

Remote WebAuthn: FIDO2 Authentication for Less Accessible Devices

Paul Wagner^a, Kris Heid^b and Jens Heider^c

Fraunhofer Institute for Secure Information Technology, Rheinstraße 75, 64295 Darmstadt, Germany

Keywords: FIDO2, Security, Authentication, Remote, Webauthn.


Abstract: Nowadays, passwords are the prevalent authentication mechanism, even though it is proven to offer insufficient protection against cyber crime. Thus, FIDO2 was released with a more secure authentication mechanism. FIDO2 enables authentication with cryptographic hardware, such as USB sticks, NFC cards or in the smartphone integrated chips. A device with FIDO2 support is required to implement the whole FIDO2 stack and offer the required interfaces for the security hardware. However, many systems like for example Smart TVs can not make use of FIDO2 due to the lack of HW interfaces or the usage of outdated software. To overcome this, we present Remote WebAuthn, which enables secure authentication on such restricted devices through a remote authentication from a secondary, FIDO2 compatible device, such as a smartphone. We evaluate our approach to have better usability compared to FIDO2 while maintaining most security advantages.


1 INTRODUCTION

Passwords are already the dominating authentication method for over the last decades, even though they are known to provide insufficient protection (Bonneau et al., 2012; Bonneau and Preibusch, 2010). Constantly changing passwords, long and random enough passwords or individual passwords for each service are accepted by users, but mostly hated concepts due to their inconvenience. Also, security enhancements such as two-factor-authentication are not well adopted by users (Voigt, 2019). This leads to the conclusion that many users sacrifice security for usability. As a way out of the password dilemma, FIDO2 authentication has been published. FIDO2 allows authentication via a hardware security key, such as a USB stick, NFC card/tag or a Bluetooth device. Such a key can be used as full authentication mechanism or as a second factor authentication together with for example a finger print or a pin to unlock the security key¹. The user connects the security key and legitimates the request for every authentication. In general, FIDO2's security is commonly well perceived and handled as the successor of passwords (Lyastani et al., 2020), the only stumbling block might be the usability among all devices. A device that should

leverage FIDO2 must on the one hand support the full WebAuthn protocol stack and on the other hand also the interfaces for the security key (NFC, USB or Bluetooth). Many IoT devices or Smart TVs also require authentication for online services, but have very restricted hardware and software interfaces hindering the usage and the spread of FIDO2. For such devices, passwords would still be required, leaving an open area for attackers. To overcome these flaws, we present Remote WebAuthn, utilizing the central elements of the WebAuthn standard. As a benefit, we are allowing a remote authentication through a secondary device, instead of connecting the security key to the device requiring authentication. Thus, the user can freely choose interface requirements and use integrated hardware keys on restricted devices. This work provides a proof of concept architecture and implementation for Remote WebAuthn in Section 3. We evaluate our concept based on sophisticated usability (Bonneau et al., 2012) and security (Schneier, 1999; Microsoft, 1999; Kohnfelder and Garg, 1999) criteria in Section 4. The evaluation is also done with password-based authentication and FIDO2, where we compare our approach against. Finally, we are concluding the pros and cons of our approach and give an outlook on future work.

^a  <https://orcid.org/0000-0002-6125-574X>

^b  <https://orcid.org/0000-0001-7739-224X>

^c  <https://orcid.org/0000-0001-8343-6608>

¹ <https://www.w3.org/TR/webauthn/>

2 AUTHORIZATION METHODS

Authentication can be reached by different means: knowledge, possession or biometric properties. Each of which can be used as a single factor authentication mechanism, or they can be combined to increase the security level.

Biometric properties like face recognition or finger prints are in practice only used for non-critical authentication or as a second factor. These mechanisms have been proven to be easily spoofed and also special hardware is required to capture these biometric properties.

Password based methods represent a simple subtype of knowledge based authentication. Because of their simplicity in implementation, they still constitute the predominant form of authentication on the web (Reese et al., 2019). There are several practices to add an extra layer of security, such as time-based tokens (OTP) and hardware keys (Reese et al., 2019). Since all of these approaches come in addition to regular password inputs and hence directly badly affect the usability (Florêncio and Herley, 2010; Bonneau et al., 2012), they are not further considered in this work.

