Oracle Corporation

Oracle VM Server for SPARC 3.6 and Oracle Solaris 11.4

Assurance Activity Report

Version 1.4

January 25, 2024

Document prepared by

Lightship Security

www.lightshipsec.com

Table of Contents

1 Introduction

1 This Assurance Activity Report (AAR) documents the evaluation activities performed by Lightship Security for the evaluation identified in Table 1: Evaluation Identifiers. The AAR is produced in accordance with National Information Assurance Program (NIAP) reporting guidelines.

1.1 Evaluation Identifiers

Table 1: Evaluation Identifiers

Scheme	Canadian Common Criteria Scheme
Evaluation Facility	Lightship Security
Developer/Sponsor	Oracle Corporation
TOE	Oracle VM Server for SPARC 3.6.2.0.57 and Oracle Solaris 11.4.57.0.1.144.3 with IDR 5391
Security Target	Oracle VM Server for SPARC 3.6 and Oracle Solaris 11.4 Security Target, v2.4, January 2024
Protection Profile	[PP] NIAP Protection Profile for Virtualization, Version: 1.1, 2021-06-14 [SV] NIAP PP-Module for Server Virtualization Systems, Version: 1.1, 2021-06-14 [CFG] PP-Configuration for Virtualization and Server Virtualization Systems, Version: 1.0, 2021-06-04 [PKG_SSH] NIAP Functional Package for Secure Shell (SSH), Version: 1.0, 2021-05-13 [PKG_TLS] NIAP Functional Package for Transport Layer Security (TLS), Version: 1.1, 2019-03-01

1.2 Evaluation Methods

2 The evaluation was performed using the methods, tools and standards identified in Table 2.

Table 2: Evaluation Methods

Evaluation Criteria	CC v3.1R5
Evaluation Methodology	CEM v3.1R5 CC and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs, Version 0.5, May 2017
Supporting Documents	Assurance Activities found in [PP], [SV-SD], [CFG], [PKG_SSH] and [PKG_TLS]
Interpretations	

[PP]	Applicable to Evaluation?
TD0615: Audit generation for hypercalls implemented in HW	Yes.
TD0721: Mapping FTA_TAB.1 to Objective	Yes.
TD0742: Updates to Certificate Revocation (FIA_X509_EXT.1) for Base Virtualization PP v1.1	Yes.
[PKG_TLS]	
TD0442: Updated TLS Ciphersuites for TLS Package	Yes.
TD0469: Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1	N/A. FCS_TLSS_EXT.1 not claimed.
TD0499: Testing with pinned certificates	Yes.
TD0513: CA Certificate loading	Yes.
TD0726: Corrections to (D)TLSS SFRs in TLS 1.1 FP	N/A. FCS_TLSS_EXT.1 and FCS_DTLSS_EXT.1 not claimed.
TD0739: PKG_TLS_V1.1 has 2 different publication dates	N/A. FCS_TLSS_EXT.1 not claimed.
TD0770: TLSS.2 connection with no client cert	N/A. FCS_TLSS_EXT.2 not claimed.
TD0779: Updated Session Resumption Support in TLS package V1.1	N/A. FCS_TLSS_EXT.1 not claimed.
[PKG_SSH]	
TD0682: Addressing Ambiguity in FCS_SSHS_EXT.1 Tests	Yes.
TD0695: Choice of 128 or 256 bit size in AES-CTR in SSH Functional Package.	Yes.
TD0732: FCS_SSHS_EXT.1.3 Test 2 Update	Yes.
TD0777: Clarification to Selections for Auditable Events for FCS_SSH_EXT.1	Yes.

Tools	Please refer to the associated Test Plan document.

1.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
[ST]	Oracle VM Server for SPARC 3.6 and Oracle Solaris 11.4 Security Target, v2.4, January 2024
[PP]	NIAP Protection Profile for Virtualization, Version: 1.1, 2021-06-14
[SV]	NIAP PP-Module for Server Virtualization Systems, Version: 1.1, 2021-06-14
[SV-SD]	Supporting Document Mandatory Technical Document PP-Module for Server Virtualization Systems, Version: 1.1, 2021-06-14
[CFG]	PP-Configuration for Virtualization and Server Virtualization Systems, Version: 1.0, 2021-06-04
[PKG_SSH]	NIAP Functional Package for Secure Shell (SSH), Version: 1.0, 2021-05-13
[PKG_TLS]	NIAP Functional Package for Transport Layer Security (TLS), Version: 1.1, 2019-03-01
[AGD]	Oracle VM Server for SPARC 3.6 and Oracle Solaris 11.4 Common Criteria Guide, Version: 1.5, January 2024
[SPARC]	Oracle VM Server for SPARC Release 3.6 Documentation – https://docs.oracle.com/cd/E93612_01/
[T8LIB]	Oracle SPARC T8 Servers Documentation Library – https://docs.oracle.com/en/servers/sparc/t8/index.html
[SOLARIS]	Oracle Solaris 11.4 Documentation Library – https://docs.oracle.com/cd/E37838_01/
[VIOP]	Logical Domains Virtual I/O Protocol Specification, revision v9, February 15, 2010 *Proprietary annex*
[SUN4V]	Oracle UltraSPARC Virtual Machine Specification, version 3033604f0239 3.0-draft7, Publication date 2012-03-13 19:43, available at: https://sun4v.github.io/downloads/hypervisor-api-3.0draft7.pdf

2 Evaluation Activities for SFRs

2.1 Security Audit (FAU)

2.1.1 FAU_GEN.1 Audit Data Generation

2.1.1.1 TSS

3 The evaluator shall check the TSS and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type shall be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP-Configuration is described in the TSS.

Findings: The evaluator found that in section 6.1 of the [ST], the author refers the reader to the table of auditable messages in section 5.3 of the [ST]. The evaluator checked each auditable event in the FAU_GEN.1 table in section 5.3.1 of the [ST] and found that all audit event types mandated by the PP are accounted for.

[ST], section 6.1.1 states, “Refer to section 3.2.4.3 of the Oracle VM Server for SPARC 3.6 and Oracle Solaris 11.4 Common Criteria Guide for material details on audit record formats.” (*n.b. document reference is referred to within this AAR as [AGD].*)

Section 3.2.4.3 of the [AGD] discusses the use of praudit as a means to display the information. The manual page for praudit(8) found in [SOLARIS] states: “*…interprets the data as audit trail records as defined in the audit.log(5) man page.*” The audit.log(5) man page, in turn provides the binary format of the logs, and praudit describes the transformation process from binary to administrator-friendly information while discussing each of the fields contained in the audit log data.

2.1.1.2 Guidance Documentation

4 The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP-Configuration. The evaluator shall examine the administrative guide and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP and PP-Modules. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are security-relevant with respect to this PP-Configuration.

Findings: The evaluator used the [AGD] and each of the additional resources provided in the Oracle documentation library to successfully configure the mechanisms necessary to enforce the requirements specified in the PP, Functional Packages and PP-Modules, such that the claimed security functionality of the TOE was exercised/demonstrated, including successful completion of all mandated testing requirements, as described in the Test Plan.

In doing so, the evaluator was able to determine that the administrative actions, that are relevant in the context of this PP-Configuration, are those that are described within sections 2, 3 and 4 of the [AGD], with additional information provided in the Oracle documentation library.

	The methodology used to determine which actions in the administrative guidance were security-relevant with respect to this PP-Configuration involved reviewing the claimed PP, Functional Packages, PP-Modules with the [ST] and attempting to use the provided guidance resources to configure the TOE such that each SFR was met as described above. Security-relevant commands were determined to be those that are necessary for the TOE to meet a given SFR. The evaluator did not note any areas where the mapping between administrator actions and SFRs were ambiguous.

2.1.1.3 Tests

5 The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed and administrative actions. For administrative actions, the evaluator shall test that each action determined by the evaluator above to be security relevant in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

6 Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

High-Level Test Description
Ensure each of the Mandatory and Selection-based auditable requirements are met by reviewing the audit records generated during testing of the associated SFRs. Verify the audit records match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.
PASS

2.1.2 FAU_SAR.1 Audit Review

2.1.2.1 Guidance Documentation

7 The evaluator shall review the operational guidance for the procedure on how to review the audit records.

Findings:	Section 3.2.4.3 of the [AGD] indicates audit records can be reviewed using the *praudit* and *auditreduce* commands.

2.1.2.2 Tests

8 The evaluator shall verify that the audit records provide all of the information specified in FAU_GEN.1 and that this information is suitable for human interpretation. The evaluation activity for this requirement is performed in conjunction with the evaluation activity for FAU_GEN.1.

Findings:	Please refer to the evaluation activities conducted as part of FAU_GEN.1.

2.1.3 FAU_STG.1 Protected Audit Trail Storage

2.1.3.1 TSS

9 The evaluator shall ensure that the TSS describes how the audit records are protected from unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.

Findings:	Section 6.1.3 of the [ST] indicates that an authorized administrator is the only subject permitted to delete audit records on the TOE.

2.1.3.2 Tests

10 The evaluator shall perform the following tests:

Test 1: The evaluator shall access the audit trail as an unauthorized Administrator and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail.

High-Level Test Description
For each of the audit log files, attempt to modify and delete them as an unprivileged user and show the attempt is rejected.
PASS

11 **Test 2:** The evaluator shall access the audit trail as an authorized Administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.

High-Level Test Description
As an authorized administrator, delete logs found in */var/audit* and */var/log* and show the deletion is successful.
PASS

2.1.4 FAU_STG_EXT.1 Off-Loading of Audit Data

2.1.4.1 FAU_STG_EXT.1.1 TSS

12 Protocols used for implementing the trusted channel must be selected in FTP_ITC_EXT.1.

Findings:	Section 6.1.4 of the [ST] indicates that TLS is used by the TSF to communicate with the audit server. TLS is one of the protocols selected in FTP_ITC_EXT.1 in section 5.3.8 of the [ST].

13 The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.

Findings:	As per section 6.1.4 of the [ST], the TOE forwards logs to an external syslog server in real-time using TLS.

2.1.4.2 FAU_STG_EXT.1.1 Guidance Documentation

14 The evaluator shall examine the operational guidance to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

Findings:	Sections 3.2.4.6 and 4 of the [AGD] describe the audit server requirements, necessary TOE configuration steps, and how to establish the trusted channel to the audit server.

2.1.4.3 FAU_STG_EXT.1.1 Tests

15 Testing of the trusted channel mechanism is to be performed as specified in the evaluation activities for FTP_ITC_EXT.1.

16 The evaluator shall perform the following test for this requirement:

- **Test 1:** The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator’s choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing.

High-Level Test Description
Login and logout of the TOE. Show that TLS records are generated between the TOE and the Syslog receiver. Show that the audit data cannot be seen in plaintext on the wire and that the user’s password is not exposed in the log messages.
PASS

2.1.4.4 FAU_STG_EXT.1.2 TSS

17 The evaluator shall examine the TSS to ensure it describes what happens when the local audit data store is full.

Findings:	Section 6.1.1 of the [ST] indicates that when the TOE’s local audit space is exhausted, the TOE will count dropped audit events. When only 1% of the audit disk storage remains, the TOE will raise a warning.

2.1.4.5 FAU_STG_EXT.1.2 Guidance Documentation

18 The evaluator shall also examine the operational guidance to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and “cleared” periodically by sending the data to the audit server.

Findings:	The evaluator consulted the man pages for rsyslog(8) and rsyslog.conf(5) found under Sections 5 and 8 of [SOLARIS] / Oracle Solaris Reference Manuals. The man pages direct an administrator to review the rsyslog distribution documentation at https://rsyslog.com/doc for the current revision (v8-stable). This led the evaluator to the following documentation https://www.rsyslog.com/doc/v8-stable/tutorials/reliable_forwarding.html and https://www.rsyslog.com/doc/v8-stable/concepts/queues.html which describe the queueing mechanisms in use. The evaluator considered the information and found that rsyslog queues information in-memory unless configured for a disk-queue. The queue is emptied to the remote endpoint as simultaneously as possible, with exceptions when the remote server is down, or the queue is backed up due to excessive size or other performance issues.

2.1.4.6 FAU_STG_EXT.1.2 Tests

19 The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behavior defined in the ST for FAU_STG_EXT.1.2.

High-Level Test Description
Review the audit records on the TOE. Use a tool to fill up the local audit space and subsequently execute actions that would result in auditable events. Review the audit records and verify the TOE dropped the recent audit data, as described in the [ST].
PASS

2.2 Cryptographic Support (FCS)

2.2.1 FCS_CKM.1 Cryptographic Key Generation

2.2.1.1 TSS

20 The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Findings:	Section 6.2.1 of the [ST] claims that RSA 2048- and 3072-bit keys are supported. The TOE supports ECDSA keys of size P-256 and P-384. It also claims in the same section that FFC schemes using safe-primes are supported. Finally, DH group 14 is supported. The key generation scheme usages are clearly delineated in the table in section 6.2.1 of the [ST] and these are consistent with the remaining SFR claims.

2.2.1.2 Guidance Documentation

21 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.

Findings:	Sections 3.3-3.3.1 and 4.2 of the [AGD] describe how to configure the TOE to use the selected key generation schemes and key sizes for the TOE's SSH server and TLS client, respectively.

2.2.1.3 Tests

22 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 Generation for FIPS PUB 186-4 RSA Schemes

23 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*.

24 Key Pair generation specifies 5 ways (or methods) to generate the primes *p* and *q*. These include:

- Random Primes:
    - Provable primes
    - Probable primes
- 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

25 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 seeds, 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.

Note:	The TOE makes use of RSA key generation schemes for TLS and SSH. The RSA key generation services are provided by A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826). The evaluator verified the certificate includes the appropriate RSA key generation algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

26 For each supported NIST curve, i.e., P-256, P-384 and P-521, 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

27 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.

Note:	The TOE makes use of ECC key generation schemes for SSH. The ECC key generation services are provided by A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826). The evaluator verified the certificate includes the appropriate ECC key generation algorithms for the claimed curves and an operating environment that corresponds to the TOE platform.

