Evaluating the Security Posture of Real-World FIDO2 Deployments

Dhruv Kuchhal
dkuchhal@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Muhammad Saad
muhsaad@paypal.com
PayPal, Inc.
Scottsdale, Arizona, USA

Adam Oest
aoest@paypal.com
PayPal, Inc.
Scottsdale, Arizona, USA

Frank Li
frankli@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

FIDO2 is a suite of protocols that combines the usability of local authentication (e.g., biometrics) with the security of public-key cryptography to deliver passwordless authentication. It eliminates shared authentication secrets (i.e., passwords, which could be leaked or phished) and provides strong security guarantees assuming the benign behavior of the client-side protocol components.

However, when this assumption does not hold true, such as in the presence of malware, client authentications pose a risk that FIDO2 deployments must account for. FIDO2 provides recommendations for deployments to mitigate such situations. Yet, to date, there has been limited empirical investigation into whether deployments adopt these mitigations and what risks compromised clients present to real-world FIDO2 deployments, such as unauthorized account access or registration.

In this work, we aim to fill in the gap by: 1) systematizing the threats to FIDO2 deployments when assumptions about the client-side protocol components do not hold, 2) empirically evaluating the security posture of real-world FIDO2 deployments across the Tranco Top 1K websites, considering both the server-side and client-side perspectives, and 3) synthesizing the mitigations that the ecosystem can adopt to further strengthen the practical security provided by FIDO2. Through our investigation, we identify that compromised clients pose a practical threat to FIDO2 deployments due to weak configurations, and known mitigations exhibit critical shortcomings and/or minimal adoption. Based on our findings, we propose directions for the ecosystem to develop additional defenses into their FIDO2 deployments. Ultimately, our work aims to drive improvements to FIDO2's practical security.

CCS CONCEPTS

• **Security and privacy → Multi-factor authentication**; *Web application security*; **Security protocols**.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3623063

KEYWORDS

FIDO2; WebAuthn; Malware; Security Measurements

ACM Reference Format:
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), November 26–30, 2023, Copenhagen, Denmark*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623063

1 INTRODUCTION

FIDO2 is a second-generation suite of protocols supporting passwordless authentication. It relies upon public-key cryptography, where the cryptographic keys, their access control, and cryptographic operations are ideally secured by a hardware-based Trusted Execution Environment (TEE). For access control to cryptographic keys, FIDO2 relies on the authentication performed locally at the user's device, supporting methods such as biometrics (e.g., fingerprint or facial scanning), and local PINs. FIDO2 has been designed and formally verified [30, 32, 56, 57] as providing strong authentication security guarantees (with prior work also demonstrating promising usability outcomes [49, 62, 65, 69]), all without the use of any shared authentication secrets (i.e., passwords). FIDO2's promise has driven real-world adoption, with support by major browsers (e.g., Chrome, Firefox, Edge, Safari), OSes (e.g., Windows, Linux, macOS, iOS, Android), and online services (e.g., Microsoft, Google, Facebook, PayPal, and eBay).

By avoiding shared secrets, FIDO2 directly eliminates the threat of credential phishing and data breaches: two of the primary credential theft vectors [78]. However, malware remains a notable threat, which has also plagued password authentication for decades (e.g., keyloggers are often used to steal passwords). As FIDO2 becomes more prominent, attackers will naturally be incentivized to prioritize this remaining attack surface. The official documentation of FIDO2 [44] also recognizes the threat of malware and thus recommends configurations that help mitigate the impact of malware. However, to date, there has been limited investigation into whether FIDO2 deployments adopt these mitigations, and what risks compromised clients pose in practice.

FIDO2's strong security guarantees also have the potential to bias the risk assumptions made by practitioners, as these guarantees depend fundamentally on the integrity of the local client operations, and may not completely hold if client-side components are

malicious, compromised (as proven by formal analysis of the protocol [56, 57]), or misconfigured. Moreover, FIDO2 does not inherently eliminate the risk of social engineering: in Section 8.2, we show that a practical social engineering attack could compromise a sensitive action such as an online payment transaction. This motivates the need for additional controls alongside FIDO2 authentication—such as in-browser integrity checks and authenticator attestation—to adequately secure such interactions.

In this work, we investigate the existing configurations of FIDO2 deployments, the degree to which they adopt the recommended mitigations, and the extent to which malicious client software and social engineering can impact them. We consider the case where legitimate clients are compromised, both when the compromise occurs during FIDO2 registration and when it occurs only during authentication after a legitimate user successfully registers.

We cover the background necessary for FIDO2 in Sections 2 and 3, describe our threat model in Section 4, and our measurement method in Sections 5. In Sections 6 - 8, we provide three key contributions across the aforementioned scenarios:

(1) Our team of four members who have prior experience with real-world FIDO2 deployments and authentication research, collaboratively systematize the threats to FIDO2 from malicious client components, expanding beyond prior work (discussed in Section 3) and FIDO2 documentation [13]. As part of our threat characterization, we identify and disclose a unique social engineering attack and browser vulnerability through which user-level malware can trick users into authenticating sensitive actions without their knowledge.
(2) Using the FIDO2 documentation, we synthesize the potential mitigations that FIDO2's stakeholders can implement to reduce the risk of unauthorized account access posed by compromised clients and further strengthen FIDO2's security in practice.
(3) We collect a snapshot of FIDO client authentication telemetry from a large financial service provider, and longitudinal measurements of FIDO Alliance's own database of authenticator metadata, to characterize the available FIDO authenticators. Building upon our characterization, we analyze the configurations of FIDO2 deployments in the Tranco Top 1K websites and evaluate their susceptibility to the threats we discuss.

Through our investigation, we find that real-world FIDO2 deployments are broadly not configured to identify compromised FIDO2 clients, and hence are vulnerable to potential attacks. For some threats, improved configurations and adopting recommended mitigations could help secure FIDO2 deployments; for other issues, we identify the lack of or significant shortcomings with existing mitigations. We compare our findings of the deployment realities with the design intentions, as outlined in FIDO's documentation, to identify long-term directions that various stakeholders of the FIDO2 ecosystem can pursue for broader impact, as well as immediate measures that FIDO2 deployments can implement to reduce their risk exposure. FIDO's documentation consists of several distinct components, each proposed by different working groups, and implemented differently by various platforms. We analyze the specifications and implementations together based on real-world measurements, and show that the assumptions and measures as outlined in the documentation do not reflect reality and require more practical solutions. Ultimately, our study is a first step towards developing defenses for practical attacks on FIDO2 deployments, which is particularly salient at this time as the protocol becomes increasingly adopted.

2 BACKGROUND ON FIDO2

FIDO2 provides public-key cryptographic protocols for authentication, using client-side FIDO2 components that interact with an online service supporting FIDO2, called the Relying Party (RP). At a high level, when a user wishes to authenticate with an RP via FIDO2, they use the client components to execute a registration protocol with the RP that results in the generation of a public/private cryptographic key pair (the FIDO2 credentials) that is stored on the client, where the public key is shared with the RP. This credential is bound to the specific user and RP, and access control to the private key is maintained by the client components, permitting key use only upon successful user authentication on the local device (e.g., via biometrics or a PIN). For subsequent authentication attempts with the RP, the client components perform a challenge-response cryptographic protocol with the RP, after locally authenticating the user. In addition to the traditional challenge-response protocol, FIDO2 also allows for RPs and clients to build custom functionality through FIDO2 extensions [71], where extension data is linked to and stored securely with the FIDO2 credential.

Client-side components of the family of FIDO protocols consist of the Authenticator, the Authenticator Specific Module (ASM), and the FIDO client. The Authenticator is the component that stores the cryptographic key materials and supports cryptographic operations, while the ASM provides a standardized software interface to the Authenticator. FIDO clients are the software clients that interact with the Authenticator and user agents during authentication. There is a wide variety of FIDO authenticator implementations, including software emulations of physical authenticators that are called virtual authenticators. Virtual authenticators by design do not provide any security guarantees, as they are primarily designed for testing and development use-cases.

The FIDO Alliance has published three specifications: the deprecated *Universal Second Factor (U2F)*, which describes the framework for using FIDO authenticators as a second factor [1], *Universal Authentication Framework (UAF)*, which is a framework for passwordless authentication [43], and the *Client to Authenticator Protocols (CTAP)*, which specifies how OSes/browsers communicate with a compliant authenticator [40]. CTAP's latest version, CTAP2, is complementary to the W3C's Web Authentication (WebAuthn) specification, which defines an API for web applications to use FIDO protocols [45]. WebAuthn supports communication by web applications with both RPs and authenticators, whereas CTAP2 provides an interface between applications and authenticators. Together, CTAP2 and WebAuthn are known as FIDO2.

In this study, we focus on the two FIDO2 protocols in the UAF framework, as U2F is deprecated. However, since FIDO2 is backward compatible with U2F, it can still be used for multi-factor authentication (MFA), which we consider in scope for our study.

RP security mechanisms: When an authenticator is registered on an RP, it generates a new key pair (the "FIDO credential") and transmits the public key and the authenticator's unique model ID (AAGUID), among other information, to the RP. The authenticator signs this output with its attestation private key, which is part of

a key pair specific to the device model burned into the device at the time of manufacturing. The attestation certificates associated with the attestation keys are also attached with the output, which chain to a trusted root certificate, conveying the provenance of the authenticator to the RP. This process cryptographically proves that a user has a specific model of authenticator when they register [3].

An RP is responsible for maintaining a FIDO Server that stores the public key credentials of users, and a database of authenticator metadata and trust anchors [44] described as follows:

- Authenticator metadata can be derived from FIDO Alliance's Metadata Service (MDS), which contains standardized information describing the characteristics of an authenticator. RPs can use this information to make interoperability or risk decisions. For example, it includes an authenticator's FIDO certification level, which indicates its attack-resistance [11]. There are five possible certifications, in increasing order of attack-resistance, namely L1 [6], L1+ [7], L2 [8], L3 [9] and L3+ [10]. In brief, the authenticators which are resistant to malware and more sophisticated OS, circuit, or chip-level attacks, owing to the presence of TEEs, qualify for L2 or above certification, whereas authenticators lacking TEEs qualify for an L1/1+ certification. The certification is awarded by the FIDO Alliance based on their proprietary security evaluation tests. We note here that all authenticators of L2 or higher consist of hardware-based TEEs, but not all authenticators consisting of hardware-based TEE are awarded L2 or above.
- Trust anchors can be derived from two primary sources, the MDS as well as known root CAs. For example, trusted certificate chains for Android's SafetyNet attestation can be found in MDS, whereas Apple publishes its own root CA for its devices [4].

