FIDO U2F Implementation Considerations

Specification Set: fido-u2f-v1.0-rd-20140209 REVIEW DRAFT

Editors:

Dirk Balfanz (balfanz@google.com)

Contributors:

Abstract:

This document lists a number of considerations for U2F implementers.

Status:

This Specification has been prepared by FIDO Alliance, Inc. **This is a Review Draft Specification and is not intended to be a basis for any implementations as the Specification may change**. Permission is hereby granted to use the Specification solely for the purpose of reviewing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this Specification for other uses must contact the FIDO Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this Specification may require licenses under third party intellectual property rights, including without limitation, patent rights. The FIDO Alliance, Inc. and its Members and any other contributors to the Specification are not, and shall not be held, responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

THIS FIDO ALLIANCE SPECIFICATION IS PROVIDED "AS IS" AND WITHOUT ANY WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTY OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1 Notation

Type names, attribute names and element names are written in *italics*.

String literals are enclosed in “”, e.g. “UAF-TLV”.

In formulas we use “|” to denote byte wise concatenation operations.

U2F specific terminology used in this document is defined in [FIDOGlossary]

1.1 Key Words

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

2 Implementation Considerations

Note: Reading the 'FIDO U2F Overview' [U2FOverview] is recommended as a background for this document.

2.1 Timing Considerations

U2F Tokens should respond to authentication and registration request as soon as possible to ensure a responsive user interface. In particular, they *should not* wait for user presence if the request message requires it. Usually, this means that U2F tokens should respond within 500ms to requests. (FIDO clients, on the other hand, should be coded more defensively, and should wait for at least 3 seconds before giving up on a U2F token.)

Once user presence is detected, U2F tokens should persist the “user presence” state for 10 seconds or until an operation which requires user presence is performed, whichever comes first.

2.2 Generation of Key Handles

U2F tokens might not store private key material, and instead might export a wrapped private key as part of the key handle. If a U2F token chooses to do this, then the following must be taken into consideration:

- The U2F token should employ a cipher that offers the best possible security on the given hardware. Sometimes, hardware offers better protections against certain attacks for “weak” ciphers (e.g., 3DES) than against “strong” ciphers (e.g., AES). Implementers should carefully weigh the pros and cons of different ciphers on the hardware platform that they’re implementing on.
- Given a particular U2F token and a relying party, the relying party should not be able to tell the difference between a key handle that was issued for a different token, and a key handle that was issued for a different relying party. (The concern is that a site, evil.com, might want to find out whether a given token has been registered for a site embarrassing.com, and would be able to do so if it had key handles from embarrassing.com if it could tell the difference.) The two error conditions (“wrong key handle” and “wrong origin (but correct key handle)”) should not be distinguishable to the relying party, through careful timings or otherwise.

2.3 Secure Key Generation

U2F tokens should follow best practices when generating private keys (i.e., use a recommended PRNG) and use a good source of entropy (which usually serves as input to

the PRNG). If no good source of entropy is available on the token, the token should combine whatever entropy there is with the challenge parameter from the request as input into the PRNG.

2.4 Challenge Parameters

The registration and authentication operations require the relying party to pass a challenge parameter to the Javascript API (as part of the SignData and EnrollData parameters – see FIDO U2F Javascript API [U2FJSAPI]). This parameter is the base64-encoding of a byte array chosen by the relying party.

Relying parties should ensure that the challenge parameter has sufficient entropy. In particular, it is recommended that the challenge parameter contains at least 8 random bytes, following the requirements in [SP800-63-1].

2.5 Revocation of Tokens

Since U2F tokens don't have device identifiers, U2F does not prescribe a way to revoke tokens (through a revocation list or similar mechanism). Instead, it is up to individual relying parties to stop honoring authentication responses that come from certain tokens.

Relying parties should give users a mechanism to report lost or stolen tokens. If the token is lost or stolen, then the relying party should stop honoring authentication responses from the token.

2.6 Token Counters

A U2F token must increase a counter each time it performs an authentication operation. This counter may be "global" (i.e., the same counter is incremented regardless of the application parameter in Authentication Request message), or per-application (i.e., one counter for each value of application parameter in the Authentication Request message).

U2F token counters should start at 0, and wrap around to 0 when they have reached their maximum value.

The counter allows relying parties to detect token cloning in certain situations. Relying parties should implement their own remediation strategies if they suspect token cloning due to non-increasing counter values (other than wrap-around).

2.7 Key Usage

Keys generated during a U2F registration must not be used for any purpose other than U2F authentications. Implementers of hardware and/or software that serves other purposes beyond U2F need to ensure that U2F keys are not used for such other purposes.

2.8 UI Considerations for FIDO Clients

FIDO Clients should implement a user interface that allows the user to get a clear indication of which relying parties are using the FIDO U2F APIs. Such APIs allow relying parties that are in possession of the user's public key to confirm the identity of the user, even if the user has removed their cookies, is using anonymizing onion routing networks, etc. In the case where the FIDO Client is a web browser, the web browser should indicate to the user which page or web origin is creating or exercising U2F keys for the user. The FIDO client might also give the user the ability to allow or deny the use of the U2F APIs for relying parties.

Bibliography

FIDO Alliance Documents:

[FIDOGlossary] Rolf Lindemann, Davit Baghdasaryan, Brad Hill, John Kemp. FIDO Technical Glossary. Version v1.0-rd-20140209, FIDO Alliance, February 2014. See http://fidoalliance.org/specs/fido-glossary-v1.0-rd-20140209.pdf

[**U2FJSAPI**] Dirk Balfanz. FIDO U2F Javascript API. Version v1.0-rd-20140209, FIDO Alliance, February 2014. See http://fidoalliance.org/specs/fido-u2f-javascript-api-v1.0-rd-20140209.pdf

[**U2FOverview**] Sampath Srinivas, Dirk Balfanz, Eric Tiffany. FIDO Universal 2nd Factor (U2F) Overview. Version v1.0-rd-20140209, FIDO Alliance, February 2014. See http://fidoalliance.org/specs/fido-u2f-overview-v1.0-rd-20140209.pdf

Other References:

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels (RFC2119), S. Bradner, March 1997

[SP800-63-1] NIST Electronic Authentication Guideline SP 800-63-1 (NIST SP 800-63-1). W. Burr et al, National Institute of Standards and Technology, December 2011