Assurance Activities Report for Wickr Enterprise Client 6.10

Version 1.0
5 April 2023

Evaluated By:

Leidos Inc.
https://www.leidos.com/civil/commercial-cyber/product-compliance
Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive
Columbia, MD 21046

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

The Developer of the TOE:
Wickr Inc.
W 31st Street
New York, NY 10001

The TOE Evaluation was Sponsored by:
Wickr Inc.
W 31st Street
New York, NY 10001

Evaluation Personnel:
Allen Sant
Anthony Apted
Armin Najafabadi
Greg Beaver
Josh J. Marciante
Pascal Patin

Common Criteria Version:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, Version 3.1, Revision 5, April 2017.
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017.

Common Evaluation Methodology Version:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017.

Protection Profiles:

- Protection Profile for Application Software, Version 1.4, 7 October 2021

Revision History

Version	Date	Description
0.1	21 October 2022	Initial draft
1.0	5 April 2023	Final version

Contents

LIST OF TABLES

1. Introduction

This document presents the results of performing assurance activities associated with the Wickr Enterprise Client 6.10 evaluation. This report contains sections documenting the performance of evaluation activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in the following document:

- *Protection Profile for Application Software*, Version 1.4, 7 October 2021 [PP_APP_v1.4]

Note that, in accordance with NIAP Policy Letter #5, all cryptography in the TOE for which NIST provides validation testing of FIPS-approved and NIST-recommended cryptographic algorithms and their individual components must be NIST validated. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity. As such, Test assurance activities associated with functional requirements within the scope of Policy Letter #5 are performed by verification of the relevant CAVP certification and not through performance of any testing as specified in the claimed PP documents.

1.1 Technical Decisions

This subsection lists the Technical Decisions that have been issued by NIAP against [PP_APP_v1.4], along with rationale as to their applicability or otherwise to this evaluation.

TD0624 – Addition of DataStore for Storing and Setting Configuration Options

This TD has been applied to this evaluation.

TD0628 – Addition of Container Image to Package Format

This TD has been applied to this evaluation.

TD0650 – Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4

N/A – the TOE does not claim VPN client functionality.

TD0655 – Mutual authentication in FTP_DIT_EXT.1 for SW App

This TD has been applied to this evaluation.

TD0664 – Testing activity for FPT_TUD_EXT.2.2

This TD has been applied to this evaluation.

TD0669 – FIA_X509_EXT.1 Test 4 Interpretation

This TD has been applied to this evaluation.

TD0709 – Number of elements for iterations of FCS_HTTPS_EXT.1

N/A – the TOE does not claim this SFR.

TD0717 – Format changes for PP_APP_V1.4

This TD has been applied to this evaluation.

TD0719 – ECD for PP APP V1.3 and 1.4

Not applicable; this TD updates the App PP to include a formal ECD which is needed for the PP itself to conform to CC Part 3. This does not change the ST or how the evaluation of the TOE is conducted..

1.2 References

[ST]	Wickr Enterprise Client Version 6.10Security Target, Version 1.0, 07 March 2023
[CCECG]	Wickr Enterprise Client Common Criteria Evaluated Configuration Guide (CCECG), Version 1.0, 6 March, 2023.
[Install]	Wickr Enterprise NIAP Version Installation and Maintenance, Version 1.30.0
[Admin]	Wickr Enterprise Administrator Guide, Version 426151b
[DUG]	Wickr Enterprise Desktop User Guide, Version 6.10
[Test]	Wickr Enterprise Client Version 6.10 Common Criteria Test Report and Procedures, Version 1.0, March 10, 2023

1.3 SAR Evaluation

The following Security Assurance Requirements (SARs) were evaluated during the evaluation of the TOE:

SAR	Verdict
ASE_CCL.1	Pass
ASE_ECD.1	Pass
ASE_INT.1	Pass
ASE_OBJ.2	Pass
ASE_REQ.2	Pass
ASE_TSS.1	Pass
ADV_FSP.1	Pass
AGD_OPE.1	Pass
AGD_PRE.1	Pass
ALC_CMC.1	Pass
ALC_CMS.1	Pass
ALC_TSU_EXT.1	Pass
ATE_IND.1	Pass
AVA_VAN.1	Pass

The evaluation work units are listed in the proprietary ETR. The evaluators note per the PP evaluation activities that many of the SARs were successfully evaluated through completion of the associated evaluation activities present in the claimed PP.

2. Security Functional Requirement Evaluation Activities

This section describes the evaluation activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The evaluation activities are derived from [PP_APP_v1.4]. NIAP Technical Decisions have been applied and are identified as appropriate.

2.1 Cryptographic Support (FCS)

2.1.1 Certificate Table

The TOE implements its own cryptography (Wickr OpenSSL version 2.0.16) for use in credential storage protection. Credential storage is accomplished through a combination of platform cryptography and TSF cryptography. The following table lists the cryptographic functions and specific algorithms implemented by the TOE, together with the associated SFRs and relevant ACVP validation number.

Function	Standard	Certificate
FCS_COP.1/SKC: Cryptographic Operation – Encryption/Decryption		
AES-GCM (256 bits)	NIST SP 800-38	#A3372
FCS_CKM.1/SK: Cryptographic Symmetric Key Generation FCS_RBG_EXT.2: Random Bit Generation from Application		
CTR_DRBG (AES)	NIST SP 800-90A	#A3372

The TOE invokes platform-provided cryptography to secure data in transit. All cryptographic services used for data in transit protection are implemented by the platform cryptographic library for the respective platforms. The TOE relies on its underlying platform to satisfy the following SFR:

- FTP_DIT_EXT.1: Protection of data in transit—the TOE invokes platform-provided functionality to encrypt all transmitted data with TLS. The platform-provided TLS implementation uses the following cipher suites:
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.

Additionally, the TOE uses a combination of platform-provided functionality and its own functionality to securely store credential data at rest (FCS_STO_EXT.1). All platform versions of the TOE use platform-provided AES to encrypt a SQL database: iOS uses Coredata, which uses Apple CoreCrypto to encrypt the database using AES-256-GCM; all other platforms use SQLCipher, which calls the respective platform cryptographic libraries to encrypt the database using AES-256-CBC. The DEK that encrypts the database is generated randomly. The DEK is encrypted by a KEK, which is not persistently stored, using AES-256-GCM. The generation of the DEK/KEK and encryption of the KEK are performed by the TSF

The following table lists, for each supported platform, the cryptographic functions and specific algorithms implemented by the TOE platform that the TOE relies on in order to satisfy FTP_DIT_EXT.1 and FCS_STO_EXT.1, together with the relevant ACVP validation numbers.

Function	Algorithm	Standard	Certificate
Windows (Windows 10 SymCrypt)			

Function	Algorithm	Standard	Certificate
Asymmetric key generation	ECC key generation (P-384, P-521)	FIPS PUB 186-4	#A2677
Key establishment	ECC based key establishment	NIST SP 800-56A	#A2677
Cryptographic hash	SHA-256, SHA-384	FIPS PUB 180-4	#A2677
Keyed-hash message authentication	HMAC-SHA-256, HMAC-SHA-384	FIPS PUB 198-1, FIPS PUB 180-4	#A2677
Digital signature generation and verification	RSA (2048-bit or greater) ECDSA (P-256, P-384)	FIPS PUB 186-4, Section 4; FIPS PUB 186-4, Section 5	#A2677
Symmetric encryption	AES-CBC (256 bits), AES-GCM (128 bits, 256 bits)	NIST SP 800-38	#A2677
Deterministic random bit generation	CTR_DRBG(AES)	NIST SP 800-90A	#A2677
macOS (Apple coreCrypto Module [User])			
Asymmetric key generation	ECC key generation (P-384, P-521)	FIPS PUB 186-4	#A2820
Key establishment	ECC based key establishment	NIST SP 800-56A	#A2820
Cryptographic hash	SHA-256, SHA-384	FIPS PUB 180-4	#A2820
Keyed-hash message authentication	HMAC-SHA-256, HMAC-SHA-384	FIPS PUB 198-1, FIPS PUB 180-4	#A2820
Digital signature generation and verification	RSA (2048-bit or greater) ECDSA (P-256, P-384)	FIPS PUB 186-4, Section 4; FIPS PUB 186-4, Section 5	#A2820
Symmetric encryption	AES-CBC (256 bits), AES-GCM (128 bits, 256 bits)	NIST SP 800-38	#A2820
Deterministic random bit generation	CTR_DRBG(AES)	NIST SP 800-90A	#A2820
iOS (Apple coreCrypto Module [User])			
Asymmetric key generation	ECC key generation (P-384, P-521)	FIPS PUB 186-4	#A2788
Key establishment	ECC based key establishment	NIST SP 800-56A	#A2786
Cryptographic hash	SHA-256	FIPS PUB 180-4	#A2789
	SHA-384		#A2788
Keyed-hash message authentication	HMAC-SHA-256	FIPS PUB 198-1 FIPS PUB 180-4	#A2789
	HMAC-SHA-384		#A2788
	RSA (2048-bit or greater)	FIPS PUB 186-4, Section 4	#A2788

Function	Algorithm	Standard	Certificate
Digital signature generation and verification	ECDSA (P-256, P-384)	FIPS PUB 186-4, Section 5	#A2788
Symmetric encryption	AES-GCM (128 bits, 256 bits)	NIST SP 800-38	#A2787
Deterministic random bit generation	CTR_DRBG(AES)	NIST SP 800-90A	#A2787
Android (Samsung BoringSSL Android)			
Asymmetric key generation	ECC key generation (P-384, P-521)	FIPS PUB 186-4	#A2351
Key establishment	ECC based key establishment	NIST SP 800-56A	#A2351
Cryptographic hash	SHA-256, SHA-384	FIPS PUB 180-4	#A2351
Keyed-hash message authentication	HMAC-SHA-256, HMAC-SHA-384	FIPS PUB 198-1, FIPS PUB 180-4	#A2351
Digital signature generation and verification	RSA (2048-bit or greater) ECDSA (P-256, P-384)	FIPS PUB 186-4, Section 4; FIPS PUB 186-4, Section 5	#A2351
Symmetric encryption	AES-CBC (256 bits), AES-GCM (128 bits, 256 bits)	NIST SP 800-38	#A2351
Deterministic random bit generation	CTR_DRBG(AES)	NIST SP 800-90A	#A2351

2.1.2 Cryptographic Asymmetric Key Generation (FCS_CKM_EXT.1)

Modified per TD0717.