After establishing the legitimacy of the authenticator based on trust anchors, the RPs can calculate the risk associated with an authenticator based on its security characteristics and decide whether to allow its registration [44].

At registration, **RPs can verifiably ascertain an authenticator's legitimacy** and the risk associated with it.

Client security mechanisms: FIDO UAF's design ensures that the FIDO credentials (i.e., cryptographic keys) can only be accessed by the user associated with the credential (through local user authentication on the user device), and by the same client application and ASM that performed the initial registration operation [46]. Conceptually, UAF specifies that when a new credential is generated, authenticators bind the private authentication key with a key provided by the ASM called the KHAccessToken, which is generated as a function of the AppID, PersonaID, ASMToken, and CallerID. AppID is an identifier for a particular user application (e.g., Android application); PersonaID is an identifier generated by the ASM to differentiate credentials registered on the same RP; ASMToken is a random ID unique to the ASM; CallerID is an identifier unique to the FIDO Client. KHAccessToken ensures that malware on the client device is unable to access keys previously registered by a legitimate application [42] (so long as the attacker lacks root OS access, in which case they can spoof these ID values). FIDO2's CTAP2 implements the equivalent of KHAcessToken via an authenticator activation PIN (authenticatorClientPIN) [40, 44].

Once a legitimate FIDO2 authenticator is registered, **non-root malware on the client can not abuse FIDO2 credentials** previously registered by a legitimate application.

3 RELATED WORK

Since FIDO2's official launch in 2018, a number of studies have investigated the usability and security properties of its protocols as a secure replacement for password-based authentication. On the usability side, several works [49, 62, 65, 69] have conducted user studies to understand how users engage with FIDO2 authentication workflows. Overall, while these works have identified valuable usability properties complemented by a general willingness among users to adopt FIDO2, a few works have also identified user concerns and misconceptions. A common user concern was identified as complexity in account recovery under FIDO2, which can drive users back to using passwords [49, 69].

On the security side, Barbosa et al. [30] provide a cryptographic security analysis of both FIDO2's WebAuthn 1 and CTAP2 protocols, confirming the authentication security of WebAuthn. Among the four security notions proposed in [30], CTAP2 was observed to have the weakest security model. Bindel et al. [32] expanded on this initial analysis to consider WebAuthn2 and CTAP 2.1, demonstrating provable security for both protocols. Meanwhile, Guan et al. [56] and Jacomme and Kremer [57] both applied ProVerif to formally analyze the assumptions that FIDO2 require for its security properties to hold. Similar to prior efforts, they confirmed the security properties of FIDO2 under explicit assumptions but identified that these properties do not necessarily hold with malicious client components. These works demonstrate FIDO2's security, but only under the assumptions of FIDO2's threat model, which does not consider malicious client components. Indeed, several works [58, 79] have identified social engineering-based attacks on FIDO that exploit weaknesses not accounted for by FIDO assumptions.

In our work, we empirically evaluate the consequences of malicious client components. While prior work demonstrated such a threat to FIDO2 theoretically exists, in our work, we empirically evaluate how this threat manifests in practice.

4 THREAT MODEL

Given our focus on understanding the impact of compromised FIDO2 client components, our threat model is centered on an attacker who is able to infect the client with malware. Such malware can be delivered remotely and at scale, permitting widespread attacks. This threat model is realistic as in practice, keyloggers have been widely used to steal user passwords [78]. We consider both a weaker threat model of malware with only user-level access (e.g., a malicious user application or browser extension) and a stronger threat model of malware that achieves elevated root access (such as through exploiting kernel vulnerabilities [86]). For user-level malware, we assume the integrity of OS-level security protections.

We note that a malware-infected client is already in a grave situation, as the attacker may have remote access to their victim's active sessions. However, if the attacker is able to leverage malware to bypass a FIDO2 authentication attempt, this affords significantly expanded capabilities through circumventing additional defenses

(e.g., step-up authentication). Such capabilities can allow attackers to execute sensitive authentication-protected actions, such as changing account recovery settings to permanently take over an account or initiate arbitrary financial transactions (rather than hijacking only ones that a user legitimately initiates). As FIDO2 becomes more popular, attackers will naturally have even more incentive to exploit it with malware, especially without the ability to rely on phishing and credential leaks.

> Malware on the client is a strong threat model which can already result in significant harm, including remote access to the user's active sessions online. **However, we focus on malware that enables an attacker to bypass a FIDO2 authentication attempt**, allowing an attacker to execute any sensitive authentication-protected action.

Our threat model does not include attacks on the FIDO2 protocol itself, which we assume is secure (aligning with prior work [30, 32, 56, 57]). We also assume that authenticators using TEEs that are certified by the FIDO Alliance at L2 or above provide a secure execution environment (per the certification criteria), such that cryptographic keys and operations cannot be leaked or tampered with. However, for authenticators of lower or no security certifications, we do not make such assumptions (as they have not been certified as providing such security properties). In addition, we do not investigate attack vectors beyond those of malicious software on the user device, including network-based attacks (FIDO2 and existing protocols such as TLS protect against such threats) and attacks directly against the RP (e.g., DoS attacks). Finally, we do not consider FIDO2 privacy concerns.

Prior works have considered similar threat models. For example, Eskandarian et al. proposed an architecture to run web applications securely within a compromised browser and compromised OS [39]. However, the design changes proposed are not practical enough to implement at scale. On the contrary, FIDO2 is fast gaining adoption as it offers a wide range of authenticators at different levels of security and interoperability.

5 MEASUREMENT METHOD

We begin by discussing how we empirically evaluate real-world FIDO2 deployments, studying the ecosystem through the lens of each of its participants: (i) online services (RPs) in the Tranco Top 1K which deploy FIDO2, (ii) FIDO Alliance's MDS which plays a central role in providing metadata and trust anchors for authenticators, and (iii) authenticators observed on mobile devices in the wild. We describe the various datasets we collect, which we use to assess the vulnerability of RPs to malware-based threats.

5.1 Relying Parties (RPs)

5.1.1 Identifying RPs deploying FIDO2. Evaluating the behavior of RPs requires identifying the websites supporting FIDO2 logins, creating user accounts on those sites, and registering authenticators (typically in the account settings) to observe the RP's FIDO2 configurations. This process is challenging to fully automate, given the broad diversity in website designs and authentication workflows. Our initial exploration involved crawling the landing sites of the Tranco Top 100K domains (using the snapshot of February 26, 2022). We searched the JS resources loaded by each site for the presence of the call made to the Credential Management API (`navigator.credentials.create`) to create a WebAuthn credential (`publicKey`) [23]. We found the pair of strings present in JS resources loaded by 135 sites, out of which for 82 sites, it was found in enterprise Identity and Access Management SDKs that the sites had imported [14, 25]. Manual analysis of the remaining revealed that barring a few sites (e.g., GitHub), the strings were found in imported JS SDKs without the WebAuthn functionality being utilized by the site. Moreover, popular sites such as Google and PayPal, which we knew supported FIDO2 [22, 24], were not detected by this method. We also attempted automated search engine queries for FIDO2 support on a site, but our manual analysis often led to false positives, such as documents or posts on a site discussing FIDO2, despite the site itself not deploying it (e.g., a Q&A page discussing FIDO2 on a commercial site). Prior work also observed similar issues when investigating MFA based on security keys [52, 53].

Measurement Stage	#
1. Domains in Tranco Top 1K	1000
2. Domains/sites that load in a browser	841
3. Sites with account login/signup pages	585
4. Sites that support FIDO2 WebAuthn	85
5. Distinct RPs for FIDO2 sites	40
6. *Distinct RPs we can evaluate*	*29*

Table 1: The number of entities found at each stage of our measurement method for identifying FIDO2 RPs to evaluate.

Therefore, we use a manual approach for identifying RPs and creating accounts and apply a semi-automated approach for assessing RP FIDO2 configurations. Given the significant manual effort required, we choose to search for FIDO2 support among the domains listed in the Tranco Top 1K domains. First, we visit each domain in a browser and manually inspect the landing page for account creation/login pages. If a site supports public account creation, we then manually look for evidence of FIDO2 support on either the account creation pages or through the links on the first page returned by a Google search (using the query *[domain] "webauthn" OR "passwordless" OR "u2f" OR "uaf" OR "fido2" OR "security key"*). We note that for all sites we found supporting FIDO2, the top search result provided the relevant information on FIDO2 support.

Table 1 provides statistics on each stage of this search process. From the initial 1K domains, we found 585 domains with account login or signup pages (note, some domains did not host a website and thus did not load). Of these domains, we identified 85 supporting FIDO2's WebAuthn, many of which mapped to the same RP (e.g., multiple Microsoft domains all use the same RP). In total, we aggregated the FIDO2-supporting domains into 40 distinct RPs. However, not all of these RPs support consumer-facing account creation (e.g., enterprise or financial services), inhibiting our investigation of their configurations. Ultimately, we could create accounts on and evaluate 29 RPs across the Tranco Top 1K. Given that only 6 of the 29 RPs are ranked beyond 500, we believe that the marginal benefit of expanding beyond the Top 1K sites is low. According to Symantec

Sitereview's domain categorization [26], these RPs span 12 diverse types of websites. 11 RPs are "Technology", 4 RPs are "Search Engines/Portals", and 3 RPs are "Finance". Other categories include "Shopping" and "Social Networking". We note that several of the RPs we study, such as Google and Facebook, are widely used as Identity Providers on other online services, indicating that their authentication practices implicitly impact many more online services than the ones we directly analyze. Interestingly, several RPs we study, such as PayPal and Google, have been influential members of the FIDO Alliance since its founding, and as such, studying their implementations provides us a representative view of real-world FIDO, along with cues for ongoing and future policy decisions.

5.1.2 RP-Requested Authenticator Properties. To evaluate the requirements each RP imposes on authenticators registering on their platform, we manually create an account on each RP's website (arbitrarily choosing one site if an RP is associated with multiple), initiate authenticator registration, and parse the requirements indicated by the RP as part of the authentication challenge request (note here that we do not need to complete the registration). Specifically, the RP's client-side script calls `navigator.credentials.create()`, passing along data fields indicating the RP's criteria for allowed authenticators, the attestation type required, and the extensions supported [23]. In our manual analysis, we found that the client-side code calling the API was often obfuscated, inhibiting static analysis. Instead, we used Chrome's in-built JS debugger [21] to set a breakpoint at the function call, so we could manually inspect the request's parameters when we trigger a registration.