Key Generation for Finite-Field Cryptography (FFC)

28 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.

29 The Parameter generation specifies two ways (or methods) to generate the cryptographic prime q and the field prime p:

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

30 and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

31 The Key generation specifies two ways to generate the private key x:

- 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 and a +1 operation, where 1<= x<=q-1.

32 The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

33 To test the cryptographic and field prime generation method for the provable primes method and 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.

34 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 shall also confirm

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

35 for each FFC parameter set and key pair.

Findings:	The TOE makes use of FFC key generation schemes for TLS. The FFC key generation services are provided by A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826). The evaluator verified the certificate includes the appropriate FFC key generation algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups

36 Testing for FFC Schemes using Diffie-Hellman group 14 and "safe-prime" groups is done as part of testing in FCS_CKM.2.1.

Findings:	Please refer to FCS_CKM.2 for descriptions of testing DH group 14 for safe prime groups.

2.2.2 FCS_CKM.2 Cryptographic Key Distribution

2.2.2.1 TSS

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

Findings:	Section 5.3.2 of the [ST] claims RSA, ECDSA and FFC safe-prime key generation in FCS_CKM.1.1. FCS_CKM.2.1 claims RSA, FFC safe-primes, and DH group 14 which is correct as ECDSA is not used for key exchange, but rather SSH host public keys. The [ST], in section 6.2.1, claims use of RSA, FFC safe-prime schemes and DH group 14 as key establishment schemes. The key establishment scheme usages are clearly delineated in the table in section 6.2.1 of the [ST].

2.2.2.2 Guidance Documentation

38 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment schemes.

Findings:	Sections 3.3-3.3.1 and 4.2 of the [AGD] describe how to configure the TOE to use the selected key establishment schemes for the TOE's SSH server and TLS client, respectively.

2.2.2.3 Tests

Key Establishment Schemes

RSAES-PKCS1-v1_5 Key Establishment Schemes

39 The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses RSAES-PKCS1-v1_5.

Findings:	Please see FCS_TLSC_EXT.1 for confirmation.

SP800-56A ECC Key Establishment Schemes

40 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

41 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.

42 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 field OI and TOE id fields.

43 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.

44 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.

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

Validity Test

46 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 other info and TOE id fields.

47 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 other information field OI, 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 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).

48 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.

Findings:	The TOE does not claim support for ECC key establishment schemes.

Diffie-Hellman Group 14

49 The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses Diffie-Hellman Group 14.

Findings:	Please see FCS_SSHS_EXT.1 for confirmation.

FFC Schemes using "safe-prime" groups (identified in Appendix D of SP 800-56A Revision 3)

50 The evaluator shall verify the correctness of the TSF's implementation of "safe-prime" groups by using a known good implementation for each protocol selected in FTP_ITC_EXT.1 that uses "safe-prime" groups. This test must be performed for each "safe-prime" group that each protocol uses.

Findings:	Please see FCS_TLSC_EXT.1 for confirmation.

2.2.3 FCS_CKM_EXT.4 Cryptographic Key Destruction

2.2.3.1 TSS

51 The evaluator shall check to ensure the TSS lists each type of key and its origin and location in memory or storage. The evaluator shall verify that the TSS describes when each type of key is cleared.

Findings:	Section 6.2.2 of the [ST] provides a statement of the keys. Section 6.2.2 of the [ST] also indicates that keys stored in volatile storage are destroyed upon removal of power; keys in non-volatile storage are destroyed when the OS deletes them. Based on a review of the applicable cryptographic protocols and key usages, the description appears to be complete and accurate: the [ST] claims TLS (as a non-authenticating client), SSH (as a server) and AES (in support of ZFS). No other encryption/decryption or digital signature generation services are claimed or described in the [ST].

2.2.3.2 Tests

52 For each key clearing situation the evaluator shall perform one of the following activities:

- The evaluator shall use appropriate combinations of specialized operational or development environments, development tools (debuggers, emulators, simulators, etc.), or instrumented builds (developmental, debug, or release) to demonstrate that keys are cleared correctly, including all intermediate copies of the key that may have been created internally by the TOE during normal cryptographic processing.
- In cases where testing reveals that third-party software modules or programming language run-time environments do not properly overwrite keys, this fact must be documented. Likewise, it must be documented if there is no practical way to determine whether such modules or environments destroy keys properly.
- In cases where it is impossible or impracticable to perform the above tests, the evaluator shall describe how keys are destroyed in such cases, to include:
    - Which keys are affected

- The reasons why testing is impossible or impracticable
- Evidence that keys are destroyed appropriately (e.g., citations to component documentation, component developer/vendor attestation, component vendor test results)
- Aggravating and mitigating factors that may affect the timeliness or execution of key destruction (e.g., caching, garbage collection, operating system memory management)

53 Use of debug or instrumented builds of the TOE and TOE components is permitted in order to demonstrate that the TOE takes appropriate action to destroy keys. These builds should be based on the same source code as are release builds (of course, with instrumentation and debug-specific code added).

Volatile Memory

High-Level Test Description
Using instrumented TLS and SSH clients, establish a TLS connection between the TOE and the remote logging server and a SSH connection between the TOE and the evaluator’s workstation. Record the TLS session key and SSH encryption keys, as output by the tools. Prior to terminating the TLS and SSH sessions, dump the system memory pages to files using the *savecore* utility. Search the dumped memory pages for the recorded TLS and SSH keys and verify the keys are found. Establish new TLS and SSH connections and record the keys, as was previously done. Terminate the TLS and SSH sessions and subsequently dump the system memory pages. Search the dumped memory pages for the recorded TLS and SSH keys, and verify the keys are not found.
PASS

Non-volatile Memory: SSH server key

High-Level Test Description
Construct an encrypted *ZFS* dataset using a KEK which is stored in a file. Store an SSH host private key on the encrypted *ZFS* filesystem. Run an instance of the SSH daemon which uses a host-key on the encrypted dataset. Unmount the *ZFS* dataset. Destroy the KEK file and attempt to remount the encrypted dataset (it will fail). Rerun the SSH daemon instance and show it fails to load due to a missing hostkey (due to a missing mount).
PASS

Non-volatile Memory: ZFS Data Encryption Key

High-Level Test Description
Construct an encrypted ZFS dataset using a KEK which is stored in a file. Write a file to the encrypted ZFS filesystem and read the file to show that basic file operations work as expected. Unmount the ZFS dataset and subsequently destroy the KEK. Attempt to remount the encrypted dataset (it will fail). Attempt to read the previously created file and show it is not available (due to a missing KEK/mount).
PASS

2.2.4 FCS_COP.1/Hash Cryptographic Operation (Hashing)

2.2.4.1 TSS

54 The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Findings:	Section 6.2.3 of the [ST] claims that the hash function is used for HMAC services and digital signature verification for trusted updates. These services are consistent with the claims made in section 5 of the [ST].

2.2.4.2 Guidance Documentation

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

Findings:	Sections 3.3.1 and 4.2 of the [AGD] describe how to configure the TOE to use the selected hashing algorithms for the TOE's SSH server and TLS client, respectively.

2.2.4.3 Tests

56 **SHA-1 and SHA-2 Tests** The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented test MACs.

57 The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

58 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.

59 **Short Messages Test Bit-oriented Mode**

60 The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

61 **Short Messages Test Byte-oriented Mode**

62 The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

63 **Selected Long Messages Test Bit-oriented Mode**

64 The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

65 **Selected Long Messages Test Byte-oriented Mode**

66 The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

67 **Pseudorandomly Generated Messages Test**

68 This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

69 **SHA-3 Tests** The tests below are derived from The Secure Hash Algorithm-3 Validation System (SHA3VS), Updated: April 7, 2016, from the National Institute of Standards and Technology.

70 For each SHA-3-XXX implementation, XXX represents d, the digest length in bits. The capacity, c, is equal to 2d bits. The rate is equal to 1600-c bits.

71 The TSF hashing functions can be implemented with one of two orientations. The first is a bit-oriented mode that hashes messages of arbitrary length. The second is a byte-oriented mode that hashes messages that are an integral number of bytes in length (i.e., the length (in bits) of the message to be hashed is divisible by 8). Separate tests for each orientation are given below.

72 The evaluator shall perform all of the following tests for each hash algorithm and orientation implemented by the TSF and used to satisfy the requirements of this PP. The evaluator shall compare digest values produced by a known-good SHA-3 implementation against those generated by running the same values through the TSF.

Short Messages Test, Bit-oriented Mode

73 The evaluators devise an input set consisting of rate+1 short messages. The length of the messages ranges sequentially from 0 to rate bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Short Messages Test, Byte-oriented Mode

74 The evaluators devise an input set consisting of rate/8+1 short messages. The length of the messages ranges sequentially from 0 to rate/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure

that the correct result is produced when the messages are provided to the TSF. The message of length 0 is omitted if the TOE does not support zero-length messages.

Selected Long Messages Test, Bit-oriented Mode

75 The evaluators devise an input set consisting of 100 long messages ranging in size from rate+ (rate+1) to rate+(100*(rate+1)), incrementing by rate+1. (For example, SHA-3-256 has a rate of 1088 bits. Therefore, 100 messages will be generated with lengths 2177, 3266, …, 109988 bits.) The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test, Byte-oriented Mode

76 The evaluators devise an input set consisting of 100 messages ranging in size from (rate+ (rate+8)) to (rate+100*(rate+8)), incrementing by rate+8. (For example, SHA-3-256 has a rate of 1088 bits. Therefore 100 messages will be generated of lengths 2184, 3280, 4376, …, 110688 bits.) The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Monte Carlo) Test, Byte-oriented Mode

77 The evaluators supply a seed of d bits (where d is the length of the message digest produced by the hash function to be tested. This seed is used by a pseudorandom function to generate 100,000 message digests. One hundred of the digests (every 1000th digest) are recorded as checkpoints. The TOE then uses the same procedure to generate the same 100,000 message digests and 100 checkpoint values. The evaluators then compare the results generated to ensure that the correct result is produced when the messages are generated by the TSF.

Findings:	The TOE makes use of cryptographic hashing for digital signature verification of trusted updates and in support of HMAC services. The hashing services are provided by A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826). The evaluator verified the certificate includes the appropriate SHA- algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

2.2.5 FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithms)

2.2.5.1 TSS

78 The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

Findings:	Section 6.2.4 of the [ST] identifies the key length, hash function, block size and MAC length output.

2.2.5.2 Tests

79 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.

80 For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known good implementation.

Findings: The TOE makes use of HMAC for TLS and SSH. This service is provided by A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826). The evaluator verified the certificate includes the appropriate HMAC-SHA algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

2.2.6 FCS_COP.1/Sig Cryptographic Operation (Signature Algorithms)

2.2.6.1 Tests

81 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.

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

82 For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

83 For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Findings: The TOE makes use of ECDSA signing and verification for SSH. These services are provided by A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826). The evaluator verified the certificate includes the appropriate ECDSA signature generation and verification algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

RSA Signature Algorithm Tests

Signature Generation Test

84 The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test, the evaluator shall generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

85 The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

86 The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, or signatures. The TOE attempts to verify the signatures and returns success or failure.

87 The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

Findings:	The TOE makes use of RSA signing and verification for TLS, SSH and trusted updates. These services are provided by A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826). The evaluator verified the certificate includes the appropriate RSA signature generation and verification algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

2.2.7 FCS_COP.1/UDE Cryptographic Operation (AES Data Encryption/Decryption)

2.2.7.1 Tests

88 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.

AES-CBC Tests

AES-CBC Known Answer Tests

89 There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

90 **KAT-1.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

91 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

92 **KAT-2.** To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

93 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

94 **KAT-3.** To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall

have 256 256-bit keys. Key *i* in each set shall have the leftmost *i* bits be ones and the rightmost *N-i* bits be zeros, for *i* in [1,N].

95 To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

96 **KAT-4.** To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

97 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

98 The evaluator shall test the encrypt functionality by encrypting an *i*-block message where 1 < *i* <=10. The evaluator shall choose a key, an IV and plaintext message of length *i* blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

99 The evaluator shall also test the decrypt functionality for each mode by decrypting an *i*-block message where 1 < *i* <=10. The evaluator shall choose a key, an IV and a ciphertext message of length *i* blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

100 **AES-CBC Monte Carlo Tests**

101 The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
    if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
    else:
        CT[i] = AES-CBC-Encrypt(Key, PT)
        PT = CT[i-1]
```

102 The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

103 The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

Findings:	AES-CBC mode with 128-bit and 256-bit keys is claimed for SSH and TLS functionality. The evaluator verified A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826) includes the appropriate AES-CBC encrypt and decrypt algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

AES-CCM Tests

104 The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

128 bit and 256 bit keys

105 **Two payload lengths**. One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).

106 **Two or three associated data lengths**. One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2^{16} bytes, an associated data length of 2^{16} bytes shall be tested.

107 **Nonce lengths**. All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.

108 **Tag lengths**. All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.

109 To test the generation-encryption functionality of AES-CCM, the evaluator shall perform the following four tests:

- **Test 1**. For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

- **Test 2**. For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

- **Test 3**. For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.

- **Test 4**. For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator shall supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.

110 To determine correctness in each of the above tests, the evaluator shall compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

111 To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator shall supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator shall supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

112 Additionally, the evaluator shall use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AES-CCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

Note: AES-CCM mode with 128-bit and 256-bit keys is claimed for ZFS functionality. The evaluator verified A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826) includes the appropriate AES-CCM generation-encryption and verification-decryption algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

AES-GCM Test

113 The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

114 **Two plaintext lengths**. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

115 **Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

116 **Two IV lengths**. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

117 The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator, or the implementation being tested, as long as it is known.

118 The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

119 The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Findings:	AES-GCM mode with 128-bit and 256-bit keys is claimed for ZFS, SSH and TLS functionality. The evaluator verified the A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826) includes the appropriate AES-GCM encrypt and decrypt algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

XTS-AES Test

120 The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

- 256 bit (for AES-128) and 512 bit (for AES-256) keys
- **Three data unit (i.e., plaintext) lengths**. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 2^{16} bits, whichever is smaller.

121 using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

122 The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

123 The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

Note:	XTS-AES is not supported by the TOE.

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

124 The evaluator shall test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- 128 and 256 bit key encryption keys (KEKs)
- **Three plaintext lengths**. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).

125 using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall use the AES-KW authenticated-encryption function of a known good implementation.

126 The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

127 The evaluator shall test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

128 One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).

129 One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

130 The evaluator shall test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

Note: AES-KW mode with 128-bit and 256-bit keys is claimed for ZFS functionality. The evaluator verified the A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826) includes the appropriate AES-GCM authenticated-encryption and authenticated-decryption algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

AES-CTR Test

- **Test 1**: Known Answer Tests (KATs)

  There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, initialization vector (IV), and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

  Test 1a: To test the encrypt functionality, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all zeros key, and the other five shall be encrypted with a 256-bit all zeros key. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input.

  Test 1b: To test the encrypt functionality, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. Five of the key values shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.

  Test 1c: To test the encrypt functionality, the evaluator shall supply the two sets of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second shall have 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have

256 256-bit pairs. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros for i in [1, N]. The ciphertext value in each pair shall be the value that results in an all zeros plaintext when decrypted with its corresponding key.

Test 1d: To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 128-bit key value of all zeros and using a 256 bit key value of all zeros, respectively, and an IV of all zeros. Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128]. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.

- **Test 2**: Multi-Block Message Test

  The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

- **Test 3**: Monte-Carlo Test

  For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine. The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