2.1.2.1 TSS Evaluation Activity

The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the "***generate no asymmetric cryptographic keys***" selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.

The evaluator inspected the application and its documentation and determined the TOE does not need asymmetric key generation services. The TOE invokes platform-provided cryptography to secure data in transit. As such, the evaluator verified the ST selects "generate no asymmetric cryptographic keys" in FCS_CKM_EXT.1.

2.1.2.2 Guidance Evaluation Activity

None.

2.1.2.3 Test Evaluation Activity

None.

2.1.3 Cryptographic Symmetric Key Generation (FCS_CKM.1/SK)

2.1.3.1 TSS Evaluation Activity

The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked.

If the application is relying on random bit generation from the host platform, the evaluator shall verify the TSS includes the name/manufacturer of the external RBG and describes the function call and parameters used when calling the external DRBG function. If different external RBGs are used for different platforms, the evaluator shall verify the TSS identifies each RBG for each platform. Also, the evaluator shall verify the TSS includes a short description of the vendor's assumption for the amount of entropy seeding the external DRBG. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data.

Section 6.2 of [ST] ("Cryptographic Support") states that the versions of the TOE implement the CTR_DRBG Deterministic Random Bit Generation (DRBG) algorithm, within the Wickr OpenSSL Version 2.0.16 cryptographic library.

Section 6.2 of [ST] ("Cryptographic Support") states the TOE maintains a unique API key it uses as a credential for server communications. The TOE stores this credential, along with sensitive data regarding messaging history and contacts, in an encrypted SQL database. All versions of the TOE use platform-provided AES to encrypt the database. The iOS version uses Coredata, which uses Apple CoreCrypto to encrypt the database using AES-256-GCM, while all other versions use SQLCipher, which calls the respective platform cryptographic libraries to encrypt the database using AES-256-CBC.

All versions of the TOE generate a 256-bit Data Encryption Key (DEK). All TOE versions invoke a platform-provided AES implementation to encrypt the SQL database using the DEK. All versions of the TOE use the TOE's AES implementation to encrypt the DEK using a 256-bit Key Encryption Key (KEK). The generation of the DEK/KEK and encryption of the KEK are performed by the TSF.

Section 6.2 of [ST] states the TOE obtains random bits by invoking the following platform APIs, depending on the TOE version:

- Windows: ProcessPRNG
- Android, macOS, iOS: `/dev/random`.

2.1.3.2 Guidance Evaluation Activity

None.

2.1.3.3 Test Evaluation Activity

None.

2.1.4 Cryptographic Operation – Encryption/Decryption (FCS_COP.1/SKC)

2.1.4.1 TSS Evaluation Activity

None.

2.1.4.2 Guidance Evaluation Activity

The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required modes and key sizes is present.

Section "TOE Description", sub-section "" of [CCECG] states the TOE uses platform-provided cryptographic services for protection of locally-stored credential data. No configuration is required to activate these services.

2.1.4.3 Test Evaluation Activity

Performed in accordance with NIAP Policy Letter #5.

Section 6.2 of [ST] ("Cryptographic Support"), Table 4 ("Cryptographic Functions") identifies the CAVP certifications verifying AES encryption and decryption, as follows.

Algorithm	Tested Capabilities	Certificates
AES-GCM as defined in NIST SP 800-38D	AES-GCM: Direction: Decrypt, Encrypt IV Generation: Internal Key Lengths: 128, 256 (bits)	CAVP #A3372

2.1.5 Random Bit Generation Services (FCS_RBG_EXT.1)

2.1.5.1 TSS Evaluation Activity

If ***use no DRBG functionality*** is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

If ***implement DRBG functionality*** is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.

If ***invoke platform-provided DRBG functionality*** is selected, the evaluator performs the following activities. The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used "correctly" for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

In FCS_RBG_EXT.1.1, the ST author has selected ***implement DRBG functionality***. The evaluator confirmed the ST includes FCS_RBG_EXT.2.

2.1.5.2 Guidance Evaluation Activity

None.

2.1.5.3 Test Evaluation Activity

If ***invoke platform-provided DRBG functionality*** is selected, the following tests shall be performed:

The evaluator shall decompile the application binary using a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

The following are the per-platform list of acceptable APIs:

Android: The evaluator shall verify that the application uses at least one of `javax.crypto.KeyGenerator` class or the `java.security.SecureRandom` class or `/dev/random` or `/dev/urandom`.

Microsoft Windows: The evaluator shall verify that `rand_s`, `RtlGenRandom`, `BCryptGenRandom`, or `CryptGenRandom` API is used for classic desktop applications. The evaluator shall verify the application uses the `RNGCryptoServiceProvider` class or derives a class from `System.Security.Cryptography.RandomNumberGenerator` API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, `CryptGenRandom` may be removed as an option as it is no longer the preferred API per vendor documentation.

Apple iOS: The evaluator shall verify that the application invokes either `SecRandomCopyBytes`, `CCRandomGenerateBytes`, or `CCRandomCopyBytes`, or uses `/dev/random` directly to acquire random

Apple macOS: The evaluator shall verify that the application invokes either `CCRandomGenerateBytes` or `CCRandomCopyBytes`, or collects random from `/dev/random`.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

In FCS_RBG_EXT.1.1, the ST author has selected ***implement DRBG functionality***. Therefore, this activity is not applicable to the TOE.

2.1.6 Random Bit Generation from Application (FCS_RBG_EXT.2)

2.1.6.1 FCS_RBG_EXT.2.1

2.1.6.1.1 TSS Evaluation Activity

None.

2.1.6.1.2 Guidance Evaluation Activity

None.

2.1.6.1.3 Test Evaluation Activity

Performed in accordance with NIAP Policy Letter #5.

Section 6.2 of [ST] ("Cryptographic Support"), Table 4 ("Cryptographic Functions") identifies the CAVP certification verifying deterministic random bit generation services implemented by the TOE, as follows.

Algorithm	Tested Capabilities	Certificates
CTR_DRBG in accordance with NIST Special Publication 800-90A	Counter DRBG Mode: AES-256	A3372

2.1.6.2 FCS_RBG_EXT.2.2

2.1.6.2.1 TSS Evaluation Activity

Documentation shall be produced – and the evaluator shall perform the activities – in accordance with Appendix D - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

The vendor provided documentation describing the entropy sources used by the TOE.

2.1.6.2.2 Evaluation Activity

None.

2.1.6.2.3 Test Evaluation Activity

In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates.

The proprietary Entropy Analysis Report (EAR) describes how the TSF extracts random data from software-based sources to ensure that an amount of entropy that is at least equal to the strength of the generated keys is present (i.e., at least 256 bits when the largest supported keys are generated) when seeding the DRBG for key generation purposes. The TOE relies on entropy sources provided by its platform.

2.1.7 Storage of Credentials (FCS_STO_EXT.1)

2.1.7.1 TSS Evaluation Activity

The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

Section 6.2 of [ST] ("Cryptographic Support") states the TOE maintains a unique API key that it uses as a credential for server communications. The TOE stores the API key in an encrypted SQL database. All versions of the TOE use platform-provided AES to encrypt the database. The iOS version uses Coredata, which uses Apple CoreCrypto to encrypt the database using AES-256-GCM. The other versions of the TOE use SQLCipher, which calls the respective platform cryptographic libraries to encrypt the database using AES-256-CBC.

The TOE uses a 256-bit Data Encryption Key (DEK) to encrypt the SQL database. The TOE generate the DEK using the CTR_DRBG algorithm implemented in the TOE's OpenSSL 2.0.16 FIPS Object Module cryptographic library,

The TOE versions use the TOE's AES implementation to encrypt the DEK using a 256-bit Key Encryption Key (KEK).

2.1.7.2 Guidance Evaluation Activity

None.

2.1.7.3 Test Evaluation Activity

For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/SKC or conditioned according to FCS_CKM.1.1/AK and FCS_CKM.1/PBKDF.

Android, iOS, macOS and Windows encryption of the Data Encryption Key that is then used to encrypt the SQL database.

For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

Platforms: Android... The evaluator shall verify that the application uses the Android **KeyStore** or the Android **KeyChain** to store certificates.

Test Results – per TQ1296, platform-provided AES is an allowable storage mechanism to protect credentials. The Android version of the TOE uses SQLCipher, which calls the Android cryptographic library (Samsung BoringSSL) to encrypt the database using AES-256-CBC (CAVP A#2351).

Platforms: Windows... The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

Test Results – per TQ1296, platform-provided AES is an allowable storage mechanism to protect credentials. The Windows version of the TOE uses SQLCipher, which calls the Windows cryptographic library (Windows 11 SymCrypt) to encrypt the database using AES-256-CBC (CAVP A#2004).

Platforms: Apple iOS... The evaluator shall verify that all credentials are stored within a **Keychain**.

Test Results – per TQ1296, platform-provided AES is an allowable storage mechanism to protect credentials. The iOS version of the TOE uses Coredata, which uses Apple CoreCrypto to encrypt the database using AES-256-GCM (CAVP A#2795).

Platforms: Apple macOS... The evaluator shall verify that all credentials are stored within **Keychain**.