Table 2 lists the 29 RPs evaluated, the authenticators allowed, and the attestations required by each RP.

5.1.3 RP Allowlisting of Authenticators. Next, we investigate whether RPs place further restrictions on the authenticators they allow on their platforms by allowlisting certain authenticator/AAGUIDs based on metadata from MDS, as recommended by FIDO to maintain an acceptable level of assurance [44]. To that end, we attempt to manually register a credential on each of the 29 RPs found in Section 5.1.2, using Chrome's built-in virtual authenticator [16] under different configurations. While there are other ways to instrument custom authenticators, such as Google's OpenSK [55], our goal is to choose an authenticator with the weakest security guarantees. Since Chrome's virtual authenticator is designed for testing WebAuthn applications, it stores the credentials (including private keys) in plaintext and allows one-click export of sensitive credentials. At registration, it provides an RP with a self-signed certificate issued by 'Chromium Authenticator Attestation'. Since it is not registered with the MDS, its attestation does not chain to any trust anchor. Thus, RPs should not allow such an authenticator to register in practice, as an attacker could easily implement a malicious virtual authenticator, or attack credentials registered via a virtual authenticator if used in real-world settings.

In our experiment, we attempt to register the virtual authenticator under different configurations, including different protocols (CTAP2 or U2F) and transports (USB, BLE, NFC, or Internal). Whenever the RP required user verification, we enabled user verification in the virtual authenticator (which does not involve real user verification). In Table 2, we list whether we successfully registered a virtual authenticator under each configuration, for all RPs.

5.2 FIDO Metadata Service (MDS)

Given the central role that the MDS plays in maintaining trust and safety in the FIDO ecosystem, we conducted over a year long longitudinal study into the operation of the MDS, from September 16, 2021, to April 13, 2023. We recorded daily snapshots of the MDS, via their APIv3 [20], as they regularly update it with new entries. Our last snapshot listed 160 authenticators, a 90% increase from the first snapshot. Out of the 160 authenticators, 120 authenticators were certified at Level 1, and only 7 were certified as Level 2 (i.e., malware resistant); the remaining are not yet certified. A list of the 7 L2 authenticators is provided in Appendix C. We identified that 117 authenticators support mere user presence (e.g., pressing a button) as a user verification mode, 64 authenticators support a local passcode that is collected outside the authenticator boundary, and 58 authenticators support no user verification at all. Note that an authenticator can support multiple user verification modes.

5.3 Real-World User Authenticators

While MDS lists available authenticators and their properties, it does not provide any information about the distribution of authenticators used in the wild. To gain visibility into real-world authenticators, we partner with a large financial services RP to collect authenticator registration telemetry recorded during a FIDO2 passwordless authentication pilot involving live user traffic. The RP chose to run the pilot only for sessions from mobile browsers, and therefore the authenticator data we collect is limited to mobile devices. To the best of our knowledge, our study is the first to study a significant sample of real-world authenticators found on users' mobile devices from the lens of a large RP. Prior work conducted a user study on FIDO2 with 29 participants (with 18 iPhones and 11 Android devices) [62].

Using the RP's authentication server logs, for all users who were enrolled into passwordless authentication, we extracted the WebAuthn response payloads received from their authenticator, on two days chosen at random during the pilot. We collected a total of 126,608 payloads on July 20, 2022, and 74,270 payloads on August 23, 2022. We then parsed the payloads to extract the corresponding attestation data, which provides information on the authenticator itself. We call this dataset the *bulk WebAuthn data*. Due to technical limitations of the server's logging mechanism, we could only retrieve the first 4KB of the payload, and some response payloads were truncated and could not be parsed. We identified that authenticators attested by Android SafetyNet issue WebAuthn responses with payload exceeding 4KB, so we attribute the responses with truncated payloads as being SafetyNet-attested. To validate these assumptions about payload truncation, we collected a separate set of WebAuthn response data containing full payloads for 14,616 sessions randomly chosen in the last week of July 2022. We call this dataset the *sampled validation WebAuthn data*.

From the 200,878 WebAuthn responses in the entire bulk WebAuthn data, we were able to parse ~ 72% of responses as coming from Apple-attested authenticators, ~ 1% without any attestation, and 1 self-attested response. About 0.5% of registrations had Apple's attestation but also an empty AAGUID – we suspect that these are Apple Passkeys [15], based on the session HTTP user agents indicating iOS 16 devices and information from Apple developer forums [50].

WebAuthn Usage	Domain Rank (↓)	RP/Service	RP Authenticator Preferences			RP Acceptance of Virtual Authenticator Configurations			
			Authenticator Attachment	Attestation Requirement	User Verification	USB	BLE	NFC	Internal
Password-less	26	Microsoft	platform	direct	✗	✗			
	74	PayPal			discouraged	✗			✓CTAP2 ✗U2F
	85	eBay	cross-platform		required	✓CTAP2 , ✗U2F			
	128	Mail.ru	platform	none	✗	✓CTAP2, ✓U2F			
	517	BestBuy			required	✗			✓CTAP2 ✗U2F
MFA	1	Google	cross-platform	direct	✗	✓U2F , ✗CTAP2			
	3	Facebook			✗	✓U2F ✓CTAP2			
	6	Twitter		none	discouraged				
	14	Yahoo		direct	preferred				
	28	GitHub	✗	none	discouraged				
	32	AWS	cross-platform	direct	preferred	✓U2F, ✓CTAP2			✗
	35	WordPress	✗			✓U2F ✓CTAP2			
	56	Chaturbate	cross-platform	✗	discouraged				
	67	Dropbox	✗	none	preferred				
	80	Cloudflare							
	151	GoDaddy		✗	discouraged				
	162	Aliyun	✗	direct					
	192	Shopify	platform	✗					
	202	AoL	cross-platform	direct	preferred				
	277	Zoho							
	296	Binance	✗						
	321	Roblox		✗	discouraged				
	419	Stripe	cross-platform			✓U2F, ✓CTAP2			✗
	490	BofA		direct		✓U2F, ✓CTAP2			
	553	NVIDIA	✗	none	required	✗			
	607	GitLab	✗		discouraged	✓U2F ✓CTAP2			
	656	Namecheap	✗	direct					
	938	BitBucket	cross-platform						
	984	Norton	✗	none	✗				

Table 2: We study the FIDO2 WebAuthn configurations of 29 real-world RPs identified from the Tranco Top 1K (see Table 1). RPs can choose to implement WebAuthn for **passwordless authentication or multi-factor authentication (MFA)**, and in our dataset, we identify 5 RPs deploying WebAuthn for passwordless, while the other 24 utilize it for MFA. When enabling FIDO2 WebAuthn for a user, RPs can declare preferences or requirements about the authenticators they allow (listed under **RP Authenticator Preferences**). First, they can specify the allowed modes of **authenticator attachment**, as an authenticator could be built-in directly into the client devices (i.e., platform) or could be externally attached/roaming (i.e., cross-platform), such as USB security keys. Note that allowing cross-platform authenticators also inherently allows platform ones as well. Second, they can indicate the degree of **attestation required**, ranging from none (i.e., no attestation requirement), indirect (e.g., attestation certificate suffices without authenticator metadata statement), and direct (i.e., full attestation of unaltered metadata statement required). Some RPs may opt for reduced attestation information to preserve user privacy. Finally, the RP can signal whether they require, prefer, or discourage **user verification**. Note that an ✗ mark denotes that a preference/requirement was not specified by an RP. For the same set of RPs, we also test whether an RP allows Google Chrome's virtual authenticator [16] to register as a legitimate authenticator, using different transports (**CTAP** and **U2F**) and protocols (**USB**, **BLE**, **NFC**, and **Internal**). Here, ✗ indicates unsuccessful registration with the virtual authenticator under a specific configuration, whereas a ✓ indicates a successful registration.

Attestation Type	Apple Anonymous	None			Android SafetyNet	Self
AAGUID (in MDS ✓/✗)	f24a8... (✗)	No AAGUID Provided (✗)		b93fd... (✓)		adce0... (✗)
No. of registrations (07/20/22 | 08/23/22)	71% | 71%	0.5% | 0.4%	0.6% | 0.9%	0.5% | 0.4%	27% | 27%	1 | 0

Table 3: Distribution of WebAuthn attestations observed at registration, during a WebAuthn pilot for mobile browser sessions at a large financial service RP, measured on two days. (Full AAGUIDs are listed in Appendix B.)

For the other responses with Apple's attestation, the AAGUID was set to `f24a8...`, but we note that this AAGUID was not listed in the MDS. The anomalous self-attested authenticator was associated with sessions using an HTTP User Agent (potentially spoofed) indicating a Chrome browser on a Mac OS X device. As a sanity check, we verified that the HTTP User Agent for the remaining ∼ 27% truncated responses did indicate Android devices. We did not observe any other attestations. Table 3 presents a summary of this dataset.

Using the sampled validation WebAuthn data, we observed that the responses consisted of ∼ 72% attested by Apple, ∼ 27% attested by Android SafetyNet, and ∼ 1% without attestation. The stark similarity in the distributions of Apple-attested and unattested responses between the sampled and the bulk dataset gives us further confidence that the ∼ 27% truncated responses in the bulk dataset should be correctly attributed to Android SafetyNet attestation.

5.4 Ethical Considerations

Our measurements evaluate FIDO2 deployments and their susceptibility to attacks. In conducting our measurements, we only test benign FIDO2 authentication attempts using different configurations on a test account, without inducing a high load on the online services and without causing any harm to the service or any real users. For identified issues, we have or are in the process of vulnerability disclosure to the relevant stakeholders in a position to employ remediations. The partner-RP telemetry we analyzed did not contain any personally-identifying information.

6 PRACTICALITY OF MALWARE THREATS

FIDO2 assumes a TEE environment for providing its security guarantees and defending against malware attacks. Here, we analyze our data on authenticator characteristics to identify whether this assumption holds true in practice for available authenticators and to determine whether malicious client components pose a realistic threat to FIDO2.

Using the latest snapshot of MDS (from Section 5.2), we find that only 7 out of 160 (4%) authenticators have L2 certification, which the FIDO Alliance has defined as offering malware resistance (with no authenticators certified at a higher level). As noted in Section 2, it is possible though for an authenticator to have secure hardware but still not be certified as malware-resistant (L2). To determine an upper bound on the population of such authenticators, we analyzed the distribution of authenticators by the security properties of their keystores. We find that ∼ 93% of authenticators are listed as backed by either TEE, hardware, or secure element keystores, while the remaining ∼ 7% authenticators are backed by software-only keystores. Therefore, at least the ∼ 7% authenticators are likely vulnerable to malware, while the others may provide some malware resistance, based on other properties such as mode of attachment.