Possession based authentication methods like FIDO2 can either be represented by integrated hardware or external security keys, which connect via USB, NFC or Bluetooth (FIDO-specs, 2020). In any case, each identity is stored as an individual key pair, that can be verified using a challenge response algorithm. While the public key should be returned to the relying party upon creation, the private key must not ever leave the hardware token. On the technical side, FIDO2 is based on two pillars: CTAP specifies the communication between hardware token and client. On the other hand, WebAuthn describes the interface between client and application. Since this is mostly a browser-website relationship, WebAuthn endpoints are typically accessible via JavaScript. The bridge between CTAP and WebAuthn can either be divided between browser and operating system or implemented in browser context completely. Since no official operating system interface is designed, browsers relying on the first approach must offer explicit FIDO2 support for the concrete system they are running on.

In both cases, FIDO2 needs explicit software support and at least one compatible interface (USB, NFC or Bluetooth) or an integrated authenticator on the host system. *Krypton* (Krypton, 2020) is close to Remote WebAuthn in some regards. It allows users to authenticate against almost every FIDO2-capable website with their smartphone. To do so, Krypton requires the installation of a desktop browser plu-

gin. It can then be paired with a smartphone, running Krypton's authenticator app. In contrast, Remote WebAuthn does not require explicit device pairing and is therefore also usable on temporary devices like kiosk terminals. Despite all components being open source, Krypton is limited to their own infrastructure, hence it is not possible to use third-party authenticator apps. In addition, only integrated authenticators can be used, making it mutual exclusive to the use of external FIDO2 security keys. This paper instead proposes an open approach, that can be used on both public and private infrastructure.

3 REMOTE WEBAUTHN

The idea behind remote WebAuthn is to bring the security of FIDO2 to devices with limited IO interfaces or software support, such as Smart TVs or public internet terminals. From the user perspective, a remote WebAuthn log in flow would look like follows: The user wants to sign in to a service on the Smart TV, such as a video streaming service. Instead of providing username and password, the TV presents a QR-code on its screen. The user scans the QR-code with his smartphone, which then opens an authentication app. This allows picking a key file previously associated with the service. Finally, by clicking the presented allow button on the smartphone, the user grants the Smart TV to access the video streaming service. The described abstract behavior is highlighted from the technical perspective in the following.

3.1 Concept

The key principle behind Remote WebAuthn is to separate the FIDO2 client across multiple systems: While the first part remains on the restricted device on which the user wants to log into a service, the second part handles any authenticator communication on a dedicated system. For demonstration, in this paper the combination of a Smart TV as service client and a mobile phone as authentication device is used. Both parts communicate over an internet tunnel in background, but hide that complexity from the applications context. The parts in Figure 1 are defined as follows:

Relying Party Client (RP Client). The *RP Client* is running on the system that should authenticate against a service, for instance a Smart TV. Therefore, it is responsible for any *Relying Party (RP)*-related communication. WebAuthn commands are always structured into two parts: The request is sent from *RP* to the authenticator and contains either a

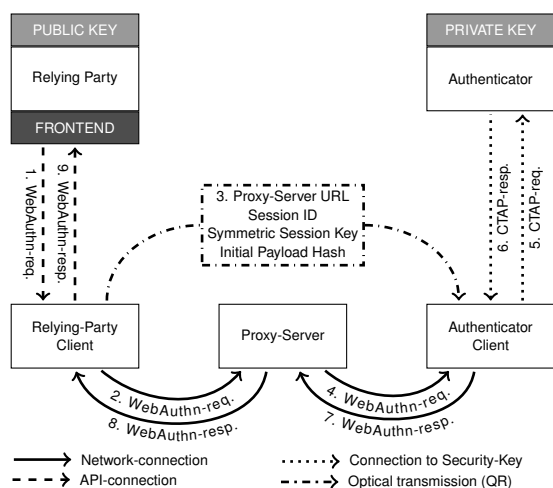


Figure 1: Concept of Remote WebAuthn.

credential creation request or a signable challenge. In reverse, the response carries the resulting cryptographic signature. Therefore, the *RP Client* always starts by submitting a request to the *Auth Client*. When receiving the response, the authentication is finished by passing the WebAuthn payload to the calling *RP*.