Test Results – per TQ1296, platform-provided AES is an allowable storage mechanism to protect credentials. The macOS version of the TOE uses SQLCipher, which calls the macOS cryptographic library (CoreCrypto) to encrypt the database using AES-256-CBC (CAVP A#919).

2.2 User Data Protection (FDP)

2.2.1 Encryption of Sensitive Application Data (FDP_DAR_EXT.1)

2.2.1.1 TSS Evaluation Activity

The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application. The evaluator shall then ensure that the following activities cover all of the sensitive data identified in the TSS.

Section 6.3 of [ST] (“User Data Protection”), Table 5 (“Sensitive Data”) lists the data considered to be sensitive by the TOE, the manner in which the data is stored, any communication (“Exchange”) of that data, and how the data is protected at rest and in transit. The TOE protects all sensitive data in accordance with either FCS_STO_EXT.1/1 or FCS_STO_EXT.1/2, depending on the specific TOE version.

If ***not store any sensitive data*** is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory. The evaluator shall also ensure that this is consistent with the filesystem test below.

The ST does not select “not store any sensitive data”, so this activity is not applicable.

2.2.1.2 Guidance Evaluation Activity

None.

2.2.1.3 Test Evaluation Activity

Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1.

Section 6.3 of [ST] states the TOE protects all sensitive data in accordance with FCS_STO_EXT.1. As such, the Test activities are not applicable.

The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted.

If ***leverage platform-provided functionality*** is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis:

Platforms: Android… The evaluator shall inspect the TSS and verify that it describes how files containing sensitive data are stored with the **MODE_PRIVATE** flag set.

The ST states that all sensitive data is to be protected in accordance with FCS_STO_EXT.1. There is no sensitive data that is not covered by FCS_STO_EXT.1 Accordingly there are no evaluation activities to be performed for this SFR.

Platforms: Microsoft Windows… The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.

The ST states that all sensitive data is to be protected in accordance with FCS_STO_EXT.1. There is no sensitive data that is not covered by FCS_STO_EXT.1 Accordingly there are no evaluation activities to be performed for this SFR.

Platforms: Apple iOS… The evaluator shall inspect the TSS and ensure that it describes how the application uses the **Complete Protection**, **Protected Unless Open**, or **Protected Until First User Authentication Data Protection Class** for each data file stored locally.

The ST states that all sensitive data is to be protected in accordance with FCS_STO_EXT.1. There is no sensitive data that is not covered by FCS_STO_EXT.1 Accordingly there are no evaluation activities to be performed for this SFR.

Platforms: Apple macOS... The macOS platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

The ST states that all sensitive data is to be protected in accordance with FCS_STO_EXT.1. There is no sensitive data that is not covered by FCS_STO_EXT.1 Accordingly there are no evaluation activities to be performed for this SFR.

2.2.2 Access to Platform Resources (FDP_DEC_EXT.1)

2.2.2.1 FDP_DEC_EXT.1.1

2.2.2.1.1 TSS Evaluation Activity

None.

2.2.2.1.2 Guidance Evaluation Activity

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.

Section “TOE Description”, sub-section “Evaluated Configuration” of [CCECG] states the TOE interacts with the underlying platform when using its network connectivity. Network usage of the TOE is authorized implicitly through user guidance; it does not make any specific requests on its own to use network services once installed. This sub-section further states the TOE may access location services in support of peer connections. This is consistent with the selections in FDP_DEC_EXT.1.1.

2.2.2.1.3 Test Evaluation Activity

Platforms: Android... The evaluator shall verify that each **uses-permission** entry in the **AndroidManifest.xml** file for access to a hardware resource is reflected in the selection.

The only platform resources that the TOE accesses are network connectivity and location services. This was clearly documented in the ST. During testing the TOE did not make any attempts to access other resources.

Platforms: Microsoft Windows... For Windows Universal Applications the evaluator shall check the **WMAppManifest.xml** file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as **ID_CAP_ISV_CAMERA**, **ID_CAP_LOCATION**, **ID_CAP_NETWORKING**, **ID_CAP_MICROPHONE**, **ID_CAP_PROXIMITY** and so on. A complete list of Windows App permissions can be found at:

- http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of the required hardware resources.

The only platform resources that the TOE accesses are network connectivity and location services. This was clearly documented in the ST. During testing the TOE did not make any attempts to access other resources.

Platforms: Apple iOS... The evaluator shall verify that either the application or the documentation provides a list of the hardware resources it accesses.

The only platform resources that the TOE accesses are network connectivity and location services. This was clearly documented in the ST. During testing the TOE did not make any attempts to access other resources.

Platforms: Apple macOS... The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

The only platform resources that the TOE accesses are network connectivity and location services. This was clearly documented in the ST. During testing the TOE did not make any attempts to access other resources.

2.2.2.2 FDP_DEC_EXT.1.2

2.2.2.2.1 TSS Evaluation Activity

None.

2.2.2.2.2 Guidance Evaluation Activity

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information repository which it accesses, identify the justification as to why access is required.

Section “TOE Description”, sub-section “Evaluated Configuration” of [CCECG] states the TOE may access photos and address book in support of peer connections. This is consistent with the selections in FDP_DEC_EXT.1.2.

2.2.2.2.3 Test Evaluation Activity

Platforms: Android... The evaluator shall verify that each **uses-permission** entry in the **AndroidManifest.xml** file for access to a sensitive information repository is reflected in the selection.

The only platform information repositories the TOE accesses are the address book and photos. This is clearly documented in the ST. During testing the TOE did not make any attempts to access other repositories.

Platforms: Microsoft Windows... For Windows Universal Applications the evaluator shall check the **WMAppManifest.xml** file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as **ID_CAP_CONTACTS**, **ID_CAP_APPOINTMENTS**, **ID_CAP_MEDIALIB** and so on. A complete list of Windows App permissions can be found at:

- http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.

The only platform information repositories the TOE accesses are the address book and photos. This is clearly documented in the ST. During testing the TOE did not make any attempts to access other repositories.

Platforms: Apple iOS... The evaluator shall verify that either the application software or its documentation provides a list of the sensitive information repositories it accesses.

The only platform information repositories the TOE accesses are the address book and photos. This is clearly documented in the ST. During testing the TOE did not make any attempts to access other repositories.

Platforms: Apple macOS... The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

The only platform information repositories the TOE accesses are the address book and photos. This is clearly documented in the ST. During testing the TOE did not make any attempts to access other repositories.

2.2.3 Network Communications (FDP_NET_EXT.1)

2.2.3.1 TSS Evaluation Activity

None.

2.2.3.2 Guidance Evaluation Activity

None.

2.2.3.3 Test Evaluation Activity

The evaluator shall perform the following tests:

Test 1: The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.

The evaluator ran a packet capture of network traffic going to and from the TOE while it was in operation. All of the captured traffic was associated with a TLS connection to a Wickr Messenger server.

Test 2: The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).

The evaluator used nmap to verify that no ports were opened by the TOE.

Platforms: Android... If "***no network communication***" is selected, the evaluator shall ensure that the application's **AndroidManifest.xml** file does not contain a **uses-permission** or **uses-permission-sdk-23** tag containing **android:name="android.permission.INTERNET"**. In this case, it is not necessary to perform the above Tests 1 and 2, as the platform will not allow the application to perform any network communication.

The ST does not select “no network communication”. As such, the evaluator performed Tests 1 and 2 on the Android version of the TOE.

2.3 Identification and Authentication (FIA)

2.3.1 X.509 Certificate Validation (FIA_X509_EXT.1/1)

2.3.1.1 FIA_X509_EXT.1.1/1

2.3.1.1.1 TSS Evaluation Activity

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Section 6.4 of [ST] (“Identification and Authentication”) states the TOE relies on platform-provided functionality for validation of X.509 certificates. Certificate and certificate path validation is performed in accordance with RFC 5280.

2.3.1.1.2 Guidance Evaluation Activity

None.

2.3.1.1.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
- by omitting the basicConstraints field in one of the issuing certificates,
- by setting the basicConstraints field in an issuing certificate to have CA=False,
- by omitting the CA signing bit of the key usage field in an issuing certificate, and
- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.

The evaluator opened a connection from the TOE to a TLS test server whose certificate was not issued by a valid CA. The TOE rejected the certificate. This test was repeated for certificates issued by a CA that lacked basicConstraints, a CA that had basicConstraints set to False, a CA that lacked the CA signing bit and a CA with a too short path length value. In each instance the TOE rejected the certificate.

Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

The evaluator opened a connection from the TOE to a TLS test server with an expired certificate. The TOE rejected the certificate.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL, OCSP, OCSP Stapling or OCSP Multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

- The evaluator shall test revocation of the node certificate.
- The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP Stapling per RFC 6066 is the only supported revocation method, this test is omitted.
- The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

For Windows and MacOS the evaluator opened a connection from the TOE to a TLS test server which used CRLs for certificate revocation checking. The TOE was able to use CRLs to check the validity of a good certificate. Additionally, it rejected a leaf certificate that had been revoked and a good leaf certificate that was issued by an intermediate CA whose certificate had been revoked.

Modified in accordance with TD0669.

Test 4: If any OCSP option is selected, the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the-middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set~~, and~~. The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection.

Note: The intent of this test is to ensure a TSF does not trust invalid revocation status information. A TSF receiving invalid revocation status information from the only advertised certificate status provider should treat the certificate whose status is being checked as invalid. This should generally be treated differently from the case where the TSF is not able to establish a connection to check revocation status information, but it is acceptable that the TSF ignore any invalid information and attempt to find another source of revocation status (another advertised provider, a locally configured provider, or cached information) and treat this situation as not having a connection to a valid certificate status provider.

For Windows, MacOS and Android the evaluator verified that the TOE would not accept a certificate whose revocation status was verified by a CRL signer with an invalid certificate.

Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

The evaluator opened a connection from the TOE to a TLS test server. The server had the first eight bytes of its certificate modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator opened a connection from the TOE to a TLS test server. The server had the last byte of its certificate modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator opened a connection from the TOE to a TLS test server. The server had its public key modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

N/A. The TOE does not claim support for EC certificates in FCS_COP.1/SIG.

Test 9: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

N/A. The TOE does not claim support for EC certificates in FCS_COP.1/SIG.

2.3.1.2 FIA_X509_EXT.1.2/1

2.3.1.2.1 TSS Evaluation Activity

None.

2.3.1.2.2 Guidance Evaluation Activity

None.

2.3.1.2.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall ensure that the certificate of at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate without the basicConstraints extension to the TOE's trust store.

This test has already been performed under FIA_X509_EXT.1.1 Test 1 (Subtest 3 – basicConstraint not set). In that test, the evaluator attempted to connect to the Wickr Messenger while the issuer of the certificate was missing the basicConstraint value, which resulted in a session termination from the TOE.

Test 2: The evaluator shall ensure that the certificate of at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension to the TOE's trust store.

This test has already been performed under FIA_X509_EXT.1.1 Test 1 (Subtest 3 – basicConstraints set to False test). In that test, the evaluator attempted to connect to the Wickr Messenger while the issuer of the certificate had a basicConstraint set to False, which resulted in a session termination from the TOE.

2.3.2 X.509 Certificate Validation (FIA_X509_EXT.1/2)

2.3.2.1 FIA_X509_EXT.1.1/2

2.3.2.1.1 TSS Evaluation Activity

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Section 6.4 of [ST] ("Identification and Authentication") states the TOE relies on platform-provided functionality for validation of X.509 certificates. Certificate and certificate path validation is performed in accordance with RFC 5280.

2.3.2.1.2 Guidance Evaluation Activity

None.

2.3.2.1.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
- by omitting the basicConstraints field in one of the issuing certificates,
- by setting the basicConstraints field in an issuing certificate to have CA=False,
- by omitting the CA signing bit of the key usage field in an issuing certificate, and
- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.

The evaluator opened a connection from the TOE to a TLS test server whose certificate was not issued by a valid CA. The TOE rejected the certificate. This test was repeated for certificates issued by a CA that lacked basicConstraints, a CA that had basicConstraints set to False, a CA that lacked the CA signing bit and a CA with a too short path length value. In each instance the TOE rejected the certificate.

Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

The evaluator opened a connection from the TOE to a TLS test server with an expired certificate. The TOE rejected the certificate.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL, OCSP, OCSP Stapling or OCSP Multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

- The evaluator shall test revocation of the node certificate.
- The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP Stapling per RFC 6066 is the only supported revocation method, this test is omitted.
- The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

For Android the evaluator opened a connection from the TOE to a TLS test server which used CRLs for certificate revocation checking. The TOE was able to use CRLs to check the validity of a good certificate. Additionally, it rejected a leaf certificate that had been revoked and a good leaf certificate that was issued by an intermediate CA whose certificate had been revoked.

Modified in accordance with TD0669.

Test 4: If any OCSP option is selected, the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the-middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set~~, and~~. The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection.

Note: The intent of this test is to ensure a TSF does not trust invalid revocation status information. A TSF receiving invalid revocation status information from the only advertised certificate status provider should treat the certificate whose status is being checked as invalid. This should generally be treated differently from the case where the TSF is not able to establish a connection to check revocation status information, but it is acceptable that the TSF ignore any invalid information and attempt to find another source of revocation status (another advertised provider, a locally configured provider, or cached information) and treat this situation as not having a connection to a valid certificate status provider.

For Windows, MacOS and Android the evaluator verified that the TOE would not accept a certificate whose revocation status was verified by a CRL signer with an invalid certificate.

Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

The evaluator opened a connection from the TOE to a TLS test server. The server had the first eight bytes of its certificate modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator opened a connection from the TOE to a TLS test server. The server had the last byte of its certificate modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator opened a connection from the TOE to a TLS test server. The server had its public key modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

N/A. The TOE does not claim support for EC certificates in FCS_COP.1/SIG.

Test 9: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

N/A. The TOE does not claim support for EC certificates in FCS_COP.1/SIG.

2.3.2.2 FIA_X509_EXT.1.2/2

2.3.2.2.1 TSS Evaluation Activity

None.

2.3.2.2.2 Guidance Evaluation Activity

None.

2.3.2.2.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall ensure that the certificate of at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate without the basicConstraints extension to the TOE's trust store.

This test has already been performed under FIA_X509_EXT.1.1 Test 1 (Subtest 3 – basicConstraint not set). In that test, the evaluator attempted to connect to the Wickr Messenger while the issuer of the certificate was missing the basicConstraint value, which resulted in a session termination from the TOE.

Test 2: The evaluator shall ensure that the certificate of at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension to the TOE's trust store.

This test has already been performed under FIA_X509_EXT.1.1 Test 1 (Subtest 3 – basicConstraints set to False test). In that test, the evaluator attempted to connect to the Wickr Messenger while the issuer of the certificate had a basicConstraint set to False, which resulted in a session termination from the TOE.

2.3.3 X.509 Certificate Validation (FIA_X509_EXT.1/3)

2.3.3.1 FIA_X509_EXT.1.1/3

2.3.3.1.1 TSS Evaluation Activity

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Section 6.4 of [ST] ("Identification and Authentication") states the TOE relies on platform-provided functionality for validation of X.509 certificates. Certificate and certificate path validation is performed in accordance with RFC 5280.

2.3.3.1.2 Guidance Evaluation Activity

None.

2.3.3.1.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
- by omitting the basicConstraints field in one of the issuing certificates,
- by setting the basicConstraints field in an issuing certificate to have CA=False,
- by omitting the CA signing bit of the key usage field in an issuing certificate, and
- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.

The evaluator opened a connection from the TOE to a TLS test server whose certificate was not issued by a valid CA. The TOE rejected the certificate. This test was repeated for certificates issued by a CA that lacked basicConstraints, a CA that had basicConstraints set to False, a CA that lacked the CA signing bit and a CA with a too short path length value. In each instance the TOE rejected the certificate.

Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

The evaluator opened a connection from the TOE to a TLS test server with an expired certificate. The TOE rejected the certificate.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL, OCSP, OCSP Stapling or OCSP Multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

- The evaluator shall test revocation of the node certificate.
- The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP Stapling per RFC 6066 is the only supported revocation method, this test is omitted.
- The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

For iOS the evaluator opened a connection from the TOE to a TLS test server which used OCSP for certificate revocation checking. The TOE was able to use OCSP to check the validity of a good certificate. Additionally, it rejected a leaf certificate that had been revoked and a good leaf certificate that was issued by an intermediate CA whose certificate had been revoked.

Modified in accordance with TD0669.

Test 4: If any OCSP option is selected, the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the-middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set~~, and~~. The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection.

Note: The intent of this test is to ensure a TSF does not trust invalid revocation status information. A TSF receiving invalid revocation status information from the only advertised certificate status provider should treat the certificate whose status is being checked as invalid. This should generally be treated differently from the case where the TSF is not able to establish a connection to check revocation status information, but it is acceptable that the TSF ignore any invalid information and attempt to find another source of revocation status (another advertised provider, a locally configured provider, or cached information) and treat this situation as not having a connection to a valid certificate status provider.

For iOS the TOE treated an OCSP responder with an invalid OCSP signing purpose the same as a responder that was unreachable.

Note: The TOE relies on iOS's certificate validation functionality which is designed to accept a TLS certificate if that certificate's OCSP responder has an invalid certificate. iOS 15 successfully completed CC certification with this functionality as VID11237. iOS behavior is documented on page 145 of their publicly available AAR.

Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

The evaluator opened a connection from the TOE to a TLS test server. The server had the first eight bytes of its certificate modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator opened a connection from the TOE to a TLS test server. The server had the last byte of its certificate modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator opened a connection from the TOE to a TLS test server. The server had its public key modified. Upon receiving the bad certificate the TOE terminated the TLS connection attempt.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

N/A. The TOE does not claim support for EC certificates in FCS_COP.1/SIG.

Test 9: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

N/A. The TOE does not claim support for EC certificates in FCS_COP.1/SIG.

2.3.3.2 FIA_X509_EXT.1.2/3

2.3.3.2.1 TSS Evaluation Activity

None.

2.3.3.2.2 Guidance Evaluation Activity

None.

2.3.3.2.3 Test Evaluation Activity

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

Test 1: The evaluator shall ensure that the certificate of at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate without the basicConstraints extension to the TOE's trust store.

This test has already been performed under FIA_X509_EXT.1.1 Test 1 (Subtest 3 – basicConstraint not set). In that test, the evaluator attempted to connect to the Wickr Messenger while the issuer of the certificate was missing the basicConstraint value, which resulted in a session termination from the TOE.

Test 2: The evaluator shall ensure that the certificate of at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension to the TOE's trust store.

This test has already been performed under FIA_X509_EXT.1.1 Test 1 (Subtest 3 – basicConstriant set to False test). In that test, the evaluator attempted to connect to the Wickr Messenger while the issuer of the certificate had a basicContraint set to False, which resulted in a session termination from the TOE.

2.3.4 X.509 Certificate Authentication (FIA_X509_EXT.2)

2.3.4.1 TSS Evaluation Activity

The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

Section 6.4 of [ST] ("Identification and Authentication") states the TOE chooses what certificates to use to validate the authenticity of a TLS server based on what is presented to it by the server as part of establishing a TLS session.