> **96% of authenticators available today did not receive FIDO Alliance's malware-resistance certification**, and thus are potentially vulnerable to malware-based attacks.

Beyond considering the distinct authenticators available, we also evaluate hardware-backed authenticators in our real-world user authenticator data from Section 5.3 (recall though that our client authenticator data is strictly from mobile clients of our partner RP). Specifically, for authenticators attested by Android's SafetyNet, the attestation information contains a field labeled `evaluationType` which indicates whether SafetyNet's device integrity evaluation is based on hardware-backed security features (`evaluationType = HARDWARE_BACKED`). We found that ∼94% of attestations in the SafetyNet WebAuthn sample were hardware-backed, indicating that a non-trivial minority of authenticators lack hardware support.

Furthermore, for hardware-backed SafetyNet attestations, ∼0.25% indicated compromise (SafetyNet's device integrity check failed) and ∼13% of the non-hardware-backed SafetyNet attestation indicated compromise. For the devices that were detected as being compromised, SafetyNet further specified that ∼49% were detected having an unlocked bootloader, ∼10% a custom ROM, and ∼5% having both. Due to the sensitivity of customer data, our insights from this data are limited to aggregate statistics, however, our partner RP did indicate that they flagged evidence of suspicious scripted activity for the sessions from these compromised devices. They noted a distinctly high volume of logins on multiple accounts from these devices, many using proxy IP addresses. **Thus, we observe that FIDO2's assumption of hardware-backed security does not hold for many authenticators, and we already find evidence of malicious activity on compromised devices.**

7 REGISTRATION-PHASE ATTACKS

As FIDO2 gains adoption, it is important to consider the threats when an authenticator is first registered, as this authenticator would be trusted for subsequent login attempts to the account. In this section, we expand upon existing FIDO2 documentation [2, 44, 45] to systematize realistic threats to FIDO2 deployments involving malicious authenticator registration. We consider two attack scenarios, malicious authenticators that are registered to a legitimate user's account, and a legitimate but vulnerable authenticator registered to the user's account which may be subsequently compromised. For each threat, we discuss what mitigation actions could be taken, and then empirically evaluate the extent to which such mitigations can or have been adopted by real-world FIDO2 deployments.

7.1 Malicious Authenticator

Here we consider the threats where the attacker aims to register their malicious (i.e., attacker-controlled) authenticator to a user's account. This allows the attacker to authenticate into the account at will, effectively taking over the account. To gain access to a user's account, the attacker could either utilize existing account takeover techniques to gain access themselves, or they could deploy malware to the user's device which already has the access required.

7.1.1 Traditional Account Takeover.

Attack Description: An attacker can exploit non-FIDO2 credentials, such as passwords (e.g., via phishing), to take over an account and register a malicious authenticator. Pre-hijacking attacks (e.g., Sudhodanan et al. [77]) could also be leveraged to register a malicious authenticator and gain persistent access to a user's account.

Measurement of Mitigation Adoption: Given the importance of initial authenticator registration, FIDO2 best practices recommend that RPs employ user verification methods, such as challenge-response verification to a user's phone or email, to ensure that the actual user is registering an authenticator. Prior work has shown that similar login challenges are effective in limiting account takeover itself [2, 38]. However, while registering our authenticators at our evaluated RPs (Sections 5.1.2 and 5.1.3), we found that out of the 29 RPs analyzed, only Aliyun, Binance, Stripe, and Bank of America (BofA) challenged our attempt to register an authenticator. They required us to enter a One-Time Password (OTP) sent over either SMS or email, and BofA additionally required the PIN of the debit card registered on the account. We note that while other services may have made risk-aware decisions not to verify our identity, prior work has demonstrated that such user verification challenges should augment risk-aware authentication to minimize the risk of account takeover [38]. Thus, RPs currently do not widely deploy such measures to limit malicious authenticator registration through traditional account takeover.

7.1.2 Phishing FIDO2 via User-Level Malware.

Attack Description: A user could inadvertently install a malicious user-level (i.e., non-root) application that misrepresents itself as the target RP's official application. Similar to a phishing site, the application could look and behave the same way as its legitimate counterpart. Once the user attempts to log into the RP through the malicious app, the attacker has access to the user account. If the attacker can directly register their own malicious authenticator for the user, then this case falls back to that of Section 7.1.1.

However, some services may require user verification when registering a new authenticator. In that case, the malicious application must trick the user into registering the malicious authenticator while thinking they are registering their real authenticator (or wait for the user to conduct an action that would require the same form of user verification). When this occurs, the application could intercept the legitimate FIDO2 registration request and instead register a malicious virtual authenticator embedded in the application itself (with the user verifying the malicious authenticator registration, thinking they are registering their actual authenticator or doing the user verification for a different action). Once registered, the malware could report the FIDO2 credentials (visible to the malware in plaintext) back to the attacker, allowing the attacker to clone a similar virtual authenticator with the same stolen credentials and remotely compromise the user's account. Since FIDO2's threat analysis does not account for virtual authenticators, this attack is not discussed in existing documentation.

Measurement of Mitigation Adoption: To mitigate this attack, FIDO2 recommends that RPs identify client authenticators via attestation (based on attested AAGUIDs), and allow only trusted authenticators to register [3]. Permitted authenticators should exclude virtual ones, which provide no meaningful security.

From our characterization of RP FIDO2 configurations (in both Sections 5.1.2 and 5.1.3), we observe that only 14/29 RPs request attestation of any kind, and the remaining RPs cannot ensure that they have trustworthy information on the authenticators being used by clients. Furthermore, we find that 27/29 RPs – including financially sensitive RPs such as BofA, PayPal, Binance, Stripe, and eBay, allow even a virtual authenticator to be registered. Thus the vast majority of evaluated RPs are vulnerable to such an attack.

While RPs are broadly not allowlisting trusted authenticators, we explore the extent to which they could, using our real-world user authenticator data (Section 5.3). From the distribution of attestation types that we observed from client authenticators (all on mobile clients of our partner RP), as listed in Table 3, we find that the vast majority provide attestation linked to either an Anonymization CA (Apple Anonymous; $\sim$ 71%) or attestation root certificates found in MDS (Android SafetyNet; $\sim$ 27%). Thus, for at least mobile devices, attestation information from most authenticators can be verified and used for allowlisting.

However, there are $\sim$ 1% of client authenticators that do not provide any attestation (and 1 self-attested authenticator), for which the RP has no trusted information. In such cases, RPs could disallow these unattested authenticators at the risk of impacting legitimate users or permitting such devices but remain exposed to this attack. We briefly note that for accounts linked to these non-attested authenticators, our partner RP indicated that their risk systems identified significantly higher volumes of risky transactions (attempts to steal funds) than typical for the average account, as well as evidence of attempted money laundering through newly created accounts. We can also explore suspicious activity on authenticators attested by Android SafetyNet, as SafetyNet provides information on the calling/authenticating application in its attestation response [17]. From our sample of WebAuthn responses attested by Android SafetyNet, we found that $\sim$ 0.1% of attestations did not include the calling application's certificate, and another $\sim$ 0.1% had a certificate different from that of Chrome on Android, the calling application it claimed. SafetyNet also indicated that these sessions came from rooted devices. Thus, we have preliminary evidence that suspicious activity from authenticators lacking proper attestation may already be occurring in practice, exposing RPs that permit such behavior.

7.1.3 FIDO2 Hijack by Root Malware.

Attack Description: Overprivileged devices, such as rooted Android devices or jailbroken iOS devices, allow malware to circumvent built-in security mechanisms enforced by the OS [59, 87]. A root-level malware can intercept and respond to a FIDO2 registration request from a malicious virtual authenticator. Unlike the malware attack described earlier, here the user can still interact

with the legitimate RP application, but their FIDO2 operations are hijacked at the root level. While a strong attack, this threat is still realistic as a non-trivial population of mobile devices is rooted [80] (as we also confirm in our measurements), and rooted devices are frequent compromise victims [74].

Measurement of Mitigation Adoption: This attack can also be mitigated by RPs verifying attestation and allowlisting trusted AAGUIDs [3], which prevents the attacker from registering a malicious authenticator. As uncovered above (in Section 7.1.2), we found that such mitigations are limited in practice. The 27/29 sites that lack trusted authenticator allowlisting are similarly vulnerable to this attack.

27/29 RPs can increase their risk awareness by formulating policies to allowlist secure and trusted authenticators.

7.2 Vulnerable Authenticator

FIDO2 authentication is available on a wide variety of devices, and the authentication security depends on characteristics of the device security, such as the availability, modality, and accuracy of user verification, matcher protection, and private key management protection. Some authenticators may be weak/vulnerable and later compromised by an attacker after FIDO2 registration. Here we consider threats where the attacker targets a vulnerable authenticator.

7.2.1 *Local Authentication Downgrade.*

Attack Description: The level of assurance an authenticator provides for a user's identity varies with the modality of local verification used. It can range from biometrics (e.g., fingerprint scan) to knowledge-based factors (e.g., PIN) to mere user presence, or even no verification at all. FIDO defines stringent biometric performance requirements that a FIDO-certified authenticator needs to adhere to [5], while prior work has shown that knowledge-based factors such as PINs can often be observed or easily guessed [66], such as by malware on the device. Therefore, the security posture of a user's account could significantly weaken if their authenticator allows for weak user verification methods (i.e., PIN), which malware could successfully bypass. Similar social engineering downgrade attacks have been demonstrated previously at the RP level [79].

Measurement of Mitigation Adoption: To mitigate this attack, FIDO2 provides the User Verification Method Extension (`uvm`) which enables the RP to know which verification methods (factors) were used for an operation [45, 48]. In case a PIN needs to be used, such as when other modalities fail (e.g., physical obstruction for biometrics), WebAuthn is expected to soon implement an extension currently supported by CTAP called `minPinLength` which conveys the minimum PIN length value of the authenticator to the RP [37, 40]. Together, this information allows an RP to implement policies/requirements on user verification methods, and/or adjudicate risk appropriately during an authentication attempt. However, from analyzing the MDS data, we see that only 1 and 8 (out of a total 160) authenticators in the MDS currently support `uvm` and `minPinLength`, respectively. Our RP partner also did not support any extensions during the pilot, so we could not measure whether clients support these extensions. Overall, we conclude that the authenticator ecosystem does not yet widely support this mitigation.