Authenticator Client (Auth Client). Supplementary to the *RP Client*, the *Auth Clients* handles the communication with the actual FIDO2 authenticator, which may be integrated or attached over USB, NFC or Bluetooth. After receiving the request submitted by the *RP Client*, the specified operation is executed and its response returned to the *RP Client*. Because of their direct camera and security hardware integration, smartphones are used as visual example in the course of this work.

Proxy Server (Proxy). Since the communication between *RP Client* and *Auth Client* equals a one-shot ping-pong game, the *Proxy* simply has to forward requests and responses to the opposite party. Thus, the main task is to coordinate and close the session as soon as the authentication completed.

The communication between client and the *Proxy* is implemented using *Web Socket Secure (WSS)*, which provides TLS transport encryption by default. To also hide exchanged data from the *Proxy*, any traffic is additionally end-to-end encrypted with AES. Both *Proxy* session and symmetric encryption keys are initialized by the *RP Client* in the first place. After obtaining a session id, it is transmitted to the *Auth Client* as QR code alongside with other relevant initialization features. Following the previously introduced device setup, the code is scanned from TV screen using a mobile phone with integrated authenticator.

In summary, the QR code contains the following data:

Base URL. When using an integrated scanner, the Base URL is only meant as fallback that shows up in case the user scans the code with a wrong scanner application. If Remote WebAuthn by contrast relies on an external scanner, the URL should instead contain a deep link, that is registered on the operating system and tells the scanner application to open the correct *Auth Client*. The consideration, whether to use integrated or external scanners, is further described in Section 3.1.1.

Proxy URL. Defines the *Proxy* to connect to via WSS.

Session ID. Identifies sessions uniquely and must therefore be issued by the *Proxy*.

Symmetric Session Key. Similar to the Session ID, the *RP Client* generates a Symmetric Session Key upon initialization. This key is used for end-to-end encryption during the whole dialogue, so that neither the *Proxy* nor underlying infrastructure gets access to exchanged data.

Initial Payload Hash. To ensure integrity, the encrypted WebAuthn payload is hashed after submission. In consequence, each QR code is also directly linked to a specific WebAuthn transaction. The following security analysis points out, that this makes phishing significantly harder for attackers.

Encrypting traffic between both clients ensures higher security, but also makes it impossible for *Proxys* to review package contents. Therefore, especially publicly driven *Proxys* are susceptible to abuse for other applications like file sharing. However, WebAuthn transactions contain only a single request-response pair, hence the *Proxy* also limits the maximum message count to a single initial message from *RP Client* to *Auth Client* and a response in reverse direction afterwards. In combination with a heuristic message size limit being just sufficient for WebAuthn payloads, abusive data-exchange scenarios become much more impractical.

3.1.1 Relying Party Client

As mentioned before, the *RP Client* builds a bridge between *RPs* and *Proxys*. There are two different approaches for a *RP Client* implementation:

Implementation as Browser Plugin. The key principle of an implementation as browser plugin is to override existing interfaces, which were formerly used for native WebAuthn authentication. After invocation, the plugin handles full communication with the smartphone, thus the whole process is transparent for the site. This allows great compatibility alongside websites, which already sup-

port WebAuthn authentication. However, using this method requires not only full-stack FIDO2 support, but the *RP Clients* also needs to explicitly support Remote WebAuthn in form of a browser plugin. This contradicts with the goal of being independent of the *RP Client* system environment.

Integration inside Website Context. To circumvent the problem of missing host support, it is also possible to fully implement Remote WebAuthn inside website context. Instead of calling default WebAuthn browser APIs, the *RP Client* embeds an own JavaScript library with a similar interface. After invocation, the website itself sets up communication with the *Proxy* and displays a QR code for initialization.

