Assurance Activities Report

for

Hypori Halo Client (iOS) 4.3

Version 1.0

February 21, 2024

Prepared by:

Leidos Inc.
https://www.leidos.com/CC-FIPS140
Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive
Columbia, MD 21046

Prepared for:

Hypori, Inc.
1801 Robert Fulton Drive, Suite 440
Reston, VA 20191

The Developer of the TOE:

Hypori, Inc.
1801 Robert Fulton Drive, Suite 440
Reston, VA 20191

The TOE Evaluation was Sponsored by:

Hypori, Inc.
1801 Robert Fulton Drive, Suite 440
Reston, VA 20191

Evaluation Personnel:

Dawn Campbell
Josh Marciante
Pascal Patin
Allen Sant

Contents

1 Introduction

This document presents results from performing assurance activities associated with the evaluation of Hypori Halo Client (iOS) 4.3. This report contains sections documenting the performance of assurance activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in the Evaluation Activities for the Protection Profile for Application Software, Version 1.4, 2021-10-07 [App PP].

1.1 Evidence

[App PP]	Protection Profile for Application Software, Version 1.4, 2021-10-07
[ST]	Hypori Halo Client (iOS) 4.3 Security Target, Version 1.0, 15 February 2024
[CCCO]	Hypori User Guide Common Criteria Configuration and Operation, Version 4.3
[ADMIN]	Hypori Halo Administrator Guide, Guide Version 1.18 (Supplementary guide for Hypori Server)

1.2 Conformance Claims

Common Criteria Versions

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, dated: April 2017.
- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, Version 3.1, Revision 5, dated: April 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Version 3.1, Revision 5, dated: April 2017.

Common Evaluation Methodology Versions

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

1.3 CAVP/ACVP Certificates

The TOE is supported on iOS 15 and 16. The TOE does not implement any cryptographic algorithms; however it does rely on its Android platform for cryptographic functionality, specifically for its implementation of TLS 1.2 (FTP_DIT_EXT.1 and by extension FCS_CKM_EXT.1, FCS_CKM.1/AK, FCS_CKM.2), which is provided by the cryptomodules within iOS. The following iOS evaluations, which cover all of the platforms claimed in the evaluated configuration, are conformant to the Common Criteria for IT Security Evaluation (ISO Standard 15408) and are listed on the NIAP Product Compliant List (PCL):

- Apple iOS 15: VID11237: https://www.niap-ccevs.org/MMO/ProductAM/st_vid11237-st.pdf
    - https://www.niap-ccevs.org/Product/Maint.cfm?AMID=1542&PID=11237
    - Apple iOS 15: iPhones, Update from v15.1.0 to v15.7.1
    - cryptomodules:

        - USR: Apple corecrypto Module v12.0 [Apple ARM, User, Software, SL1] (User Space)
        - KRN: Apple corecrypto Module v12.0 [Apple ARM, Kernel, Software, SL1] (Kernel Space)
        - SKS: Apple corecrypto Module v12.0 [Apple ARM, Secure Key Store, Hardware, SL2]
        - SEP: Secure Enclave Processor (SEP) Hardware v2.0
    - Operating System—iOS 15.0 (tested using iOS 15.7.1
    - Processor— iPhone SE A1723 (A9 Processor)
    - Validation Report Number: CCEVS-VR-VID11237-2022
    - Certificate Date: 2022.11.30
- Apple iOS 16: 11349: https://www.niap-ccevs.org/MMO/Product/st_vid11349-st.pdf
    - https://www.niap-ccevs.org/Product/Compliant.cfm?PID=11349
    - Apple iOS 16: iPhone
    - Cryptomodules (CAVP Module Implementation Names as drawn from Appendix A.3):
        - USR: Apple corecrypto Module v13.0 [Apple ARM, User, Software, SL1] (User Space)
        - KRN: Apple corecrypto Module v13.0 [Apple ARM, Kernel, Software, SL1] (Kernel Space)
        - SKS-FW: Apple corecrypto Module v13.0 [Apple ARM, Secure Key Store, Hardware, SL2] (SKS)
        - SKS-HW: Apple corecrypto Module v2.0 [A11 Bionic]:
            - Apple Secure Enclave Processor Hardware DRBG (Apple A11)
    - Operating System—iOS 16 (tested version 16.3)
    - Processor— A11 Bionic
    - Validation Report Number: CCEVS-VR-VID11349-2023
    - Certificate Date: 2023.10.10

Each iOS evaluations on the PCL, as identified above, demonstrates the included libraries have the necessary cryptographic functions and by extension, the CAVPs. The evaluator verified the platform-provided cryptography satisfies the cryptographic requirements identified in the TOE ST, using primarily method #1 in #10 Frequently Asked Question in Policy Letter 5 Addendum 1 that states, "If the platform has been evaluated and is on the NIAP Product Compliant List (PCL), the evaluator may rely on the Security Target of the evaluated platform to verify the functionality was evaluated." The addendum further states that the previously certified evaluation's ST and screen shot evidence of the previously evaluated ST showing how the new evaluation's cryptographic requirements are met must be provided in the ETR. This evidence is provided in the proprietary ETR. The CAVPs satisfying the SFRs in the TOE's ST are as follows: FTP_DIT_EXT.1 (A3428, A2788, A2797, A2848, A2786, A3426, A2845, A3686, A4109, A4106); FCS_CKM_EXT.1 / FCS_CKM.1/AK (A3428, A2788, A2797, A2848, A2786, A3426, A3686, A4109); and FCS_CKM.2 (A3426, A2786, A2845, A4106).

1.4 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 Assurance Activities

This section describes the assurance activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The assurance activities are derived from the [App PP] and modified by applicable NIAP Technical Decisions. Assurance activities for SFRs not claimed by the TOE have been omitted.

Evaluator notes, such as changes made due to NIAP Technical Decisions, are in bold text. Bold text is also used within assurance activities to identify when they are mapped to individual SFR elements rather than the component level.

