Oracle Corporation

Oracle Linux 7.6 UEK 5 KVM & Virtualization Manager 4.3

Assurance Activity Report

Version 1.2

March 2023

Document prepared by

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 Linux 7.6 UEK 5 KVM & Virtualization Manager 4.3
Security Target	Oracle Linux 7.6 UEK 5 KVM & Virtualization Manager 4.3 Security Target, v2.3, March 2023.
Protection Profile	[PP] NIAP Protection Profile for Virtualization, Version: 1.0, 2016-11-17 [SV_EP] NIAP Protection Profile for Virtualization Extended Package Server Virtualization, Version: 1.0, 2016-11-17 [SSH_EP] NIAP Extended Package for Secure Shell (SSH), Version: 1.0, 2016-02-19

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 v3.1R4
Evaluation Methodology	CEM v3.1R4
Supporting Documents	Assurance Activities found in [PP], [SV_EP] and [SSH_EP]
Interpretations	**PP_BASE_VIRTUALIZATION_V1.0** TD0139: Clarification of testing for FDP_RIP_EXT.2 TD0206: Testing for Non-Existence of Disconnected Virtual Devices TD0230: ALC Assurance Activities for Server Virtualization and Base Virtualization PPs TD0249: Applicability of FTP_ITC_EXT.1

	TD0250: Hypercall Controls - FPT_HCL_EXT.1 Clarification
	TD0264: Clarification of Auditable Events for FPT_RDM_EXT.1
	TD0360: AD Server configuration in FMT_MOF_EXT.1
	TD0363: Access Banner and applicability to programmatic interfaces
	TD0431: Modification to Cipher Suites for TLS
	TD0432: Corrections to FIA_AFL_EXT.1
	TD0443: FPT_VDP_EXT.1 Clarification for Assurance Activity
	TD0526: Updates to Certificate Revocation (FIA_X509_EXT.1)
	TD0567: Security Objectives Rationale, SFR Rationale, and Implicitly Satisfied SFRs
	TD0617: TLSC wildcard testing
	EP_SV_V1.0
	TD0360: AD Server configuration in FMT_MOF_EXT.1
	TD0568: SFR Rationale
	PP_SSH_EP_V1.0
	TD0240: FCS_COP.1.1(1) Platform provided crypto for encryption/decryption
	TD0331: SSH Rekey Testing
	TD0332: Support for RSA SHA2 host keys
	TD0420: Conflict in FCS_SSHC_EXT.1.1 and FCS_SSHS_EXT.1.1
	TD0446: Missing selections for SSH
	TD0598: Expanded AES Modes in FCS_COP for App PP *Note: This TD only applies when PP_SSH_EP is extending the Protection Profile for Application Software.*
Tools	Please refer to the test plan.

1.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
[ST]	Oracle Linux 7.6 UEK 5 KVM & Virtualization Manager 4.3 Security Target, v2.3, March 2023.
[AGD]	Oracle Linux 7.6 UEK 5 KVM & Virtualization Manager 4.3 Common Criteria Guide, v1.6, March 2023
[OL7]	Oracle Linux 7 Information Library https://docs.oracle.com/en/operating-systems/oracle-linux/7/
[OLVM]	Oracle® Linux Virtualization Manager Getting Started Guide https://www.oracle.com/a/ocom/docs/olvm43/olvm-43-gettingstarted.pdf
[OLVM_START]	Oracle Linux Virtualization Manager Getting Started https://docs.oracle.com/en/virtualization/oracle-linux-virtualization-manager/getstart/getstarted-manager-install.html
[OVIRT]	oVirt Administration Guide (upstream OLVM documentation) https://www.ovirt.org/documentation/administration_guide/
[KVM]	Oracle® Linux KVM User's Guide https://docs.oracle.com/en/operating-systems/oracle-linux/kvm-user/
[PORTAL]	oVirt Introduction to the VM Portal https://www.ovirt.org/documentation/introduction_to_the_vm_portal/
[ADMIN]	Oracle® Linux Virtualization Manager Administration Guide https://www.oracle.com/a/ocom/docs/olvm43/olvm-43-administration.pdf
[VMM_GUIDE]	oVirt Virtual Machine Management Guide https://www.ovirt.org/documentation/virtual_machine_management_guide/
[OS_GUIDE]	Oracle Linux v7.6 Common Criteria Guidance Document, v0.9, September 1, 2020
[VIRT_DEV]	Oracle KVM Virtual Devices

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 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, “Logs that are generated by the TOE follow the type and format identified in the following link: https://access.redhat.com/articles/4409591.” The evaluator examined the audit logs identified in the link and confirmed every audit event mandated by the PP were described.

