A Security Evaluation of FIDO's UAF Protocol in Mobile and Embedded Devices

Christoforos Panos[1], Stefanos Malliaros[2], Christoforos Ntantogian[2], Angeliki Panou[2] and Christos Xenakis[2]

[1] Department of Informatics & Telecommunications, University of Athens, Greece
cpanos@di.uoa.gr
[2] Department of Digital Systems, University of Piraeus, Greece
{stefmal, dadoyan, apanou, xenakis}@unipi.gr

Abstract. The FIDO (Fast Identity Online) Universal Authentication Framework is a new authentication mechanism that replaces passwords, simplifying the process of user authentication. To this end, FIDO transfers user verification tasks from the authentication server to the user's personal device. Therefore, the overall assurance level of user authentication is highly dependent on the security and integrity of the user's device involved. This paper analyses the functionality of FIDO's UAF protocol and identifies a list of critical vulnerabilities that may compromise the authenticity, privacy, availability, and integrity of the UAF protocol, allowing an attacker to launch a number of attacks, such as, capturing the data exchanged between a user and an online service, impersonating a user at any UAF compatible online service, impersonating online services to the user, and presenting fake information to the user's screen during a transaction.

Keywords: Authentication, FIDO, security analysis, trusted computing, TPM, remote attestation, TrustZone, mobile and embedded devices.

1 Introduction

The most traditional form of authentication, i.e., one-factor, password-based authentication, has become a deficient and inconvenient solution for the modern-day user, who must keep track and maintain an ever-growing list of login credentials and passwords. In 2014, an average user had 25 accounts and performed logins 8 times a day, using 6.5 passwords [1]. Even more importantly, password-based authentication is becoming less secure. Users typically rely on low entropy passwords so that they are easy to remember. Furthermore, recent password breaches resulted in large password lists (55000 accounts from Twitter [2], 450000 accounts from Yahoo [3], and 6.5 million from LinkedIn [4]), which, in conjunction with today's abundant computing power, made password cracking a viable attack vector. The FIDO (Fast Identity Online) Alliance [5], a new industry working group, has been founded to define an open, interoperable set of authentication mechanisms that reduces the reliance on passwords and addresses the limitations and vulnerabilities of existing authentication schemes.

The FIDO set of specifications supports multifactor authentication (MFA) and public key cryptography. Two protocols are being developed, namely the universal second factor authentication (U2F) [6] and the universal authentication framework (UAF) [7]. Both protocols may either work in conjunction, or independently. The U2F protocol augments the security of existing password authentication mechanisms by adding a second factor to user login, and, therefore, does not alleviate the use of passwords. A user logs on using a username and a password, while the protocol prompts the user to present a second factor device for authentication. The UAF protocol, on the other hand, offers password-less authentication.

The operation of the UAF protocol involves the communication of two computing entities, one maintained by the service provider that requires a user's authentication and one controlled by the user that must be authenticated. The service provider is referred as the relying party and is typically composed of a web server and a UAF server. The web server provides a front-end interface to the users, while the UAF server is responsible for communicating UAF protocol messages to a user's device. On the client side, the users' computing entity consists of one or more UAF authenticators, the UAF client and user agent software. The UAF authenticator is an entity connected or integrated within user devices responsible for (i) user authentication and (ii) the generation and association of key pairs with relying parties. The UAF client constitutes the user-side endpoint of the UAF protocol (typically a browser plugin) and its main responsibility is the interaction and coordination of UAF protocol operations with the UAF authenticators on one end and the relying party on the other end. Finally, the user agent software is the software used by the end user (such as a browser or an application).

During the UAF protocol's operation, a user initially registers his/her device to a relying party, using one or more local authentication mechanisms such as a biometric scan, based on the authenticator policy imposed by the relying party. At this stage, the following operations take place: (i) the relying party utilizes an attestation mechanism to validate the legitimacy of the UAF authenticator(s) hosted by the user's device, and (ii), the UAF authenticator(s), associate the biometric scan with a newly generated key pair, retain the private key, and register the device to the relying party using the public key. Once registered, the user simply repeats the local authentication action whenever it is required to be authenticated to the service. The device's UAF authenticator verifies the user based on the authentication action, while the relying party verifies the device by transmitting a challenge, which is signed by the previously generated private key. Therefore, the overall assurance level of user authentication is highly dependent on the security and integrity of the user's device involved.

The UAF protocol provides several important advantages over traditional authentication mechanisms: it offers strong authentication (due to its reliance on public key cryptography); it simplifies the registration and authentication procedure; it alleviates the need for maintaining passwords, dealing with complex password rules, or going through password recovery procedures; and it strengthens user privacy, since all identifying information is stored locally, at the user's device. However, the operation of UAF relies on one fundamental assumption: the entities responsible for most of UAF's critical functionality, namely the UAF authenticator and the UAF client, are trusted and cannot be tampered by a malicious attacker. If either of these entities is compromised,

then, an attacker would be able to launch a number of critical attacks, such as, capture the data exchanged between a user and an online service, impersonate a user at any UAF compatible online service, impersonate online services to the user, present fake information to the user's screen during a transaction, access private keys used for authentication, to name a few, essentially compromising the authenticity, privacy, availability, and integrity of the UAF protocol.