2.1 Cryptographic Support (FCS)

2.1.1 FCS_CKM_EXT.1[1] Cryptographic Key Generation Services

2.1.1.1 TSS Assurance 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 examined the application and its associated developer documentation and determined the TOE requires asymmetric key generation services, since it uses cryptographic protocols for communication with external IT entities.

Section 6.1.1 of [ST] (FCS_CKM_EXT.1") states the TOE requires asymmetric key generation services to provide secure communications to the Virtual Device on the Hypori Server. The Hypori Halo Client invokes platform-provided functionality for asymmetric key generation. As part of installation, a user adds a TLS client certificate and the RSA or Elliptic Curve key pairs to the platform's key store. The platform generates all ephemeral TLS keys without direct Hypori Client action. Section 5.2.1.1 of [ST] ("FCS_CKM_EXT.1 Cryptographic Key Generation Services") specifies the TOE invokes platform-provided functions for asymmetric key generation. The evaluation activities have been performed as stated in the selection-based requirements.

2.1.1.2 Guidance Assurance Activity

None.

2.1.1.3 Test Assurance Activity

None.

2.1.2 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation

2.1.2.1 TSS Assurance Activities

The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies

[1] Modified by TD0717

more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

[ST] Section 6.1.2 (“FCS_CKM.1/AK”) and 6.1.3 (“FCS_CKM.2”) identify the supported key sizes for establishing communications to the Hypori server as P-256, P-384 Elliptic Curve keys and RSA 2048, 3072, and 4096 keys. Section 6.1 indicates the Hypori Client uses platform TLS services for secure communication with the Hypori Virtual Device on the Hypori server.

If the application “invokes platform-provided functionality for asymmetric key generation,” then the evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.

Section 6.1.1 of [ST] (“FCS_CKM_EXT.1”) states the TOE requires asymmetric key generation services to provide secure communications to the Hypori Server. Section 6.1.2 (“FCS_CKM.1/AK”) states that the TOE calls the iOS API to create a dictionary from which the RSA and ECC cryptographic key pairs are defined. The same platform API is used for both RSA and ECC key generation. Section 6.7.1 of [ST] (“FTP_DIT_EXT.1”) describes the API calls used by the TOE to invoke the platform-provided functionality and Section 6.1.2 of [ST] (“FCS_CKM.1/AK”) states the iOS platforms call the Apple corecrypto Module libraries for the platform to create the ECC and RSA keys.

2.1.2.2 Guidance Assurance Activity

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

[CCCO] Section 3 states that ciphersuites are determined by choice of Android, iOS, or Windows version, not the Hypori Client configuration and that no configuration is required to use the supported cryptographic algorithms and key strengths.

2.1.2.3 Test Assurance Activities

If the application “implements asymmetric key generation,” then the following test activities shall be carried out.

Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:
   - Provable primes
   - Probable primes

2. Primes with Conditions:
   - Primes p1, p2, q1,q2, p and q shall all be provable primes
   - Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

  - Primes p1, p2, q1,q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- n = p·q,
- p and q are probably prime according to Miller-Rabin tests,
- GCD(p-1,e) = 1,
- GCD(q-1,e) = 1,
- $2^{16} \le e \le 2^{256}$ and e is an odd integer,
- $|p-q| > 2^{nlen/2 - 100}$,
- $p \ge 2^{nlen/2 -1/2}$,
- $q \ge 2^{nlen/2 -1/2}$,
- $2^{(nlen/2)} < d < LCM(p-1,q-1)$,
- e·d = 1 mod LCM(p-1,q-1).

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P-384 and P521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x:
Private Key:

- len(q) bit output of RBG where 1 ≤x ≤ q-1
- len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1≤ x≤q-1.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- $g \neq 0,1$
- q divides p-1
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

for each FFC parameter set and key pair.

The application does not implement asymmetric key generation, therefore the assurance activity is not applicable.

2.1.3 FCS_CKM.2 Cryptographic Key Establishment

2.1.3.1 TSS Assurance Activity

Modified by TD0717
The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in **FCS_CKM.1.1/AK**. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

[ST] Section 6.1.3 states that the TOE invokes platform provided RSA and ECC key establishment schemes for establishing communications to the Hypori server. These selections in FCS_CKM.2.1 correspond with RSA and ECC key generation selections in FCS_CKM.1.1/AK.

2.1.3.2 Guidance Assurance Activity

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

[CCCO] Section 3 states that the ciphersuites are determined by choice of Android, iOS, or Windows version, not the Hypori Client configuration, and that no configuration is required to use the supported cryptographic algorithms and key strengths.

2.1.3.3 Test Assurance Activities

Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE’s public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information (OtherInfo) and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF’s implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party’s valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator’s public keys, the TOE’s public/private key pairs, MACTag, and any inputs used in the KDF, such as the OtherInfo and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the OtherInfo field, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties’ static public keys, both parties’ ephemeral public keys and the TOE’s static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

The TOE uses platform-provided CAVP-certified libraries. See Section 1.3 for the applicable ACVP certificates.

2.1.4 FCS_RBG_EXT.1 Random Bit Generation Services

2.1.4.1 TSS Assurance Activities

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.

In FCS_RBG_EXT.1, the ST author has selected use no DRBG functionality. The evaluator inspected the application and developer documentation and confirmed the TOE itself does not use any 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.

[ST] In FCS_RBG_EXT.1, the ST author has not selected *implement DRBG functionality*. Therefore this is not applicable and the ST has accurately not included the FCS_RBG_EXT.2 elements.

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.

[ST] "invoke platform-provided DRBG functionality**"** is not selected In FCS_RBG_EXT.1, therefore this activity is not applicable.

2.1.4.2 Guidance Assurance Activity

None.

2.1.4.3 Test Assurance 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.

Linux: The evaluator shall verify that the application collects random from /dev/random or /dev/urandom.

Oracle Solaris: The evaluator shall verify that the application collects random from /dev/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).