Section 4.4.1 of [Install] ("Hostname and TLS") states the Wickr Server administrator can choose to upload a custom certificate and key to be used by the Server to authenticate itself to TLS clients and provides a screen shot of the Admin Web GUI where this can be done. Alternatively, Section 5 of [Install] ("Configure Wickr Enterprise") describes how the Server administrator can upload a separate custom certificate to authenticate the Wickr Server to Wickr Clients.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described.

Section 6.4 of [ST] states in the event that the revocation status of a certificate cannot be verified, a connection will not be established. There is no distinction between trusted channels.

If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

The statement of FIA_X509_EXT.2.2 in Section 5.2.3 of [ST] ("Identification and Authentication") selects "not accept the certificate". Therefore, there is no capability for the administrator to specify the default action, and no requirement for operational guidance to provide such instructions.

2.3.4.2 Guidance Evaluation Activity

None.

Note, one of the TSS activities places a requirement on the Guidance.

The evaluator shall check … any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

[CCECG] Section "Software Download, Installation and Configuration Download and Installation > Certificate Configuration" provides the instructions to import the certificate needed to validate a TLS server certificate and to install a CA certificate on the device. The instructions are platform specific.

Section 4.4.5 of [Install] ("Preflight Checks") provides the guidance to run checks to ensure that the system meets the requirements for running Wickr Enterprise.

2.3.4.3 Test Evaluation Activity

The evaluator shall perform the following test for each trusted channel:

Test 1: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

For Windows, MacOS and Android the evaluator had the TOE attempt to connect to a TLS server whose certificate used CRLs for validation. The ability of the TOE to retrieve the CRL and verify the server's certificate was demonstrated via wire capture. After this the CRL was deleted from the CDP and the test was repeated. The Android instance of the TOE terminated the connection while the MacOS and Windows instances of the TOE continued the connection which is consistent with the ST.

Test 2: The evaluator shall demonstrate that an invalid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity cannot be accepted.

This test has been performed in conjunction with FIA_X509_EXT.1.1 Test 3. In that test, the evaluator confirmed that the TOE properly reached out to the revocation server to perform CRL functions against the certificate that was provided. Furthermore, the TOE properly accepted the traffic when it received valid certificate, and it also properly terminated the session when it received the revocation checks of a revoked certificate.

2.4 Security Management (FMT)

2.4.1 Secure by Default Configuration (FMT_CFG_EXT.1)

2.4.1.1 FMT_CFG_EXT.1.1

2.4.1.1.1 TSS Evaluation Activity

The evaluator shall check the TSS to determine if the application requires any type of credentials and if the application installs with default credentials.

Section 6.5 of [ST] ("Security Management") states the TOE does not have any default credentials. The platform's file system protects all TOE components from unauthorized access. The platform administrator registers the user.

2.4.1.1.2 Guidance Evaluation Activity

None.

2.4.1.1.3 Test Evaluation Activity

If the application uses any default credentials the evaluator shall run the following tests.

Test 1: The evaluator shall install and run the application without generating or loading new credentials and verify that only the minimal application functionality required to set new credentials is available.

Test 2: The evaluator shall attempt to clear all credentials and verify that only the minimal application functionality required to set new credentials is available.

Test 3: The evaluator shall run the application, establish new credentials and verify that the original default credentials no longer provide access to the application.

The TOE does not use any default credentials. As such, these tests are not applicable.

2.4.1.2 FMT_CFG_EXT.1.2

2.4.1.2.1 TSS Evaluation Activity

None.

2.4.1.2.2 Guidance Evaluation Activity

None.

2.4.1.2.3 Test Evaluation Activity

The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

Platforms: Android… The evaluator shall run the command `find -L . -perm /002` inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

The evaluator found no world-writable files in the TOE's data directories.

Platforms: Microsoft Windows… The evaluator shall run the `SysInternals` tools, `Process Monitor` and `Access Check` (or tools of equivalent capability, like `icacls.exe`) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. For Windows Universal Applications the evaluator shall consider the requirement met because of the AppContainer sandbox.

The evaluator found no world-writable files in the TOE's data directories.

Platforms: Apple iOS… The evaluator shall determine whether the application leverages the appropriate Data Protection Class for each data file stored locally.

Data protection is used by default for iOS apps, there is nothing to be enabled/disabled or verified for this test.

Platforms: Apple macOS… The evaluator shall run the command `find . -perm +002` inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

The evaluator found no world-writable files in the TOE's data directories.

2.4.2 Supported Configuration Mechanism (FMT_MEC_EXT.1)

2.4.2.1 TSS Evaluation Activity

The evaluator shall review the TSS to identify the application's configuration options (e.g. settings) and determine whether these are stored and set using the mechanisms supported by the platform or implemented by the application in accordance with the PP-Module for File Encryption. At a minimum the TSS shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an SFR.

Section 6.5 of [ST] ("Security Management") states the Wickr Server interfaces with the client to communicate configuration changes through a JSON API on the client. Specifically, the server can be used to configure the number of authentication failures that are allowed for a user to access the client.

The TOE allows local users to configure logging, which includes whether logging is enabled, whether enhanced logging is enabled, and where logs are stored. The user can also delete stored logs through the TOE. The TOE also has a 'secure shredder' function that will overwrite all memory and disk space used by files that are opened through the application.

The TOE stores configuration settings that are not sensitive in the manner specified below. This includes user identification information (username/email), the current invalid password retry count, flags for whether the user is registered, needs to be shown a different onboarding screen, or (for iOS/Android) whether the user has granted permission for the application to use device features, and basic key/value data such as recently used emoji characters:

- Android—the TOE stores all data other than user credentials in a custom **SharedPreferences** implementation that stores an encrypted blob (as opposed to plaintext key/value pairs) using the Java Native Interface library. The files are all in the protected/private app folder, which requires root privilege to access from outside of the TOE.
- iOS—the TOE stores configuration settings using the user defaults system. The app folder uses **FileProtectionCompleteUntilFirstUserAuthentication** to enable encryption. Registration configuration information is stored as a JSON-encoded file in the app document folder. The TOE

saves this file with the **NSDataWritingFileProtectionCompleteUntilFirstUserAuthentication** flag, so the file is only decrypted if the user has unlocked the phone.

- macOS—the TOE stores configuration settings using the `NSUserDefaults` plist format.
- Windows—the TOE stores configuration settings in the registry under `HKEY_USERS\[USER ID]\Software\Wickr Enterprise`.

Conditional: If "***implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption***" is selected, the evaluator shall ensure that the TSS identifies those options, as well as indicates where the encrypted representation of these options is stored.

The ST does not make this selection.

2.4.2.2 Guidance Evaluation Activity

None.

2.4.2.3 Test Evaluation Activity

If "***invoke the mechanisms recommended by the platform vendor for storing and setting configuration options***" is chosen, the method of testing varies per platform as follows:

Modified in accordance with TD0624.

Platforms: Android... The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least one ~~XML~~ file exists at location `/data/data/package/shared_prefs/` (for **SharedPreferences**) and/or `/data/data/package/files/datastore` (for **DataStore**) ~~reflects the changes made to the configuration to verify that the application used `SharedPreferences` and/or `PreferenceActivity` classes for storing configuration data,~~ where `package` is the Java package of the application. For **SharedPreferences** the evaluator shall examine the XML file to make sure it reflects the changes made to the configuration to verify that the application used **SharedPreferences** and/or **PreferenceActivity** to store the configuration data. For **DataStore** the evaluator shall use a protocol buffer analyzer to examine the file to make sure it reflects the changes made to the configuration to verify that the application used **DataStore** to store the configuration data.

The evaluator verified that the contents of the SharedPreferences and PreferenceActivity folders were changed when configuration changes were made to the TOE.

Platforms: Microsoft Windows... The evaluator shall determine and verify that Windows Universal Applications use either the `Windows.Storage` namespace, `Windows.UI.ApplicationSettings` namespace, or the `IsolatedStorageSettings` namespace for storing application specific settings. For .NET applications, the evaluator shall determine and verify that the application uses one of the locations listed in https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/ for storing application specific settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the `SysInternals` tool `ProcMon` and make changes to its configuration. The evaluator shall verify that `ProcMon` logs show corresponding changes to the Windows Registry or `C:\ProgramData\` directory.

The evaluator used the Process Monitor tool to verify that configuration changes to the TOE were written to the registry.

Platforms: Apple iOS... The evaluator shall verify that the app uses the `user defaults system` or `key-value store` for storing all settings.

The user defaults system is used to store configuration settings.

Platforms: Apple macOS... The evaluator shall verify that the application stores and retrieves settings using the `NSUserDefaults` class.

The evaluator confirmed that TOE stored and retrieves settings in the proper NSUserDefaults location.

If "***implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption***" is selected, for all configuration options listed in the TSS as being stored and protected using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

The ST does not make this selection.

2.4.3 Specification of Management Functions (FMT_SMF.1)

2.4.3.1 TSS Evaluation Activity

None.

2.4.3.2 Guidance Evaluation Activity

The evaluator shall verify that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

As described in Section 6.5 of [ST] ("Security Management"), the TOE supports the following security-relevant management functions:

- Configuration of the number of authentication failures allowed for a user to access the TOE
- Configuration of logging
- Deletion of stored logs
- Secure overwrite of file storage ("secure shredder").

Section "Software Download, Installation and Configuration", subsection "Download and Installation" of [CCECG] also describes the function to configure the number of authentication failures for a user to access the TOE.

Section "Software Download, Installation and Configuration", subsection "Logging Configuration" of [CCECG] describes the function to configure logging on the TOE, including enabling logging, enabling enhanced logging, configuring the location for log file storage, and deletion of stored logs.

Section "Software Download, Installation and Configuration", subsection "Secure Wipe" of [CCECG] describes the function to securely overwrite ("wipe", or "shred") storage locations used by files opened within the TOE.