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. 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. 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.

Findings:	The evaluator used Table 12 from the [ST] to determine all administrative actions necessary to enforce the requirements specified in the PP. The evaluator then examined the administrative guidance documentation and determined that all commands, subcommands, scripts, and configuration files are clearly identified in the administrative guidance documentation to enable the administrator to configure the mechanisms implemented in the TOE that are necessary to enforce the requirements in the PP.

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
The security mechanisms were tested as part of the evaluation, and it was confirmed that the TOE generated all of the audit records for the events listed and administrative actions.
Findings: PASS

2.1.2 FAU_SAR.1 Audit Review

2.1.2.1 Guidance Documentation

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

Findings:	[AGD] Section 3.3 provides a link that describes the type and format of the logs generated by the TOE and the evaluator determined this information is suitable for human interpretation. [AGD] Section 3.2.1 Syslog Configuration, describes the configuration to send audit records to an external syslog server. Audit records can be reviewed on the syslog server.

2.1.2.2 Tests

8 The assurance activity for this requirement is performed in conjunction with the assurance activity for FAU_GEN.1.

Findings:	The assurance activity for this requirement is performed in conjunction with the assurance activity for 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:	In section 6.1.3 of the [ST], it states only administrators may access or delete the audit log files.

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
Log in as an unauthorized administrator (user) and attempt to modify and delete the audit records. Confirm that the user account is restricted from modifying or deleting the audit records.
Findings: 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
Log into the TOE with an authorized administrator account and attempt to delete the audit records. Confirm that the audit records are successfully deleted. Confirm that only the records authorized for deletion are deleted.
Findings: 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:	The ST claims that the audit log offloading is implemented using SSH. This is reflected in section 5.3.9 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:	In section 6.1.4, the TOE forwards logs to a syslog server via SSH.

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

Findings:	Please refer to the test activities for FTP_ITC_EXT.1.

2.1.4.2 FAU_STG_EXT.1.1 Guidance Documentation

15 The evaluator shall also 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:	[AGD] Section 3.2.1 Syslog Configuration, describes the configuration to send audit records to an external syslog server. The evaluator confirmed it describes how to establish the trusted channel to the audit server, as well as any requirements on the audit server and configuration of the TOE needed to communicate with the audit server.

2.1.4.3 FAU_STG_EXT.1.1 Tests

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

a) 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
Record the software name and version used on the audit server during testing. Start a packet capture software (Wireshark) and record traffic between the TOE and the audit server. On the TOE perform actions that will generate audit events that will send traffic to the audit server. Review the packet capture to confirm that the audit data is not sent in the clear.
Findings: 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:	[ST] section 6.1.4 states, "When the local audit data store is full, the TOE stops logging locally and continues to send log records to the remote log server."

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:	[AGD] section 3.2.1 states, "When syslog is configured in accordance with this section, logs will be sent to the remote syslog server as soon as they are generated. If the remote server is not available, the logs will not be sent to the server."

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
Generate events until /var/log fills up and then show that the TOE complies with the ST and drops new audit data. This complies with the behavior defined in the ST for FAU_STG_EXT.1.2.
Findings: 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-, 3072-bit keys are supported. It also claims in the same section that FFC schemes with keys of size 2048- and 3072-bits are supported. Finally, DH group 14 is supported. The [ST], in section 6.2.1, claims use of RSA, FFC schemes and DH group 14. The scheme usages are clearly delineated.