In this paper, we perform an informal security analysis in which we identify several attack vectors that can be set to compromise the legitimate operation of the UAF protocol, including the ability of an attacker to: (i) gain unprivileged access to the cryptographic material stored within the UAF authenticator, and (ii) highjack either the UAF authenticator or the UAF client. Our analysis concentrates on the client-side UAF protocol functionality, which includes the most critical protocol entities, namely, the storage location for the authentication keys and the entities performing the authentication operation (i.e., the UAF authenticators and the UAF client). These entities typically operate in a consumer platform such as a mobile device, which is susceptible to a variety of attacks such as malware and viruses, its users deploy unsupervised software, and the deployed operating systems may be susceptible to several vulnerabilities. On the other hand, the server-side entities of the UAF protocol (i.e., the relying party) rely on widely adopted functionality typically associated with web servers (such as the use of TLS cryptographic protocols). Furthermore, we investigate and identify how an attacker can circumvent the security measures provided by the UAF protocol. Finally, we provide a threat analysis and investigate the impairment that may be caused by an attacker in the event of a successful exploitation of the UAF protocol. Overall the contributions of this paper are the following:

- We define and comprehensively analyze the client-side operation of the UAF protocol, including any associated security measures proposed by the UAF protocol specifications.
- We perform, to the best of our knowledge, the first undisclosed security analysis of the UAF protocol. Based on this analysis, we identify several critical attack vectors that can be exploited by a malicious entity in order to compromise the authenticity, privacy, availability, and integrity provided by the UAF protocol.
- Our analysis also reveals vulnerabilities in the security measures employed by the UAF protocol, as well as to entities outside the scope of the UAF protocol, which, can be exploited to circumvent the security measures, or, target the legitimate operation of the UAF protocol.
- Based on our security analysis, we identify and categorize the critical assets related to the UAF protocols' secure operation.
- We perform a threat analysis in which we investigate and identify several critical attacks that can be deployed by a malicious entity exploiting the attack vectors identified in our security evaluation, including user and relying party impersonation, phishing, and the disclosure of encrypted data.

The rest of this paper is organized as follows. In section 2 we analyze the functionality of the UAF protocol, outline the entities involved in the process of user registration and authentication, present the security measures proposed by the UAF specifications

and evaluate existing literature related to the evaluation of FIDO's security features and functionality. In section 3, we perform a security analysis of the UAF protocol, in which we identify several vulnerabilities and limitations that may be exploited by an attacker in order to circumvent any security measures and compromise the legitimate operation of the UAF protocol. Furthermore, we perform a threat analysis, with the goal of identifying the critical assets of the UAF protocol as well as the threats resulting from the attacks identified in the security evaluation. Finally, section 4 contains the conclusions.

2 Background

In this section, we first provide an overview of the UAF protocol's functionality. This overview covers the most critical aspects of the protocol's operations, including the process of registering and authenticating a user to a relying party. A more detailed analysis of the UAF protocol exists in [7]. Furthermore, in section 2.2, we survey any literature associated with the security evaluation of the UAF protocol. We outline the security measures proposed by the FIDO Alliance, analyze several entities associated with the security of the UAF protocol, and identify any associated vulnerabilities and limitations.

2.1 UAF protocol operations

The UAF protocol (see Figure 1) encompasses three major operations, namely, registration, authentication, and deregistration. During the registration operation, the UAF protocol allows a user to register to a relying party using one or more UAF authenticators. Once registration is complete, the user can then invoke the authentication operation, in which the relying party prompts for a user authentication using the UAF authenticator previously used during the registration operation. Finally, in the deregistration operation, the relying party can trigger the deletion of the authentication key material and remove the user from its list of authenticated users.