2.4.3.3 Test Evaluation Activity

The evaluator shall test the application's ability to provide the management functions by configuring the application and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

The evaluator used the TOE's interface to configure logging, delete stored logs, use the secure shredder and configure authentication lockout.

2.5 Privacy (FPR)

2.5.1 User Consent for Transmission of Personally Identifiable Information (FPR_ANO_EXT.1)

2.5.1.1 TSS Evaluation Activity

The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted.

Section 6.6 of [ST] ("Privacy") states the TOE never transmits known PII over a network. However, the user may direct the TOE to transmit data that contains PII.

2.5.1.2 Guidance Evaluation Activity

None.

2.5.1.3 Test Evaluation Activity

If ***require user approval before executing*** is selected, the evaluator shall run the application and exercise the functionality responsible for transmitting PII and verify that user approval is required before transmission of the PII.

The ST does not make this selection.

2.6 Protection of the TSF (FPT)

2.6.1 Anti-Exploitation Capabilities (FPT_AEX_EXT.1)

2.6.1.1 FPT_AEX_EXT.1.1

2.6.1.1.1 TSS Evaluation Activity

The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

Section 6.7 of [ST] ("Protection of the TSF") states the vendor uses the DYNAMICBASE compiler flag to enable address space layout randomization (ASLR) when compiling the TOE.

2.6.1.1.2 Guidance Evaluation Activity

None.

2.6.1.1.3 Test Evaluation Activity

The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform. For those platforms requiring the same application running on two different systems, the evaluator may alternatively use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings.

Platforms: Android… The evaluator shall run the same application on two different Android systems. Both devices do not need to be evaluated, as the second device is acting only as a tool. Connect via ADB and inspect `/proc/PID/maps`. Ensure the two different instances share no memory mappings made by the application at the same location.

An inspection of PID maps shows that different instances of the TOE do not share memory mappings.

Platforms: Microsoft Windows… The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

VMMap showed that different instances of the TOE do not share memory mappings.

Platforms: Apple iOS… The evaluator shall perform a static analysis to search for any `mmap` calls (or API calls that call `mmap`), and ensure that no arguments are provided that request a mapping at a fixed address.

Mmap showed that different instances of the TOE do not share memory mappings.

Platforms: Apple macOS… The evaluator shall run the same application on two different Mac systems. The evaluator shall then compare their memory maps using `vmmap PID` to ensure the two different instances share no mapping locations.

VMMap showed that different instances of the TOE do not share memory mappings.

2.6.1.2 FPT_AEX_EXT.1.2

2.6.1.2.1 TSS Evaluation Activity

None.

2.6.1.2.2 Guidance Evaluation Activity

None.

2.6.1.2.3 Test Evaluation Activity

The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

Platforms: Android… The evaluator shall perform static analysis on the application to verify that

- `mmap` is never be invoked with both the `PROT_WRITE` and `PROT_EXEC` permissions, and
- `mprotect` is never invoked.

The evaluator examined the TOE’s source code and verified that mmap and mprotect are never invoked.

Platforms: Microsoft Windows… The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the `/NXCOMPAT` flag was used during compilation to verify that DEP protections are enabled for the application.

The evaluator used BinScope to verify that the TOE passes NXCheck.

Platforms: Apple iOS… The evaluator shall perform a static analysis on the application to verify that `mprotect` is never invoked with the `PROT_EXEC` permission.

The evaluator examined the TOE's source code and verified that PROT_EXEC and mprotect are never used.

Platforms: Apple macOS… The evaluator shall perform a static analysis on the application to verify that `mprotect` is never invoked with the `PROT_EXEC` permission.

The evaluator examined the TOE's source code for all uses of mprotect and verified that it was never invoked with PROT_EXEC permissions.

2.6.1.3 FPT_AEX_EXT.1.3

2.6.1.3.1 TSS Evaluation Activity

None.

2.6.1.3.2 Guidance Evaluation Activity

None.

2.6.1.3.3 Test Evaluation Activity

The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

Platforms: Android… Applications running on Android cannot disable Android security features, therefore this requirement is met and no evaluation activity is required.

As noted in [PP_APP_v1.4], the Android platform meets this requirement.

Platforms: Microsoft Windows… If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defenderexploit-guard/customize-exploit-protection.

The evaluator examined security configuration settings on a platform running the TOE. No Windows security features had to be turned off in order to install or run the TOE.

If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

The OS platform does not support EMET.

Platforms: Apple iOS… Applications running on iOS cannot disable security features, therefore this requirement is met and no evaluation activity is required.

As noted in [PP_APP_v1.4], the Apple iOS platform meets this requirement.

Platforms: Apple macOS… The evaluator shall ensure that the application can successfully run on macOS without disabling any security features.

There are no significant security features which can be deactivated on MacOS. As the following screenshot shows no security settings had to be altered to install or run the TOE.

2.6.1.4 FPT_AEX_EXT.1.4

2.6.1.4.1 TSS Evaluation Activity

None.

2.6.1.4.2 Guidance Evaluation Activity

None.

2.6.1.4.3 Test Evaluation Activity

The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:

Platforms: Android… The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored under `/data/data/package/` where package is the Java package of the application.

The evaluator confirmed that the /data/data/com.wickrenterprise.debugger directory did not have any executable files in its sub directory.

Platforms: Microsoft Windows… For Windows Universal Applications the evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

The evaluator used acceschk to examine the Program Files directory where Wickr's executable files are located. They verified that there are no user-modifiable files in that directory. The full output of the accesschk tool is in the file below.

Platforms: Apple iOS… The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

As noted in [PP_APP_v1.4], the Apple iOS platform meets this requirement.

Platforms: Apple macOS… The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

An examination of the TOE's application data directories confirmed that the TOE does not have any executable files that have user writeable files in the same directories.

2.6.1.5 FPT_AEX_EXT.1.5

2.6.1.5.1 TSS Evaluation Activity

None.

2.6.1.5.2 Guidance Evaluation Activity

None.

2.6.1.5.3 Test Evaluation Activity

The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

Platforms: Microsoft Windows… Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS and verify that the `/GS` flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of `/GS`.

The TOE's developer showed that /GS was used during TOE compilation.

2.6.2 Use of Supported Services and APIs (FPT_API_EXT.1)

2.6.2.1 TSS Evaluation Activity

The evaluator shall verify that the TSS lists the platform APIs used in the application.

Section 6.7 of [ST] ("Protection of the TSF") references Appendix A.1 of [ST] ("Platform APIs") for the lists of platform APIs used by each platform version of the TOE. The evaluator confirmed Appendix A.1 of [ST] lists platform APIs used by the Android, iOS, macOS, and Windows versions of the TOE.

2.6.2.2 Guidance Evaluation Activity

None.

2.6.2.3 Test Evaluation Activity

The evaluator shall then compare the list with the supported APIs (available through e.g. developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

All APIs used by the TOE were shown to be properly documented and valid.

2.6.3 Use of Third Party Libraries (FPT_LIB_EXT.1)

2.6.3.1 TSS Evaluation Activity

None.

2.6.3.2 Guidance Evaluation Activity

None.

2.6.3.3 Test Evaluation Activity

The evaluator shall install the application and survey its installation directory for dynamic libraries. The evaluator shall verify that libraries found to be packaged with or employed by the application are limited to those in the assignment.

The evaluator verified that all of the libraries included with the TOE were consistent with those documented in the ST.

2.6.4 Software Identification and Versions (FPT_IDV_EXT.1)

2.6.4.1 TSS Evaluation Activity

If **"other version information"** is selected the evaluator shall verify that the TSS contains an explanation of the versioning methodology.

Section 5.2.6 of [ST] ("Protection of the TSF (FPT)") selects "other version information" in FPT_IDV_EXT.1.1 and completes the assignment with "major.minor.release". Section 6.7 of [ST] ("Protection of the TSF") states the TOE can display its current version in major.minor.release format.

2.6.4.2 Guidance Evaluation Activity

None.

2.6.4.3 Test Evaluation Activity

The evaluator shall install the application, then check for the existence of version information. If ***SWID tags*** is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

As described in the ST the TOE does not use SWID tags. The TOE is versioned using a major.minor.release versioning system. FPT_TUD_EXT.1.2 shows that each instance of the TOE displays versioning information in that format.

2.6.5 Integrity for Installation and Update (FPT_TUD_EXT.1)

2.6.5.1 FPT_TUD_EXT.1.1

2.6.5.1.1 TSS Evaluation Activity

None.

2.6.5.1.2 Guidance Evaluation Activity

The evaluator shall check to ensure the guidance includes a description of how updates are performed.

Section "Software Download, Installation and Configuration", subsection "Download and Installation" of [CCECG] also includes a description of how TOE updates are performed.

2.6.5.1.3 Test Evaluation Activity

The evaluator shall check for an update using procedures described in either the application documentation or the platform documentation and verify that the application does not issue an error. If it is updated or if it reports that no update is available this requirement is considered to be met.

Mobile instances of the TOE demonstrated the ability to check for updates through the appropriate platform app store. Desktop instances of the TOE were able to check for updates through the TOE itself.

2.6.5.2 FPT_TUD_EXT.1.2

2.6.5.2.1 TSS Evaluation Activity

None.

2.6.5.2.2 Guidance Evaluation Activity

The evaluator shall verify guidance includes a description of how to query the current version of the application.

Section “Software Download, Installation and Configuration”, subsection “Download and Installation” of [CCECG] also includes a description of how to query the current version of the TOE.

2.6.5.2.3 Test Evaluation Activity

The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

The evaluator demonstrated that the TOE’s user interface displays the version number using the major.minor.release format.

2.6.5.3 FPT_TUD_EXT.1.3

2.6.5.3.1 TSS Evaluation Activity

None.

2.6.5.3.2 Guidance Evaluation Activity

None.