Since one premise of this paper is to bring the security enhancements of FIDO2 also to outdated devices like Smart TVs, only the latter approach is further considered in the course of this work.

3.1.2 Authenticator Client

As *RP Client* counterpart, the *Auth Client* joins an existing *Proxy* session and forwards incoming requests to an attached authenticator. Any required initialization data is transmitted via QR code, hence the *Auth Client* must provide an option for reading them. The scanner module can be implemented both externally and internally. When using the first approach, authentication QR codes are preceded by a deep link prefix, that allows external scanners invoking the target authenticator application. Since deep links can be used from any foreign application context, this opens a new attack vector on the other hand. Alternatively, the latter approach describes a scanner baked into the binary, so that QR code data never leaves the application sandbox.

3.1.3 Authentication Flow

Figure 1 shows a simplified communication flow for Remote WebAuthn transactions. For starting a basic authentication, the *RP* backend initially generates a cryptographic challenge. In parallel, the frontend-embedded *RP Client* module opens a connection with a *Proxy*. After receiving the Session ID, the *RP Client* then builds a WebAuthn request object using the cryptographic backend challenge. Combining the static configuration data, Session ID and the WebAuthn request hash, a QR showing a derived URL can be presented to the user. Even before scanning, the generated WebAuthn object can be sent out to the *Proxy*. When the user picks up his smartphone to scan the shown QR code, the *Auth Client* opens a *Proxy* connection using the stored information. The *Proxy* then

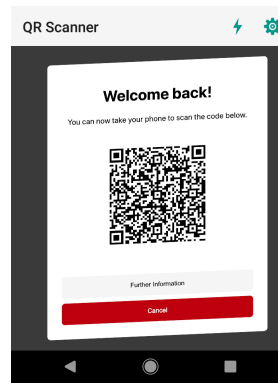


Figure 2: Scan QR code from restricted device on which to authenticate user on.

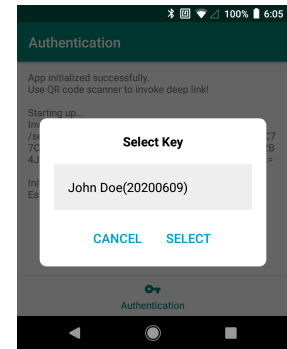


Figure 3: Remote authentication: Choose identity key from authenticator app.

releases the submitted WebAuthn request, so that the *Auth Client* can answer it subsequently. Depending on the *Auth Client* implementation and the requirements specified inside the WebAuthn request, the user may have to provide an additional biometric or knowledge-based feature. The generated response is sent back to the *Proxy* and then forwarded to the *RP Client*. Finally, the *RP* backend can verify the WebAuthn response and its encapsulated cryptographic signature using the users stored public key credential.

3.2 Example Implementation

The following briefly describes our implementation. The source code (RemoteWebAuthn, 2020) can be reviewed by the interested reader to get a deeper insight.

3.2.1 Proxy Server

A *Proxy* initializes and manages Remote WebAuthn communication between the *RP Client* and *Auth Client*. Besides the unique identifier and connection handles to both clients, each session contains a state machine and a deferred payload cache as assets. The state indicates, whether a WebAuthn request and response have been already transmitted. In addition, the payload cache temporarily stores the request in case the *Auth Client* has not yet been connected.

The *Proxy* is designed to operate in a completely passive way. Therefore, outgoing *Event Messages* are always a reaction of either incoming *Request Messages* or status changes such as a *Auth Client* join. The following description provides an overview over all *Event Messages*. For each message, the concrete trigger action is specified:

session_assignment. As soon as the connection between *RP Client* and *Proxy* is established, a

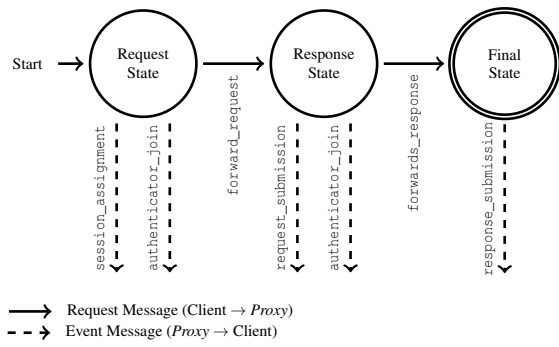


Figure 4: Proxy state machine.