```
For AES-ECB mode
# Input: PT, Key
for i = 1 to 1000:
    CT[i] = AES-ECB-Encrypt(Key, PT)
    PT = CT[i]
```

  The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

Findings:	AES-CTR mode with 128-bit and 256-bit keys is claimed for SSH functionality. The evaluator verified the A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826) includes the appropriate AES-CTR encrypt and decrypt algorithms for the claimed key sizes and an operating environment that corresponds to the TOE platform.

131 If "invoke platform-provided" is selected, the evaluator confirms that SSH connections are only successful if appropriate algorithms and appropriate key sizes are configured. To do this, the evaluator shall perform the following tests:

- **Test 1**: [Conditional: TOE is an SSH server] The evaluator shall configure an SSH client to connect with an invalid cryptographic algorithm and key size for each listening SSH socket connection on the TOE. The evaluator initiates SSH client connections to each listening SSH socket connection on the TOE and observes that the connection fails in each attempt.

- **Test 2**: [Conditional: TOE is an SSH client] The evaluator shall configure a listening SSH socket on a remote SSH server that accepts only invalid cryptographic algorithms and keys. The evaluator uses the TOE to attempt an SSH connection to this server and observes that the connection fails.

Findings:	There is no option for “invoke platform-provided” in this SFR. This appears to be a copy/paste from another Protection Profile. SSH testing is conducted as part of FCS_SSH_EXT.1 in the [PKG_SSH] module. In any event, because of the fact that these tests are only performed if the “invoke platform-provided” functionality is selected, these tests are skipped.

2.2.8 FCS_ENT_EXT.1 Entropy for Virtual Machines

2.2.8.1 TSS

132 The evaluator shall verify that the TSS describes how the TOE provides entropy to Guest VMs, and how to access the interface to acquire entropy or random numbers. The evaluator shall verify that the TSS describes the mechanisms for ensuring that one VM does not affect the entropy acquired by another.

Findings:	As per section 6.2.7 of the [ST], the TOE exposes a paravirtualized hardware device to guest VMs via the host /dev/random device, which itself is fed by a high-speed hardware noise source. Section 6.2.7 of the [ST] further provides information that the paravirtualized device is protected by mechanisms described in FDP_HBI_EXT.1 and that host-only devices are used to feed the /dev/random backend.

2.2.8.2 Tests

133 The evaluator shall perform the following tests:

- **Test 1:** The evaluator shall invoke entropy from each Guest VM. The evaluator shall verify that each VM acquires values from the interface.

Note:	This test is conducted as part of Test 2 below.

- **Test 2:** The evaluator shall invoke entropy from multiple VMs as nearly simultaneously as practicable. The evaluator shall verify that the entropy used in one VM is not identical to that invoked from the other VMs.

High-Level Test Description
Provision two identical virtual machines in the VS (referred to here as A and B) and start them. Within VM A, acquire entropy from the host-based entropy source. After a short time, do the same within VM B. Disable acquisition of entropy samples in VM B, and subsequently in VM A. Perform a similarity check between the entropy samples acquired between A and B. The test is considered failed if VM A and B acquire the same entropy. Due to the overlapping nature of the starting sequence, entropy samples in VM B will be a strict subset of those in VM A under failure conditions.

High-Level Test Description
Verify that the collected entropy samples form VM B are not found in those from VM A.
PASS

2.2.9 FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

134 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix E - Entropy Documentation and Assessment.

2.2.9.1 Tests

135 The evaluator shall also perform the following tests, depending on the standard to which the RBG conforms.

136 The evaluator shall perform 15 trials for the RBG implementation. If the RBG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RBG functionality.

137 If the RBG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. “generate one block of random bits” means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

138 If the RBG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

139 The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

- Entropy input: the length of the entropy input value must equal the seed length
- Nonce: If a nonce is supported (CTR_DRBG with no df does not use a nonce), the nonce bit length is one-half the seed length.
- Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is supported, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.
- Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

Findings: The Oracle Solaris KCF Hash_DRBG and Oracle OpenSSL 3.0.8 FOM Hash_DRBG claimed in the ST are included in C1895 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?product=12702) and A4216 (https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?validation=36826), respectively.

2.3 User Data Protection (FDP)

2.3.1 FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

2.3.1.1 TSS

140 The evaluator shall ensure that the TSS provides evidence that hardware-based isolation mechanisms are used to constrain VMs when VMs have direct access to physical devices, including an explanation of the conditions under which the TSF invokes these protections.

Findings: In section 6.3.1 of the [ST], the TSS claims that the TOE uses logical domains to provide access to physical resources. The logical domains are isolated through the SPARC hardware. This isolation mechanism is summarized at a high-level in Chapters 1.2 and 1.3 in [SUN4V].

2.3.1.2 Guidance Documentation

141 The evaluator shall verify that the operational guidance contains instructions on how to ensure that the platform-provided, hardware-based mechanisms are enabled.

Findings: Section 3.4.1 of the [AGD] describes how to ensure the platform-provided, hardware-based mechanisms are enabled. A reference to [SPARC] is included, which provides further details on configuring the platform-provided, hardware-based mechanisms.

2.3.2 FDP_PPR_EXT.1 Physical Platform Resource Controls

2.3.2.1 TSS

142 The evaluator shall examine the TSS to determine that it describes the mechanism by which the VMM controls a Guest VM's access to physical platform resources. This description shall cover all of the physical platforms allowed in the evaluated configuration by the ST. It should explain how the VMM distinguishes among Guest VMs, and how each physical platform resource that is controllable (that is, listed in the assignment statement in the first element) is identified to an Administrator.

Findings: In section 6.3.1 of the [ST], the TSS claims that the TOE uses logical domains to provide access to physical resources. The logical domains are isolated through the SPARC hardware.

According to section 6.3.2 of the [ST], domains are identified using an ID number and an alpha-numeric name. When a VM is created or edited by an administrator, the above devices are either added/configured (allowed) or not added/configured (denied) to the VM by assigning the PCI end-device to the corresponding PCI bus ID, and then assigning the bus ID to the domain.

143 The evaluator shall ensure that the TSS describes how the Guest VM is associated with each physical resource, and how other Guest VMs cannot access a physical resource without being granted explicit access. For TOEs that implement a robust interface (other than just "allow access" or "deny access"), the evaluator shall ensure that the TSS describes the possible operations or modes of access between a Guest VM's and physical platform resources.

Findings:	The [ST] in section 6.3.2 indicates that resources are protected from being shared between Guest VMs by enforcing PCI bus isolation.

144 If physical resources are listed in the second element, the evaluator shall examine the TSS and operational guidance to determine that there appears to be no way to configure those resources for access by a Guest VM. The evaluator shall document in the evaluation report their analysis of why the controls offered to configure access to physical resources can't be used to specify access to the resources identified in the second element (for example, if the interface offers a drop-down list of resources to assign, and the denied resources are not included on that list, that would be sufficient justification in the evaluation report).

Findings:	According to section 5.3.3 of the [ST], the TOE explicitly denies access to the Integrated Lights-Out Management (ILOM) function. This is further clarified in section 6.3.2 of the [ST] by indicating that the ILOM cannot be assigned to PCI bus IDs or directly to domains which prevents VMs from gaining access to it.

2.3.2.2 Guidance Documentation

145 The evaluator shall examine the operational guidance to determine that it describes how an administrator is able to configure access to physical platform resources for Guest VMs for each platform allowed in the evaluated configuration according to the ST. The evaluator shall also determine that the operational guidance identifies those resources listed in the second and third elements of the component and notes that access to these resources is explicitly denied/allowed, respectively.

Findings:	Section 3.4.1 of the [AGD] describes how to ensure the platform-provided, hardware-based mechanisms are enabled. The section states access to the ILOM function by Guest VMs is explicitly denied. Furthermore, Section 3.4.1 includes a reference to [SPARC] / Oracle VM Server for SPARC 3.6 Administration Guide, wherein detailed information about configuring physical platform resources for Guest VMs can be found starting in Section 4. Information on PCI bus configuration for Guest VMs can be found in Sections 6-10.

2.3.2.3 Tests

146 Using the operational guidance, the evaluator shall perform the following tests for each physical platform identified in the ST:

- **Test 1:** For each physical platform resource identified in the first element, the evaluator shall configure a Guest VM to have access to that resource and show that the Guest VM is able to successfully access that resource.

High-Level Test Description
Configure a Guest VM to have access to each of the grantable physical resources. Start the Guest VM and show that it can access the platform resources.
PASS

- **Test 2:** For each physical platform resource identified in the first element, the evaluator shall configure the system such that a Guest VM does not have access to that resource and show that the Guest VM is unable to successfully access that resource.

High-Level Test Description
Configure a Guest VM to not have access to each of the grantable physical resources. Start the Guest VM and show that it cannot access the platform resources.
PASS

- **Test 3** [conditional]: For TOEs that have a robust control interface, the evaluator shall exercise each element of the interface as described in the TSS and the operational guidance to ensure that the behavior described in the operational guidance is exhibited.

Note:	The TOE does not have a robust control interface.

- **Test 4** [conditional]: If the TOE explicitly denies access to certain physical resources, the evaluator shall attempt to access each listed (in FDP_PPR_EXT.1.2) physical resource from a Guest VM and observe that access is denied.

High-Level Test Description
Verify the ILOM physical platform resource is accessible from the Control Domain and is not accessible from a Guest VM.
PASS

- **Test 5** [conditional]: If the TOE explicitly allows access to certain physical resources, the evaluator shall attempt to access each listed (in FDP_PPR_EXT.1.3) physical resource from a Guest VM and observe that the access is allowed. If the operational guidance specifies that access is allowed simultaneously by more than one Guest VM, the evaluator shall attempt to access each resource listed from more than one Guest VM and show that access is allowed.

Note:	The TOE does not explicitly allow access to physical platform resources by Guest VMs.

2.3.3 FDP_RIP_EXT.1 Residual Information in Memory

2.3.3.1 TSS

147 The evaluator shall ensure that the TSS documents the process used for clearing physical memory prior to allocation to a Guest VM, providing details on when and how this is performed. Additionally, the evaluator shall ensure that the TSS documents the conditions under which physical memory is not cleared prior to allocation to a Guest VM, and describes when and how the memory is cleared.

Findings:	In section 6.3.3 of the [ST] the TSS claims that SPARC hardware is responsible for clearing memory upon allocation.

	There are no conditions where memory clearing is not performed.

2.3.4 FDP_RIP_EXT.2 Residual Information on Disk

2.3.4.1 TSS

148 The evaluator shall ensure that the TSS documents how the TSF ensures that disk storage is zeroed upon allocation to Guest VMs. Also, the TSS must document any conditions under which disk storage is not cleared prior to allocation to a Guest VM. Any file system format and metadata information needed by the evaluator to perform the below test shall be made available to the evaluator, but need not be published in the TSS.

Findings:	Section 6.3.4 of the [ST] claims that virtual disks are zeroized upon creation. In goes on to state that shared virtual disks are not cleared prior to allocation. The same section provides information on metadata found in the V5 disk format.

2.3.4.2 Tests

149 The evaluator shall perform the following test:

- **Test 1:** On the host, the evaluator creates a file that is more than half the size of a connected physical storage device (or multiple files whose individual sizes add up to more than half the size of the storage media). This file (or files) shall be filled entirely with a nonzero value. Then, the file (or files) shall be released (freed for use but not cleared). Next, the evaluator (as a VS Administrator) creates a virtual disk at least that large on the same physical storage device and connects it to a powered-off VM. Then, from outside the Guest VM, scan through and check that all the non-metadata (as documented in the TSS) in the file corresponding to that virtual disk is set to zero.

High-Level Test Description
Construct a file which takes up more than half of the physically provisioned disk space and fill it with a non-zero value. Delete the file. Create a new Guest VM but keep it powered down. Create a new virtual disk that is at least as large as half the size of the physically provisioned disk space and assign it to the Guest VM. Without powering on the Guest VM, scan the newly created virtual disk to determine if the user-data sections have been appropriately cleared prior to first use.
PASS

2.3.5 FDP_VMS_EXT.1 VM Separation

2.3.5.1 TSS

150 The evaluator shall examine the TSS to verify that it documents all inter-VM communications mechanisms (as defined above), and explains how the TSF prevents the transfer of data between VMs outside of the mechanisms listed in FDP_VMS_EXT.1.1.

Findings:	The [ST] claims, in section 6.3.5, that Guest VMs can communicate with one another via virtual networking. This claim is consistent with the SFR in section 5.3.3 of the [ST]. The [ST] section 6.3.5 claims the network interface must be explicitly configured for the VM. An administrator can configure to connect or disconnect VMs from the network.

2.3.5.2 Guidance

151 The evaluator shall examine the operational guidance to ensure that it documents how to configure all inter-VM communications mechanisms, including how they are invoked and how they are disabled.

Findings:	Section 3.4.2 of [AGD] describes how to configure all inter-VM communication mechanisms, namely, though virtual networking. The section includes a reference to [SPARC] / Oracle VM Server for SPARC 3.6 Administration Guide, with detailed information on virtual networking found in section 13.

2.3.5.3 Tests

152 The evaluator shall perform the following tests for each documented inter-VM communications channel:

- **Test 1:**

  a. Create two VMs without specifying any communications mechanism or overriding the default configuration.

  b. Test that the two VMs cannot communicate through the mechanisms selected in FDP_VMS_EXT.1.1.

  c. Create two new VMs, overriding the default configuration to allow communications through a channel selected in FDP_VMS_EXT.1.1.

  d. Test that communications can be passed between the VMs through the channel.

  e. Create two new VMs, the first with the inter-VM communications channel currently being tested enabled, and the second with the inter-VM communications channel currently being tested disabled.

  f. Test that communications cannot be passed between the VMs through the channel.

  g. As an Administrator, enable inter-VM communications between the VMs on the second VM.

  h. Test that communications can be passed through the inter-VM channel.

  i. As an Administrator again, disable inter-VM communications between the two VMs.

  j. Test that communications can no longer be passed through the channel.