2.6.5.3.3 Test Evaluation Activity

The evaluator shall verify that the application's executable files are not changed by the application.

Platforms: Apple iOS... The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

For all other platforms, the evaluator shall perform the following test:

Test 1: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

For the Android, Windows and MacOS instances of the TOE the evaluator generated hashes of the TOE's executable files before and after running the TOE. None of the executable files were changed by the TOE.

2.6.5.4 FPT_TUD_EXT.1.4

2.6.5.4.1 TSS Evaluation Activity

The evaluator shall verify that the TSS identifies how updates to the application are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.

Section 6.7 of [ST] ("Protection of the TSF") states the vendor (i.e., the authorized source) digitally signs updates for the Android platform version of the TOE using a 4096-bit RSA key and digitally signs updates for the iOS, macOS, and Windows platform versions using a 2048-bit RSA key.

Section 6.7 of [ST] states the TOE platform provides the means to verify and apply TOE updates. The macOS and Windows versions of the TOE periodically check for updates on the vendor's support site. The Android and iOS versions of the TOE perform periodic (hourly) checks for updates to their respective app stores.

2.6.5.4.2 Guidance Evaluation Activity

None.

2.6.5.4.3 Test Evaluation Activity

None.

2.6.5.5 FPT_TUD_EXT.1.5

2.6.5.5.1 TSS Evaluation Activity

The evaluator shall verify that the TSS identifies how the application is distributed. If "***with the platform***" is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If "***as an additional package***" is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

Section 6.7 of [ST] states the TOE is distributed as an additional software package to the platform. Refer to Section 2.6.6 for evaluation activities associated with FPT_TUD_EXT.2.

2.6.5.5.2 Guidance Evaluation Activity

None.

2.6.5.5.3 Test Evaluation Activity

None.

2.6.6 Integrity for Installation and Update (FPT_TUD_EXT.2)

2.6.6.1 FPT_TUD_EXT.2.1

2.6.6.1.1 TSS Evaluation Activity

None.

2.6.6.1.2 Guidance Evaluation Activity

None.

2.6.6.1.3 Test Evaluation Activity

Modified in accordance with TD0628.

<u>If a container image is claimed the evaluator shall verify that application updates are distributed as container images. If the format of the platform-supported package manager is claimed,</u> the evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:

Platforms: Android… The evaluator shall ensure that the application is packaged in the Android application package (APK) format.

The Android instance of the TOE was shown to be distributed as an APK file.

Modified in accordance with TD0628.

Platforms: Microsoft Windows… The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the Windows Universal Application package (.APPX) format. See https://msdn.microsoft.com/en-us/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.

The Windows instance of the TOE was shown to be distributed as a .MSI file.

Modified in accordance with TD0628.

Platforms: Apple iOS… The evaluator shall ensure that the application is packaged in the IPA format.

As shown in FPT_TUD_EXT.1 the TOE is distributed through the Apple app store.

Modified in accordance with TD0628.

Platforms: Apple macOS… The evaluator shall ensure that application is packaged in the DMG format, the PKG format, or the MPKG format.

The MacOS instance of the TOE was shown to be distributed as a .dmg file.

2.6.6.2 FPT_TUD_EXT.2.2

2.6.6.2.1 TSS Evaluation Activity

None.

2.6.6.2.2 Guidance Evaluation Activity

None.

2.6.6.2.3 Test Evaluation Activity

Modified in accordance with TD0664.

Platforms: Android… The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

As noted in [PP_APP_v1.4], the Android platform meets this requirement.

~~**Platforms: Microsoft Windows…** The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.~~

Platforms: Apple iOS… The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

As noted in [PP_APP_v1.4], the Apple iOS platform meets this requirement.

~~**Platforms: Apple macOS…** The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.~~

All Other Platforms… The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.

Windows: Before installing the TOE, the evaluator recorded the path of every file on the platform filesystem. The evaluator then installed the TOE, rebooted the system, and logged in to the TOE. The evaluator then uninstalled the TOE and compared the resulting filesystem to the initial record. The evaluator confirmed …

MacOS: Before installing the TOE, the evaluator recorded the path of every file on the platform filesystem. The evaluator then installed the TOE, rebooted the system, and logged in to the TOE. The evaluator then uninstalled the TOE and compared the resulting filesystem to the initial record. The evaluator confirmed …

2.6.6.3 FPT_TUD_EXT.2.3

2.6.6.3.1 TSS Evaluation Activity

The evaluator shall verify that the TSS identifies how the application installation package is signed by an authorized source. The definition of an authorized source must be contained in the TSS.

Section 6.7 of [ST] ("Protection of the TSF") states the vendor (i.e., the authorized source) digitally signs installation packages for the Android platform version of the TOE using a 4096-bit RSA key and digitally signs installation packages for the iOS, macOS, and Windows platform versions using a 2048-bit RSA key.

2.6.6.3.2 Guidance Evaluation Activity

None.

2.6.6.3.3 Test Evaluation Activity

None.

2.7 Trusted Path/Channels (FTP)

2.7.1 Protection of Data in Transit (FTP_DIT_EXT.1)

2.7.1.1 TSS Evaluation Activity

For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.

Section 5.2.7 of [ST] ("Trusted Path/Channels (FTP)") specifies the application shall invoke platform-provided functionality to encrypt all transmitted data with TLS. Section 6.8 of [ST] ("Trusted Path/Channels") states the TOE uses platform-provided TLS for authentication and messaging communication.

Section 6.8 of [ST] states the macOS and Windows versions of the TOE invoke the following platform TLS interfaces through calls made by the Qt library included with the TOE:

- macOS: Apple Secure Transport
- Windows: Schannel

The iOS version of the TOE calls the URLSession API, which implements cryptographic functionality using the CoreCrypto library. The Android platform version of the TOE uses the okhttp third-party library, which makes calls to the platform BoringSSL for protocol and cryptographic functionality.

2.7.1.2 Guidance Evaluation Activity

None.

2.7.1.3 Test Evaluation Activity

The evaluator shall perform the following tests:

Test 1: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS, TLS, DTLS, SSH, or IPsec in accordance with the selection in the ST.

The evaluator examined all traffic going to and from the TOE when it was opening a connection to a Wickr server. TLS was used to protect data going to and from the server.

Test 2: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.

This test was run in conjunction with FTP_DIT_EXT.1 Test 1. That test shows that all data going between the TOE and a Wickr server is encrypted by TLS and not transmitted in the clear.

Test 3: The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

This test was run in conjunction with FTP_DIT_EXT.1 Test 1. That test shows that all data going between the TOE and a Wickr server is encrypted by TLS and not transmitted in the clear. There is no plaintext data being sent where credentials could be found.

Platforms: Android… If "***not transmit any data***" is selected, the evaluator shall ensure that the application's `AndroidManifest.xml` file does not contain a `uses-permission` or `uses-permission-sdk-23` tag containing `android:name="android.permission.INTERNET"`. In this case, it is not necessary to perform the above Tests 1, 2, or 3, as the platform will not allow the application to perform any network communication.

The ST does not select “not transmit any data”. The evaluator performed Tests 1, 2, and 3 on the Android version of the TOE.

Platforms: Apple iOS… If "***encrypt all transmitted data***" is selected, the evaluator shall ensure that the application's `Info.plist` file does not contain the `NSAllowsArbitraryLoads` or `NSExceptionAllowsInsecureHTTPLoads` keys, as these keys disable iOS's Application Transport Security feature.

The ST does not select “encrypt all transmitted data”. Therefore, this activity is not applicable to the Apple iOS version of the TOE.

3. Security Assurance Requirement Assurance Activities

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

3.1.1.1 Assurance Activity

There are no specific assurance activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

The Assurance Activities identified above provided sufficient information to determine the appropriate content for the TSS section and to perform the assurance activities. Since these are directly associated with the SFRs, and are implicitly already done, no additional documentation or analysis is necessary.

3.2 Guidance Documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

3.2.1.1 Assurance Activity

Some of the contents of the operational guidance will be verified by the assurance activities in Section 5.1 and evaluation of the TOE according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE. The documentation must describe the process for verifying updates to the TOE by verifying a digital signature – this may be done by the TOE or the underlying platform. The evaluator shall verify that this process includes the following steps: Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory). Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature. The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

The Android, Windows, iOS and macOS versions of the TOE provide cryptographic functions implemented in Wickr OpenSSL version 2.0.16. Section "TOE Description", sub-section "Evaluated Configuration" of [CCECG] states the TOE uses platform-provided cryptographic services for protection of locally-stored credential data. No configuration is required to activate these services.

The guidance provided by [CCECG] includes a description of how updates are performed. The description covers how to obtain the update and make it accessible to the TOE and how to initiate the update process. Refer to Section "Software Download, Installation and Configuration". The update process is the same for all TOE versions. When the TOE advises the user an update is available, the user can click a button labeled "Update" that launches the installer on the platform. The process to install an update is the same as the process for a fresh installation. The platform installer verifies the digital signature on the upgrade package as part of the upgrade process. The user can discern the success or otherwise of an upgrade attempt by viewing the running TOE version on the main menu page of the user interface.

Section “Logical Boundaries” of [CCECG] describes the security functionality of the TOE that falls within the scope of evaluation, while section “Excluded from the Evaluation” lists specific TOE functionality not covered by the evaluation.

3.2.2 Preparative Procedures (AGD_PRE.1)

3.2.2.1 Assurance Activity

As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

The TOE in its evaluated configuration is supported on four platforms (Android, Windows, iOS, macOS) that are adequately addressed in the guidance documentation. Section “Physical Boundaries” of [CCECG] states the TOE consists of the Wickr Enterprise Client application and list the supported platform versions and identifies the specific platforms on which the TOE is evaluated.