session_assignment event containing the session identifier is emitted.

authenticator_join. This event has no functional purpose, but notifies about *Auth Clients* joining the session and can therefore used for visual progress updates.

request_submission. The request_submission carries the WebAuthn request to the *Auth Client*, whenever the *RP Client* submits a forward_request.

response_submission. In reverse to request_submission, response_submission redirects incoming WebAuthn responses via forward_response back to the *RP Client*.

RP Clients initialize new sessions using the /session endpoint, which then emits a session_assignment event containing the session identifier needed for QR code creation. The *Auth Client* can instead use the /session/{session_id} endpoint to join with its particular session id.

3.2.2 Relying Party Client

As mentioned before, the *RP Client* is designed as JavaScript library for the purpose of this paper. Therefore, the only requirement is a JavaScript-capable browser environment, which allows the *RP Client* running even on elderly devices like Smart TVs. The full functionality is encapsulated inside a Tunnel class, which takes the basic configuration via constructor and forwards authentication requests on function calls.

3.2.3 Authenticator Client

The related source code project (RemoteWebAuthn, 2020) implements a simple version of an *Auth Client* as Android app, which relies on an external scanner and hence registers a system-wide rwa: deep link prefix. In this implementation, the request is then passed to a software authenticator library for demon-

Table 1: Usability comparison chart.

		PW	FIDO2	RWA
U1	Memorywise-Effortless	X	✓	✓
U2	Scalable-for-Users	(✓)	(✓)	✓
U3	Nothing-to-Carry	✓	X	(✓)
U4	Physically-Effortless	X	X	X
U5	Easy-to-Learn	✓	✓	✓
U6	Efficient-to-Use	✓	✓	✓
U7	Infrequent-Errors	X	✓	✓
U8	Easy-Recovery-from-Loss	✓	✓	✓

✓ applies (✓) mostly applies X doesn't apply

stration purposes. For real-world implementations, more secure options using integrated security hardware or attached security keys should be chosen.

4 EVALUATION

This chapter compares Remote WebAuthn with FIDO2 and the password based authentication. The following analysis subdivides into two main sections: First, the usability is compared using criteria from existing literature. In contrast, the second part focuses on threat analysis by deducing appropriate scenarios and applying them to all introduced approaches.

4.1 Usability

For usability comparison, the criteria of (Bonneau et al., 2012) are applied. The following enumeration first summarizes the definitions and extends U2 by the aspect of scalability using multiple devices. Then, Table 1 applies the introduced criteria to all three approaches.

U1 Memorywise-Effortless. The authentication does not require a password input. The password based authentication of course does not match this criterion, as the password itself represents the primary asset. For FIDO2 and Remote WebAuthn however, depending on the used authenticator memory-effortless features like biometry can be chosen.

U2 Scalable-for-Users. Using multiple accounts or devices does not lead to an increased workload for users. When targeting the highest possible security for password based authentication, each relying party has to have its own passphrase. FIDO2 on the other hand may need multiple authenticators when using various clients with different interface requirements. Therefore, both approaches only match one of two criteria. By contrast, Remote WebAuthn matches the criterion completely by extending FIDO2 with the possibility to use any client with any authenticator.

Table 2: Threat comparison chart.

	PW	FIDO2	RWA
T1 Auth. with stolen identity	X	✓	(✓)
T2 Auth. with compromised identity	✓	✓	✓
T3 Denial of performed auth.	X	✓	(✓)
T4 Auth. information exposure	✓	✓	✓

✓ applies (✓) mostly applies X doesn't apply

U3 Nothing-to-Carry. Users are not required to carry a physical thing for authentication. This is only fully true for the password based authentication, that does only require remembering a passphrase. Instead, Remote WebAuthn requires carrying a smartphone as everyday object, whereas FIDO2 needs a dedicated piece of hardware.