In our RP characterization (Sections 5.1.2 and 5.1.3), only 3/29 RPs required user verification, allowing authenticators on other RPs to return a success without any local verification of the user's identity. No sites we studied declared support for the aforementioned extensions, with only 9/29 RPs supporting any FIDO2 extension. Thus, RPs are not checking for robust user verification methods, and are vulnerable to local authentication downgrade attacks.

7.2.2 *Compromise of Vulnerable Authenticators.*

Attack Description: For a legitimate authenticator, if user verification can be circumvented by malware, or its attestation or private keys can be compromised, either remotely or with physical access to the authenticator, the authenticator is known to be vulnerable [47]. An attacker can exploit such a vulnerability to compromise the authenticator's security and potentially take over an account.

Measurement of Mitigation Adoption: FIDO2 recommends that RPs monitor the MDS for security notifications declaring such vulnerabilities in authenticators registered with FIDO [44]. Ideally, an RP should also periodically keep monitoring if vulnerabilities are reported for previously registered authenticators, and in the case a vulnerability is discovered, treat the authentication attempts from those authenticators as risky, until patched.

In our longitudinal measurement of the MDS starting September 2021 (in Section 5.2), spanning over a year, we did not record any security notifications for any authenticator. Yet, we are aware of existing authenticator vulnerability. For example, Shakevsky et al. demonstrated an IV reuse attack on Android's hardware-backed Keystore in Samsung's flagship smartphones, leading to the compromise of private FIDO2 credentials, and a bypass of FIDO2-based authentication [73]. Following their disclosure, Samsung published CVE-2021-25490 with High severity and issued a patch in October 2021. However, neither *Samsung Pass*, Samsung's identity management service, nor *Android with SafetyNet* – both authenticators registered with FIDO – reflected a vulnerability in the MDS. In fact, as of May 2023, *Samsung Pass*'s MDS entry has not been updated since 2018, and *Android with SafetNet*'s entry's last update was in 2020 – both when they were first registered.

As an increasing number of devices and services bank upon the security guarantees provided by TEEs, other TEE implementations have also been similarly targeted [33, 34]. The lack of updates signals that the MDS may not be a trustworthy source of information to make accurate decisions about vulnerable authenticators.

7.2.3 *Credentials stolen from Virtual Authenticator.*

Attack Description: Users could install an application (or a Chrome extension [76]) that includes an authenticator and allows the user to manage (export and sync) their FIDO2 credentials. While some users might choose this for its usability, it provides little security guarantees, as the FIDO credentials reside in user space and can be stolen by malware. Stolen credentials can be easily seeded in a cloned virtual authenticator for the attacker to gain access.

Measurement of Mitigation Adoption: RPs should disallow virtual authenticators. As seen in Section 7.1.2, 27/29 RPs permitted virtual authenticator, and are exposed to this threat.

Authenticators need to implement FIDO2 extensions (e.g., `uvm`) which enable RPs to **ensure trusted user verification**. RPs also need to **actively track vulnerabilities in authenticators**, such as by monitoring public CVEs.

8 AUTHENTICATION-PHASE ATTACKS

In this section, we investigate how malicious software on the client can affect FIDO2 authentication assuming that a malicious or vulnerable authenticator is not registered to the user's account. In this setting, the attacker is not able to compromise the authenticator (unlike in Section 7), and thus can only target the FIDO2 authentication phase. We consider first the case where malware attempts to leverage existing legitimately-registered credentials to authenticate, which is only possible with a limited set of authenticator properties. Outside of those conditions, malware cannot directly utilize existing credentials for authentication. Instead, malware must involve the user through a social engineering attack that results in the user authenticating during insecure situations. Towards that, we identify a unique social engineering attack where malware can trick users into authenticating sensitive actions without them realizing. For both categories of authentication-phase attacks, we discuss what mitigations exist to address them and the extent to which real-world FIDO2 deployments are configured to do so.

8.1 Targeting Existing Legitimate Credentials

Attack Description: As discussed in Section 2, FIDO2 has built-in protection (KHAccessToken/authenticatorClientPIN) to prevent a user-level malware from accessing keys previously registered by a legitimate application, and thus user-level malware is not able to interfere with the FIDO2 authentication phase.

For root-level malware, as long as the FIDO2 Client resides as a Trusted Application in the TEE, OS integrity checks will fail when the FIDO2 client attempts to access the keystore, denying the malware access to the credentials. However, in the case of roaming authenticators (e.g., a USB security key) where the FIDO2 Client is a root application on the OS, root malware could bypass local user verification (by spoofing a successful verification) to the authenticator and raise the request to the authenticator when it is plugged in. If there is no user verification at the roaming authenticator itself (e.g., button push, biometric validation), the malware could successfully trigger an authentication using the user's FIDO2 credentials, without them realizing it.

Measurement of Mitigation Adoption. RPs can declare preference for the authenticator attachment mode during the authenticator registration phase (through one of the parameters passed during credential creation). Their preference can be based on various factors including their intention to support cross-platform authentication, and the impact of their choice on user experience [85]. To address this threat, RPs could declare that they require `platform` authenticators, disallowing roaming authenticators. However, roaming authenticators provide key usability benefits such as enabling users to carry their credentials and use them to authenticate on multiple devices. Therefore, it is reasonable for RPs to allow roaming authenticators in order to reduce friction for users and encourage broader adoption. When allowing roaming authenticators, RPs can use MDS metadata, which lists the user verification methods supported by authenticators, to make a risk-aware policy/decision based on the strength of the user verification method (or lack thereof). For example, the RP could challenge authentication attempts from roaming authenticators with weak user verification.

From our analysis of RPs (Table 2), only 5/29 RPs declare a preference for `platform` authenticators, of which two (Mail.ru and Shopify) allowed registering a virtual authenticator with external transports. Thus, most RPs are potentially exposed to this attack on users with roaming authenticators which lack user verification.

We investigated how common such roaming authenticators were. In our April 13, 2023, MDS snapshot, we found that 139/160 authenticators were capable of acting as roaming authenticators. Of those, only 4 can locally authorize an authentication without requiring any user verification at all. Thus, the vast majority of roaming authenticators (even with user verification as simple as user presence) could mitigate this issue in practice, limiting the exposure of RPs.

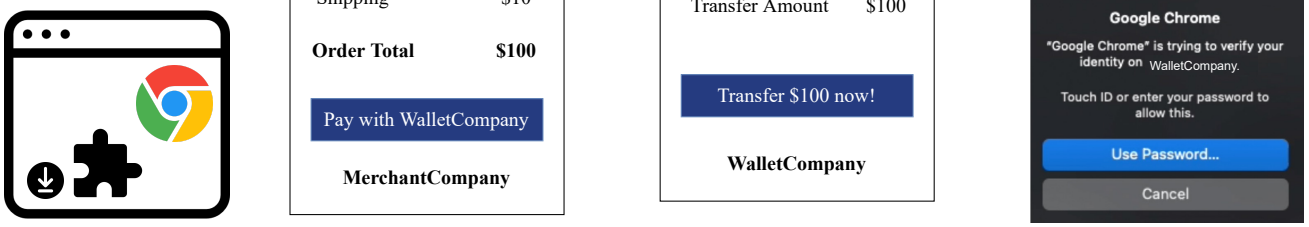
Root-level malware can bypass FIDO-based authentication on external authenticators, which lack on-device user verification.

8.2 Social Engineering Attack to Authenticate Sensitive Action

Here, we consider the scenario where the FIDO Client resides as a Trusted Application in the TEE, so the malware (regardless of user-level or root-level) cannot directly trigger an authentication. Instead, it must leverage a social engineering approach to cause an attacker-desired authentication, as FIDO2's workflow involves a human-in-the-loop.

Prior work has proposed such a social engineering attack on FIDO2. Jubur et al. demonstrated an attack where multiple near-concurrent and indistinguishable 2FA authentication prompts can trick the user into authorizing the attacker-initiated attempt [58], similar to "MFA fatigue" and "push phishing" attacks [27, 72]. However, their attack requires a synchronized timing of when the user would attempt to log in, and hence the authors consider their attack a targeted one rather than a scalable one. Without an attack that could be executed automatically and at scale, one could argue that this is a limited real-world threat. However, we investigate and identify a unique social engineering attack that can be executed in an automated fashion by malware on the client device, thus raising this threat's practicality.

We observe that in practice, real-world implementations of FIDO2 lack explicit user consent for a specific action, as originally recommended by FIDO [64]. As a result, users lack clarity about what specific action they are authenticating. In addition, online services apply Risk-Backed Authentication (RBA) systems that users lack transparency into, and thus users are unable to accurately predict when they may be asked to (re-)authenticate [52, 83]. We combine these two observations to construct a unique social engineering attack that tricks a user into authenticating an attacker-initiated sensitive action (which the user does not realize is the action being authenticated), at the same time as when the user has initiated a non-sensitive action (but does not realize that the action does not actually require re-authentication). Sensitive actions could vary



(a) User downloads and installs a malicious browser extension [61].

(b) With an active session on WalletCo., user attempts to make a transaction on another merchant site, say MerchantCo.

(c) Extension detects the transaction and attempts attacker-initiated action (e.g., transferring funds to the attacker's wallet) in a background tab. This triggers a FIDO2 authentication prompt.

(d) User is tricked into authenticating the attacker's transaction, mistaking it as authenticating their own transaction (which did not actually require authentication).

Figure 1: Using WalletCo. as an example target RP, we demonstrate how a malicious browser extension could trick the user into authenticating an attacker-initiated sensitive action on WalletCompany, such as transferring funds to the attacker's wallet. We demonstrate this attack with PayPal (WalletCo.) and eBay (MerchantCo.). Attack details are outlined in Section 8.

from RP to RP – we broadly define it as an action that a user performs on their account which requires re-authentication or a step-up/challenge authentication due to its sensitive nature. For example, financial and online banking sites often consider financial transactions to new parties as sensitive, and many other online services designate access to or change of personal profile settings as sensitive (e.g., changing a recovery email address). As a real-world example, Bank of America requires user verification when transferring funds above a threshold, and supports FIDO2 authentication for user verification [68].

Attack Setup/Assumptions: Conceptually, our attack makes no assumption about the platform and authenticator properties. Given the KHAccessToken/authenticatorClientPIN protection discussed in Section 2, user-level malware, which would have a different (untrusted) AppID (and hence a different KHAccessToken/authenticatorClientPIN), would not be able to trigger an authentication using existing FIDO2 credentials. However, in the web context, browser extensions are different because they act under the same AppID as the credential-registering application (that of the browser), and have the same scope of control as the actual user. Also, given that malicious browser extensions are already known vectors for social engineering attacks [61], we choose to construct our attack via a Chrome extension. (Note that as root-level malware can bypass the KHAccessToken/authenticatorClientPIN protection, spoofing the AppID, it can conduct a similar attack.)