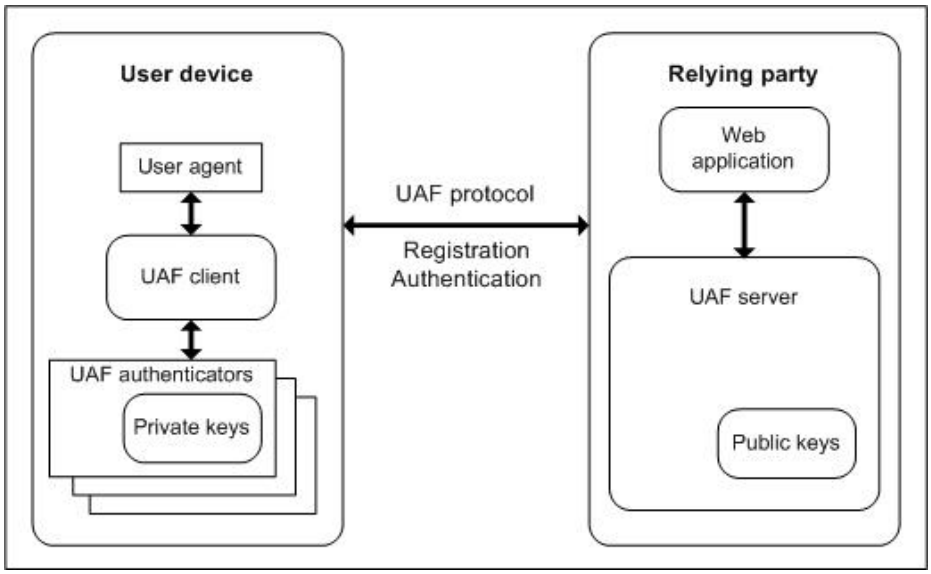

Fig. 1. the UAF protocol

The UAF registration operation. The registration operation is initiated when a user requests a registration to a relying party, either through a compatible application or through a browser. The relying party replies to the registration request by transmitting a registration message with the following parameters: the AppID, the authenticator policy, the server generated challenge, and the username to the UAF client residing in the user's device (illustrated in Figure 2). The AppID parameter is used by the UAF client to determine if the calling application (or website) is authorized to use the UAF protocol and it is associated with a key pair by the UAF authenticator (during key generation), so that access to the generated key pair is limited to its respective application. The authenticator policy lists the type of UAF authenticators required by the relying party, while the server generated challenge is a random nonce value used to protect against replay attacks. Finally, the username parameter is used by the UAF authenticator to distinguish key pairs that belong to the same application (or website), but to different users.

Once the UAF client receives the registration message from the relying party, it first identifies the calling app (or website) and then determines (based on the AppID parameter) whether the associated application is trusted and allowed to proceed with a registration request. To accomplish this, the UAF client queries the relying party for the trusted facet list (i.e., a list of all the approved entities related to the calling app) and, based on this list, decides whether registration will proceed or not. For example, if the registration request was initiated by an application, then the trusted facet list will contain a signature of the calling application that the UAF client can use to verify the app. If, on the other hand, the registration was initiated by a website, then the trusted facet list will contain all the associated and approved domain names. Subsequently, the UAF

client will check the authenticator policy parameter and generate a key registration request to the set of UAF authenticator(s) mandatory by the policy. If the required UAF authenticators are not present in the user's device, then the registration operation will be canceled.

The UAF client communicates with the UAF authenticator(s) using the authenticator specific module (ASM), a software associated with a UAF authenticator that provides a uniform interface between the hardware and the UAF client software. At this stage, the UAF client performs the following operations: it first calls the UAF authenticator in order to compute the final challenge parameter (FCP), which is a hash of the AppID and the server challenge. Then, it generates the KHAccessToken, which is an access control mechanism for protecting an authenticator's UAF credentials from unauthorized use. It is created by ASM by mixing various sources of information together. Typically, KHAccessToken contains the following four data items: AppID, PersonaID, ASMToken and CallerID. The AppID is provided by the relying party and it is contained within every UAF message. The PersonaID is obtained by ASM from the operating system, and, typically, a different PersonaID is assigned to every user account. The ASMToken is a random generated secret which is maintained and protected by ASM. In a typical implementation ASM will randomly generate an ASMToken when it is first executed and will store this secret until it is uninstalled. CallerID is the calling UAF client's platform assigned ID. Once the FCP and the KHAccessToken are computed, the UAF client will send the key registration request to the UAF authenticator including the FCP, the KHAccessToken, and the username parameter.

Following the reception of a key registration request by a UAF authenticator, the later will first prompt the user for authentication, and, then, generate a new key pair (Uauth.pub, Uauth.priv), store it on its secure storage, and associate it with the received username and KHAccessToken. Subsequently, the UAF authenticator will create the key registration data (KRD) object containing the FCP, the newly generated user public key (Uauth.pub), and the authenticator's attestation ID (AAID), which is a unique identifier assigned to a model, class or batch of UAF authenticators, and it is used by the relying party to identify a UAF authenticator and attest its legitimacy. Once the KRD is generated, the UAF authenticator will sign it using its attestation private key and return to the UAF client a key registration reply (which the later forwards to the relying party) that encompasses: the signed KRD, the AAID, Uauth.pub, and its attestation certificate (Certattest). Upon the reception of the key registration reply by the relying party, the later cryptographically verifies the KRD object, uses the AAID to identify if the UAF authenticator is a legitimate authenticator with a valid (i.e., unrevoked) attestation certificate, and, finally, stores the Uauth.pub key in a database for the purposes of user authentication in any subsequent authentication requests.