“invoke platform-provided DRBG functionality” is not selected In FCS_RBG_EXT.1, therefore this activity is not applicable.

2.1.5 FCS_STO_EXT.1 Storage of Credentials

2.1.5.1 TSS Assurance 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.

In Section 6.1.5 of [ST] (“FCS_STO_EXT.1”), Table 8 (“Persistent Credential Use and Storage”) states the TOE stores the following credential in the iOS Keychain:

- User TLS client private key—used for TLS mutual authentication.

2.1.5.2 Guidance Assurance Activity

None.

2.1.5.3 Test Assurance Activity

Modified by TD0717

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_EXT.1/PBKDF**.

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

Android: The evaluator shall verify that the application uses the Android KeyStore or the Android

KeyChain to store certificates.

Microsoft 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.

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

Linux: The evaluator shall verify that all keys are stored using Linux keyrings.

Oracle Solaris: The evaluator shall verify that all keys are stored using Solaris Key Management Framework (KMF).

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

The evaluator verified via static analysis that the TOE stored credentials within a Keychain.

2.2 User Data Protection (FDP)

2.2.1 FDP_DAR_EXT.1 Encryption of Sensitive Application Data

2.2.1.1 TSS Assurance 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.

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.

Section 6.2.1 of [ST] (“FDP_DAR_EXT.1”) states the sensitive data processed by the TOE consists of the User TLS client private key. This is consistent with the selection in FDP_DAR_EXT.1 of “***protect sensitive data in accordance with FCS_STO_EXT.1.***” Since FDP_DAR_EXT.1.1 does not select “not store any sensitive data” this part is not applicable.

2.2.1.2 Guidance Assurance Activity

None.

2.2.1.3 Test Assurance Activity

Modified by TD0756:

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.

If "implement functionality to encrypt sensitive data as defined in the PP-Module for File Encryption" or "protect sensitive data in accordance with FCS_STO_EXT.1" is selected, 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.

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.

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.

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.

Linux: The Linux 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.

Oracle Solaris: The Solaris 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.

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 only sensitive data is the user TLS client private key covered by FCS_STO_EXT.1, therefore, this activity is not applicable.

2.2.2 FDP_DEC_EXT.1 Access to Platform Resources

2.2.2.1 TSS Assurance Activity

FDP_DEC_EXT.1.1 and FDP_DEC_EXT.1.2

None.

2.2.2.2 Guidance Assurance Activities

FDP_DEC_EXT.1.1

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.

The evaluator examined the list of system resources identified in [CCCO] Section 4.2 and verified it is consistent with those indicated in the selections.

The FDP_DEC_EXT.1.1 claim in [ST] section 5.2.2.2 (“Access to Platform Resources (FDP_DEC_EXT.1)”) selects the following platform hardware resources that the TOE requests permission to access: network connectivity; camera; microphone; location services, fingerprint scanner.

Chapter 1 of [CCCO] (“Introduction and System Overview”) identifies the purpose of the TOE is to enable users to connect their physical mobile device to the virtual mobile device (“Virtual Device”) on the Hypori server, thus justifying the TOE’s need to access network connectivity. Section 4.2 of [CCCO] (“iOS Permissions”) lists the permissions the TOE requests to access other required resources, as follows:

- Network connectivity—section 4.2.8 of [CCCO] (“Cellular Data”) states the TOE uses the mobile device’s permission to allow the user to set the client to only use Wi-Fi by turning this feature off. If enabled the Hypori client will use either Wi-Fi or cellular data.
- Camera—section 4.2.2 of [CCCO] (“Take pictures and videos (Camera)”) states the TOE uses remote access to the device’s camera to support multimedia applications that use the camera in the Virtual Device. It can also use the camera when scanning a QR code during account provisioning. The user is prompted for access to the camera when the application is first started. Section 4.2.7 (“FaceID/TouchID”) states the TOE uses the mobile device’s permission to support biometric scanners as an additional option for authentication. Both the server and the client device have to enable this option for it to be usable. FaceID uses the camera.
- Microphone—section 4.2.4 of [CCCO] (“Record audio”) states the TOE provides access to the device’s microphone to support apps in the Virtual Device that require audio input.
- Location service—section 4.2.3 of [CCCO] (“GPS and network-based location”) states the TOE provides access to the GPS sensors and the Wi-Fi location services of the mobile device for authentication with the Hypori server and for apps in the Virtual Device that require these services.
- Fingerprint scanner – Section 4.2.7 of [CCCO] (“FaceID/TouchID”) states the TOE uses the mobile device’s permission to support biometric scanners as an additional option for authentication. Both the server and the client device have to enable this option for it to be usable. TouchID uses the fingerprint scanner.

The evaluator examined the list of system resources identified in [CCCO] Section 4.2 and verified it is consistent with those indicated in the selections. Subsequent subsections of 4.2 provide justification for why the TOE needs access to these resources. The [CCCO] lists several other permissions; however the PP states that all resources do not need to be identified in the ST if they are considered ordinarily used by any application such as central processing units, main memory, displays, input devices; and the Selections should be expressed in a manner consistent with how the application expresses its access needs to the underlying platform. Based on the descriptions provided, the evaluator determined that the remaining permissions are not accessing any hardware other than memory or display, which fall into the ordinarily used category as described in the PP App Note. As such, these other permissions are not and do not need to be listed in the SFR.

FDP_DEC_EXT.1.2

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.

The evaluator reviewed the guidance documentation and did not identify any requirement for the TOE to access sensitive information repositories on the platform, consistent with the selection in FDP_DEC_EXT.1.2. This is consistent with the selection in FDP_DEC_EXT.1.2 in Section 5.2.2.2 of [ST] (“Access to Platform Resources (FDP_DEC_EXT.1)”) of “no sensitive information repositories”.

2.2.2.3 Test Assurance Activities

FDP_DEC_EXT.1.1

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.

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.