The attack setup requires the user to have FIDO2 passwordless authentication enabled on their account at a target RP. When they interact with the RP in a browser and perform an action that the user might consider sensitive (requiring re-authentication), but in fact is not, the malicious extension will simultaneously initiate a sensitive action on the RP in the background. The user will be tricked into authenticating the attacker's sensitive action mistaking it for a re-authentication challenge for their own action. Note that as our attack assumes malware on the device, the attacker already has access to an authenticated session. However, our focus is rather on attacking the step-up authentication challenge/RBA.

For our attack, the threat model assumes that the authentication prompt raised by the OS, as seen in Figure 1d, cannot be compromised. However, root-level malware can compromise such prompts, and trick the user into authenticating an attacker-controlled session for an RP different from the one that the user is interacting with.

Attack Description: To describe our attack concretely, we assume WalletCompany to be an example target RP deploying FIDO2 for passwordless authentication. WalletCompany provides checkout functionality at merchant sites as well as person-to-person (P2P) transfer of funds. We choose the user-initiated action to be a checkout on MerchantCompany, a merchant site, and the attacker-initiated sensitive action to be P2P transfer of funds to the attacker's wallet on WalletCompany. The attack should also work on other RPs for other combinations of user-initiated sensitive actions.

Once the malicious Chrome extension is installed (Figure 1a), it waits for the user to visit the merchant site to make a transaction. This could be achieved by inserting a content script that records all elements which are clicked. When the user checks out with WalletCompany, as seen in Figure 1b, the click is detected and the extension spawns a background Chrome process to initiate a transfer of funds to the attacker's own wallet (Figure 1c) – since the Chrome instance has an active session on WalletCompany, no authentication is required by WalletCompany at this stage. However, the P2P transfer of funds to a previously unseen wallet is typically a sensitive action and would trigger re-authentication. The user will see a WebAuthn authentication prompt within seconds of attempting to check out with WalletCompany on MerchantCompany, as seen in Figure 1d. Since RBA systems are neither transparent nor well-understood [52], the user will be tricked into believing that

the re-authentication prompt is for their checkout on MerchantCompany. Once they authenticate, the transfer of funds would be authorized, and the user would not notice anything suspicious because their checkout would have succeeded anyways.

Attack Proof-of-Concept (PoC): For simplicity, we demonstrate our concrete example attack using PayPal and eBay, which are leading mobile wallet and e-commerce companies respectively – via a malicious browser extension on Chrome 103.0.5060.53 on a 2019 MacBook Pro running OS X 12.4. We enabled passwordless authentication in one of the author's PayPal account, registering with Mac's Touch ID via Chrome. We also log into PayPal before the attack executes, establishing an active authenticated session. We will use the terms WalletCo. and MerchantCo. for PayPal and eBay respectively, to describe the attack as it can be generalized over other similar applications.

To demonstrate that an authentication prompt can be raised by a background process spawned by an extension, we present a PoC Chrome extension in Appendix Listing 1. The extension requires permissions `tabs` and `scripting` to open the target RP's site in a tab and execute JS scripts in its context, as well as `host_permissions` to our target RP (WalletCo.). The extension opens WalletCo.'s WebAuthn login page in a new background window and simulates a login button click to raise a WebAuthn authentication prompt to the user. The user provides their biometrics, authorizing the attacker-controlled session. Other than the prompt, the user never sees anything and their only action is providing their biometrics. The attack is automated and takes only seconds to execute (e.g., time for the page load and for the user to provide biometrics).

Measurement of Mitigation Adoption: To our knowledge, there are only a limited number of existing mitigations to this attack, so RPs and users are broadly vulnerable. However, we believe there are tractable directions for better mitigations.

For e-commerce specifically, the W3C Working Group has developed Secure Payment Confirmation [75], which is available as the `payment` extension in WebAuthn and is fully supported by Chrome. It provides transaction confirmation functionality to financial services RPs, with permission to perform registration and authentication ceremonies on behalf of the RP on merchant sites. The transaction confirmation includes information such as `payeeName`, `payeeOrigin`, `currency` and `value`, among other things – thus mitigating the potential for identity confusion in the context we demonstrated. However, in our real-world measurement of RPs (Section 7), we did not find any RP supporting the `payment` extension. In our MDS snapshot, we found that only 15% authenticators supported a transaction confirmation display.

In the absence of protocol-level protection, user agents could also mitigate such an attack by employing certain sanity checks before raising a WebAuthn request to the OS. For example, our attack can be mitigated if Chrome restricts `navigator.credentials` API requests to be made only by a page that is in the user's focus, and if RPs specify that user verification by the authenticator is either required or preferred (and not discouraged). To our knowledge, existing browsers do not implement such policies.

Additionally, in line with Prakash et al.'s proposal to modify the authentication prompt design [70] to counter Jubur et al.'s attack, RPs could send out-of-band notifications, such as a push notification to the user's mobile device, indicating the transaction details. We are not aware of any RPs currently doing so.

Attack Disclosure: We disclosed this attack via Google's Bug Bounty Program and recommended that Chrome restrict `navigator.credentials` API usage to the page in focus to prevent such misuse. Google's response was that while confusion with WebAuthn interactions across tabs is a known issue, our attack demonstrating the bypass of step-up authentication is plausible, and a fix for it will be prioritized [12]. They noted that they would need to work with RPs to gracefully handle the case of restoring a set of tabs on a Chrome restart, as some of them might initiate re-authentication from the background.

The **lack of trusted transaction confirmation information can be exploited by social engineering attacks** to trick the user into authenticating actions they do not intend to. Improved UI and user education could be stopgap solutions until the ecosystem matures to defend against such attacks.

9 CONCLUDING REMARKS

In this work, we evaluated the security posture of real-world FIDO2 deployments across the Tranco Top 1K, particularly focusing on the impact of compromised clients. Here we draw on our findings to synthesize key takeaways, as well as propose high-level recommendations for FIDO2 deployments.

Compromised clients are a realistic and salient threat to FIDO2 deployments. In Section 6, we found that 96% of authenticators present in FIDO MDS are not certified to be malware-resistant, so a significant population of existing authenticators could potentially be compromised by a motivated attacker. From real-world authenticator telemetry (via our partner RP), we found evidence already of system integrity compromise on ~0.25% of hardware-backed clients and ~13% of non-hardware-backed clients. Our evaluation of RPs in Sections 7 and 8 revealed how RPs have not yet adopted recommended mitigations to combating compromised clients, making them potential targets. In fact, our partner RP already has observed online abuse from active FIDO-authenticated sessions on compromised devices. Thus, our study highlights that compromised clients are a salient threat that must be accounted for by FIDO2 deployments. As FIDO2 gains adoption, attackers will be even further incentivized to utilize this attack surface.

Improvements to mitigations are needed. Through our study, we uncovered various shortcomings with the existing FIDO2 mitigations to compromised clients. For example, a key recommendation is to maintain an assurance level for registered authenticators by allowlisting either known authenticator AAGUIDs or authenticators with particular attributes. However, in Section 7.1.2, we found that nearly all (27/29) RPs do not enforce such a policy. To the best of our knowledge, Microsoft is the only RP that mentions a "Key restrictions policy" for passwordless authentication [67] (while BofA states that they require a FIDO2-certified authenticator, but we found that in reality, they do not verify if an authenticator is actually certified). While some RPs may intentionally allow all authenticators to encourage broader adoption[1], security-sensitive

[1]Passkeys are user-friendly FIDO2 credentials [18] designed to sync across devices. They are not device-specific, and as such, currently do not support attestation [36].

RPs (e.g., BofA, PayPal, Stripe) should ideally implement such a policy and adhere to strong authentication requirements (e.g., the European Union's Second Payment Services Directive [19]). We attempted to share our findings with all 27 RPs via their vulnerability disclosures forms. In our disclosure, we briefly described our threat model as well as reproducible steps to register an untrusted authenticator to their site. PayPal worked with us to implement the recommendations. For the other RPs, the most common response (7 RPs) was justifying their choice as a design decision, followed by 4 RPs which requested implementation details for the malware we described in theory, and 3 RPs which considered malware out of scope. We did not hear back from 4 RPs, and we were unable to contact the remaining 8 RPs (5 of which did not accept our report due to an insufficient reputation score on HackerOne).

Another key recommendation is that RPs make risk decisions based on authenticator metadata available from the FIDO MDS. However, we found the MDS to be incomplete, as the metadata for popular authenticators (e.g., Apple devices) was missing. We also identified in Section 7.2.2 that the MDS is not promptly updated with authenticator vulnerability information. These findings motivate the need for better MDS maintenance so that it provides reliable authenticator information to support the FIDO2 ecosystem.

We found that limited mitigations exist for malware-driven social engineering attacks, such as the one described in Section 8.2, which can exploit the human-in-the-loop even if trusted authenticators are registered. The FIDO Alliance had initially proposed gathering explicit user consent for actions (i.e. 'Transaction Confirmation' [64]), which would have mitigated such attacks by allowing an RP to confirm that the transaction is exactly what the user intended. Subsequently, the W3C Working Group proposed the extension `txAuthSimple` in WebAuthn's first specification [41], allowing RPs to specify a prompt to be shown to users on a trusted display (e.g., the prompt in Figure 1d). However, it was removed from WebAuthn's second specification, due to a lack of client implementation support [54]. However, our results highlight the dire need for such mechanisms to be widely supported, allowing RPs to securely convey to users what they are authorizing. (In fact, some developers have similarly expressed their concerns with `txAuthSimple`'s removal and advocated for it to be officially supported [54].) In the meantime, UI/UX improvements (e.g., out-of-band notifications) can help users understand the action they are authorizing, reducing the impact of social engineering attacks.

Risk policies can harden FIDO2. For securing traditional password-based authentication, online services have employed risk-based authentication (RBA) models for years [52]. Given the continued threat of compromised clients, as online services move towards FIDO2-based authentication, we believe that they will benefit from similar risk-based policies built on FIDO2-specific features.