FDP_VMS_EXT.1.2 is met if communication is unsuccessful in step (b). FDP_VMS_EXT.1.3 is met if communication is successful in step (d) and unsuccessful in step (f).

High-Level Test Description
Create two VMs without providing any networking communications between them. Attempt to invoke the communications channel to transmit data between the Guest VMs and show this fails. Enable networking on each of the Guest VMs and then restart them to show that they can communicate between the shared channel. On one VM, disable the networking interface. Show that the Guest VMs can no longer communicate. Re-enable the networking interface and show that the Guest VMs can communicate once again.
PASS

2.3.6 FDP_VNC_EXT.1 Virtual Networking Components

2.3.6.1 TSS

153 The evaluator shall examine the TSS (or a proprietary annex) to verify that it describes the mechanism by which virtual network traffic is ensured to be visible only to Guest VMs configured to be on that virtual network.

Findings:	The TSS in section 6.3.6 of the [ST] indicates that an administrator is required to ensure that guest VMs are configured to be part of the virtual network (i.e. logical separation enforced by hardware means). Members of the network group can communicate with one another.

2.3.6.2 Guidance Documentation

154 The evaluator must ensure that the Operational Guidance describes how to create virtualized networks and connect VMs to each other and to physical networks.

Findings:	Section 3.4.2 and 3.4.4 of [AGD] describe how to configure virtual and physical networking for inter-VM and physical network communications. The sections include a reference to [SPARC] / Oracle VM Server for SPARC 3.6 Administration Guide, with detailed information on virtual and physical networking for VMs found in Section 13. A reference to [SOLARIS] / Solaris 11.4 Network Administration Cheatsheet is provided in Section 3.4.4 of [AGD], wherein detailed information on virtual and physical networking for the Solaris 11.4 OS can be found.

2.3.6.3 Tests

155 **Test 1**: The evaluator shall assume the role of the Administrator and attempt to configure a VM to connect to a network component. The evaluator shall verify that the attempt is successful. The evaluator shall then assume the role of an unprivileged

user and attempt the same connection. If the attempt fails, or there is no way for an unprivileged user to configure VM network connections, the requirement is met.

High-Level Test Description
As a privileged administrator, log into the control domain and configure a Guest VM to make use of virtual networking and show that the configuration is accepted. As an unprivileged user, log into the control domain and attempt to configure a Guest VM to make use of virtual networking and show that the configuration is not accepted.
PASS

156 **Test 2**: The evaluator shall assume the role of the Administrator and attempt to configure a VM to connect to a physical network. The evaluator shall verify that the attempt is successful. The evaluator shall then assume the role of an unprivileged user and make the same attempt. If the attempt fails, or there is no way for an unprivileged user to configure VM network connections, the requirement is met.

High-Level Test Description
As a privileged administrator, log into the control domain and configure a Guest VM to make use of physical networking and show that the configuration is accepted. As an unprivileged user, log into the control domain and attempt to configure a Guest VM to make use of physical networking and show that the configuration is not accepted.
PASS

2.4 Identification and Authentication (FIA)

2.4.1 FIA_AFL_EXT.1 Authentication Failure Handling

2.4.1.1 Tests

157 The evaluator shall perform the following tests for each credential selected in FIA_AFL_EXT.1.1:

158 The evaluator will set an Administrator-configurable threshold n for failed attempts, or note the ST-specified assignment.

- **Test 1:** The evaluator will attempt to authenticate remotely with the credential n-1 times. The evaluator will then attempt to authenticate using a good credential and verify that authentication is successful.

Note: This test is covered in Test 2 below.

- **Test 2:** The evaluator will make n attempts to authenticate using a bad credential. The evaluator will then attempt to authenticate using a good credential and verify that the attempt is unsuccessful. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.

High-Level Test Description
Configure the TOE to permit a certain number of login attempts before locking the user account. Using a bad credential, attempt to login multiple times to lock the user account. Attempt to login to the locked user account using a good credential and show that the login attempt fails.

High-Level Test Description
PASS

After reaching the limit for unsuccessful authentication attempts the evaluator will proceed as follows:

- **Test 1:** If the Administrator action selection in FIA_AFL_EXT.1.2 is selected, then the evaluator will confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote Administrator's access results in successful access (when using valid credentials for that Administrator).

Note:	This functionality is covered in the previous test case.

- **Test 2:** If the time period selection in FIA_AFL_EXT.1.2 is selected, the evaluator will wait for just less than the time period configured and show that an authentication attempt using valid credentials does not result in successful access. The evaluator will then wait until just after the time period configured and show that an authentication attempt using valid credentials results in successful access.

Note:	This functionality is covered in the previous test case.

2.4.2 FIA_UAU.5 Multiple Authentication Mechanisms

2.4.2.1 Tests

159 If 'username and password authentication' is selected, the evaluator will configure the VS with a known username and password and conduct the following tests:

- **Test 1**: The evaluator will attempt to authenticate to the VS using the known username and password. The evaluator will ensure that the authentication attempt is successful.

Note:	The evaluator demonstrated successful authentication with a known username and password for SSH during the testing for FIA_AFL_EXT.1.

- **Test 2**: The evaluator will attempt to authenticate to the VS using the known username but an incorrect password. The evaluator will ensure that the authentication attempt is unsuccessful.

Note:	The evaluator demonstrated unsuccessful authentication with a known username and incorrect password for SSH during the testing for FIA_AFL_EXT.1.

160 If 'username and PIN that releases an asymmetric key' is selected, the evaluator will examine the TSS for guidance on supported protected storage and will then configure the TOE or OE to establish a PIN which enables release of the asymmetric key from the protected storage (such as a TPM, a hardware token, or isolated execution environment) with which the VS can interface. The evaluator will then conduct the following tests:

- **Test 1**: The evaluator will attempt to authenticate to the VS using the known user name and PIN. The evaluator will ensure that the authentication attempt is successful.

- **Test 2**: The evaluator will attempt to authenticate to the VS using the known user name but an incorrect PIN. The evaluator will ensure that the authentication attempt is unsuccessful.

Note:	The TOE does not claim username and PIN functionality and therefore these test cases are not conducted.

161 If ‘X.509 certificate authentication‘ is selected, the evaluator will generate an X.509v3 certificate for an Administrator user with the Client Authentication Enhanced Key Usage field set. The evaluator will provision the VS for authentication with the X.509v3 certificate. The evaluator will ensure that the certificates are validated by the VS as per FIA_X509_EXT.1.1 and then conduct the following tests:

- **Test 1**: The evaluator will attempt to authenticate to the VS using the X.509v3 certificate. The evaluator will ensure that the authentication attempt is successful.

- **Test 2**: The evaluator will generate a second certificate identical to the first except for the public key and any values derived from the public key. The evaluator will attempt to authenticate to the VS with this certificate. The evaluator will ensure that the authentication attempt is unsuccessful.

Note:	The TOE does not claim X.509 certificate authentication functionality and therefore these test cases are not conducted.

162 If ‘SSH public-key credential authentication‘ is selected, the evaluator shall generate a public-private host key pair on the TOE using RSA or ECDSA, and a second public-private key pair on a remote client. The evaluator shall provision the VS with the client public key for authentication over SSH, and conduct the following tests:

- **Test 1**: The evaluator will attempt to authenticate to the VS using a message signed by the client private key that corresponds to provisioned client public key. The evaluator will ensure that the authentication attempt is successful.

- **Test 2**: The evaluator will generate a second client key pair and will attempt to authenticate to the VS with the private key over SSH without first provisioning the VS to support the new key pair. The evaluator will ensure that the authentication attempt is unsuccessful.

Note:	The above SSH public-key credential test cases are conducted as part of FCS_SSHS_EXT.1.

2.4.3 FIA_UIA_EXT.1 Administrator Identification and Authentication

2.4.3.1 TSS

163 The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a “successful logon.” The evaluator shall examine the operational guidance to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-

shared keys, tunnels, certificates) to logging in are described. For each supported login method, the evaluator shall ensure the operational guidance provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the operational guidance provides sufficient instruction on limiting the allowed services.

Findings:	Section 6.4.4 of the [ST] provides information on the logon process for each login method as well as describing what constitutes a successful logon.

2.5 Security Management (FMT)

2.5.1 FMT_MOF_EXT.1 Management of Security Functions Behavior

2.5.1.1 TSS

164 The evaluator shall examine the TSS and Operational Guidance to ensure that it describes which security management functions require Administrator privilege and the actions associated with each management function. The evaluator shall verify that for each management function and role specified in the FMT_MOF_EXT.1.1 Server Virtualization Management Functions Table (Table 3), the defined role is able to perform all mandatory functions as well as all optional or selection-based functions claimed in the ST.

Findings:	Section 6.5.1 of the [ST] points back to an authoritative table in the SFR in section 5.3.5 of the [ST] indicating which functions are provided to which roles. The evaluator considered each of the functions defined and found that all mandatory functions and optional and selection-based functions were properly claimed. Those which were not claimed are marked with an "N".

2.5.1.2 Guidance Documentation

165 The evaluator shall examine the Operational Guidance to ensure that it describes how the Administrator or User are able to perform each management function that the ST claims the TOE supports.

Findings:	The evaluator examined the [AGD], [SPARC] and [SOLARIS] guidance resources and determined that they describe how the Administrator or User can perform each claimed management function, as listed in Table 12 of the [ST].

166 The evaluator shall verify for each claimed management function that the Operational Guidance is sufficiently detailed to allow the function to be performed.

Findings:	The evaluator used the [AGD], [SPARC] and [SOLARIS] guidance resources to exercise each of the claimed management functions and, in doing so, determined that they were sufficiently detailed to allow the function to be performed by an Administrator or User.

2.5.1.3 Tests

167 The evaluator shall test each management function for each role listed in the FMT_MOF_EXT.1.1 Server Virtualization Management Functions Table (Table 3) in the ST to demonstrate that the function can be performed by the roles that are authorized to do so and the result of the function is demonstrated. The evaluator shall also verify for each claimed management function that if the TOE claims not to provide

a particular role with access to the function, then it is not possible to access the TOE as that role and perform that function.

High-Level Test Description
For each of the claimed management functions which have not been tested in other test cases, perform the appropriate function as an administrator. Verify the expected result occurs and that the appropriate audit log entry is generated. As a non-administrative user, attempt to perform the same function, where applicable. Verify the expected result occurs and that the appropriate audit log entry is generated.
PASS

2.5.2 FMT_SMO_EXT.1 Separation of Management and Operational Networks

2.5.2.1 TSS

168 The evaluator shall examine the TSS to verify that it describes how management and operational traffic is separated.

Findings:	Section 6.5.2 of the [ST] indicates that separated networks can be implemented via virtual and physical networking.

2.5.2.2 Guidance Documentation

169 The evaluator shall examine the operational guidance to verify that it details how to configure the VS to keep Management and Operational traffic separate.

Findings:	Section 3.4.4 of [AGD] describes how to configure the VS to keep Management and Operational traffic separate. The section includes a reference to the [SPARC] / Oracle VM Server for SPARC 3.6 Administration Guide and [SOLARIS] / Solaris 11.4 Network Administration Cheatsheet guidance resources, which provide detailed information on configuration of virtual and physical networks for VMs and the Solaris 11.4 OS.

2.5.2.3 Tests

170 The evaluator shall configure the TOE as documented in the guidance. If separation is logical, then the evaluator shall capture packets on the management network. If plaintext Guest network traffic is detected, the requirement is not met.

171 If separation uses trusted channels, then the evaluator shall capture packets on the network over which traffic is tunneled. If plaintext Guest network traffic is detected, the requirement is not met.

172 If data encryption is used, then the evaluator shall capture packets on the network over which the data is sent while a VM or other large data structure is being transmitted. If plaintext VM contents are detected, the requirement is not met.

High-Level Test Description
Capture traffic on the Management Network interface and generate traffic to and from the management components. At the same time, generate traffic between Guest VMs on the Guest Network and traffic between Guest VMs and an external physical device on the Guest Network. Verify Guest VM traffic does not appear on the Management Network.

High-Level Test Description
PASS

2.6 Protection of the TSF (FPT)

2.6.1 FPT_DVD_EXT.1 Non-Existence of Disconnected Virtual Devices

2.6.1.1 Tests

173 The evaluator shall connect a device to a VM, then from within the guest scan the VM's devices to ensure that the connected device is present--using a device driver or other available means to scan the VM's I/O ports or PCI Bus interfaces. (The device's interface should be documented in the TSS under FPT_VDP_EXT.1.) The evaluator shall remove the device from the VM and run the scan again. This requirement is met if the device's interfaces are no longer present.

High-Level Test Description
Configure a Guest VM to have access to at least one of the devices claimed in FPT_VDP_EXT.1 and show that the device is accessible. Then reconfigure the Guest VM to remove the device and show that the Guest VM is no longer able to access it.
PASS

2.6.2 FPT_EEM_EXT.1 Execution Environment Mitigations

2.6.2.1 TSS

174 The evaluator shall examine the TSS to ensure that it states, for each platform listed in the ST, the execution environment-based vulnerability mitigation mechanisms used by the TOE on that platform. The evaluator shall ensure that the lists correspond to what is specified in FPT_EEM_EXT.1.1.

Findings:	In section 6.6.2 of the [ST], the document describes that for the management and configuration components of the TOE which operate on the Solaris trusted control domain, the Solaris OS includes address space layout randomization (ASLR), memory execution protection, and stack and heap overflow protection to protect running components. As guest virtual machines operate independently of the Solaris control domain, guest VMs are isolated using mechanisms implemented by the SPARC hardware.

2.6.3 FPT_HAS_EXT.1 Hardware Assists

2.6.3.1 TSS

175 The evaluator shall examine the TSS to ensure that it states, for each platform listed in the ST, the hardware assists and memory-handling extensions used by the TOE on that platform. The evaluator shall ensure that these lists correspond to what is specified in the applicable FPT_HAS_EXT component.