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

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

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

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

The evaluator checked the documentation provided by the vendor to ensure that there was a list of hardware resources the application uses. This can be found in section 4.2 of [CCCO].

FDP_DEC_EXT.1.2

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.

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

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

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

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

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

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

The evaluator checked the guidance documentation provided by the vendor and confirmed it does not identify any sensitive information repositories on the platform that the TOE would need to access, consistent with the selection in FDP_DEC_EXT.1.2.

2.2.3 FDP_NET_EXT.1 Network Communications

2.2.3.1 TSS Assurance Activity

None.

2.2.3.2 Guidance Assurance Activity

None.

2.2.3.3 Test Assurance Activities

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 opened the application and attempted to access the backend Hypori Services. When the TOE application reached out to the backend services, the traffic was secured and encrypted ensuring that no plaintext was sent/received.

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).

After the application was opened, an Nmap scan was performed against the iOS devices to detect for open ports. The scan for TCP identified that only ports known to be used by iOS itself were open; the TOE opened no ports itself. The scan for UDP identified all ports as “open|filtered” (meaning nmap could not determine if a port was open or filtered, which occurs for scan types in which ports give no response), or “closed”. The third selection was not selected, so these results match the expected result of no open ports at the behest of the TOE.

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 usespermission-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 TOE does not run on Android platforms and therefore this activity is not applicable, and the evaluation team performed Tests 1 and 2 as specified above.

2.3 Identification and Authentication (FIA)

2.3.1 X.509 Certificate Validation (FIA_X509_EXT.1)

2.3.1.1 TSS Assurance Activity

FIA_X509_EXT.1.1
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 6.3.1 of [ST] ("FIA_X09_EXT.1") states the iOS platform performs certificate path validation in accordance with RFC 5280 as part of the TLS service. It recursively builds certificate chains until a valid chain is found or all possible paths are exhausted. The chain begins at the leaf certificate and ends in the final trusted root certificate. The certificate path validation algorithm is described in an Apple Tech Note, available here:

https://developer.apple.com/library/content/technotes/tn2232/_index.html#//apple_ref/doc/uid/DTS40012884-CH1-SECTRUSTEVALUATIONFUNDAMENTALS.

FIA_X509_EXT.1.2
None.

2.3.1.2 Guidance Assurance Activity

FIA_X509_EXT.1.1 and FIA_X509_EXT.1.2
None.

2.3.1.3 Test Assurance Activities

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

FIA_X509_EXT.1.1
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 presented the TOE with several certificate chains made up of four certificates: a trusted root CA, an intermediate CA signed by the root CA, a subordinate CA signed by the intermediate CA, and a leaf certificate signed by the subordinate CA. Each chain had been modified to exhibit one of the following problems in turn:

- The subordinate CA omitted the Basic Constraints extension but signed the leaf certificate.
- The subordinate CA contained the Basic Constraints extension and had the CA flag set to False but signed the leaf certificate. (This test case also serves as the test case for when "one of the issuing certificates is not a CA certificate", because the CA flag being set to False implicitly categorizes the certificate as an End Entity certificate, which is not a CA certificate).
- The subordinate CA omitted the CA signing bit but signed the leaf certificate.
- The intermediate CA signed the subordinate CA while having a path length of 0. The subordinate CA then signed the leaf certificate.

In all the previously described cases the TOE rejected the certificate paths. The evaluator then presented the TOE with a fifth certificate chain consisting of a trusted root CA, a valid intermediate CA signed by the root CA, a valid subordinate CA signed by the intermediate CA, and a valid leaf certificate signed by the subordinate CA. The TOE accepted this certificate path. The evaluator then omitted the intermediate CA from the server's certificate file, preventing it from advertising the necessary chain of intermediate CAs to complete the path to the trusted root CA for the client. The TOE then rejected the certificate chain, even though the certificates had not changed.

FIA_X509_EXT.1.1

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

The evaluator verified the TOE rejects an expired certificate.

FIA_X509_EXT.1.1

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 RFC6066 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.

The TOE was tested against OCSP as indicated in the selection. The evaluator verified that a revoked certificate both at the node and intermediate CA level resulted in the function failing.

FIA_X509_EXT.1.1

Modified by TD0780:

Test 4: If any OCSP option is selected, the evaluator **shall configure the TSF to reject certificates if it cannot access valid status information, if so configurable. Then 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. 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.**

The TSF contained no configurable behavior when valid status information cannot be retrieved. The evaluator ensured that the TOE only had the ability to determine revocation information via OCSP by generating a certificate chain which contained only OCSP Distribution Points and no other revocation information sources. The evaluator configured an OCSP responder server throughout the evaluation to issue OCSP responses with a 5-minute time-to-live. The evaluator thus waited five minutes to ensure any previous responses expired. The evaluator then configured the OCSP responder to sign its OCSP responses with a certificate issued by the CA whose certificates were being checked for validity that did not contain the OCSP Signing key usage bit. The evaluator verified that the TOE rejected the OCSP response. The evaluator verified that, on iOS 16, the connection was terminated.

Note: The TOE on iOS 15 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.

FIA_X509_EXT.1.1

Test 5: The evaluator shall 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 verified that the TOE terminated a connection after receiving a certificate with its first eight bytes modified.

FIA_X509_EXT.1.1

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 verified that the TOE terminated a connection after receiving a certificate with its last byte modified.

FIA_X509_EXT.1.1

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 verified that the TOE terminated a connection after receiving a certificate with its public key modified.

FIA_X509_EXT.1.1

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.

The TOE does not claim support for EC certificates in FCS_COP.1/Sig, thus this test is not applicable.

FIA_X509_EXT.1.1

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 8 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 8, 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.

The TOE does not claim support for EC certificates in FCS_COP.1/Sig, thus this test is not applicable.

FIA_X509_EXT.1.2

The tests described must be performed in conjunction with the other certificate services assurance 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.