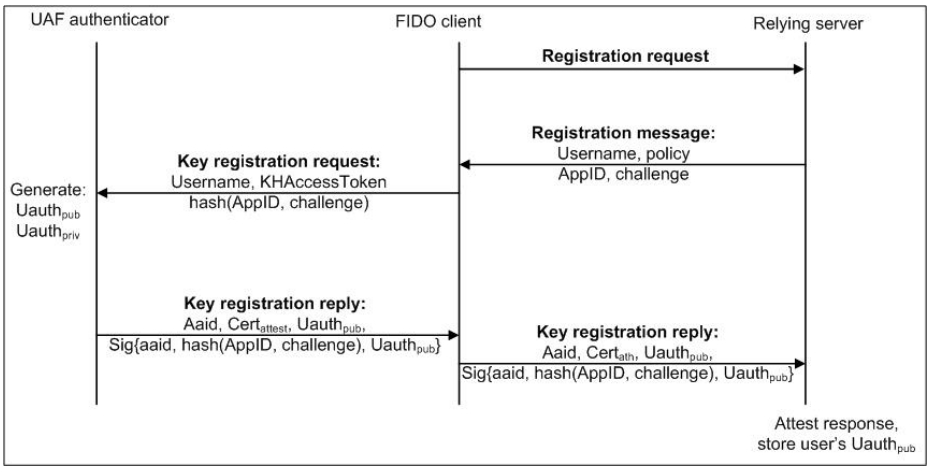

Fig. 2. the UAF registration operation

The UAF authentication operation. The authentication operation (illustrated in Figure 3) is initiated when a user requests a service that requires authentication to a relying party, either through a compatible application or through a browser (in a similar fashion with the registration operation outlined above). The relying party replies to the authentication request by transmitting an authentication message with the following parameters: the AppID, the authenticator policy, and a server generated challenge, to the UAF client residing in the user's device. The UAF client receiving the authentication request, first identifies the calling app (or website) and then determines (based on the AppID parameter) whether the associated application is trusted and allowed to proceed with the authentication request. Subsequently, the UAF client checks the authenticator policy parameter and sends a key authentication request to the set of UAF authenticator(s) mandatory by the policy. If the required UAF authenticators are not present in the user's device, then the authentication operation will be canceled. Using ASM, the UAF client performs the following operations: it first calls the UAF authenticator in order to compute the FCP, which is a hash of the AppID and the server challenge. Then, it retrieves the KHAccessToken, and finally, sends the key authentication request to the UAF authenticator(s) including the FCP and the KHAccessToken.

Following the reception of a key authentication request by a UAF authenticator, the later will first check if the UAF client is authorized to request an authentication for that particular user key, based on KHAccessToken. If the UAF client is authorized, then the UAF authenticator will prompt the user for authentication, and, then, retrieve the associated Uauth.priv from its secure key storage. Subsequently, the UAF authenticator will create the SignedData object containing the FCP, a newly generated nonce, and a Sign Counter (cntr). The cntr variable is a monotonically increasing counter, incremented on every sign request performed by the UAF authenticator for a particular user key pair. This value is then used by the relying party to detect cloned authenticators. Once the SignedData object is generated, the UAF authenticator will sign it using the Uauth.priv key and return to the UAF client a key authentication reply (which the later forwards

to the relying party) that encompasses: the signed object SignedData, the FCP, the nonce n, and the counter cntr. Finally, upon the reception of the key authentication reply by the relying party, the later first retrieves Uauth.pub from its database, cryptographically verifies the signedData object, and stores the value of the cntr counter. If the verification of the SignedData object succeeds, then the user is successfully authenticated.

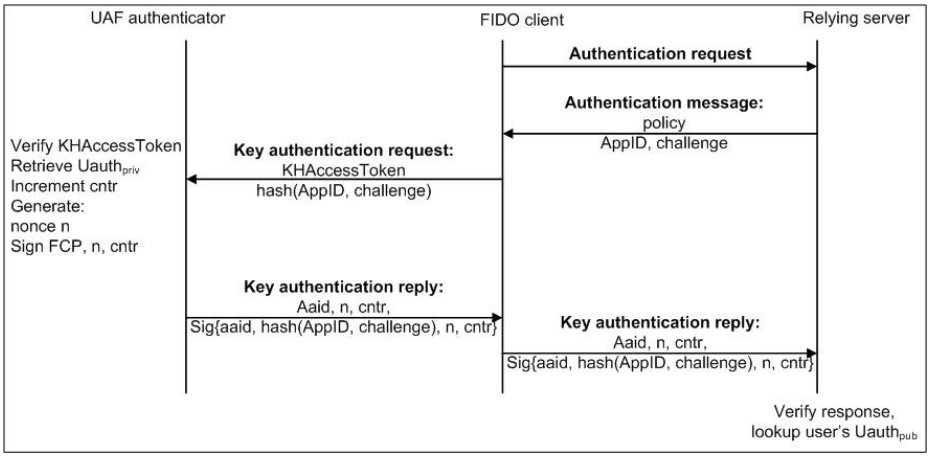

Fig. 3. the UAF authentication operation

2.2 Related Work

The literature includes some recent work that elaborates on the security of FIDO. In [22], the authors pinpoint and evaluate a set of trust requirements of FIDO protocols. Based on their analysis, the authors reach the conclusion that the FIDO solution does not solve the trust requirements of previous online identity management solutions (e.g., passwords) and instead it has shifted these requirements to other components in its architecture. In [23], three attacks are presented for FIDO UAF, but the authors do not elaborate extensively on the assumptions required to perform these attacks.

The FIDO security reference [9] outlines a list of assets that must be protected against malicious behavior and provides a limited set of security requirements with the goal of protecting these assets. We would like to point out that these requirements are optional and vendors receiving FIDO certification are not obliged to implement them. A variety of vendors such as Samsung, LG, Qualcomm, and Huawei [8] have already received FIDO certification, however, their implementations are proprietary, and, therefore, not open to 3rd party evaluation. Per FIDO specifications, the critical assets of the UAF protocol are the private key of the authentication key pair, the private key of the UAF authenticator attestation key pair, and the UAF authenticator attestation authority private key [7]. Furthermore, the UAF protocol specifications incorporate the following (optional) security requirements: the authentication keys must be securely stored within a UAF authenticator and thus protected against any misuse, users must authenticate themselves to the UAF authenticator before the authentication keys are

accessed, the UAF authenticators may support authenticator attestation using a shared attestation certificate, and a UAF authenticator may implement a secure display mechanism (also referred as transaction confirmation mechanism), which can be used by the UAF client for displaying transaction data to the user. Therefore, the UAF specifications do not incorporate any mechanisms that safeguard the cryptographic material stored in the UAF authenticators, or protect against attacks that may target the UAF client. Instead, the responsibility for the design and implementation of any security measures that protect these critical entities is passed on to the vendors.