A key challenge for RPs during FIDO2 authentication is the reduced risk telemetry, which hampers their ability to detect anomalous login attempts. Risk signals from password authentication, including autofill/typing behavior, incorrect password attempts, and mouse movements [51], lack equivalents for FIDO2 authentications. Instead, FIDO2-specific signals are needed, such as round-trip communication time with the authenticator. For instance, an RP might decide to allow authenticators without attestation requirements to reduce user friction, but challenge authentication attempts with anomalous round-trip communication times with the authenticator. While Whalen et al. argued that a successful FIDO2 credential creation represents a cryptographic attestation of human signals [81], identity confusion attacks such as the one demonstrated in Section 8.2 violate that assumption (even if using a secure authenticator).

Interestingly, we have already observed some evidence of RPs modifying the risk profile for users who authenticate via FIDO2, primarily considering such user authentications to be less risky. Namecheap indicated that users registering a FIDO2 authenticator would never see a CAPTCHA [60], while BofA said that a FIDO2 authenticator can be used as an alternative to SMS-OTPs to verify high-value transactions [68]. Namecheap's waiver of bot defense is similar in concept to Cloudflare's Cryptographic Attestation of Personhood (CAP) recently proposed by Whalen et al. [81], which uses FIDO2's attestation to distinguish human-originated traffic from bot traffic. However, we found in Section 7.2.3 that as Namecheap and BofA both allow registering virtual authenticators, FIDO2 authentications are not inherently less risk, and these RPs remain exposed to automated abuse and malware threats.

We propose that the security community investigate augmenting existing authentication risk models [51, 63, 82, 84] to include additional risk signals on whether a FIDO2 authentication attempt was initiated by the intended user. FIDO2 defines an extensible extension interface, which allows one to define custom extensions based on a specific use-case [71]. Future work could explore an extension that runs within the trusted FIDO2 Client and fingerprints the user's interaction with the authenticator, to distinguish between a real user versus automated activity. The FIDO2 Client's privileged access could further be utilized to fingerprint the user's presence via kernel-level events [28], device characteristics [29, 31], or even hardware-level features [35]. Ultimately, such risk models could help harden FIDO2 deployments in practice.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback. This work was supported in part by grants from PayPal, Inc. and the National Science Foundation award CNS-2055549. The opinions expressed in this paper do not necessarily reflect those of the research sponsors.

REFERENCES

[1] 2017. *Universal 2nd Factor (U2F) Overview.* https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.html.
[2] 2018. *FIDO Alliance - Enterprise Adoption Best Practices.* https://media.fidoalliance.org/wp-content/uploads/Enterprise_Adoption_Best_Practices_Lifecycle_FIDO_Alliance.pdf.
[3] 2018. *FIDO TechNotes: The Truth about Attestation.* https://fidoalliance.org/fido-technotes-the-truth-about-attestation/.
[4] 2021. *Apple Private PKI.* https://www.apple.com/certificateauthority/private/.
[5] 2021. *FIDO Biometric Requirements.* https://fidoalliance.org/specs/biometric/requirements.
[6] 2022. *Authenticator Level 1 - FIDO Alliance.* https://fidoalliance.org/certification/authenticator-certification-levels/authenticator-level-1.
[7] 2022. *Authenticator Level 1+ - FIDO Alliance.* https://fidoalliance.org/authenticator-level-1/.
[8] 2022. *Authenticator Level 2 - FIDO Alliance.* https://fidoalliance.org/certification/authenticator-certification-levels/authenticator-level-2/.
[9] 2022. *Authenticator Level 3 - FIDO Alliance.* https://fidoalliance.org/certification/authenticator-certification-levels/authenticator-level-3/.
[10] 2022. *Authenticator Level 3+ - FIDO Alliance.* https://fidoalliance.org/certification/authenticator-certification-levels/authenticator-level-3-plus/.

[11] 2022. *Certified Authenticator Levels - FIDO Alliance.* https://fidoalliance.org/certification/authenticator-certification-levels.

[12] 2022. *Chromium Issue 1341134.* https://bugs.chromium.org/p/chromium/issues/detail?id=1341134.

[13] 2022. *FIDO Authentication Specifications.* https://fidoalliance.org/specifications/download.

[14] 2022. *Okta Sign-in Widget.* https://github.com/okta/okta-signin-widget/blob/master/docs/classic.md.

[15] 2022. Passkeys. https://developer.apple.com/passkeys/.

[16] 2022. *WebAuthn: Emulate authenticators - Chrome Developers.* https://developer.chrome.com/docs/devtools/webauthn.

[17] 2023. *Android Developers - SafetyNet Attestation API.* https://developer.android.com/training/safetynet/attestation#use-response-server.

[18] 2023. FIDO Alliance - Passkeys. https://fidoalliance.org/passkeys/.

[19] 2023. *FIDO Alliance - PSD2 Compliance.* https://fidoalliance.org/psd2-compliance/.

[20] 2023. *FIDO Alliance Metadata Service.* https://fidoalliance.org/metadata/.

[21] 2023. *Google DevTools: Debug JavaScript.* https://developer.chrome.com/docs/devtools/javascript.

[22] 2023. *Google: Use a security key for 2-Step Verification.* https://support.google.com/accounts/answer/6103523.

[23] 2023. *MDN: CredentialsContainer.create().* https://developer.mozilla.org/en-US/docs/Web/API/CredentialsContainer/create.

[24] 2023. *PayPal: Two-Factor Authentication.* https://developer.paypal.com/braintree/articles/risk-and-security/control-panel-security/two-factor-authentication.

[25] 2023. *SAP Customer Data Cloud: accounts.auth.fido.register JS.* https://help.sap.com/docs/SAP_CUSTOMER_DATA_CLOUD/8b8d6fffe113457094a17701f63e3d6a/4594b321af26476ba6156c3dafd8428f.html.

[26] 2023. *Symantec Sitereview.* https://sitereview.bluecoat.com/#/.

[27] Lawrence Abrams. 2022. *MFA Fatigue: Hackers' new favorite tactic in high-profile breaches.* https://www.bleepingcomputer.com/news/security/mfa-fatigue-hackers-new-favorite-tactic-in-high-profile-breaches/.

[28] Muhammad Ejaz Ahmed, Hyoungshick Kim, Seyit Camtepe, and Surya Nepal. 2021. Peeler: Profiling Kernel-Level Events to Detect Ransomware. In *European Symposium on Research in Computer Security (ESORICS).*

[29] Furkan Alaca and Paul C Van Oorschot. 2016. Device fingerprinting for augmenting web authentication: classification and analysis of methods. In *ACM Annual Computer Security Applications Conference (ACSAC).*

[30] Manuel Barbosa, Alexandra Boldyreva, Shan Chen, and Bogdan Warinschi. 2021. Provable Security Analysis of FIDO2. In *Annual International Cryptology Conference.* Springer.

[31] Adam Bates, Ryan Leonard, Hannah Pruse, Daniel Lowd, and Kevin RB Butler. 2014. Leveraging USB to Establish Host Identity Using Commodity Devices. In *Network and Distributed System Security Symposium (NDSS).*

[32] Nina Bindel, Cas Cremers, and Mang Zhao. 2023. FIDO2, CTAP 2.1, and WebAuthn 2: Provable security and post-quantum instantiation. In *IEEE Symposium on Security and Privacy (IEEE S&P).*

[33] David Cerdeira, José Martins, Nuno Santos, and Sandro Pinto. 2022. REZONE: Disarming TrustZone with TEE Privilege Reduction. In *USENIX Security Symposium.*

[34] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. 2020. SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems. In *IEEE Symposium on Security and Privacy (S&P).*

[35] Sanjeev Das, Jan Werner, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose. 2019. SoK: The Challenges, Pitfalls, and Perils of Using Hardware Performance Counters for Security. In *IEEE Symposium on Security and Privacy (S&P).*

[36] Apple Developer. 2022. Touch ID - Attestation. https://developer.apple.com/forums/thread/708982.

[37] Google Developers. 2022. *Intent to Ship: WebAuthn minPinLength extension.* https://groups.google.com/a/chromium.org/g/blink-dev/c/VnXR-U3jROc.

[38] Periwinkle Doerfler, Kurt Thomas, Maija Marincenko, Juri Ranieri, Yu Jiang, Angelika Moscicki, and Damon McCoy. 2019. Evaluating Login Challenges as a Defense Against Account Takeover. In *The World Wide Web Conference (WWW).*

[39] Saba Eskandarian, Jonathan Cogan, Sawyer Birnbaum, Peh Chang Wei Brandon, Dillon Franke, Forest Fraser, Gaspar Garcia, Eric Gong, Hung T Nguyen, Taresh K Sethi, et al. 2019. Fidelius: Protecting User Secrets from Compromised Browsers. In *IEEE Symposium on Security and Privacy (S&P).*

[40] Armstrong et al. 2022. *Client to Authenticator Protocol (CTAP) - FIDO Alliance Proposed Standard.* https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-errata-20220621.html.

[41] Brand et al. 2019. *Web Authentication: An API for accessing Public Key Credentials, Level 1.* https://www.w3.org/TR/webauthn-1/.

[42] Baghdasaryan et al. 2020. *FIDO UAF Authenticator-Specific Module API - FIDO Alliance.* https://fidoalliance.org/specs/fido-uaf-v1.2-ps-20201020/fido-uaf-asm-api-v1.2-ps-20201020.html.

[43] Baghdasaryan et al. 2020. *FIDO UAF Protocol Specification - FIDO Alliance Proposed Standard.* https://fidoalliance.org/specs/fido-uaf-v1.2-ps-20201020/fido-uaf-protocol-v1.2-ps-20201020.html.

[44] Baghdasaryan et al. 2022. *FIDO Security Reference - FIDO Alliance Proposed Standard.* https://fidoalliance.org/specs/common-specs/fido-security-ref-v2.1-ps-20220523.html.

[45] Bradley et al. 2022. *Web Authentication: An API for accessing Public Key Credentials, Level 3.* https://w3c.github.io/webauthn.

[46] Hill et al. 2022. *FIDO AppID and Facet Specification - FIDO Alliance Proposed Standard.* https://fidoalliance.org/specs/common-specs/fido-appid-and-facets-v2.1-ps-20220523.html.

[47] Jack et al. 2021. *FIDO Metadata Service - FIDO Alliance Proposed Standard.* https://fidoalliance.org/specs/mds/fido-metadata-service-v3.0-ps-20210518.html.

[48] Jones et al. 2022. *Registries for Web Authentication - Internet Assigned Numbers Authority.* https://www.iana.org/assignments/webauthn/webauthn.xhtml.