The evaluator configured a certificate chain of four certificates in the test for FIA_X509_EXT.1.1 Test 1 in which one of the intermediate CAs omitted the Basic Constraints extension. The evaluator verified that the TOE failed to validate the path when performing validation of the peer certificate.

FIA_X509_EXT.1.2

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.

The evaluator configured a certificate chain of four certificates in the test for FIA_X509_EXT.1.1 Test 1 in which one of the intermediate CAs had its Basic Constraints set to CA=False. The evaluator verified that the TOE failed to validate the path when performing validation of the peer certificate.

2.3.2 X.509 Certificate Authentication (FIA_X509_EXT.2)

2.3.2.1 TSS Assurance Activity

FIA_X509_EXT.2.1

The evaluator shall examine 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 6.3.2 of [ST] ("FIA_X509_EXT.2") describes how the server and user's certificates are obtained and used. The certificates are stored in the platform's key store during initial configuration.

The Hypori Client presents the TLS client certificate to the Hypori server to authenticate a TLS connection. The Hypori Client uses iOS platform certificate path validation services with the server's CA certificate to validate the certificate presented by the Hypori server.

[CCCO] Section 7.2, and subsection ("*Importing a Server CA Certificate"*) describes how to obtain the server CA and user certificates.

FIA_X509_EXT.2.1

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 6.3.2 of [ST] states the TOE uses platform certificate path validation services with the CA certificate to validate the certificate presented by the Hypori server. The TOE relies on the iOS platform for certificate validation services using OCSP and is configured to fail the connection if the certificate has been revoked or the connection to the OCSP responder fails.

2.3.2.2 Guidance Assurance Activity

FIA_X509_EXT.2.1

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 FIA_X509_EXT.2.2 claim in [ST] section 5.2.7.2 ("X.509 Certificate Authentication (FIA_X509_EXT.2)") 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.

2.3.2.3 Test Assurance Activities

FIA_X509_EXT.2.1

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.

The first part of this test was covered by FIA_X509_EXT.1.1 Test 3 when testing a non-revoked chain. In that test the TOE performed certificate validation by, in part, connecting to the OCSP responder (a non-TOE IT entity). The evaluator then severed the connection the OCSP responder and verified that the TOE rejected the certificate when the responder could not be contacted.

FIA_X509_EXT.2.1

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 was covered by FIA_X509_EXT.1.1 Test 3 when testing a revoked chain. In that test the TOE performed certificate validation by, in part, connecting to the OCSP responder (a non-TOE IT entity). When the OCSP responder returned a revoked response, the TOE invalidated the certificate and rejected the connection.

2.4 Security Management (FMT)

2.4.1 FMT_CFG_EXT.1 Secure by Default Configuration

2.4.1.1 TSS Assurance Activity

FMT_CFG_EXT.1.1

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 6.4.1 of [ST] (“FMT_CFG_EXT.1”) states the TOE’s credentials consist of the user TLS client private key. The Hypori Halo Client installer does not include a default client key. The TOE obtains and stores the certificate and private key from the server during initial configuration. The user is not able to access any TOE functionality prior to the installation of the TLS client certificate and private key.

FMT_CFG_EXT.1.2

None.

2.4.1.2 Guidance Assurance Activity

FMT_CFG_EXT.1.1 and FMT_CFG_EXT.1.2

None.

2.4.1.3 Test Assurance Activities

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

FMT_CFG_EXT.1.1

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.

The TOE does not utilize default credentials, thus this test is not applicable.

FMT_CFG_EXT.1.1

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.

The TOE does not utilize default credentials, thus this test is not applicable.

FMT_CFG_EXT.1.1

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 utilize default credentials, thus this test is not applicable.

FMT_CFG_EXT.1.2

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.

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.

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.

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

Linux: 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.

Oracle Solaris: 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.

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 verified via static analysis that the appropriate Data Protection Classes are utilized.

2.4.2 FMT_MEC_EXT.1 Supported Configuration Mechanism

2.4.2.1 TSS Assurance 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.

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.

Section 6.4.2 of [ST] ("FMT_MEC_EXT.1") states the TOE uses iOS NSUserDefaults for saving configuration data for the application. The account options stored in NSUserDefaults consists of the Hypori Server hostname (URL), port number of the Hypori Server, Account Name, and the email address. The Hypori Halo Client policies downloaded from the Hypori server are also stored in NSUserDefaults.

The ST does not select "implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption", therefore last part of the activity is not applicable.

2.4.2.2 Guidance Assurance Activity

None.

2.4.2.3 Test Assurance Activities

Modified by TD0747

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:

Android: **The evaluator shall inspect the TSS and verify that it describes what Android API is used (and provides a link to the documentation of the API) when storing configuration data.**

The evaluator shall run the application ~~and make security-related changes to its configuration. The evaluator shall check that at least one XML file at location /data/data/package/shared_prefs/ 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~~ **verify that the behavior of the TOE is consistent with where and how the API documentation says the configuration data will be stored.**

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.

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

Linux: The evaluator shall run the application while monitoring it with the utility strace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that strace logs corresponding changes to configuration files that reside in /etc (for systemspecific configuration), in the user's home directory (for user-specific configuration), or /var/lib/ (for configurations controlled by UI and not intended to be directly modified by an administrator).

Oracle Solaris: The evaluator shall run the application while monitoring it with the utility dtrace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that dtrace logs corresponding changes to configuration files that reside in /etc (for systemspecific configuration) or in the user's home directory(for user-specific configuration).

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

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 evaluator verified that the app uses the user defaults system or key-value store for storing all settings.

The ST does not select "*implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption*", therefore this part is not applicable.

2.4.3 FMT_SMF.1 Specification of Management Functions

2.4.3.1 TSS Assurance Activity

None.