One solution to address the security requirements of the UAF specifications and provide a secure operational environment for the UAF authenticators, is the incorporation of trusted computing platform technologies [10]. The trusted computing platform constitutes specialized hardware that provides a variety of services, such as secure input/output, device authentication, integrity measurement, sealed storage, remote attestation, cryptographic acceleration, protected execution, root of trust, and digital rights management. Two prevalent platforms for trusted computing currently exist [10], the Trusted Platform Module (TPM) [11], which is based on the specifications created by the Trusted Computing Group, and the TrustZone (TZ) platform [12], created by the ARM corporation. The TPM is a co-processor, which provides basic cryptographic capabilities like random number generation, hashing, protected storage of sensitive data (e.g. secret keys), asymmetric encryption, as well as generation of signatures. The TPM platform presents some significant limitations [10]: (i) the need for a separate module increases the cost of a device; (ii) it cannot be deployed on legacy devices; (iii) it does not protect against runtime attacks; (iv) it relies on the assumption that a TPM cannot be tampered; (v) the physical size and energy consumption requirements make it an unsuitable solution for mobile and embedded devices; (vi) in case of a TPM compromise, the hardware module must be physically replaced; and (vii) the supported cryptographic algorithms have been found to pose security concerns (i.e., SHA-1), and are not well suited for resource restricted devices (i.e., RSA).

The TrustZone platform, is part of ARM's processor cores and system on chip (SoC) reference architecture. The associated hardware is part of the SoC silicon, and thus, it does not require any additional hardware. The primary objective of TrustZone is to establish a hardware-enforced security environment providing code isolation, that is, a clear separation between trusted software, which is granted access to sensitive data like secret keys, and other parts of the embedded software. To achieve this, the TrustZone platform provides two virtual processing cores with different privileges and a strictly controlled communication interface, enabling the creation of two distinct execution environments, encapsulated by hardware. Nevertheless, to the best of our knowledge, Samsung is the only certified vendor that implements a UAF authenticator using the TrustZone platform [13]. Furthermore, this approach only protects the UAF authenticator, while the UAF client is still susceptible to a variety of attacks (analyzed in detail in section 3.2). Finally, extensive literature has shown that the TrustZone platform itself is not immune to weakness and vulnerabilities [14][15][18][19].

3 UAF security analysis

In the following section, we provide an informal security analysis in which we manually identify several attack vectors that can beset to compromise the legitimate operation of the UAF protocol, including the ability of an attacker to: (i) gain unprivileged access to the cryptographic material stored within the UAF authenticator, and (ii) highjack either the UAF authenticator or the UAF client. Furthermore, we investigate and identify how an attacker can circumvent the security measures provided by the UAF protocol, including the authenticator attestation mechanism, the transaction confirmation mechanism, the trusted facet list, and the sign counter. Finally, we provide a threat analysis and investigate the impairment that may be caused by an attacker in the event of a successful exploitation of the UAF protocol.

3.1 UAF protocol vulnerabilities and limitations

As we previously analyzed in section 2.1, two UAF protocol entities, namely the UAF authenticator and the UAF client, reside at the client's device. These entities are responsible for most of UAF's critical functionality, including the authentication of users, the creation and maintenance of the cryptographic material used for either the attestation of the UAF authenticator or the authentication to a relying party, the presentation of UAF related information to the user, and the initiation and management of both the registration and authentication procedures. Therefore. if either of these entities is compromised, an attacker would be able to launch several critical attacks, compromising the authenticity, privacy, availability, and integrity of the UAF protocol. Subsequently, in sections 3.1.1 and 3.1.2, we identify several vulnerabilities present in the specifications of the UAF authenticator and the UAF client, respectively.

UAF authenticator vulnerabilities. The first and most apparent attack vector of the UAF protocol is the authentication keys. Therefore, an attacker may attempt to (directly or indirectly) gain unprivileged access to these keys. As we previously mentioned in section 2.1.1, the responsibility of storing the authentication keys lies with the UAF authenticator and based on the UAF protocol security requirements, the UAF authenticator utilises some form of secure/privileged storage. However, it has been shown in the literature that such types of key storage solutions can still be compromised [16]. As we mentioned in section 2.2, UAF authenticators typically rely on trusted computing platforms for the storage of cryptographic material. Cooijmans et al [15] have shown that on several widely adopted trusted computing platforms, an attacker with privileged rights can gain the ability of using encrypted credentials by moving them to a different directory, which designates a malicious application as the owner of the credentials. Finally, an attacker may also attempt to indirectly gain access to the authentication keys, by fully compromising the UAF authenticator(s). Based on the literature, an attacker can gain full access to a trusted computing platform by performing an integrated circuit attack (i.e., ICA) [14]. One limitation of this attack is the requirement to have physical access to the user's device. However, once the attack is performed, the attacker can then create a cloned UAF authenticator, alleviating any further need for the original user's device.