U4 Physically-Effortless. No physical interaction, like pressing a hardware button, is required in order to authenticate. All introduced approaches require some form of physical interaction, either for typing a password or for providing biometric features.

U5 Easy-to-Learn. Users need no domain specific knowledge for performing authentication. This criterion is fulfilled for all three approaches for different reasons: The password based authentication is established on the internet for decades and hence already well-known for most users. FIDO2 by contrast just requires plugging in an authenticator and performing a simple action like a button press. Allowing fine-grained progress tracking, Remote WebAuthn on the other hand enables relying parties to guide the user step by step through the app installation and authentication process.

U6 Efficient-to-Use. The duration of a single authentication process is appropriately short. While it is hard to objectively measure the average duration of each approach, all approaches can be conducted in less than ten seconds as rule of thumb.

U7 Infrequent-Errors. Authentication tries do not fail too often, if conducted properly. Since technical failures are hard to predict, this analysis is limited to human mistakes only. Therefore, FIDO2 and Remote WebAuthn have an advantage over the password based authentication, as both approaches allow alternatives to error-prone password typing.

U8 Easy-Recovery-from-Loss. Users can easily recover their lost keys. In all cases, not the authentication providers, but the RPs itself are responsible for managing lost credentials.

4.2 Threats

This section compares the security across all introduced authentication methods. For this purpose, vari-

ous attack scenarios are described first. For each case, the different authentication variants are compared and finally, the chosen scenarios are justified using the STRIDE model (Microsoft, 1999).

T1 Authentication with Stolen Identity. In this scenario, an attacker tries to impersonate the victim by stealing his identity. This can either be achieved by hijacking an existing authentication attempt, or by passing a malicious authentication request to the victim. The latter case additionally allows to choose any *RP* instead of just the user-intended target service. For every approach, an attacker could either try to phish a foreign authentication by spoofing the *RP* identity (T1a) or by intercepting the traffic between *RP* and client directly (T1b).

Password based: An attacker can simply entice the victim to enter his password on a malformed input form, for instance by sending a malicious email or recreating the *RP* webpage under a similar domain. In most scenarios, stolen passwords even allow permanent access until their explicit invalidation. Since this method is only founded on knowledge, there is no effective countermeasure for scenario T1a. T1b on the other hand can be closed by implementing proper transport encryption.

FIDO2: Since FIDO2 is based on public key cryptography, permanent takeovers require physical access to the linked security key. Therefore, the only possibility left, is to gain a temporary authentication by stealing or manipulation WebAuthn transaction data. In theory, an attacker could rebuild the *RP* webpage and forward the original authentication data to the user. Referring to T1b, another possibility is to replace the authentication request in transit. Both scenarios are covered by the FIDO2 security reference SG-2 and SG-3 (Lindemann, 2018).

Remote WebAuthn: On the *RP Client* side, Remote WebAuthn extends FIDO2 by communication with a central *Proxy*. While the FIDO2 subset is already covered by its own analysis above, the additional transfer can be divided into two assets:

- The whole traffic between *RP Client*, *Proxy* and *Auth Client* cannot be accessed due to Symmetric Session Key encryption. Therefore, no WebAuthn data can be spoofed or modified during transport.
- The initialization data on the other contains all relevant data concerning the *Proxy* session. For stealing a foreign identity, the attacker can create an own *RP* session and forward the initialization QR code to the victim using a trusted medium. This is especially problematic when using deep links for authenticator invocation, which can be invoked from any application or website. How-

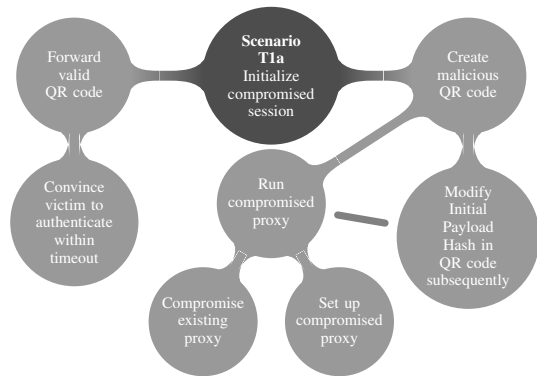


Figure 5: Attack tree for Remote WebAuthn scenario T1a.

ever, users still have to confirm each authentication explicitly. In order to decrease fraud, initialization QR codes also contain the WebAuthn requests cryptographic hash, which limits code validity to the FIDO2 transaction timeout. This measure prevents popular mail phishing and requires the attacker to create and forward the QR code or deep link in realtime instead.