Findings:	Section 6.6.3 of the [ST] indicates that no binary translations or shadow page tables are required.

2.6.4 FPT_HCL_EXT.1 Hypercall Controls

2.6.4.1 TSS

176 The evaluator shall examine the TSS (or proprietary TSS Annex) to ensure that all hypercall functions are documented at the level necessary for the evaluator to run the below test. Documentation for each hypercall interface must include: how to invoke the interface, parameters and legal values, and any conditions under which the interface can be invoked (e.g., from guest user mode, guest privileged mode, during guest boot only).

Findings:	Section 6.6.4 of the [ST] states, "Hypercalls are enabled by default and cannot be disabled." Furthermore, the section provides a reference to the [SUNV4] external public resource, which documents the hypercall functions of the TOE. The section indicates a summary of all hypercalls can be found in section A.5 of [SUNV4] and detailed descriptions for each can be found in chapters 11-27 and 31. [SUNV4] provides significant information on the hypercall interface, methods of invocation, parameters and legal value (ranges or conditions) and whether the hypercall can be executed by guest VMs or from supervisory VMs. For example, in chapter 25 (Cryptographic Services), section 25.1 is information pertinent to trusted domains and section 25.2 is pertinent to untrusted domains (e.g. Guest VMs).

2.6.4.2 Guidance Documentation

177 There is no operational guidance for this component.

2.6.4.3 Tests

178 The evaluator shall perform the following test:

179 For each hypercall interface documented in the TSS or proprietary TSS Annex, the evaluator shall attempt to invoke the function from within the VM using an invalid parameter (if any). If the VMM or VS crashes or generates an exception, or if no error is returned to the guest, then the test fails. If an error is returned to the guest, then the test succeeds.

High-Level Test Description
Using a test harness within a Guest VM, attempt to call out to the hypercall interface with an invalid parameter and show that the hypercall returns an error back to the Guest VM.
PASS

2.6.5 FPT_RDM_EXT.1 Removable Devices and Media

2.6.5.1 TSS

180 The evaluator shall examine the TSS to ensure it describes the association between the media or devices supported by the TOE and the actions that can occur when switching information domains.

Findings:	Section 6.6.5 of the [ST] indicates that removable media must be associated by means of explicit configuration. Since the removable device is on a specific PCI bus; and since the PCI bus must be assigned explicitly to a VM by an administrator at VM configuration time, there are no instances where a removable device (and therefore removable media) can be transferred to another information domain without explicit reconfiguration by the administrator to reassign the PCI buses.

2.6.5.2 Guidance Documentation

181 The evaluator shall examine the operational guidance to ensure it documents how an administrator or user configures the behavior of each media or device.

Findings:	Sections 6, 7, 8, 9, 10, 11 and 13 under the [SPARC] / Oracle VM Server for SPARC 3.6 Administration Guide guidance resource provides detailed information on how an administrator or user can configure the behaviour of each media or device.

2.6.5.3 Tests

182 The evaluator shall perform the following test for each listed media or device:

- **Test 1:** The evaluator shall configure two VMs that are members of different information domains, with the media or device connected to one of the VMs. The evaluator shall disconnect the media or device from the VM and connect it to the other VM. The evaluator shall verify that the action performed is consistent with the action assigned in the TSS.

High-Level Test Description
Configure a Guest VM without any explicit PCI device assignment and another Guest VM with an explicit PCI device assignment. Show that the PCI device can only be accessed by the Guest VM to which the device was assigned. Reassign the PCI device to the other Guest VM. Show that the PCI device can only be accessed by the Guest VM to which the device was assigned. Connect a removable device to a VM via an assigned PCI device. Show that the removable device is accessible only from the VM to which the PCI device is assigned.
PASS

2.6.6 FPT_TUD_EXT.1 Trusted Updates to the Virtualization System

2.6.6.1 TSS

183 The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system software. Updates to the TOE either have a hash associated with them, or are signed by an authorized source. The evaluator shall verify that the description includes either a digital signature or published hash verification of the software before installation and that installation fails if the verification fails. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the update, and the actions that take place for both successful and unsuccessful verification. If digital

signatures are used, the evaluator shall also ensure the definition of an authorized source is contained in the TSS.

Findings:	According to section 6.6.6 of the [ST] the TOE leverages 2048-bit RSA digital signature mechanism for verification of packages, which is included within a package's metadata. Packages are signed by Oracle-issued private keys and the corresponding trusted public keys are stored on the TOE within X.509 certificates under /etc/certs/CA. The TSS claims that the digital signature is checked prior to package installation. Failures in digital signatures prevent the package from being installed; successful digital signature checks allow the package to be installed. Packages are provided by the Image Packaging System (IPS) which are stored in repositories which are populated by IPS publishers. The default primary IPS publisher is for Solaris and is published at https://pkg.oracle.com/solaris/release/. However, end-users often implement their own IPS repository within their own network which can include packages copied from the official Oracle repositories.

184 If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or administrator guidance) describes how the certificates are installed/updated/selected, if necessary.

Findings:	The [ST] does not indicate that a certificate-based mechanism is used for digital signature verification of software updates. However, according to section 6.6.6 of the [ST], X.509 certificates are used as a means of storing trusted public keys used to verify the digital signatures.

2.6.6.2 Tests

185 The evaluator shall perform the following tests:

- **Test 1**: The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that it is successfully installed on the TOE. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update.

High-Level Test Description
Using a network-based package repository, query for updates, install a package and verify that the package was installed.
PASS

- **Test 2**: The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

    1) A modified version (e.g., using a hex editor) of a legitimately signed or hashed update

2) An image that has not been signed/hashed

3) An image signed with an invalid hash or invalid signature (e.g., by using a different key as expected for creating the signature or by manual modification of a legitimate hash/signature)

High-Level Test Description
Using a package in the IPS manager, modify the binary to fail the signature verification activity. Attempt to install the binary and show it fails. Using a package in the IPS manager, remove the signature from the package. Attempt to install the binary and show it fails. Finally, attempt the install operation again with a known-good package and signature and show that the installation succeeds.
PASS

2.6.7 FPT_VDP_EXT.1 Virtual Device Parameters

2.6.7.1 TSS

186 The evaluator shall examine the TSS to ensure it lists all virtual devices accessible by the guest OS. The TSS, or a separate proprietary document, must also document all virtual device interfaces at the level of I/O ports or PCI Bus interfaces - including port numbers (absolute or relative to a base), port name, address range, and a description of legal input values.

Findings:	Section 6.6.7 of the [ST] indicates that a small number of virtualized devices are provided to guest VMs. These include virtual disks, virtual networking, and virtual SCSI host-bus adaptors (HBA). Proprietary documentation in [VIOP] provided the specifics of the interface protocol and was found to contain the necessary information regarding how guest VMs call into virtualized devices. Rather than being specified using “ports” or “PCI bus interfaces”, virtual devices in SPARC are implemented using specific LDC hypercalls as described in section 1.5 and 1.7 of [SUN4V]. These hypercalls form the fundamental virtual channels through which physical devices or abstracted services communicate with the guest domains. The SPARC Virtual I/O (VIO) protocol is a mechanism for implementing virtual I/O devices in a SPARC server. It allows virtual devices, such as network interfaces or disk controllers, to be created and assigned to guest domains. The VIO protocol provides a layer of abstraction between the guest domains and the physical I/O devices, allowing for greater flexibility in managing and sharing hardware resources. The hypervisor specification in [SUN4V] and the protocol described in the [VIOP] document are at a granularity that would allow a reader to implement a virtual device driver connected to a physical device or abstracted virtual service running in another domain.

187 The TSS must also describe the expected behavior of the interface when presented with illegal input values. This behavior must be deterministic and indicative of parameter checking by the TSF.

Findings:	As per section 6.6.7 of the [ST], illegal parameters are rejected.

188 The evaluator must ensure that there are no obvious or publicly known virtual I/O ports missing from the TSS.

Findings:	Section 6.6.7 of the [ST] does not appear to omit any obvious or publicly known I/O ports. The fundamental mechanism of access to virtual devices (as described in [VIOP]) is actually via logical domain channels (LDCs) using hypercalls. The hypercalls specification in [SUN4V] provides a complete input and output specification.

189 There is no expectation that evaluators will examine source code to verify the "all" part of the evaluation activity.

2.6.7.2 Tests

190 For each virtual device interface, the evaluator shall attempt to access the interface using at least one parameter value that is out of range or illegal. The test is passed if the interface behaves in the manner documented in the TSS. Interfaces that do not have input parameters need not be tested. This test can be performed in conjunction with the tests for FPT_DVD_EXT.1.

Note:	Virtual device interfaces are established entirely by LDC hypercalls described in section 22 of [SUN4V]. The hypercalls are LDC_TX_QCONF (0xe0), LDC_TX_QINFO (0xe5), LDC_TX_GETSTATE (0xe2), LDC_TX_SET_QTAIL (0xe3), LDC_RX_QCONF (0xe4), LDC_RX_QINFO (0xe5), LDC_RX_GET_STATE (0xe6), LDC_RX_SET_QHEAD (0xe7), LDC_SET_MAP_TABLE (0xea), LDC_GET_MAP_TABLE (0xeb), LDC_COPY (0xec), LDC_MAPIN (0xed), LDC_UNMAP (0xee), and LDC_REVOKE (0xef). Test cases in FPT_HCL_EXT.1 exercise the LDC hypervisor calls with invalid data that return exit codes back to the guest, consistent with [SUN4V]. Therefore, FPT_HCL_EXT.1 exercises each of the necessary virtual device interfaces necessary to meet this test case.

2.6.8 FPT_VIV_EXT.1 VMM Isolation from VMs

2.6.8.1 TSS

191 The evaluator shall verify that the TSS (or a proprietary annex to the TSS) describes how the TSF ensures that guest software cannot degrade or disrupt the functioning of other VMs, the VMM or the platform. And how the TSF prevents guests from invoking higher-privilege platform code, such as the examples in the note.

Findings:	Section 6.6.8 of the [ST] claims that the TSF prevents guest VMs from degrading or disrupting the functioning of other VMs by virtue of segregating low-level resource access and provisioning functions to either the ILOM or the control domain. The ILOM is a separate part of the system with its own CPU and memory, inside the physical chassis, running its own embedded OS. Only the control domain has the ability to reconfigure platform resources (CPU, memory, PCI).

2.7 TOE Access (FTA)

2.7.1 FTA_TAB.1 TOE Access Banner

2.7.1.1 Tests

192 The evaluator shall configure the TOE to display the advisory warning message "TEST TEST Warning Message TEST TEST". The evaluator shall then log out and confirm that the advisory message is displayed before login can occur.

High-Level Test Description
As an authorized administrator, change the banner as directed and ensure that subsequent logins are presented with this banner before a successful session is established.
PASS

2.8 Trusted path/channels (FTP)

2.8.1 FTP_ITC_EXT.1 Trusted Channel Communications

2.8.1.1 TSS

193 The evaluator will review the TSS to determine that it lists all trusted channels the TOE uses for remote communications, including both the external entities and remote users used for the channel as well as the protocol that is used for each.

Findings:	The TSS in section 6.8.1 of the [ST] lists the trusted channels that the TOE uses for remote communications. These are consistent with the selections in FTP_ITC_EXT.1 in section 5.3.8 of the [ST]. The TSS indicates that TLS is used for syslog and SSH is used for the CLI (administrative access).

2.8.1.2 Tests

194 The evaluator will configure the TOE to communicate with each external IT entity and type of remote user identified in the TSS. The evaluator will monitor network traffic while the VS performs communication with each of these destinations. The evaluator will ensure that for each session a trusted channel was established in conformance with the protocols identified in the selection.

High-Level Test Description
Capture traffic on the management network of the TOE and login via SSH, thereby generating traffic on each trusted channel (SSH, Rsyslog). Verify the traffic does not contain any plaintext packets between the remote endpoints using those claimed channels and that the network traces correspond to the protocols identified in the claims.
PASS

2.8.2 FTP_UIF_EXT.1 User Interface: I/O Focus

2.8.2.1 TSS

195 The evaluator shall ensure that the TSS lists the supported user input devices.

Findings:	Section 6.8.3 of the [ST] states that the TOE supports keyboard devices over the SSH CLI.

2.8.2.2 Guidance Documentation

196 The evaluator shall ensure that the operational guidance specifies how the current input focus is indicated to the user.

Findings:	Section 3.4.3 of [AGD] specifies how the current input focus is indicated to the user.

2.8.2.3 Tests

197 For each supported input device, the evaluator shall demonstrate that the input from each device listed in the TSS is directed to the VM that is indicated to have the input focus.

High-Level Test Description
Create two guest domains, vm01 and vm02. Attempt to access the local console for vm01. Show that the UI unambiguously indicates which logical domain is being accessed.
PASS

2.8.3 FTP_UIF_EXT.2 User Interface: Identification of VM

2.8.3.1 TSS

198 The evaluator shall ensure that the TSS describes the mechanism for identifying VMs to the user, how identities are assigned to VMs, and how conflicts are prevented.

Findings:	In section 6.8.4 of the [ST], VMs are given unique names at provisioning time and any attempt to duplicate a VM is prevented by the administrative CLI toolset.

2.8.3.2 Tests

199 The evaluator shall perform the following test:

200 The evaluator shall attempt to create and start at least three Guest VMs on a single display device where the evaluator attempts to assign two of the VMs the same identifier. If the user interface displays different identifiers for each VM, then the requirement is met. Likewise, the requirement is met if the system refuses to create or start a VM when there is already a VM with the same identifier.

High-Level Test Description
Attempt to create a VM with the same name as a VM from the previous test case. Show that the VM cannot be created. Attempt to rename a VM to have the same name as a VM from the previous test case. Show that the VM cannot be renamed.
PASS

3 Evaluation Activities for Optional Requirements

201 No optional requirements have been selected by this evaluation.

4 Evaluation Activities for Selection-Based Requirements

4.1.1 FCS_TLS_EXT.1 TLS Protocol

4.1.1.1 Guidance Documentation

202 The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Findings:	The evaluator reviewed the selection under FCS_TLS_EXT.1 of section 5.3.2 of the [ST] and confirmed it was consistent with those made under FTP_ITC_EXT.1 of section 5.3.8 of the [ST] and the guidance given in sections 3.2.4.6 and 4 of the [AGD].

4.1.2 FCS_TLSC_EXT.1 TLS Client Protocol

4.1.2.1 FCS_TLSC_EXT.1.1 TSS

203 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

Findings:	Section 6.2.9 of the [ST] provides the list of TLS ciphersuites the TOE is restricted in accepting. This list is equivalent to the ciphersuites provided in section 5.3.2 of the [ST].

4.1.2.2 FCS_TLSC_EXT.1.1 Guidance Documentation