2.4.3.2 Guidance Assurance 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 6.4.3 of [ST] ("FMT_SMF.1"), the TOE supports the following management functions:

- Setting the following account options
- Applying configuration policies from the Hypori server

Sections 7 and 7.2 of [CCCO] provide the instructions for provisioning with details on how the user create an account and set the following account configuration values on the client: hostname and port number for the Hypori server, a name for the account, and an email address. Section 7 details how when using the "Add Account" screen with QR code or OTP options, the Hypori Halo Client acquires the user's credential from the Hypori provisioning server and installs it into the iOS Keystore System on the client. Section 7.2 provides the specific instructions for iOS credential provisioning.

Section 5 of the [CCCO] provides an example Client policy that when configured the TOE will download and apply. Additional details on the configuration policies are provided in the "Administrator Guide Hypori Halo", Version 1.18, 2023, Chapter 6.

2.4.3.3 Test Assurance 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 verified that configuration policies from the Hypori server were applied to clients; that the account options could be initially set on the device; and that the account name could be configured (modified after initial configuration). The account name is the only account option that is configurable after initial installation.

2.5 Privacy (FPR)

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

2.5.1.1 TSS Assurance Activity

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

The evaluator examined the TSS documentation and determined that no PII is transmitted. Section 6.5.1 of [ST] (“FPR_ANO_EXT.1”) confirms this with the statement the TOE does not transmit PII over a network.

2.5.1.2 Guidance Assurance Activity

None.

2.5.1.3 Test Assurance Activities

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

This activity is not applicable since the TOE does not handle PII.

2.6 Protection of the TSF (FPT)

2.6.1 FPT_AEX_EXT.1 Anti-Exploitation Capabilities

2.6.1.1 TSS Assurance Activity

Modified by TD0798

FPT_AEX_EXT.1.1
The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled. **If any explicitly-mapped exceptions are claimed, the evaluator shall check that the TSS identifies these exceptions, describes the static memory mapping that is used, and provides justification for why static memory mapping is appropriate in this case.**

Section 6.6.1 of [ST] (“FPT_AEX_EXT.1”) states the vendor enables address space layout randomization (ASLR) through the use of the `–fPIE –pie` compiler flags. No explicitly-mapped exceptions are identified.

FPT_AEX_EXT.1.2, FPT_AEX_EXT.1.3, FPT_AEX_EXT.1.4, and FPT_AEX_EXT.1.5
None.

2.6.1.2 Guidance Assurance Activity

FPT_AEX_EXT.1.1, FPT_AEX_EXT.1.2, FPT_AEX_EXT.1.3, FPT_AEX_EXT.1.4, and FPT_AEX_EXT.1.5
None.

2.6.1.3 Test Assurance Activities

FPT_AEX_EXT.1.1

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.

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.

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.

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.

Linux: The evaluator shall run the same application on two different Linux systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Oracle Solaris: The evaluator shall run the same application on two different Solaris systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

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.

The evaluator verified via static analysis that the TOE did not call mmap with a fixed address.

FPT_AEX_EXT.1.2

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

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

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

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.

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

Linux: The evaluator shall perform static analysis on the application to verify that both

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

- mprotect is never invoked with the PROT_EXEC permission.

Oracle Solaris: The evaluator shall perform static analysis on the application to verify that both

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

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

The evaluator verified via static analysis that the TOE never calls mprotect with the PROT_EXEC permission.

FPT_AEX_EXT.1.3

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

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

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.

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).

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

Linux: The evaluator shall ensure that the application can successfully run on a system with either SELinux or AppArmor enabled and in enforce mode.

Oracle Solaris: The evaluator shall ensure that the application can run with Solaris Trusted Extensions enabled and enforcing.

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

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

FPT_AEX_EXT.1.4

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:

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.

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.

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

Linux: 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.

Oracle Solaris: 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.

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.

The iOS platform forces applications to write all data within the application working directory (sandbox). As such, this requirement is deemed to be satisfied.

FPT_AEX_EXT.1.5

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

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.

For PE, the evaluator will disassemble each and ensure the following sequence appears:

```
mov rcx, QWORD PTR [rsp+(...)]
xor rcx, (...)
call (...)
```

.

For ELF executables, the evaluator will ensure that each contains references to the symbol **__stack_chk_fail**.

Tools such as Canary Detector may help automate these activities.

No test is defined for iOS applications, therefore this activity is not applicable.

2.6.2 FPT_API_EXT.1 Use of Supported Services and APIs

2.6.2.1 TSS Assurance Activity

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

Section 9 of [ST] (“Appendix: iOS APIs”) lists the platform APIs used by the TOE.

2.6.2.2 Guidance Assurance Activity

None.

2.6.2.3 Test Assurance 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.

The evaluator examined the API documentation and verified that all of the API's that were referenced in Appendix A of the [ST] had valid documentation sites available.

2.6.3 FPT_IDV_EXT.1 Software Identification and Versions

2.6.3.1 TSS Assurance Activity

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

Section 5.2.6.3 of [ST] ("Software Identification and Versions (FPT_IDV_EXT.1)") selects "other version information" in FPT_IDV_EXT.1.1 and completes the assignment with "iOS version information and Hypori internal versioning scheme". Section 6.6.3 of [ST] ("FPT_IDV_EXT.1") explains the TOE versioning methodology. The TOE is identified and versioned by Apple App Store version identifiers in conjunction with internal Hypori build information.

2.6.3.2 Guidance Assurance Activity

None.

2.6.3.3 Test Assurance Activities

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.

The TOE does not utilize SWID tags so that portion of the test is not applicable. The evaluator verified that the version information was shown in conjunction with FPT_TUD_EXT.1.2.

2.6.4 FPT_LIB_EXT.1 Use of Third Party Libraries

2.6.4.1 TSS Assurance Activity

None.

2.6.4.2 Guidance Assurance Activity

None.

2.6.4.3 Test Assurance Activities

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 performed static analysis on the TOE's build environment and verified that all libraries present were claimed.

2.6.5 FPT_TUD_EXT.1 Integrity for Installation and Update

2.6.5.1 TSS Assurance Activities

FPT_TUD_EXT.1.1, FPT_TUD_EXT.1.2, and FPT_TUD_EXT.1.3

None.