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:	[OS_GUIDE] section 5.3 states, “The OS supports public key authentication. In order to generate a public and public key pair, the following command should be used: # ssh-keygen Press Enter each time that the command prompts you to enter a passphrase. Use the ssh-copy-id script to append the public key in the local ~/.ssh/id_rsa.pub file to the ~/.ssh/authorized_keys file on the remote system. You can now use the OpenSSH utilities to access the remote system without supplying a password. As the script suggests, you should use ssh to log into the remote system to verify that the ~/.ssh/ authorized_keys file contains only the keys for the systems from which you expect to connect.” [AGD] section 3.4 states, “To generate SSH public and private keys used in the evaluated configuration, enter the following commands: ssh-keygen -t rsa -b 2048 ssh-keygen -t rsa -b 3072 “To generate TLS public and private keys used in the evaluated configuration, enter the following commands: openssl genrsa -out <server_private.key> 2048 openssl genrsa -out <server_private.key> 3072” [AGD] section 3.2.2 states, “For the DHE cipher suites, the FFC keys are determined by the RSA X.509 key sizes (i.e. 2048 and 3072 bit keys). For instructions on replacing the oVirt engine CA certificate, follow these instructions: https://ovirt.org/documentation/administration_guide/index.html#replacing-manager-apache-ca-certificate.”

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 seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

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.

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 2 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 2 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/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

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 must 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:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed key generation.

2.2.2 FCS_CKM.2 Cryptographic Key Establishment

2.2.2.1 TSS

36 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 both RSA and FFC key generation in FCS_CKM.1.1. The same claims are made in FCS_CKM.2.1. The [ST], in section 6.2.1, claims use of RSA, FFC schemes and DH group 14. The scheme usages are clearly delineated.

37 *(This AA was buried in a test AA below for SP800-56B.)* The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

Findings:	Section 6.2.1 of the [ST] states, “For RSA-based key establishment, the TOE acts as a recipient for TLS.”

38 *(This AA was buried in a test AA below for SP800-56B when the TOE acts as a receiver.)* The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE shall not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations.

Findings:	Section 6.2.1 of the [ST] states, “In the event of a decryption error, the TOE only logs/outputs aggregate generic error messages and does not reveal the particular error that occurred.”

2.2.2.2 Guidance Documentation

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

Findings:	Section 3.2.2 of the [AGD] describes how to configure the TLS ciphersuites that include how to configure the TOE to use the selected key establishment schemes. Section 3.2.3 of the [AGD] describes how to configure the SSH Server and SSH Client to use the Diffie-hellman-group14-sha1 key establishment scheme.

2.2.2.3 Tests

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

41 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

42 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.

43 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.

44 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.

45 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.

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

Validity Test

47 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.

48 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/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

49 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:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed key agreement/establishment schemes.

SP800-56B Key Establishment Schemes

50 If the TOE acts as a sender, the following assurance activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

- To conduct this test, the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted

ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

Findings:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed key agreement/establishment schemes.

51 If the TOE acts as a receiver, the following assurance activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

- To conduct this test, the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

Findings:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed key agreement/establishment schemes.

52 If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

Findings:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed key agreement/establishment schemes.

53 If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

Findings:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed key agreement/establishment schemes.

2.2.3 FCS_CKM_EXT.4 Cryptographic Key Destruction

2.2.3.1 TSS

54 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 when the key is no longer required; keys in non-volatile storage are destroyed after the invocation of an API call. 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 server) and SSH (in client and server capacities). No other encryption or decryption services are claimed or described.

2.2.3.2 Tests

55 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 3rd-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).

56 Note: using 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. It is expected that these builds are based on the same source code as are release builds (of course, with instrumentation and debug-specific code added).