204 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the product so that TLS conforms to the description in the TSS.

Findings:	The evaluator reviewed the TLS client description given in section 6.2.9 of the [ST] and confirmed it was consistent with the TLS client configuration given in section 4.2 of the [AGD].

4.1.2.3 FCS_TLSC_EXT.1.1 Tests

205 The evaluator shall also perform the following tests:

- **Test 1**: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

High-Level Test Description
Configure a TLS server in the TOE environment to offer a single supported ciphersuite. Restart the rsyslog client on the TOE and verify a successful TLS connection to the test server is made using that ciphersuite. Repeat the test for all claimed ciphersuites.
PASS

- **Test 2**: The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.

  The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

High-Level Test Description
Construct two X.509 certificates, one missing the serverAuth permission in the extendedKeyUsage extension and another missing the extendedKeyUsage extension altogether. In turn, serve each certificate to the TOE from a test server. Verify the TOE rejects the invalid certificate in each case and fails to establish the connection. Show that the TOE audits the reason for the failure.
PASS

- **Test 3**: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 4**: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 5**: The evaluator shall perform the following modifications to the traffic:

- **Test 5.1**: Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 5.2**: Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 5.3**: [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 5.4**: Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 5.5**: [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 5.6**: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

- **Test 5.7**: Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.

High-Level Test Description
Using a custom tool, perform the test as directed. Show that the TOE fails to negotiate a TLS handshake.
PASS

4.1.2.4 FCS_TLSC_EXT.1.2 TSS

206 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported (e.g. Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the product.

Findings:	Section 6.2.9 of the [ST] states that the TOE will compare application-configured reference identifiers to compare with those found in X.509 certificates. The TSS states that Subject Alternative Name DNS names and IP addresses are supported in the evaluated configuration. Wildcards are supported and certificate pinning is not supported.

4.1.2.5 FCS_TLSC_EXT.1.2 Guidance Documentation

207 The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Findings:	The evaluator reviewed the TLS client configuration given in section 4.2 of the [AGD] and verified instructions for setting the TLS server reference identifier were included. Specifically, the section states that the reference identifier can be configured through the “StreamDriverPermittedPeers” option in the rsyslog configuration file.

4.1.2.6 FCS_TLSC_EXT.1.2 Tests

208 **[TD0499]** The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7.

- **Test 1**: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

  Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

High-Level Test Description
Create an X.509 certificate which meets the test requirements and deliver it to the TOE from a test TLS server. Show that the connection fails to be established and no application data flows from the TOE client to the test server. Show that the appropriate audit message is received.
PASS

- **Test 2**: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

High-Level Test Description
Create an X.509 certificate which meets the test requirements and deliver it to the TOE from a test TLS server. Show that the connection fails to be established and no application data flows from the TOE client to the test server. Show that the appropriate audit message is received.
PASS

- **Test 3**: [conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

High-Level Test Description
Create an X.509 certificate which meets the test requirements and deliver it to the TOE from a test TLS server. Show that the connection succeeds and application data flows from the TOE client to the test server.

High-Level Test Description
PASS

- **Test 4**: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

High-Level Test Description
Create an X.509 certificate which meets the test requirements and deliver it to the TOE from a test TLS server. Show that the connection succeeds and application data flows from the TOE client to the test server.
PASS

- **Test 5**: The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.

  - **Test 5.1**: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

High-Level Test Description
Create an X.509 certificate which meets the test requirements and deliver it to the TOE from a test TLS server. Modify the TOE reference identifier to be "foo.bar.example.com". Show that the connection fails to be established and no application data flows from the TOE client to the test server. Show that the appropriate audit message is received. Repeat for both the CN and a DNS SAN type.
PASS

  - **Test 5.2**: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

High-Level Test Description
Modify the TOE reference identifier to be "foo.example.com". Deliver an X.509 certificate that meets the requirements from a test TLS server to the TOE. Show that the connection succeeds and application data flows from the TOE client to the test server.

High-Level Test Description
Modify the TOE reference identifier to be the apex name of “example.com”. Show that the connection fails and no application data flows from TOE client to the test server. Show that the appropriate audit message is received. Modify the TOE reference identifier to be “foo.bar.example.com”. Show that the connection fails and no application data flows from TOE client to the test server. Show that the appropriate audit message is received. Repeat for both the CN and a DNS SAN type.
PASS

- **Test 5.3**: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.

High-Level Test Description
Modify the TOE’s reference identifier to be “foo.com” and deliver the certificate described by the test from a test TLS server to the TOE. Show that the connection fails and application data does not flow from the TOE client to the test server. Modify the TOE’s reference identifier to be “bar.foo.com” and deliver the certificate described by the test from a test TLS server to the TOE. Show that the connection fails and application data does not flow from the TOE client to the test server. Repeat for both the CN and a DNS SAN type. Show that the appropriate audit messages are received in each case.
PASS

- **Test 5.4**: [conditional]: If wildcards are not supported, the evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection fails.

Note:	The TOE supports wildcards and therefore this test does not apply.

- **Test 6**: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

Note:	The TOE does not support URI or SrvName reference identifiers and therefore this test does not apply.

- **Test 7**: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

Note:	The TOE does not support pinned certificates and therefore this test does not apply.

4.1.2.7 FCS_TLSC_EXT.1.3 TSS

209 If the selection for authorizing override of invalid certificates is made, then the evaluator shall ensure that the TSS includes a description of how and when user or administrator authorization is obtained. The evaluator shall also ensure that the TSS describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.

Findings:	Section 5.3.2 of the [ST] indicates that the TOE does not support any override mechanisms in the evaluated configuration.

4.1.2.8 FCS_TLSC_EXT.1.3 Tests

210 The evaluator shall demonstrate that using an invalid certificate (unless excepted) results in the function failing as follows, unless excepted:

- **[TD0513]** Test 1a: The evaluator shall demonstrate that a server using a certificate with a valid certification path successfully connects.

- **[TD0513]** Test 1b: The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.

Note:	Test 1a and 1b are conducted as part of FIA_X509_EXT.2 Test 1.

- **[TD0513]** Test 1c [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.

Note:	Test 1c is conducted as part of FIA_X509_EXT.2 Test 1.

- **Test 2**: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.

Note:	Test 2 is conducted as part of FIA_X509_EXT.1 Test 3.

- **Test 3**: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.

Note:	Test 3 is conducted as part of FIA_X509_EXT.1 Test 2.

- **Test 4**: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.

Note:	Test 4 is conducted as part of FCS_TLSC_EXT.1.2 test cases.

4.1.3 FCS_SSH_EXT.1 SSH Protocol

4.1.3.1 FCS_SSH_EXT.1.1 TSS

211 The evaluator shall ensure that the selections indicated in the ST are consistent with selections in this and subsequent components. Otherwise, this SFR is evaluated by activities for other SFRs.

Findings:	Section 6.2.10 of the [ST] contains the necessary information and has been found to be consistent with the selections made in section 5.3.2 of the [ST].

4.1.3.2 FCS_SSH_EXT.1.1 Guidance

212 There are no guidance evaluation activities for this component. This SFR is evaluated by activities for other SFRs.

4.1.3.3 FCS_SSH_EXT.1.1 Tests

213 There are no test evaluation activities for this component. This SFR is evaluated by activities for other SFRs.

4.1.3.4 FCS_SSH_EXT.1.2 TSS

214 The evaluator shall check to ensure that the authentication methods listed in the TSS are identical to those listed in this SFR component; and, ensure if password-based authentication methods have been selected in the ST then these are also described; and, ensure that if keyboard-interactive is selected, it describes the multifactor authentication mechanisms provided by the TOE.

Findings:	Section 6.2.10 of the [ST] contains the necessary information and has been found to be consistent with the selections made in section 5.3.2 of the [ST]. Password-based authentication is selected in section 5.3.2 and is described in section 6.2.10.

4.1.3.5 FCS_SSH_EXT.1.2 Guidance

215 The evaluator shall check the guidance documentation to ensure the configuration options, if any, for authentication mechanisms provided by the TOE are described.

Findings:	Section 3.3.1 in [AGD] describes the configuration options for the SSH authentication mechanisms provided by the TOE. The section includes a reference to the SSHD section of the [SOLARIS] guidance resource which provides detailed information on configuration of the SSHD service.

4.1.3.6 FCS_SSH_EXT.1.2 Tests

216 **Test 1**: [conditional] If the TOE is acting as SSH Server:

a. The evaluator shall use a suitable SSH Client to connect to the TOE, enable debug messages in the SSH Client, and examine the debug messages to determine that only the configured authentication methods for the TOE were offered by the server.

High-Level Test Description
Using OpenSSH's SSH client, connect to the TOE to show the advertised authentication methods of the TOE SSH server. Verify only the configured authentication methods for the TOE were offered by the server.
PASS

b. [conditional] If the SSH server supports X509 based Client authentication options:

   a. The evaluator shall initiate an SSH session from a client where the username is associated with the X509 certificate. The evaluator shall verify the session is successfully established.

   b. Next the evaluator shall use the same X509 certificate as above but include a username not associated with the certificate. The evaluator shall verify that the session does not establish.

   c. Finally, the evaluator shall use the correct username (from step a above) but use a different X509 certificate which is not associated with the username. The evaluator shall verify that the session does not establish.

Findings:	The TOE does not claim use of X.509 for SSH Client authentication.

217 **Test 2**: [conditional] If the TOE is acting as SSH Client, the evaluator shall test for a successful configuration setting of each authentication method as follows:

a. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established.

b. Next, the evaluator shall use bad authentication data (e.g. incorrectly generated certificate or incorrect password) and ensure that the connection is rejected.

218 Steps a-b shall be repeated for each independently configurable authentication method supported by the server.

Findings:	The TOE does not claim SSH Client functionality.

219 **Test 3**: [conditional] If the TOE is acting as SSH Client, the evaluator shall verify that the connection fails upon configuration mismatch as follows:

a. The evaluator shall configure the Client with an authentication method not supported by the Server.

b. The evaluator shall verify that the connection fails.

220 If the Client supports only one authentication method, the evaluator can test this failure of connection by configuring the Server with an authentication method not supported by the Client.

Findings:	The TOE does not claim SSH client functionality.

221 In order to facilitate this test, it is acceptable for the evaluator to configure an authentication method that is outside of the selections in the SFR.

Findings:	The TOE does not claim SSH client functionality.

4.1.3.7 FCS_SSH_EXT.1.3 TSS

222 The evaluator shall check that the TSS describes how “large packets” are detected and handled.

Findings:	Section 6.2.10 of the [ST] indicates that packets greater than 256KB are dropped.

4.1.3.8 FCS_SSH_EXT.1.3 Tests

223 **Test 1**: The evaluator shall demonstrate that the TOE accepts the maximum allowed packet size.

High-Level Test Description
Send a packet from the SSH client to the TOE SSH server that is exactly the claimed maximum size and show that the TOE accepts the packet. Send a packet from the SSH client to the TOE SSH server that is one byte larger than the defined maximum and show the TOE drops the packet.
PASS

224 **Test 2**: This test is performed to verify that the TOE drops packets that are larger than size specified in the component.

a. The evaluator shall establish a successful SSH connection with the peer.

b. **[TD0732]** Next the evaluator shall craft a packet that is slightly larger than the maximum size specified in this component and send it through the established SSH connection to the TOE. The packet should not be greater than the maximum packet size + 16 bytes. If the packet is larger, the evaluator shall justify the need to send a larger packet.

c. **[TD0732]** The evaluator shall verify that the packet was dropped by the TOE. The method of verification will vary by the TOE. Examples_include reviewing the TOE audit log for a dropped packet audit or observing the TOE terminates the connection.

Note:	Test 2 is performed in the previous test case.

4.1.3.9 FCS_SSH_EXT.1.4 TSS

225 The evaluator will check the description of the implementation of SSH in the TSS to ensure the encryption algorithms supported are specified. The evaluator will check

the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.10 of the [ST] contains the necessary information and has found to be consistent with the selections made in section 5.3.2 of the [ST].

4.1.3.10 FCS_SSH_EXT.1.4 Guidance

226 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	Section 3.3.1 in [AGD] describes the configuration options for the SSH encryption algorithms provided by the TOE. The section includes a reference to the SSHD(8) Section of the [SOLARIS] / Oracle Solaris Reference Manuals guidance resource which provides detailed information on configuration of the SSHD service's allowed encryption algorithms. The section also references the SSHD_CONFIG(5) Section of the [SOLARIS] / Oracle Solaris Reference Manuals, wherein further information on configuring the SSH encryption is provided.

4.1.3.11 FCS_SSH_EXT.1.4 Tests

227 The evaluator shall perform the following tests.

228 If the TOE can be both a client and a server, these tests must be performed for both roles.

Note:	The TOE does not claim SSH client functionality.

- **Test 1**: The evaluator must ensure that only claimed algorithms and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall establish an SSH connection with a remote endpoint. The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers only the algorithms defined in the ST for the TOE for SSH connections. The evaluator shall perform one successful negotiation of an SSH connection and verify that the negotiated algorithms were included in the advertised set. If the evaluator detects that not all algorithms defined in the ST for SSH are advertised by the TOE or the TOE advertises additional algorithms not defined in the ST for SSH, the test shall be regarded as failed.

  The data collected from the connection above shall be used for verification of the advertised hashing and shared secret establishment algorithms in FCS_SSH_EXT.1.5 and FCS_SSH_EXT.1.6 respectively.

High-Level Test Description
Connect to the TOE via SSH and subsequently terminate the connection. Review the traffic between the TOE and the SSH client. Verify the connection was successful, the TOE offers only the algorithms defined in the ST and that the TOE terminates the connection appropriately.