When utilizing a cloned UAF authenticator, an attacker must then evade the security mechanisms of the UAF protocol, implemented on the purpose of identifying such malicious behavior. Recall from section 2.1 that the UAF protocol incorporates two security mechanisms that safeguard the operation of the UAF authenticator: (i) an attestation mechanism, in which the UAF authenticator must prove its legitimacy by providing an attestation signature during the registration process and (ii) a sign counter (cntr) mechanism, which is a monotonically increasing counter, incremented on every sign request performed by the UAF authenticator for a particular user key pair and used by the relying party to detect cloned UAF authenticators.

Regarding the attestation mechanism, we have identified three approaches that can be used by an attacker to circumvent detection. In the first method, an attacker may utilize the extracted attestation key from the compromised UAF authenticator and perform registration requests to relying parties, impersonating the legitimate user. Since the attestation keys for each UAF authenticator are not unique (i.e., a group of UAF authenticators share the same attestation key pair), the malicious behavior cannot be easily detected by the relying party. If, however, the attestation keys are revoked by the device's vendor, then there is a risk of detection by the relying party. A second method that can be used by an attacker when employing a cloned authenticator is to avoid the attestation mechanism all together. This can be achieved by exploiting a limitation in the attestation process. Recall from section 2.1 that the attestation process takes place only during the registration operation. Therefore, an attacker may allow the legitimate UAF authenticator to perform the registration process, and, subsequently, without the users' knowledge, use the cloned authenticator to authenticate itself to the relying party, masquerading as the legitimate user. Finally, an attacker may use the cloned UAF authenticator temporarily to collect personal information related to the legitimate user, and, then, register at other relying parties using a different, non-cloned UAF authenticator. Subsequently, since the attestation procedure takes place at a non-cloned authenticator, there is no risk of revocation, while the attacker retains the ability to impersonate the legitimate user to any relying party.

On the other hand, the second security measure proposed by the UAF specifications (i.e., sign counter), can be circumvented by an attacker, if the later actively attempts to perform an authentication operation immediately after the completion of cloning a UAF authenticator. Recall from section 2.1 that during the authentication operation, a relying party will assume a UAF authenticator is legitimate if the sign counter encapsulated in the key authentication reply is equal to the sign counter maintained by the relying party incremented by one. Therefore, a race condition evolves between the legitimate and the cloned UAF authenticator, since only the UAF authenticator that manages to perform an authentication request first, will be considered legitimate by the relying party (while the second authenticator will attempt to authenticate using an older value of the sign counter). Thus, an attacker can circumvent this security measure by performing an authentication request to the relying party as soon as the UAF authenticator is cloned, maximizing his chances of winning the race condition.

UAF client vulnerabilities. The second critical entity of the UAF protocol that resides at a user's device is the UAF client. Recall from section 2 that the UAF client acts as an intermediator between the relying party on one hand and the UAF authenticator

on the other and it is responsible for most of UAF's protocol operations, short of generating the encryption keys or performing cryptographic operations. Furthermore, the UAF client is implemented entirely in software, making it an ideal candidate for software attacks. Even more importantly, the UAF protocol does not incorporate any security measures that safeguard the UAF client from attacks or verifies that a user's device operates a legitimate version of the client. The UAF protocol specifications propose the execution of the UAF client in a "privileged" environment, however, since the client is typically embedded within a browser either fully or as a plug-in, it is de-facto implemented as a normal application.

The simplest method of delivering a malicious UAF client to a user's device is by deceiving the user to install the application voluntarily. Common delivery methods include attachments in e-mails or browsing a malicious website that installs software after the user clicks on a pop-up. Other methods of compromising a UAF client is through malicious software residing at the user's device (such as a virus, worm. trojan, or root kit) or by exploiting an operating system vulnerability. The latter, enables the execution of a plethora of attacks such as spoofing of inter-process communication, privilege escalation, return-oriented programming, or code injection attacks. For example, in a variety of sources such as [17][20][21], the authors demonstrate methodologies for accomplishing privilege escalation in the android operating system, one of the most widely used platforms, which includes a variety of privilege protection mechanisms, such as application specific sandboxing and Mandatory Access Control (MAC) policies. Furthermore, in the most recent versions of android, privilege escalation is typically achieved using system less root [20], which is the process of gaining escalated privileges without any modification to the system partition, thus evading detection by any security mechanisms that validate an operation system through a checksum of its system partition (i.e., a common security mechanism used by most of the trusted computing platforms).

3.2 Threat analysis

In the following section, we provide a threat analysis based on the vulnerabilities identified in section 3.1. First, we outline the assets that reside at the client side, which are critical for the legitimate operation of the UAF protocol. In this list, we also include assets that are not part of the UAF protocol, such as, the underlying operating system and the utilization of a trusted computing platform, since (as we have shown in section 3.1), they are detrimental to the security of the UAF protocol. We then investigate the consequences that may be caused in the event of a successful exploitation of the vulnerabilities identified in section 3.1. Table 1 provides a summary of the critical assets related to the UAF protocols' secure operation, the threats identified in the threat analysis, and, the consequences induced by the threats, if the later are carried out successfully by an attacker.