FPT_TUD_EXT.1.4

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 6.6.5 of [ST] ("FPT_TUD_EXT.1, FPT_TUD_EXT.2") states the TOE is distributed as an IPA file for iOS devices. The vendor (Hypori, who is the authorized source for the TOE and TOE updates) digitally signs the installation package and all updates, which Apple then re-signs. The iOS platform will only install a package from the Apple App Store if it has a valid signature from both Apple and the app developer. The user obtains the installation package through Apple App Store or their enterprise IT group, and obtains updates using the platform's update mechanism or from their enterprise IT group.

FPT_TUD_EXT.1.5

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 5.2.6.5 of [ST] ("Integrity for Installation and Update (FPT_TUD_EXT.1)") claims that the application is distributed "as an additional software package to the platform OS". [ST] section 6.6.5 ("FPT_TUD_EXT.1, FPT_TUD_EXT.2") describes the distribution as via an IPA file for iOS devices. Refer to section **Error! Reference source not found.** below for the evaluation activities associated with FPT_TUD_EXT.2.

2.6.5.2 Guidance Assurance Activities

FPT_TUD_EXT.1.1

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

Section 6 of [CCCO] ("Updates and Update Verification") describes how TOE updates are performed. Hypori distributes the TOE as an .IPA file for iOS devices. Users can obtain the installation package through the Apple App Store or the enterprise IT group of the user. Users obtain TOE updates using iOS update mechanisms or from their IT group. The Hypori Halo Client (iOS) installation package is signed by Hypori and Apple re-signs it. On iOS devices, iOS will only install a package from the Apple App Store if it has a valid signature from both Apple and the app developer.

If the application is installed using the Apple App Store, it may be updated automatically if the Apple App Store is configured to do so. If it is not, selecting the "update" option for the application in the Store application will verify that the application package is valid and install it over the older version.

FPT_TUD_EXT.1.2

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

Section 9 of [CCCO] ("Verify Version of the Hypori Client") describes how the user can query the current version of the TOE.

FPT_TUD_EXT.1.3, FPT_TUD_EXT.1.4, and FPT_TUD_EXT.1.5

None.

2.6.5.3 Test Assurance Activities

FPT_TUD_EXT.1.1

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.

The evaluator traveled to the app store and did a search of the application. When the application was shown, there was no update available.

FPT_TUD_EXT.1.2

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 installed the application. After the application was installed, the evaluator verified that the version information could be shown. The evaluator confirmed the current version of the TOE matched the documentation.

FPT_TUD_EXT.1.3

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

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 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.

Because the iOS platform forces applications to write all data within the application working directory, this requirement is considered to be met and no test activity is required.

FPT_TUD_EXT.1.4 and FPT_TUD_EXT.1.5

None.

2.6.6 FPT_TUD_EXT.2 Integrity for Installation and Update

2.6.6.1 TSS Assurance Activity

FPT_TUD_EXT.2.1 and FPT_TUD_EXT.2.2

None.

FPT_TUD_EXT.2.3

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 6.6.5 of [ST] ("FPT_TUD_EXT.1, FPT_TUD_EXT.2") states the TOE is distributed as an IPA file for iOS devices. The vendor (Hypori, who is the authorized source for the TOE and TOE updates) digitally signs the installation package and all updates, which Apple then re-signs. The iOS platform will only install a package from the Apple App Store if it has a valid signature from both Apple and the app developer.

2.6.6.2 Guidance Assurance Activity

FPT_TUD_EXT.2.1, FPT_TUD_EXT.2.2, and FPT_TUD_EXT.2.3

None.

2.6.6.3 Test Assurance Activities

Modified by TD0628

FPT_TUD_EXT.2.1

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, the evaluator shall verify that application updates are distributed in the **correct** format. This varies per platform:

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

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.

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

Linux: The evaluator shall ensure that the application is packaged in the format of the package management infrastructure of the chosen distribution. For example, applications running on Red Hat and Red Hat derivatives shall be packaged in RPM format. Applications running on Debian and Debian derivatives shall be packaged in DEB format.

Oracle Solaris: The evaluator shall ensure that the application is packaged in the PKG format.

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

The Apple App Store requires applications to be packaged in the IPA format. The TOE is available and installed via the App Store, thus this requirement is implicitly met.

FPT_TUD_EXT.2.2
Modified by TD0664

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

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

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.

The iOS platform forces applications to write all data within the application working directory (sandbox). As such, this requirement is deemed to be satisfied.

FPT_TUD_EXT.2.3

None.

2.7 Trusted Path/Channels (FTP)

2.7.1 FTP_DIT_EXT.1 Protection of Data in Transit

2.7.1.1 TSS Assurance 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 5.2.7.1 of [ST] ("FTP_DIT_EXT.1 Protection of Data in Transit") claims that the application shall invoke platform-provided functionality to encrypt all transmitted data with TLS. Section 6.7.1 of [ST] ("FTP_DIT_EXT.1") states the TOE leverages the following calls to invoke the platform-provided functionality:

- PLIST/"No arbitrary loads" is a mechanism to tell the iOS platform that the Hypori Halo Client will not make any HTTP or non-TLS protected communications. It is set in a PLIST file that is bundled inside the Apple IPA file. The PLIST is not a classic API call, but a mechanism for the Hypori Halo Client to constrain how it can invoke remote services,
- The calls outgoing on NSURLSession will invoke App Transport Security (ATS).
- The calls outgoing on SWIFTNIO use Network.Framework and DO NOT invoke ATS.

2.7.1.2 Guidance Assurance Activity

None.

2.7.1.3 Test Assurance Activities

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 used the application as a TLS client during the testing of FDP_NET_EXT.1.1. The evaluator inspected packet captures and confirmed traffic was encrypted with TLS.

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.

The evaluator used the application as a TLS client during the testing of FDP_NET_EXT.1.1. The evaluator inspected packet captures and confirmed traffic was encrypted and that no sensitive data was 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.

