024].

[28] Mark Russinovitch (Microsoft). 2024. Sysmon - Sysinternals. https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon [Online; accessed 01. Feb. 2024].

[29] Aditi Srivastava Paolo Matarazzo. 2023. Support for passkeys in Windows - Windows Security. https://learn.microsoft.com/en-us/windows/security/identity-protection/passkeys/ [Online; accessed 8. Jan. 2024].

[30] passkeydeveloper. 2023. passkey-authenticator-aaguids. https://github.com/passkeydeveloper/passkey-authenticator-aaguids/blob/main/aaguid.json [Online; accessed 02. Feb. 2024].

[31] RR Safin, AS Abdiraman, AM Nurusheva, and LS Aldasheva. 2022. Comparison of information security methods of information-communication infrastructure: Multi-Factor Authentication. (2022).

[32] Avinash Singh, Hein S. Venter, and Adeyemi R. Ikuesan. 2018. Windows registry harnesser for incident response and digital forensic analysis. *Australian Journal of Forensic Sciences* 52, 3 (Dec. 2018), 337–353. https://doi.org/10.1080/00450618.2018.1551421

[33] Athanasios Vasileios Grammatopoulos, Ilias Politis, and Christos Xenakis. 2021. A web tool for analyzing FIDO2/WebAuthn Requests and Responses. In *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES 2021)*. ACM. https://doi.org/10.1145/3465481.3469209

[34] Tarun Kumar Yadav and Kent Seamons. 2024. A Security and Usability Analysis of Local Attacks Against FIDO2. In *Proceedings 2024 Network and Distributed System Security Symposium (NDSS 2024)*. Internet Society. https://doi.org/10.14722/ndss.2024.24327