High-Level Test Description
Non-Volatile TLS and SSH Private keys: Build a new SSH host key and show that the key pattern bytes are found in non-volatile memory. Then execute the TSFI command to zeroize the key file and show that the key pattern bytes are no longer found in non-volatile memory. Without loss of generality, the same process and conclusions applies equivalently to the TLS private key. **Volatile SSH Server Session keys:** Capture the SSH session keys between a test SSH client and the TOE SSH server. Keeping the SSH connection open, dump volatile memory with the help of a memory tap. A search of this volatile memory images will show the keys. Then perform the same test again, but this time terminate the SSH connection before dumping RAM. A search of this volatile memory images will not show the keys. **Volatile SSH Client Session keys:** Capture the SSH session keys between a test SSH server and the TOE SSH client. Keeping the SSH connection open, dump volatile memory with the help of a memory tap. A search of this volatile memory images will show the keys. Then perform the same test again, but this time terminate the SSH connection before dumping RAM. A search of this volatile memory images will not show the keys. **Volatile TLS Server Session keys:** Capture the TLS session key between a test TLS client and the TOE TLS server. Keeping the TLS connection open, dump volatile memory with the help of a memory tap. A search of this volatile memory images will show the keys. Then perform the same test again, but this time terminate the TLS connection before dumping RAM. A search of this volatile memory images will not show the keys.
Findings: PASS

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

2.2.4.1 Tests

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

58 **AES-CBC Tests**

59 **AES-CBC Known Answer Tests**

60 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.

61 **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.

62 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.

63 **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.

64 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.

65 **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].

66 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.

67 **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].

68 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.

69 **AES-CBC Multi-Block Message Test**

70 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.

71 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.