Critical assets related to the UAF protocols' secure operation. As we mentioned in section 2.2, the UAF specifications [9] provide a limited list of assets that must be protected in an implementation of the UAF protocol. These assets include the private key of the authentication key pair, the private key of the UAF authenticator attestation

key pair, and the UAF authenticator attestation authority private key. However, as we have seen in section 3.1, an attacker may also target several other assets that are either part of the UAF protocol, or they are integral in its secure operation. In particular, an attacker may either target the UAF authenticator(s) or the UAF client that are present in a legitimate users' device. Furthermore, an attacker may indirectly compromise the secure operation of the UAF protocol by exploiting existing vulnerabilities (i) at the underlying operating system in which the UAF protocol is executed, or (ii) at the trusted computing platform (typically the TrustZone platform), used for the hardware-assisted protection of the encryption keys and the operation of the UAF authenticator(s).

Threat evaluation. Based on the security analysis in section 3.1, the private keys stored in the UAF authenticator, namely the attestation private key and the authentication private keys pose a critical attack vector of the UAF protocol. Recall from section 2.2 that these keys are used by the UAF authenticator to sign registration and authentication replies, respectively. On the other hand, the relying party uses these signed replies to authenticate the UAF authenticator and verify its legitimacy. Therefore, if an attacker compromises the attestation private key, he would then be capable of impersonating the legitimate user by registering to other relying parties on the users' behalf, without the latter's consent (including fraudulent relying parties). In order to have access to the authentication keys associated with the malicious registrations and to avoid detection by the user, the attacker will have to import the attestation private key to a cloned and silent authenticator, i.e., an authenticator that appears to have been manufactured by the same vendor as the legitimate one and does not prompt the user for any action during the registration and authentication operations of the UAF protocol. On the other hand, if the attacker compromises one or more authentication private keys, he would then be capable of impersonating the legitimate user by authenticating as the user to relying parties. The attacker is limited, however, to relying parties that the legitimate user has already registered. Nevertheless, once authenticated, the attacker can then collect personal data related to the legitimate user and stored at the relying party, as well as perform transactions with the relying party without the users' consent.

An attacker may also attempt to indirectly gain access to the attestation and authentication keys, by fully compromising the UAF authenticator(s) residing at the device of a legitimate user. This can be accomplished in the following ways: the user unwillingly installs a malicious authenticator to his/her device, the attacker compromises the UAF authenticator by targeting the UAF authenticators' underlying trusted computing platform, and, the attacker gains physical access to the device and either installs a malicious authenticator, or tampers with the legitimate UAF authenticator(s) installed on the device. As a result, any subsequent registration and authentication requests will be captured by the malicious authenticator, enabling the attacker to impersonate the legitimate user, collect personal data, and perform transactions on the users' behalf, similarly to the cloned authenticator threat we analyzed previously. Furthermore, the attacker can also extract the attestation and authentication keys, in order to create a cloned authenticator that resides outside the device of the user.

The UAF client signifies another critical attack vector identified in the security evaluation. An attacker may attempt to compromise the UAF client by exploiting one or more of the following vulnerabilities: gaining physical access to the user's device and

manually installing a malicious client, deceiving the user to install the malicious client voluntarily, using other malicious software residing at the user's device (such as a virus, worm. trojan, or root kit) in order to install the malicious client, or by exploiting an operating system vulnerability. Having successfully compromised the UAF client, an attacker is then capable of launching several additional attacks against the UAF protocol, such as: allowing itself or other malicious applications to perform registration/authentication operations without the user's consent, enforce the use of the weakest/less secure UAF authenticator during a legitimate registration process, direct a user to a fake or malicious relying party, and defeat the user consent, transaction confirmation, and trusted facet list security measures of the UAF protocol. Recalling from section 2.1.1, during the registration operation, the UAF client is responsible for initiating registration requests, determining if applications (or websites) are authorized to use the UAF protocol, present a UI to the user, and directing the relying party challenge to the UAF authenticator based on the authenticator policy transmitted by the relying party (i.e., based on the trusted facet list). Since the UAF client is the only entity responsible for assessing the trusted facet list, it can allow the registration operation for any website, or from any application, regardless of what is enforced by the trusted facet list security measure. Therefore, the user may unwillingly be redirected to a malicious relying party masqueraded as a legitimate one, so that personal/valuable information can be phished by an attacker. Furthermore, as we mentioned previously, it is the UAF client's responsibility for presenting a UI to the user, and, therefore, even if the user's device incorporates a transaction confirmation security mechanism, the confirmation will always be true, since the mechanism validates if the information provided to the user is tampered/modified/spoofed after leaving the UAF client, and not if the later modified the displayed content. Finally, a malicious UAF client may forward a relying party challenge to the weakest UAF authenticator (preferably one with a low entropy secret). Subsequently, during authentication, the attacker could attempt to discover the secret and access the user's account without the legitimate users' consent.

Table 1. threats related to the UAF protocol and their associated consequences

Asset	Threat	Consequences
Attestation private key	Attacker gains access to the attestation keys	Create a fake authenticator
Authentication private key	Attacker gains access to the authentication keys	Attempt to obtain user data from the relying party by guessing the counter
UAF authenticator	User installs a malicious authenticator	Impersonate user, capture user data, register the user to a fraudulent relying party
TrustZone, UAF authenticator	Attacker compromises the trusted computing platform	Create cloned authenticator, impersonate user, compromise the UAF authenticator
UAF client, UAF authenticator, TrustZone	Attacker gains physical access to a user's device	Create cloned authenticator, impersonate user, compromise the UAF authenticator, install malicious UAF client
UAF authenticator	Attacker employs a cloned authenticator	Impersonate user, capture user data, register the user to a fraudulent relying party