T2 Authentication with Compromised Identity.

Instead of impersonating someone’s identity, an attacker tries answering to a foreign authentication request with his own credentials. Afterwards, users may expose personal data to the attackers account, since they are not aware of using a malicious account.

Password based: The attacker could modify the password in transit and hence authenticate against the *RP* instead of the user itself. This attack can be foiled by securing the traffic with transport encryption.

FIDO2: If the attacker gets access to the communication between *RP* and client, he might be able to answer the authentication request with an own security key. This scenario is covered by security goal SG-11 (Lindemann, 2018).

Remote WebAuthn: In addition to the described FIDO2 risks, there are the same conceivable scenarios for injecting a foreign authentication as in T1. Under the assumption, that the attacker cannot access the QR code containing the Symmetric Session Key, WebAuthn transaction data cannot be stolen or modified in transit. Therefore, for performing such an attack the attacker must get inside the victims physical range in order to scan the code.

T3 Denial of Performed Authentications. In this attack, the victim itself represents the attacker by denying an actually performed authentication approach. To succeed, there must be possibilities for

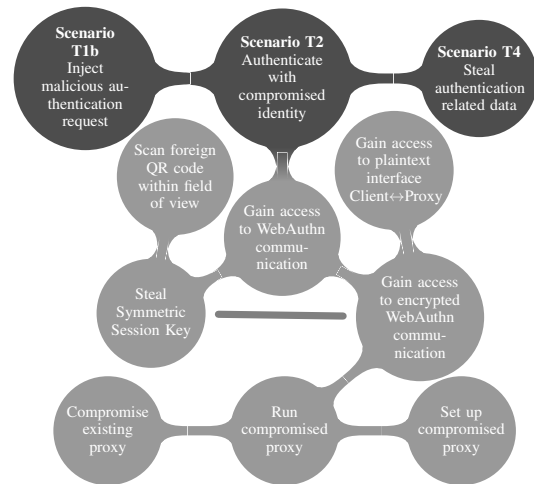


Figure 6: Attack tree for Remote WebAuthn scenarios T1b, T2, T4.

attackers to steal foreign identities. Therefore, this attack vector is only possible if and only if there’s a derivation for attack tree T1 in the respective scenario. For FIDO2 especially, this risk is covered by security goal SG-14 (Lindemann, 2018).

T4 Authentication Related Information Exposure.

The goal of an Information Exposure attack is not hijacking foreign identities. Instead, the authentication process may expose user related information such as usernames or mail addresses, that help disclosing the victims identity.

Password based. Especially weakly chosen passwords may contain personal information such as the date of birth or the children’s names. This makes it easy to draw conclusions about the victim’s identity. However, this threat can be defeated by conducting proper transport encryption.

FIDO2. If the attacker gets access to the transmitted WebAuthn request, he might be able to extract personal data like the mail address or the username. This scenario is covered by security goal SG-8 (Lindemann, 2018).

Remote WebAuthn. The Remote WebAuthn initialization data do not contain any privacy related information. Therefore, only WebAuthn transaction traffic may include sensitive data. Besides sniffing the encrypted network traffic, this attack necessarily requires access to the Symmetric Session Key for decryption. Consequently, this access cannot be performed without stealing the victims QR code.

4.3 Threat Justification with STRIDE

Spoofing. A spoofing attack targets the authentication integrity by stealing and reproducing the user’s

identity. In the context of authentication, this can be achieved following several attacks: The attacker can either try to impersonate the *RP* in order to send false authentication requests to the user (*T1*). Alternatively, he can try to imitate the users identity to authenticate against the *RP* (*T2*).