3.3 Tests (ATE)

3.3.1 Independent Testing – Conformance (ATE_IND.1)

3.3.1.1 Assurance Activity

The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP’s Assurance Activities.

While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS, SSH). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a “fail” and “pass” result (and the supporting details), and not just the “pass” result.

The TOE was tested at Leidos’s Columbia, MD location from November 2022 to March 2023. The procedures and results of this testing are available in the DTR document.

3.4 Vulnerability Assessment (AVA)

3.4.1 Vulnerability Survey (AVA_VAN.1)

3.4.1.1 Assurance Activity

The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious.

The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

For Windows, Linux, macOS and Solaris: The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious.

The evaluation team performed a search of the National Vulnerability Database (https://nvd.nist.gov/).

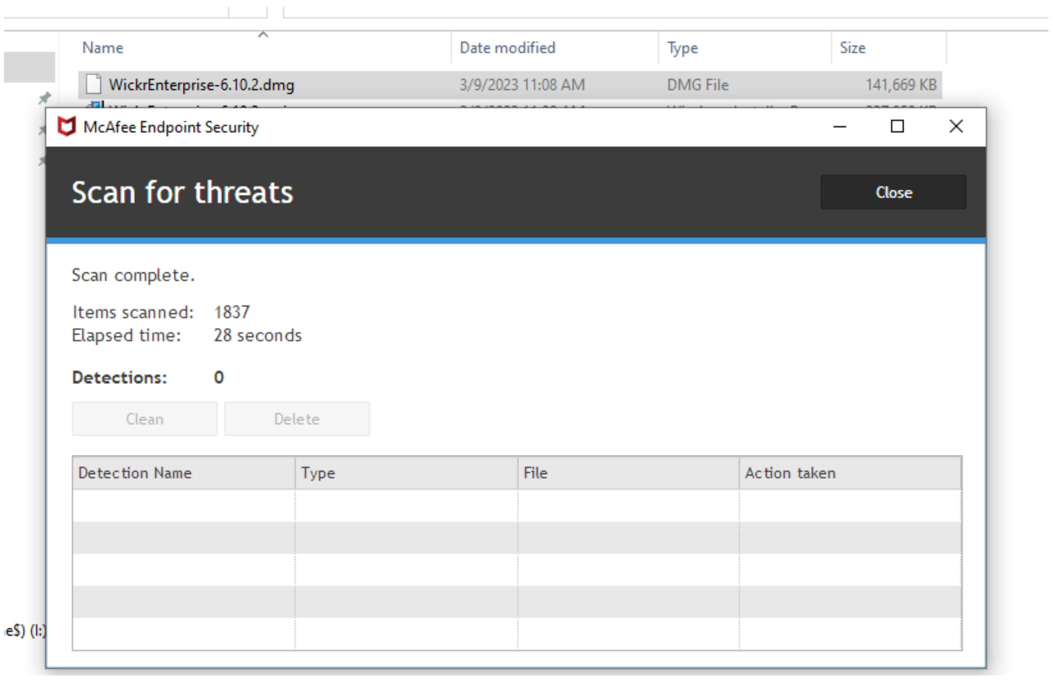
The evaluation team performed searches on 2/23/2023 and repeated the searches on 4/05/2023, using the following search terms:

- Wickr
- Encrypted Service
- zero trust
- OpenSSL 1.0.2zg
- OpenSSL 2.0.16
- symcrypt
- Apple coreCrypto Module
- BoringSSL
- Third Party Libraries identified in Section A.2 of the Security Target

No vulnerabilities were identified for the TOE.

The evaluator ran a virus scan with up to date virus definitions against the Windows and macOS TOE executables and verified that no files were flagged as malicious.

macOS Virus Scan



Name	Date modified	Type	Size
WickrEnterprise-6.10.2.dmg	3/9/2023 11:08 AM	DMG File	141,669 KB

McAfee Endpoint Security

Scan for threats

Close

Scan complete.

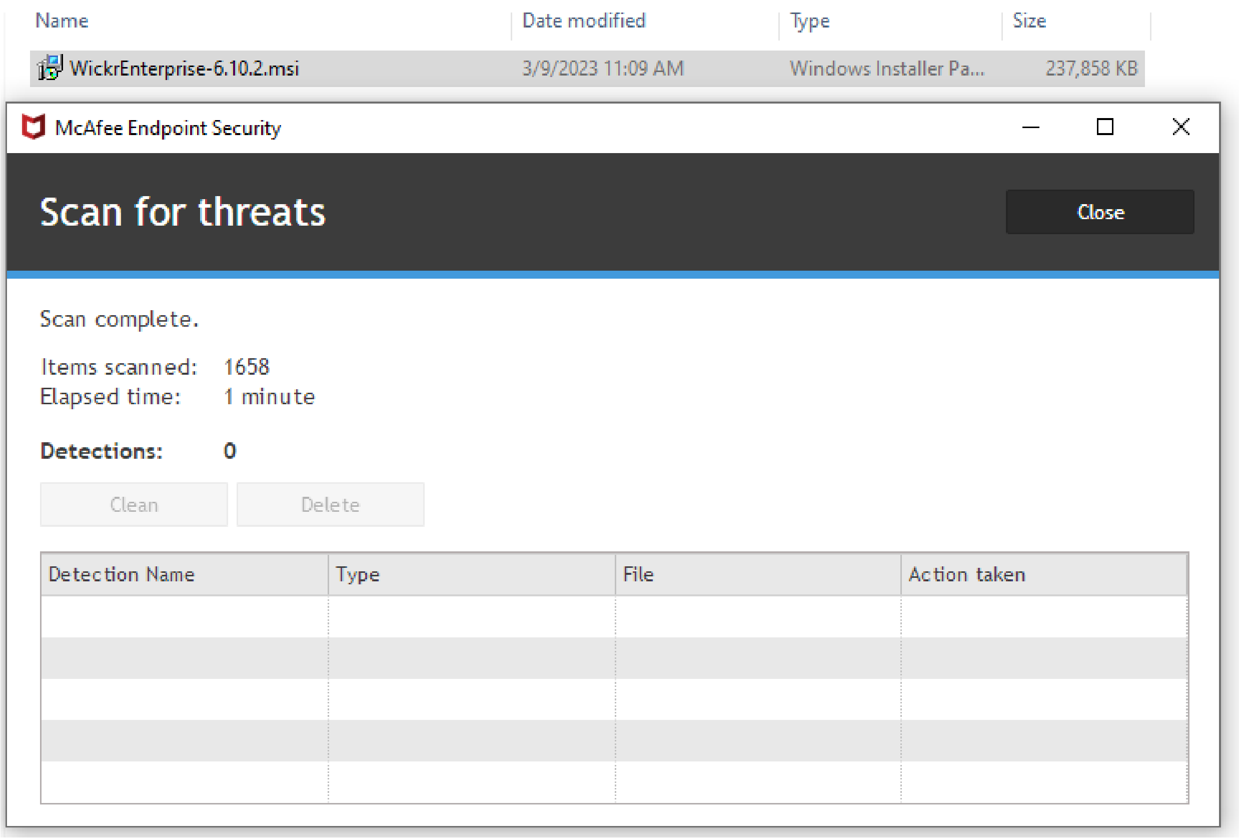
Items scanned: 1837
Elapsed time: 28 seconds

Detections: 0

Clean Delete

Detection Name	Type	File	Action taken

Windows Virus Scan



Name	Date modified	Type	Size
WickrEnterprise-6.10.2.msi	3/9/2023 11:09 AM	Windows Installer Pa...	237,858 KB

McAfee Endpoint Security

Scan for threats

Scan complete.

Items scanned: 1658
Elapsed time: 1 minute

Detections: 0

Detection Name	Type	File	Action taken

The evaluation team determined that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.

3.5 Life-Cycle Support (ALC)

3.5.1 Labeling of the TOE (ALC_CMC.1)

3.5.1.1 Assurance Activity

The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Section 1.1 of [ST] ("Security Target, TOE and CC Identification") includes the TOE identification. The TOE is identified in terms of the software included in the evaluated configuration.

Wickr Enterprise Client 6.10. The platform-specific versions of the TOE include:

- Wickr Enterprise Client for Windows 6.10.2
- Wickr Enterprise Client for macOS 6.10.2
- Wickr Enterprise Client for iOS 6.10.0

- Wickr Enterprise Client for Android 6.10.0

The TOE version is consistent with the version number of the TOE identified in [CCECG] and the version identified by the TOE sample received for testing.

3.5.2 TOE Coverage (ALC_CMS.1)

3.5.2.1 Assurance Activity

The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

The evaluator shall ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

As described in Section 3.5.1 above, the evaluator confirmed the TOE is labelled with unique software version identifiers. Section 6.7 of [ST] ("Protection of the TSF") describes how each TOE version uses security features and APIs provided by its platform. This includes data execution protection, stack-based buffer overflow protection, and compatibility with platform security features.

3.5.3 Timely Security Update (ALC_TSU_EXT.1)

3.5.3.1 Assurance Activity

The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall verify that this description addresses the entire application. The evaluator shall also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days.

The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

Section 6.1 of [ST] ("Timely Security Updates") describes the timely security update process used by the developer to create and deploy TOE security updates. The description encompasses the entirety of the TOE.

Wickr normally provides releases on a quarterly basis. Bugs may result in additional releases on accelerated schedules. The releases include bug fixes and security updates for all platform versions of the TOE. Additionally, when updates are made to bundled third-party capabilities, they are obtained by Wickr and included in releases. Wickr support personnel contact the POCs for affected customers. The only mechanism to deploy security updates is through maintenance releases. Upon discovery of a vulnerability, the impact will be assessed for priority based on the severity of the bug. The target timeline for releases ranges from 48 hours for critical bugs to 90 days for low severity bugs. Security reports are communicated from customers to Customer Support through an HTTPS form on the HackerOne platform.