UAF client	User installs a malicious client	Register to a fraudulent relying party, phishing – lead to malicious websites, downgrade authentication policy, capture user data, circumvent transaction confirmation security mechanism, allow malicious apps to register/impersonate the user
Operating system	Attacker can execute privileged code at the user's device	Compromise the UAF client

4 Conclusions

The UAF protocol provides several important advantages over traditional authentication mechanisms, such as strong authentication and a simplified registration and authentication procedure. However, the UAF protocol also transfers user authentication operations from the server-side to the client-side. Therefore, the critical functionality of the UAF protocol typically operates in a consumer platform such as a mobile device, which is susceptible to a variety of attacks such as malware and viruses, its users deploy unsupervised software, and the deployed operating systems may be susceptible to several vulnerabilities. In this paper, we have provided a comprehensive security analysis of the UAF protocol and have identified several vulnerabilities that may be exploited by an attacker in order to compromise the authenticity, privacy, availability, and integrity of the UAF protocol. More specifically, we have investigated methods of attacking the two entities of the UAF protocol residing at a user's device, namely, the UAF authenticator and the UAF client, including the ability of an attacker to gain unprivileged access to the cryptographic material stored within the UAF authenticator and highjack either the of these two entities. Furthermore, we have investigated and identified how an attacker can circumvent the security measures provided by the UAF protocol, including the authenticator attestation mechanism, the transaction confirmation mechanism, the trusted facet list, and the sign counter. Finally, we provided a threat analysis in which we analyze the impairment that may be caused by an attacker in the event of a successful exploitation of the UAF protocol. Based on our threat analysis, by exploiting the identified vulnerabilities, an attacker would be able to capture the data exchanged between a user and an online service, impersonate a user at any UAF compatible online service, impersonate online services to the user, perform transactions on the users' behalf without the latter's consent, present fake information to the user's screen during a transaction, re-direct the user to a fraudulent relying party during registration, force the use of a weak or malicious UAF authenticator, allow malicious applications to register to a legitimate relying party, and access private keys used for authentication of a user.

Acknowledgments. This research has been funded by the European Commission in part of the ReCRED project (Horizon H2020 Framework Programme of the European Union under GA number 653417).

References

1. Das, Anupam, et al. "The Tangled Web of Password Reuse." NDSS. Vol. 14. 2014
2. 55K Twitter Passwords Leaked. http://www.newser.com/story/145750/55k-twitter-passwords-leaked.html
3. Yahoo hacked, 450,000 passwords posted online. http://www.cnn.com/2012/07/12/tech/web/yahoo-users-hacked
4. 6.46 million LinkedIn passwords leaked online. http://www.zdnet.com/blog/btl/6-46-million-linkedin-passwords-leaked-online/79290.
5. FIDO Alliance. Fido security reference. http://www.fidoalliance.org/specifications
6. Srinivas, Sampath, et al. "Universal 2nd factor (U2F) overview." FIDO Alliance Proposed Standard (2015): 1-5
7. F.I.D.O. Alliance, "FIDO UAF Protocol Specification v1.1: FIDO Alliance Proposed Standard." (2016)
8. F.I.D.O. Alliance, “FIDO Certified Products”, https://fidoalliance.org/certification/fido-certified-products/ , last accessed June 5, 2017
9. FIDO Alliance. Fido security reference (2014). www.fidoalliance.org/specifications
10. Panos, Christoforos, et al. "A specification-based intrusion detection engine for infrastructure-less networks." Computer Communications 54 (2014): 67-83
11. Trusted Computing Platform Alliance. TCPA main specification v. 1.2. http://www.trustedcomputing.org
12. Winter, Johannes. "Trusted computing building blocks for embedded linux-based ARM trustzone platforms." Proceedings of the 3rd ACM workshop on Scalable trusted computing. ACM, 2008
13. Common Criteria for Information Technology Security Evaluation, “SAMSUNG SDS FIDO Server Solution V1.1 Certification Report,” (2016)
14. C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert, “Breaking and entering through the silicon,” in Computer and Communications Security (CCS), pp. 733–744, 2013
15. Cooijmans, Tim, Joeri de Ruiter, and Erik Poll. "Analysis of secure key storage solutions on Android." Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices. ACM, 2014
16. Cooijmans, Tim, et al. "Secure key storage and secure computation in Android." Master's thesis, Radboud University Nijmegen (2014)
17. Davi, Lucas, et al. "Privilege escalation attacks on android." International Conference on Information Security. Springer Berlin Heidelberg, 2010
18. Shen, Di. "Exploiting Trustzone on Android." Black Hat USA (2015)
19. Rosenberg, Dan. "Qsee trustzone kernel integer over flow vulnerability." Black Hat conference. 2014
20. Abhishek, P. C. "Student Research Abstract: Analysing the Vulnerability Exploitation in Android with the device-mapper-verity (dm-verity)." (2017)
21. Does, Thom, and Mike Maarse. "Subverting Android 6.0 fingerprint authentication." (2016)
22. Ijlal Loutfi and Audun Jøsang, "FIDO Trust Requirements", 20th Nordic Conference, Stockholm, Sweden, 2015
23. Kexin Hu, Zhenfeng Zhang, "Security analysis of an attractive online authentication standard: FIDO UAF protocol", IEEE China Communications, Vol. 13, No. 12, 2016.