[49] Florian M Farke, Lennart Lorenz, Theodor Schnitzler, Philipp Markert, and Markus Dürmuth. 2020. You still use the password after all – Exploring FIDO2 Security Keys in a Small Company. In *Symposium on Usable Privacy and Security (SOUPS).*

[50] Apple Developer Forums. 2022. *get webauthn attestation statement on Safari.* https://developer.apple.com/forums/thread/713195.

[51] David Freeman, Sakshi Jain, Markus Dürmuth, Battista Biggio, and Giorgio Giacinto. 2016. Who Are You? A Statistical Approach to Measuring User Authenticity.. In *Network and Distributed System Security Symposium (NDSS).*

[52] Anthony Gavazzi, Ryan Williams, Engin Kirda, Long Lu, Andre King, Andy Davis, and Tim Leek. 2023. A Study of Multi-Factor and Risk-Based Authentication Availability. In *USENIX Security Symposium (USENIX Security).*

[53] Sanam Ghorbani Lyastani, Sven Bugiel, and Michael Backes. 2023. A Systematic Study of the Consistency of Two-Factor Authentication User Journeys on Top-Ranked Websites. In *Network and Distributed System Security Symposium (NDSS).*

[54] GitHub. 2020. *Remove unimplemented extensions.* https://github.com/w3c/webauthn/issues/1386.

[55] GitHub. 2022. *google/openSK.* https://github.com/google/OpenSK.

[56] Jingjing Guan, Hui Li, Haisong Ye, and Ziming Zhao. 2022. A Formal Analysis of the FIDO2 Protocols. In *European Symposium on Research in Computer Security (ESORICS).*

[57] Charlie Jacomme and Steve Kremer. 2021. An Extensive Formal Analysis of Multi-factor Authentication Protocols. In *ACM Transactions on Privacy and Security (TOPS).*

[58] Mohammed Jubur, Prakash Shrestha, Nitesh Saxena, and Jay Prakash. 2021. Bypassing push-based second factor and passwordless authentication with human-indistinguishable notifications. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[59] Ansgar Kellner, Micha Horlboge, Konrad Rieck, and Christian Wressnegger. 2019. False Sense of Security: A Study on the Effectivity of Jailbreak Detection in Banking Apps. In *IEEE European Symposium on Security and Privacy (EuroS&P).*

[60] Namecheap Knowledgebase. 2022. *How can I enable/disable Two-Factor Authentication?* https://www.namecheap.com/support/knowledgebase/article.aspx/9253/45/how-can-i-enabledisable-twofactor-authentication/.

[61] Takashi Koide, Daiki Chiba, and Mitsuaki Akiyama. 2020. To Get Lost is to Learn the Way: Automatically Collecting Multi-step Social Engineering Attacks on the Web. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[62] Leona Lassak, Annika Hildebrandt, Maximilian Golla, and Blase Ur. 2021. "It's Stored, Hopefully, on an Encrypted Server": Mitigating Users' Misconceptions About FIDO2 Biometric WebAuthn. In *USENIX Security Symposium.*

[63] Song Li and Yinzhi Cao. 2020. Who Touched My Browser Fingerprint?: A Large-scale Measurement Study and Classification of Fingerprint Dynamics. In *ACM Internet Measurement Conference (IMC).*

[64] Rolf Lindemann and Bill Leddy. 2022. *FIDO Transaction Confirmation White Paper.* https://media.fidoalliance.org/wp-content/uploads/2020/08/FIDO-Alliance-Transaction-Confirmation-White-Paper-08-18-DM.pdf.

[65] Sanam Ghorbani Lyastani, Michael Schilling, Michaela Neumayr, Michael Backes, and Sven Bugiel. 2020. Is FIDO2 the kingslayer of user authentication? A comparative usability study of FIDO2 passwordless authentication. In *IEEE Symposium on Security and Privacy (S&P).*

[66] Philipp Markert, Daniel V Bailey, Maximilian Golla, Markus Dürmuth, and Adam J Aviv. 2020. This PIN can be easily guessed: Analyzing the security of smartphone unlock PINs. In *IEEE Symposium on Security and Privacy (S&P).*

[67] Microsoft. 2022. *Plan a passwordless authentication deployment in Azure Active Directory.* https://learn.microsoft.com/en-us/azure/active-directory/authentication/howto-authentication-passwordless-deployment.

[68] Bank of America: Privacy and Security. 2022. *Secured Transfer.* https://www.bankofamerica.com/security-center/faq/additional-security-features/.

[69] Kentrell Owens, Olabode Anise, Amanda Krauss, and Blase Ur. 2021. User perceptions of the usability and security of smartphones as FIDO2 roaming authenticators. In *Symposium on Usable Privacy and Security (SOUPS).*

[70] Jay Prakash, Clarice Chua Qing Yu, Tanvi Ravindra Thombre, Andrei Bytes, Mohammed Jubur, Nitesh Saxena, Lucienne Blessing, Jianying Zhou, and Tony QS

Quek. 2021. Countering Concurrent Login Attacks in "Just Tap" Push-based Authentication: A Redesign and Usability Evaluations. In *IEEE European Symposium on Security and Privacy (EuroS&P)*.
[71] Florentin Putz, Steffen Schön, and Matthias Hollick. 2021. Future-proof web authentication: Bring your own FIDO2 extensions. In *International Workshop on Emerging Technologies for Authorization and Authentication*. Springer.
[72] DUO Security. 2022. *Guide to push phishing defense and best practices.* https://help.duo.com/s/article/7615.
[73] Alon Shakevsky, Eyal Ronen, and Avishai Wool. 2022. Trust Dies in Darkness: Shedding Light on Samsung's TrustZone Keymaster Design. In *USENIX Security Symposium*.
[74] Yun Shen, Nathan Evans, and Azzedine Benameur. 2016. Insights into rooted and non-rooted android mobile devices with behavior analytics. In *ACM Symposium on Applied Computing*.
[75] Rouslan Solomakhin and Stephen McGruer. 2022. *Secure Payment Confirmation.* https://www.w3.org/TR/secure-payment-confirmation/.
[76] Chrome Web Store. 2022. *WebDevAuthn.* https://chrome.google.com/webstore/detail/webdevauthn/aofdjdfdpmfeohecddhgdjfniggddpd.
[77] Avinash Sudhodanan and Andrew Paverd. 2022. Pre-hijacked accounts: An Empirical Study of Security Failures in User Account Creation on the Web. In *USENIX Security Symposium*.
[78] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, et al. 2017. Data breaches, phishing, or malware? Understanding the risks of stolen credentials. In *ACM Conference on Computer and Communications Security (CCS)*.
[79] Enis Ulqinaku, Hala Assal, AbdelRahman Abdou, Sonia Chiasson, and Srdjan Capkun. 2021. Is Real-time Phishing Eliminated with FIDO? Social Engineering Downgrade Attacks against FIDO Protocols. In *USENIX Security Symposium*.
[80] Roman Unuchek. 2017. *Kaspersky Daily: Rooting your Android.* https://usa.kaspersky.com/blog/android-root-faq/11581/.
[81] Tara Whalen, Thibault Meunier, Mrudula Kodali, Alex Davidson, Marwan Fayed, Armando Faz-Hernández, Watson Ladd, Deepak Maram, Nick Sullivan, Benedikt Christoph Wolters, et al. 2022. Let The Right One In: Attestation as a Usable CAPTCHA Alternative. In *Symposium on Usable Privacy and Security (SOUPS)*.
[82] Stephan Wiefling, Markus Dürmuth, and Luigi Lo Iacono. 2021. What's in Score for Website Users: A Data-Driven Long-Term Study on Risk-Based Authentication Characteristics. In *International Conference on Financial Cryptography and Data Security (FC)*.
[83] Stephan Wiefling, Luigi Lo Iacono, and Markus Dürmuth. 2019. Is This Really You? An Empirical Study on Risk-Based Authentication Applied in the Wild. In *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer.
[84] Stephan Wiefling, Tanvi Patil, Markus Dürmuth, and Luigi Lo Iacono. 2020. Evaluation of risk-based re-authentication methods. In *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer.
[85] Leon Würsching, Florentin Putz, Steffen Haesler, and Matthias Hollick. 2023. FIDO2 the Rescue? Platform vs. Roaming Authentication on Smartphones. In *Conference on Human Factors in Computing Systems (CHI)*.
[86] Luyi Xing, Xiaorui Pan, Rui Wang, Kan Yuan, and XiaoFeng Wang. 2014. Upgrading Your Android, Elevating My Malware: Privilege Escalation through Mobile OS Updating. In *IEEE Symposium on Security and Privacy (S&P)*.
[87] Hang Zhang, Dongdong She, and Zhiyun Qian. 2015. Android Root and its Providers: A Double-Edged Sword. In *ACM Conference on Computer and Communications Security (CCS)*.

A PROOF-OF-CONCEPT CHROME EXTENSION

```
// trigger when the extension is installed/loaded
chrome.runtime.onInstalled.addListener(async () => {
    // open new Chrome browser window
    let window = await chrome.windows.create({
        url: 'https://www.walletcompany.com/signin-
     webAuthn' ,
        focused: false,
        state: 'minimized'
    });
    // wait for the page to load
    setTimeout(() => {
        chrome.scripting.executeScript({
            target: {
                tabId: window.tabs[0].id
            },
        // instrument an authentication attempt
            function: () => {
                document.getElementById('logIn_start').
     click();
            }
        });
    }, 1000); // wait 1 second
});
```

Listing 1: `background.js` for the Chrome extension PoC that visits WalletCo.'s WebAuthn login page in a new background window, and simulates a click on the login button, to raise a WebAuthn authentication prompt to the user.

B AUTHENTICATOR AAGUIDS

The following are the complete AAGUIDs of the authenticators observed in Table 3:

- **Apple:** f24a8e70-d0d3-f82c-2937-32523cc4de5a.
- **Android:** b93fd961-f2e6-462f-b122-82002247de78.
- **Packed (suspected Apple Touch ID):** adce0002-35bc-c60a-648b-0b25f1f05503.
- **Chrome's Virtual Authenticator:** 01020304-0506-0708-0102-030405060708.

C LEVEL 2 AUTHENTICATORS

The following are the authenticators certified at Level 2 (i.e., malware resistant), as mentioned in Section 5.2.

- Feitian BioPass FIDO Security Key.
- Feitian MultiPass FIDO Security Key.
- Feitian ePass FIDO Security Key.
- GoTrust Idem Key FIDO2 Authenticator.
- GoTrust Idem Kev U2F Authenticator.
- Precision InnalT Key FIDO 2 Level 2 certified.
- eWBM eFA310 FIDO2 Authenticator.