PASS

- **Test 2**: For the connection established in Test 1, the evaluator shall terminate the connection and observe that the TOE terminates the connection.

Note:	Test 2 is covered in Test 1 above.

- **Test 3**: The evaluator shall configure the remote endpoint to only allow a mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the attempt fails.

High-Level Test Description
Attempt to connect to the TOE via SSH using the 3des-cbc cipher and show the connection fails.
PASS

4.1.3.12 FCS_SSH_EXT.1.5 TSS

229 The evaluator will check the description of the implementation of SSH in the TSS to ensure the hashing algorithms supported are specified. The evaluator will check the TSS to ensure that the hashing algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.10 of the [ST] contains the necessary information and has found to be consistent with the selections made in section 5.3.2 of the [ST].

4.1.3.13 FCS_SSH_EXT.1.5 Guidance

230 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	Section 3.3.1 in [AGD] describes the configuration options for the SSH server on the TOE. The section includes a reference to the SSHD section of the [SOLARIS] guidance resource which provides detailed information on configuration of the SSHD service's MAC algorithms and other mechanisms.

4.1.3.14 FCS_SSH_EXT.1.5 Tests

- **Test 1:** The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.

Note:	This test is conducted in FCS_SSH_EXT.1.4, Test 1.

- **Test 2:** The evaluator shall configure an SSH peer to allow only a hashing algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected.

High-Level Test Description
Attempt to connect to the TOE via SSH using the hmac-md5 integrity algorithm with a supported ciphersuite permitting its use and show that the algorithm is not supported.
PASS

4.1.3.15 FCS_SSH_EXT.1.6 TSS

231 The evaluator will check the description of the implementation of SSH in the TSS to ensure the shared secret establishment algorithms supported are specified. The evaluator will check the TSS to ensure that the shared secret establishment algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.10 of the [ST] contains the necessary information and has found to be consistent with the selections made in section 5.3.2 of the [ST].

4.1.3.16 FCS_SSH_EXT.1.6 Guidance

232 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	Section 3.3.1 in [AGD] describes the configuration options for the SSH server on the TOE. The section includes a reference to the SSHD section of the [SOLARIS] guidance resource which provides detailed information on configuration of the SSHD service's key exchange algorithms and other mechanisms.

4.1.3.17 FCS_SSH_EXT.1.6 Tests

- **Test 1**: The evaluator shall use the test data collected in FCS_SSH_EXT.1.4, Test 1 to verify that appropriate mechanisms are advertised.

Note:	This test is conducted in FCS_SSH_EXT.1.4, Test 1.

- **Test 2**: The evaluator shall configure an SSH peer to allow only a key exchange method that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection and observe that the connection is rejected.

High-Level Test Description
Attempt to connect to the TOE via SSH using the diffie-hellman-group1-sha1 key exchange algorithm and show that the algorithm is not supported.
PASS

4.1.3.18 FCS_SSH_EXT.1.7 TSS

233 The evaluator will check the description of the implementation of SSH in the TSS to ensure the KDFs supported are specified. The evaluator will check the TSS to ensure that the KDFs specified are identical to those listed for this component.

Findings:	Section 6.2.10 of the [ST] contains the necessary information and has found to be consistent with the selections made in section 5.3.2 of the [ST].

4.1.3.19 FCS_SSH_EXT.1.8 TSS

234 The evaluator shall check the TSS to ensure that if the TOE enforces connection rekey or termination limits lower than the maximum values that these lower limits are identified.

Findings:	Section 6.2.10 of the [ST] indicates that the TOE will rekey at the lesser of 1 hour or 1 GB of aggregate transmitted/received data.

235 In cases where hardware limitation will prevent reaching data transfer threshold in less than one hour, the evaluator shall check the TSS to ensure it contains:

a. An argument describing this hardware-based limitation and

b. Identification of the hardware components that form the basis of such argument.

236 For example, if specific Ethernet Controller or Wi-Fi radio chip is the root cause of such limitation, these subsystems shall be identified.

Findings:	This consideration is not applicable to this TOE.

4.1.3.20 FCS_SSH_EXT.1.8 Guidance

237 The evaluator shall check the guidance documentation to ensure that if the connection rekey or termination limits are configurable, it contains instructions to the administrator on how to configure the relevant connection rekey or termination limits for the TOE.

Findings:	Section 3.3.1 in [AGD] describes the configuration options for the SSH server on the TOE. The section includes a reference to the SSHD section of the [SOLARIS] guidance resource which provides detailed information on configuration of the SSHD service's rekey limit and other parameters.

4.1.3.21 FCS_SSH_EXT.1.8 Tests

238 The test harness needs to be configured so that its connection rekey or termination limits are greater than the limits supported by the TOE -- it is expected that the test harness should not be initiating the connection rekey or termination.

- **Test 1**: Establish an SSH connection. Wait until the identified connection rekey limit is met. Observed that a connection rekey or termination is initiated. This may require traffic to periodically be sent, or connection keep alive to be set, to ensure that the connection is not closed due to an idle timeout.

High-Level Test Description
Set the volume and time-based rekey limits on the TOE to 1GB and 1 minute, respectively. Connect to the TOE via SSH and, while keeping the session alive, wait until the time-based rekey limit is met. Verify the TOE initiates a connection rekey or terminates the connection upon reaching the limit. Ensure that the connection is not closed due to an idle timeout.

High-Level Test Description
PASS

- **Test 2**: Establish an SSH connection. Transmit data from the TOE until the identified connection rekey or termination limit is met. Observe that a connection rekey or termination is initiated.

High-Level Test Description
Set the volume and time-based rekey limits on the TOE to 500MB and 1 hour, respectively. Connect to the TOE via SSH and, force the TOE to transmit more data back to the SSH client than the client generates to the TOE. Verify the TOE initiates a connection rekey or terminates the connection upon reaching the volume-based rekey limit.
PASS

- **Test 3**: Establish an SSH connection. Send data to the TOE until the identified connection rekey limit or termination is met. Observe that a connection rekey or termination is initiated.

High-Level Test Description
Set the volume and time-based rekey limits on the TOE to 500MB and 1 hour, respectively. Connect to the TOE via SSH and, force the client to transmit more data to the SSH server than the client receives from the TOE. Verify the TOE initiates a connection rekey or terminates the connection upon reaching the volume-based rekey limit.
PASS

4.1.4 FCS_SSHS_EXT.1 SSH Protocol - Server

4.1.4.1 FCS_SSHS_EXT.1 TSS

239 No activities.

4.1.4.2 FCS_SSHS_EXT.1 Guidance

240 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections with the TOE.

Findings:	Section 3.3.1 in [AGD] describes the configuration options for the SSH server on the TOE. The section includes a reference to the SSHD section of the [SOLARIS] guidance resource which provides detailed information on configuration of the SSHD service's host key algorithms and other mechanisms.

4.1.4.3 FCS_SSHS_EXT.1 Tests

241 **[TD0682]** The evaluator shall perform the following tests:

242 **[TD0682]** Test 1: The evaluator shall use a suitable SSH Client to connect to the TOE and examine the list of server host key algorithms in the SSH_MSG_KEXINIT packet sent from the server to the client to determine that only the configured server authentication methods for the TOE were offered by the server.

High-Level Test Description
For each of the claimed host key algorithms (those used by the client to authenticate the server), attempt to make a good connection to the TOE server and disconnect the session. Verify the client successfully authenticates the server using the requested/advertised host key algorithm and that the SSH connection is successful.
PASS

243 **[TD0682]** Test 2: The evaluator shall test for a successful configuration setting of each server authentication method as follows. The evaluator shall initiate a SSH session using the authentication method configured and verify that the session is successfully established. Repeat this process for each independently configurable server authentication method supported by the server.

High-Level Test Description
For each supported public key authentication algorithms, attempt to connect to the TOE via SSH using public/private key as the authentication mechanism. Verify the server successfully authenticates the client using the appropriate public key algorithm, and that the SSH connection is successful and subsequently terminated. Connect to the TOE via SSH using username and password as the authentication mechanism. Verify the server successfully authenticates the client and that the SSH connection is successful and subsequently terminated.
PASS

244 **[TD0682]** Test 3: The evaluator shall configure the peer to only allow an authentication mechanism that is not included in the ST selection. The evaluator shall attempt to connect to the TOE and observe that the TOE sends a disconnect message.

High-Level Test Description
Attempt to connect to the TOE via SSH using an unsupported authentication mechanism. Verify the connection fails.
PASS

4.1.5 FIA_PMG_EXT.1 Password Management

4.1.5.1 Guidance Documentation

245 The evaluator shall examine the operational guidance to determine that it provides guidance to security administrators in the composition of strong passwords, and that it provides instructions on setting the minimum password length.

Findings:	Section 3.2.4.2 of [AGD] describes how administrators are able to set the password policy to enforce various password complexity requirements for administrators and users. The section includes a link to the Securing Systems and Attached Devices in Oracle Solaris 11.4 section of the [SOLARIS] guidance resource, which provides detailed information on setting password parameters via the password policy including minimum password length.

	[AGD], Section 3.2.4.2 also provides a link to the passwd (1) Section in [SOLARIS] / Oracle Solaris Reference Manuals which includes detailed instructions on setting specific password complexity requirements and policies using passwd. The Managing Password Information subsection of the [SOLARIS] / Securing Systems and Attached Devices in Oracle Solaris 11.4 guidance resource states, “Your organization should have a password policy that follows industry standards. Users must choose their passwords carefully and follow your site's password policy.”

4.1.5.2 Tests

246 The evaluator shall also perform the following test.

- **Test 1**: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible combinations of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

High-Level Test Description
Set the password complexity rules and then attempt to set various passwords to show that they are accepted or not. Verify the password policy configuration and password change attempts are audited appropriately.
PASS

4.1.6 FIA_X509_EXT.1 X.509 Certificate Validation

4.1.6.1 TSS

247 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.

Findings:	Section 6.4.5 of the [ST] provides information on how X.509 certificates are checked. The check of validity is performed when “…an X.509 certificate is presented…”. Section 6.4.5 also claims that X.509 certificates are only presented when using TLS connections for the log offloading functionality (as per section 6.1.4 of the [ST]) and code signing for trusted updates. The certificates in the path are evaluated according to the various checks and rules provided in section 6.4.5.

248 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. 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.

Findings:	Section 6.4.5 of the [ST] claims that when the TSF cannot establish a connection to the CRL to determine the validity of a certificate, the TSF shall not accept the

	certificate. This is the default behaviour of the TOE and the administrator is not able to specify the default action.

4.1.6.2 Tests

249 The tests described must be performed in conjunction with the other Certificate Services evaluation activities, including the uses listed in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

- **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.

High-Level Test Description
Show that when the chain is properly constructed, the TOE TLS client can connect. For each of the certificates/certificate paths described in the test, show that the TOE fails to connect to the TLS server when such certificates are used.
PASS

Note:	The test case to remove trust in one of the CA certificates and show that the function fails is performed in FIA_X509_EXT.2.

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

High-Level Test Description
Using a TOE TLS client, connect to a TLS server which will return an expired certificate and show the connection fails.

High-Level Test Description
Using an expired CA certificate in the trust store, show that the TOE TLS client fails to connect to the TLS server.
PASS

- **[TD0742] Test 3**: (conditional, performed except for use cases identified in exceptions that cannot be configured to allow revocation) 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, and then a test is performed for each method. The evaluator has to only test one up in the trust chain (future revisions may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator shall then attempt the test with a certificate that will be revoked (for each method chosen in the selection) and verify that the validation function fails.If the exceptions are configurable, the evaluator shall attempt to configure the exceptions to allow revocation checking for each function indicated in FIA_X509_EXT.2.

High-Level Test Description
Ensure the current CRL is empty. Verify that a TLS connection is successful with empty CRLs (i.e. no revoked certificates). Revoke the leaf certificate and attempt the connection again. Verify the connection now fails due to the certificate being revoked.
PASS

- **Test 4**: If any OCSP option is selected, the evaluator shall present a delegated OCSP certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set and verify that validation of the CRL fails.

High-Level Test Description
Using the TOE TLS client, connect to the TLS server and verify that the TLS client will fail to validate the CRL when the CRL is signed by a CA which does not have the proper policy flag extension set.
PASS

- **Test 5**: (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.

High-Level Test Description
Construct a chain of three ECDSA certificates: a leaf, an intermediate CA and a trust anchor. Show that the leaf and chain are valid when connected to. Create a clone of the Intermediate CA, such

High-Level Test Description
that the public key is explicitly defined rather than being a named curve. Show that the leaf and chain are not validated correctly when connected to.
PASS

- **Test 6**: (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 5 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 5, 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.

Note:	The functionality described in Test 6 is tested in the previous test case.

4.1.7 FIA_X509_EXT.2 X.509 Certificate Authentication

4.1.7.1 TSS

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

Findings:	Section 6.4.5 claims that all X.509 certificates are maintained within the trust store, found under the `/etc/certs/CA` directory.

251 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. If the requirement states that the administrator specifies the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

Findings:	Section 6.4.5 of the [ST] claims that when the TSF cannot establish a connection to the CRL to determine the validity of a certificate, the TSF shall not accept the certificate.

4.1.7.2 Tests

252 The evaluator shall perform Test 1 for each function listed in FIA_X509_EXT.2.1 that requires the use of certificates:

- **Test 1**: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

High-Level Test Description
Show that when the chain is properly constructed, the TOE TLS client can connect to the server. Remove trust in one of the CA certificates and show that the function fails.
PASS

- **Test 2**: The evaluator shall demonstrate that using a valid certificate requires that certificate validation checking 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.

High-Level Test Description