The TOE does not transmit user credentials over the network, thus this test is not applicable.

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 usespermission-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.

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 evaluator verified that the specified keys did not appear in the TOE’s Info.plist file.

3 Security Assurance Requirements

3.1 Class ADV: Development

3.1.1 ADV_FSP.1 Basic Functional Specification

There are no specific evaluation 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 2 Security Functional Requirement Assurance Activities, 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 evaluation 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 Class AGD: Guidance Documents

3.2.1 AGD_OPE.1 Operational User Guidance

3.2.1.1 TSS Assurance Activity

None defined.

3.2.1.2 Guidance Assurance Activity

Some of the contents of the operational guidance will be verified by the evaluation activities in Section 2 Security Functional Requirement Assurance Activities 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 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 TOE does not provide any cryptographic functions.

As stated in section 2.6.5.2 of this document, Chapter 6 of [CCCO] provides guidance for installing the TOE and TOE updates, including instructions for obtaining the TOE and TOE updates, initiating the update process, the process for verifying updates to the TOE by verifying a digital signature, and determining whether or not the update was successful.

[CCCO] section 1.1 provides an overview of the product depicting the TOE as the Hypori Client. [CCCO] section 2 “Common Criteria Evaluation” indicates that the evaluation concentrated on demonstrating that the Hypori Client conforms to the security requirements specified in *Protection Profile for Application Software* v1.4. It specifically states that the functionality described in this guidance documentation is limited to the security functionality described in the Security Target. Other product functionality is not applicable to the claimed Protection Profile and was therefore not examined as part of the Common Criteria evaluation of the Hypori Client product.

3.2.1.3 Test Assurance Activity

None defined.

3.2.2 AGD_PRE.1 Preparative Procedures

3.2.2.1 TSS Assurance Activity

None defined.

3.2.2.2 Guidance 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 iOS versions 15 and 16. The guidance documentation adequately addresses these releases.

3.2.2.3 Test Assurance Activity

None defined.

3.3 Class ALC: Life-Cycle Support

3.3.1 ALC_CMC.1 Labeling of the TOE

3.3.1.1 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 as Hypori Halo Client (iOS) 4.3. The title page of [CCCO] identifies the TOE version as 4.3, while Section 2 of [CCCO] (“Common Criteria Evaluation”) identifies multiple times the TOE as being the

Hypori Client version 4.3 (note that [CCCO] covers all three evaluated versions of Hypori Client—Android, iOS, and Windows).

The vendor maintains a web site (www.hypori.com) providing general information advertising the Hypori Halo and capabilities of Hypori Client, without identifying specific product versions. The vendor product suite consists solely of the Hypori Halo and therefore the information in the ST is sufficient to distinguish the evaluated version of the product from any unevaluated versions as there is only one solution on their website.

The evaluator checked the operational guidance and TOE samples received for testing and observed that the version number is consistent with that in the ST.

3.3.2 ALC_CMS.1 TOE CM Coverage

3.3.2.1 TSS 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 evaluation 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.

As described in Section **Error! Reference source not found.**.1 above, the evaluator confirmed the TOE is labelled with its unique software version identifier.

3.3.2.2 Guidance Assurance Activity

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.

Section 6.6 of [ST] ("Protection of the TSF") describes how the TOE uses security features provided by the iOS platform. This includes address space layout randomization (ASLR) and stack-based buffer overflow protection.

To enable ASLR and stack protection on the iOS Hypori Halo Client, Hypori builds with the -fPIE -pie and the -fstack-protector-strong flags. As the iOS platform version of the TOE is packaged as an IPA file, the compilation has already been done by default. As described in Section **Error! Reference source not found.**.1 above, the evaluator confirmed the TOE is labelled with its unique software version identifier.

3.3.2.3 Test Assurance Activity

None defined.

3.3.3 ALC_TSU_EXT.1 Timely Security Updates

3.3.3.1 TSS 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 6.8 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.

The vendor provides customers with timely updates. A customer chooses their preferred communication. The vendor's Support Department will notify customers of updates using each customer's preferred communication mechanism. Application changes may be pushed to end users via the Apple App Store like any other application or via an enterprise application store internal to a customer. Typical delivery times for security updates are 5 to 10 business days.

The vendor maintains an on-line Support Portal. Every customer is registered with the Support Portal. The vendor notifies each customer of a new security report on the Support Portal using the customer's preferred communication mechanism. The vendor secures the Support Portal via TLS and user authentication. Each customer contact must log in with their specific credentials in order to see the security reports.

3.3.3.2 Guidance Assurance Activity

None defined.

3.3.3.3 Test Assurance Activity

None defined.

3.4 Class ATE: Tests

3.4.1 ATE_IND.1 Independent Testing – Conformance

3.4.1.1 TSS Assurance Activity

None defined.

3.4.1.2 Guidance Assurance Activity

None defined.

3.4.1.3 Test 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 evaluation activities.

While it is not necessary to have one test case per test listed in an evaluation 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 effect; 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 (e.g 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. The evaluation team compiled a detailed test plan and report with a complete set of activities that follow the [App PP]. The procedures and results of this testing are available in the DTR document.

3.5 Class AVA: Vulnerability Assessment

3.5.1 AVA_VAN.1 Vulnerability Survey

3.5.1.1 TSS Assurance Activity

None defined.

3.5.1.2 Guidance Assurance Activity

None defined.

3.5.1.3 Test 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 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 following online sources:

- National Vulnerability Database (https://nvd.nist.gov/)
- US-CERT Vulnerability Notes Database (https://www.kb.cert.org/vuls/)

The searches were performed on October 30, 2023, January 8, 2024, and February 15th, 2024 using the following search terms:

- Hypori
- Hypori Client
- Hypori Halo
- Android Cloud Environment
- Thin Client
- Virtual Mobile Infrastructure
- The identity of each of the third-party libraries listed in Section 5.2.6.4 of [ST].

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