Tampering. Whereas any involved system is considered to be secure, the Tampering attack is limited to the modification of data in transit. As Figure 1 outlines, this can be achieved by either modifying the transmitted authentication data or its request.

Similar to the second Spoofing attack vector, an attacker can modify the authentication request in transit to obtain an authentication from the victim. Alternatively, it is also possible to pretend a false identity by faking a compromised authentication. Therefore, scenarios T1 and T2 are appropriate representations for this threat as well.

Repudiation. In this scenario, the victim, which is also the attacker at the same time, denies being responsible for an actually executed authentication. Therefore, this attack vector is equivalent to scenario T3.

Information Disclosure. Information Disclosure attacks target the protection goal of confidentiality. In this context, possible weak points are the authentication and its request in transit. Generally spoken it can be divided, whether the stolen data can be directly used for a following attack or for identity inference only. While the first case is identical to a *Spoofing* attack and hence covered by scenarios T1 and T2, the latter one is described by T4.

Denial of Service. The goal of a *Denial of Service* attack is to cause security vulnerabilities by disturbing the services availability. Since authentication providers only build the base for other applications, the concrete risk of an outage does not represent a relevant attack vector for this work.

Elevation of Privilege. None of the introduced approaches offer an inherent right management, hence this vector does not match the purpose of authentication. However, attackers may gain higher permission by stealing foreign identities (*T1*).

5 CONCLUSION

In this paper, we've described Remote WebAuthn, a novel authentication method related to FIDO2. Remote WebAuthn focuses on providing security close to FIDO2, on devices with restricted capabilities or interfaces and limited software support, such as Smart TVs. This publication covers the theoretical working mechanisms as well as an example implementation

which is also publicly available online for everyone to use. The example implementation (RemoteWebAuthn, 2020) was designed, such that device manufacturers could implement our method with ease. We have thoroughly evaluated usability based on Bonneau's sophisticated aspects (Bonneau et al., 2012) as well as the security based on Microsoft's STRIDE model (Microsoft, 1999). We concluded, that the security level of our approach is higher than password based authentication and slightly weaker compared to FIDO2. However, our method provides a higher usability compared to both alternatives. As future work, we see the development of an example implementation of an iOS client among the already existing Android client. With the publication of this paper as well as the example implementation, we're looking forward establishing a new open standard for restricted devices such as Smart TVs and increase the overall security of such products.

REFERENCES

- Bonneau, J., Herley, C., Van Oorschot, P. C., and Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE S&P*.
- Bonneau, J. and Preibusch, S. (2010). The password thicket: Technical and market failures in human authentication on the web. In *WEIS*.
- FIDO-specs (2020). Fido specifications overview. <https://fidoalliance.org/specifications>.
- Florêncio, D. and Herley, C. (2010). Where do security policies come from? In *SOUPS*.
- Kohnfelder, L. and Garg, P. (1999). The threats to our products. *Microsoft Interface, Microsoft Corporation*.
- Krypton (2020). Krypton website. <https://krypt.co>.
- Lindemann, R. (2018). Fido security reference. <https://fidoalliance.org/specs/fido-v2.0-rd-20180702/fido-security-ref-v2.0-rd-20180702.pdf>.
- Lyastani, S. G., Schilling, M., Neumayr, M., Backes, M., and Bugiel, S. (2020). Is FIDO2 the kingslayer of user authentication? A comparative usability study of FIDO2 passwordless authentication. In *IEEE S&P*.
- Microsoft (1999). The stride threat model. [https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878\(v=cs.20\)](https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)).
- Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J., and Seamons, K. (2019). A usability study of five two-factor authentication methods. In *SOUPS*.
- RemoteWebAuthn (2020). Remote WebAuthn source code. <https://github.com/pkwagner/remote-webauthn>.
- Schneier, B. (1999). Attack trees. *Dr. Dobb's journal*.
- Voigt, J. (2019). 5 sicherheitstipps zum safer internet day. <https://germany.googleblog.com/2019/02/5-sicherheitstipps-zum-safer-internet-day.html>.