With the TOE configured to do CRL checking via the rsyslog-crl service, initiate a TLS connection from the rsyslog service on the TOE to the evaluator's workstation. Show that the TOE communicates with the CRL server on service startup to obtain an up-to-date copy of the CRL, and that the subsequent TLS connection is successful. Repeat the above with the CRL server on the evaluator's workstation stopped. Verify the TOE fails to connect to the CRL server and that the subsequent TLS connection to the logging server fails (no connection initiated by the TOE).
PASS

4.1.8 FTP_TRP.1 Trusted Path

4.1.8.1 TSS

253 The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Findings:	Section 6.8.2 of the [ST] indicates that the TOE offers one method of remote administration: via a CLI protected by SSH. The SSH protocol is selected in FTP_ITC_EXT.1 (for use in FTP_TRP.1) in section 5.3.8 of the [ST].

4.1.8.2 Guidance Documentation

254 The evaluator shall confirm that the operational guidance contains instructions for establishing the remote administrative sessions for each supported method.

Findings:	Section 3.2.1 of the [AGD] describes the usage of SSH to establish remote administrative sessions. [SOLARIS] / Managing Secure Shell Access in Oracle Solaris 11.4 and [SOLARIS] / Oracle Solaris Reference Manuals / ssh (1) provide detailed instructions on establishing remote administrative sessions with the TOE via SSH.

4.1.8.3 Tests

255 The evaluator shall also perform the following tests:

- **Test 1**: The evaluators shall ensure that communications using each specified (in the operational guidance) remote administration method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.

Note:	The evaluator followed the administrative guidance to set up the connections to ensure that communication was successful.

- **Test 2**: For each method of remote administration supported, the evaluator shall follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish remote administrative sessions without invoking the trusted path.

High-Level Test Description
Perform a port scan of the device and determine if there are any remote administrative interfaces available outside of the SSH CLI interface.
PASS

- **Test 3:** The evaluator shall ensure, for each method of remote administration, the channel data is not sent in plaintext.

High-Level Test Description
Using a packet sniffer, show that the channel data is not being sent in plaintext for each of the TSFI.
PASS

- **Test 4:** The evaluator shall ensure, for each method of remote administration, modification of the channel data is detected by the TOE.

High-Level Test Description
Using a custom tool, modify traffic destined to the TOE's remote administration interfaces and show that the modifications are detected.
PASS

256 Additional evaluation activities are associated with the specific protocols.

5 TOE Security Assurance Requirements

5.1 Class ASE: Security Target Evaluation

257 As per ASE activities defined in [CEM] plus the TSS assurance activities defined for any SFRs claimed by the TOE.

Findings:	See above sections and content in the Evaluation Technical Report (ETR).

5.2 Class ADV: Development

258 The information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (TSS) portion of the ST. The TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The Assurance Activities contained in Section 5.2 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

Findings:	See above sections and content in the Evaluation Technical Report (ETR).

5.3 Class AGD: Guidance Documents

259 The guidance documents will be provided with the developer's security target. Guidance must include a description of how the authorized user verifies that the Operational Environment can fulfil its role for the security functionality. The documentation should be in an informal style and readable by an authorized user.

260 Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes

- instructions to successfully install the TOE in that environment; and
- instructions to manage the security of the TOE as a product and as a component of the larger operational environment.

261 Guidance pertaining to particular security functionality is also provided; specific requirements on such guidance are contained in the assurance activities specified with individual SFRs where applicable.

Findings:	See above sections and content in the Evaluation Technical Report (ETR).

5.3.1 AGD_OPE.1 Operational User Guidance

262 Some of the contents of the operational guidance will be verified by the evaluation activities in Section 5.2 [of the PP] and evaluation of the TOE according to the CEM. The following additional information is also required.

263 The operational guidance shall contain instructions for configuring the password characteristics, number of allowed authentication attempt failures, the lockout period times for inactivity, and the notice and consent warning that is to be provided when authenticating.

Findings: Section 3.2.4.2 of the [AGD] describes how administrators are able to set the password policy to enforce various password complexity requirements for administrators and users.

Section 3.2.4.1 of the [AGD] describes how administrators are able to configure the number of allowed authentication attempt failures.

Section 3.2.2 of the [AGD] describes how administrators are able to configure the lockout period times for inactivity.

The subsection, Securing Logins and Passwords of the [SOLARIS] / Securing Systems and Attached Devices in Oracle Solaris 11.4 guidance resource, provides instructions on configuring the notice and consent warning to be provided prior to authentication.

264 The operational guidance shall contain step-by-step instructions suitable for use by an end-user of the VS to configure a new, out-of-the-box system into the configuration evaluated under this Protection Profile.

Findings: Please refer to the identical work unit described in AGD_PRE.1 below.

265 The documentation shall describe the process for verifying updates to the TOE, either by checking the hash or by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

- Instructions for querying the current version of the TOE software.

Findings: Section 2.3 of [AGD] provides instructions for querying the current version of the TOE software.

- For hashes, a description of where the hash for a given update can be obtained. For digital signatures, instructions for obtaining the certificate that will be used by the FCS_COP.1/SIG mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means.

Findings: The Securing Systems and Attached Devices in Oracle Solaris 11.4 and Updating Systems and Adding Software in Oracle Solaris 11.4 sections of the [SOLARIS] guidance resource provides detailed information on the usage/configuration of certificates for digital signature verification of updates.

- 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).

Findings:	Section 2.4 of the [AGD] provides instructions on how updates are obtained. The Updating Systems and Adding Software in Oracle Solaris 11.4 section of the [SOLARIS] guidance resource provides detailed information about the update procedure/configuration.

- Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.

Findings:	Section 2.4 of the [AGD] provides instructions on how updates are initiated and how to discern the result of the update process. The Updating Systems and Adding Software in Oracle Solaris 11.4 section of the [SOLARIS] guidance resource provides detailed information about the update procedure/configuration.

5.3.2 AGD_PRE.1 Preparative Procedures

266 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 (that is, combination of hardware and operating system) claimed for the TOE in the ST.

Findings:	The [AGD], [SPARC], [SOLARIS] and [T8LIB] guidance resources are specific to the Oracle VM Server for SPARC 3.6 and Solaris 11.4 OS, which is the only claimed platform for the TOE in the [ST].

267 The operational guidance shall contain step-by-step instructions suitable for use by an end-user of the VS to configure a new, out-of-the-box system into the configuration evaluated under this Protection Profile.

Findings:	The evaluator used the [AGD], [SPARC], [SOLARIS] and [T8LIB] guidance resources to configure a new, out-of-the-box system into the configuration evaluated under this Protection Profile, in a similar manner as would be done by an end-user of the VS, and determined the provided instructions were suitable for this purpose.

5.4 Class ALC: Life-Cycle Support

268 At the assurance level specified for TOEs conformant to this PP, life-cycle support is limited to an examination of the TOE vendor’s development and configuration management process in order to provide a baseline level of assurance that the TOE itself is developed in a secure manner and that the developer has a well-defined

process in place to deliver updates to mitigate known security flaws. This is a result of the critical role that a developer's practices play in contributing to the overall trustworthiness of a product.

5.4.1 ALC_CMC.1 Labeling of the TOE

269 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.

Findings: The [ST] contains the product name and reference in section 1.2. The same information can be found in section 1.3.2 of [AGD].

Oracle maintains a website for advertising Solaris and the information in the ST is sufficient to distinguish the product. Specifically, the version on the web site is 11.4. The SRUs and IDRs are only available to those with support contracts and they would be able to acquire the TOE through those channels.

The documentation provided by Oracle for the TOE is clearly designated for Solaris 11.4.

270 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.

Findings: The TOE was verified using the instructions provided in section 2.3 of the [AGD] and was found to be consistent with the versions claimed in the [ST].

271 If the vendor maintains a website advertising the TOE, the evaluator shall examine the information on the website to ensure that the information in the ST is sufficient to distinguish the product.

Findings: Oracle maintains a website for advertising Solaris and Oracle VM Server for SPARC. The evaluator examined the website and determined that the information in the [ST] is sufficient to distinguish the product to distinguish the product amongst the information provided on the website.

Specifically, the version of Solaris on the web site is 11.4 and the version of Oracle VM Server for SPARC is 3.6. The specific SRUs and IDRs are only available to those with support contracts and they would be able to acquire the TOE through those channels.

5.4.2 ALC_CMS.1 TOE CM Coverage

272 The evaluator shall ensure that the developer has identified (in public-facing development guidance for their 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 are invoked (e.g., compiler and linker 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.

Findings:	The Solaris 11.4 documentation library is provided in [SOLARIS]. Within this library, there are significant resources available to developers. Specifically, in the section “Developing Applications For Use With Oracle Solaris”, there is information about several application development environments. For application development environments which produce binary machine code, the linker ld(1) provides link-time flags to explicitly enable aslr, nxstack and nxheap security extensions (-z sx=aslr -z sx=nxstack -z sx=nxheap). Note, however, that these security extensions are available by default in the TOE even if these flags are not provided (as per sxadm(8) which can be found by reviewing [SOLARIS] under “man pages section 8: System Administration Commands”). The nxstack, nxheap and aslr link-time security extensions are available by default in the TOE even if these flags are not provided (as per the information provided in sxadm(8) which can be found by reviewing the section, “man pages section 8: System Administration Commands” in [SOLARIS]. The Oracle VM Server for SPARC 3.6 documentation library is provided in [SPARC]. Within this library, there are significant resources available to developers. Specifically, in the section “Oracle VM Server for SPARC 3.6 Developer's Guide” there is information about several application development environments and security considerations, specific to the VM server. Note that protection mechanisms are inherited from the underlying Solaris 11.4 OS. The evaluator determined that adequate information is given within the developer resources such that each TSF is uniquely identified their mapping to specific requirements in the [ST] is not ambiguous.

5.4.3 ALC_TSU_EXT.1 Timely Security Updates

273 This component requires the TOE developer, in conjunction with any other necessary parties, to provide information as to how the VS is updated to address security issues in a timely manner. The documentation describes the process of providing updates to the public from the time a security flaw is reported/discovered, to the time an update is released. This description includes the parties involved (e.g., the developer, hardware vendors) and the steps that are performed (e.g., developer testing), including worst case time periods, before an update is made available to the public.

274 ALC_TSU_EXT.1.1C: The description shall include the process for creating and deploying security updates for the TOE software/firmware.

Findings:	Section 5.4.2 of the [ST] provides links to the developer’s “timely security update methodology”.

275 ALC_TSU_EXT.1.2C: The description shall express the time window as the length of time, in days, between public disclosure of a vulnerability and the public availability of security updates to the TOE.

Findings:	The developer’s “timely security update methodology” website described in section 5.4.2 of the [ST] notes that it is Oracle’s policy to announce security fixes as much as possible only when the fixes are available for all affected and supported product version and platform combinations. The same website further notes that “Minor

	delays in patch availability for up to two weeks from the announcement date generally due to technical issues during the production or testing of the patch”.

276 ALC_TSU_EXT.1.3C: The description shall include the mechanisms publicly available for reporting security issues pertaining to the TOE.

Findings:	Reporting is described in the “security vulnerability reporting procedures” website described in section 5.4.2 of the [ST]. The suggested method in the website is emailing the “secalert_us@oracle.com”. The PGP key is published with the email address.

5.5 Class ATE: Tests

277 Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this PP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

5.5.1 ATE_IND.1 Independent Testing – Conformance

278 Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operation) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 5.1 are being met, although some additional testing is specified for SARs in Section 5.2. The evaluation activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this PP.

279 The evaluator shall prepare a test plan and report documenting the testing aspects of the system. While it is not necessary to have one test case per test listed in an evaluation activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

280 The Test Plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

281 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 evaluators are 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) is 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 cryptographic engines to be used. The cryptographic

algorithms implemented by these engines are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS/HTTPS, SSH).

282 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.

Findings: Please refer to Tests assurance activities above in the previous sections which satisfy this work unit. The detailed results are contained in the submitted Test Plan and a summary of the testing activities is further described within the Evaluation Technical Report (ETR).

Platform coverage and equivalency arguments are provided in the Test Plan.

Explicit identification of cryptographic engines used to provide algorithms for the in-scope cryptographic operations and protocols is included in the Test Plan.

The Test Plan maintains a "journal" of the test results where necessary to showcase failures and actions taken to bring the test results up to a passing grade.

5.6 Class AVA: Vulnerability Assessment

283 For the first generation of this Protection Profile, the evaluation lab is expected to survey open sources to learn what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, evaluators will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future PPs.

284 As with ATE_IND 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 determine the vulnerabilities that have been found in virtualization in general, as well as those that pertain to the particular TOE. 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. For example, if the vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this PP. 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.

Findings: A Vulnerability Test Plan was generated as part of the evaluation effort.