72 **AES-CBC Monte Carlo Tests**

73 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]
```

74 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.

75 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:	Refer to Table 4 in the Security Target for CAVP certificate related to AES.

76 **AES-CCM Tests**

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

78 **128 bit and 256 bit keys**

79 **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).

80 **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.

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

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

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

84 **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.

85 **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.

86 **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.

87 **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.

88 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.

89 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.

90 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.

Findings:	Refer to Table 4 in the Security Target for CAVP certificate related to AES.

91 **AES-GCM Test**

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

93 **128 bit and 256 bit keys**

94 **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.

95 **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.

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

97 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.

98 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.

99 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:	The TOE does not claim AES-GCM in the evaluated configuration.

100 **XTS-AES Test**

101 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.

102 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.

103 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.

104 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.

Findings:	The TOE does not claim AES-XTS in the evaluated configuration.

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

106 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).

107 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.

108 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.

109 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:

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

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

112 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.

Findings:	Refer to Table 4 in the Security Target for CAVP certificate related to AES.

2.2.5 FCS_COP.1(2) Cryptographic Operation (Hashing)

2.2.5.1 TSS

113 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 6.2.5 of the [ST] claims that the hash function is used for signature services and HMAC services. These services are consistent with the claims made in section 5 of the [ST].

2.2.5.2 Guidance Documentation

114 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:	Section 3.2.2 of the [AGD] describes how to configure the TLS ciphersuites that include how to configure the TOE for the required hash sizes. Section 3.2.3 of the [AGD] describes how to configure the SSH Server and SSH Client for the required hash sizes.

2.2.5.3 Tests

115 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 testmacs.

116 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.

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

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

119 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.

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

121 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.

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

123 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.

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

125 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.

126 **Pseudorandomly Generated Messages Test**

127 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.

Findings:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed cryptographic hashing.

2.2.6 FCS_COP.1(3) Cryptographic Operation (Signature Algorithms)

2.2.6.1 Tests

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

129 **ECDSA Algorithm Tests**

130 **ECDSA FIPS 186-4 Signature Generation Test**

131 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.

132 **ECDSA FIPS 186-4 Signature Verification Test**

133 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.

134 **RSA Signature Algorithm Tests**

135 **Signature Generation Test**

136 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.

137 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.

138 **Signature Verification Test**

139 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, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

140 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:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed signature generation and verification.

2.2.7 FCS_COP.1(4) Cryptographic Operation (Keyed Hash Algorithms)

2.2.7.1 TSS

141 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 6.2.7 of the ST identifies the key length, hash function, block size and MAC length output.

2.2.7.2 Tests

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

143 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:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed HMACs.

2.2.8 FCS_ENT_EXT.1 Entropy for Virtual Machines

2.2.8.1 TSS

144 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 VM.

Findings:	As per section 6.2.8 of the [ST], the TOE exposes a paravirtualized hardware device to guest VMs via VirtIO RNG. It is backed by the host's /dev/urandom device. Section 6.2.8 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/urandom backend.

2.2.8.2 Tests

145 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.

High-Level Test Description
This was accomplished as part of FCS_ENT_EXT.1 Test 2.
Findings: PASS

- 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
Use a script to invoke entropy from multiple VMs (nearly) simultaneously. Review the output of the entropy to confirm that the entropy used in one VM is not identical to that invoked from the other VMs.
Findings: PASS

2.2.9 FCS_RBG_EXT.1 Cryptographic Operation (Random Bit Generation)

146 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Annex D, Entropy Documentation and Assessment.

2.2.9.1 Tests

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

148 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.

149 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).

150 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 re-seed. The final value is additional input to the second generate call.

151 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 Derivation Function 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 support, 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:	Refer to Table 4 in the Security Target for CAVP certificate related to claimed RBG operations.

2.3 User Data Protection (FDP)

2.3.1 FDP_HBI_EXT.1 Hardware-Based Isolation Mechanisms

2.3.1.1 TSS

152 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 leverages processor architectural mechanisms to isolate access to physical devices. The [ST], in section 2.4, claims the TOE runs on Intel processors. The TOE leverages Intel's VT-x and VT-d capabilities. Section 6.3.1 claims these mechanisms are always on and cannot be disabled.

2.3.1.2 Guidance Documentation

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

Findings:	[AGD] section 3.1.1 states, "Use the following commands to verify that the mechanisms are enabled: # cat /proc/cpuinfo # virt-host-validate "The cat /proc/cpuinfo command should contain the "vmx" flag to show Intel VT-x virtualization support is enabled in the BIOS. "The virt-host-validate command should be displayed as follows: QEMU: Checking for hardware virtualization :PASS …"

2.3.2 FDP_PPR_EXT.1 Physical Platform Resource Controls

2.3.2.1 TSS

154 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 is described.

Findings:	Section 6.3.2 of the [ST] states that the TOE uses Intel's VT-x and VT-d mechanisms to intercept access to all physical hardware resources and emulate those attempts in terms of virtual hardware. This interception is fundamental to virtualization and is not configurable.

155 This description shall cover all of the physical platforms allowed in the evaluated configuration by the ST.

Findings:	The [ST], in section 2.4, claims the TOE runs on Intel processors. Section 6.3.2 of the [ST] provides hardware-backed mechanisms appropriate to those processor vendors.

156 This description shall include 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.

Findings:	Section 6.3.2 indicates that Guest VMs are identified using an ID number and using an alphanumeric name. The TSS lists the set of controllable physical platform resources, and this list matches the set described in the SFR.

157 The evaluator shall ensure that the TSS describes how the Guest VM is associated with each physical resources, and how other Guest VMs cannot access a physical resource without being granted explicit access.

Findings:	Section 6.3.2 indicates that Guest VMs must be explicitly configured to use a physical resource.

158 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 VMs and physical platform resources.

Findings:	The TOE does not implement a robust access control mechanism and therefore this information is not included in the TSS.

159 If physical resources are listed in the second element, the evaluator shall examine the TSS … to determine that there appears to be no way to configure those resources for access by a Guest VM.

Findings:	According to section 5.3.3 of the [ST], the TOE is not capable of explicitly denying access to specific physical platform resources and therefore no information is required to be in the TSS.

2.3.2.2 Guidance Documentation

160 If physical resources are listed in the second element, the evaluator shall examine the […] operational guidance to determine that there appears to be no way to configure those resources for access by a Guest VM.

Findings:	No physical platform resources are listed in the second element.

161 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.

Findings:	The evaluator examined the operational guidance and determined 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. [VMM_GUIDE] / 5.5 – Hot Plugging vCPUs, provides further information on configuring access to the host CPUs. [VMM_GUIDE] / 5.4 – Virtual Memory, provides instructions on configuring access to the host Memory. [ADMIN] / 3 – Attaching and Configuring a Logical Network to a Host Network Interface, provides instructions on how to configure access to the host Network Adapter (physical NIC).

162 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:	There are no physical platform resources explicitly allowed or denied listed in the second and third elements of the component.

2.3.2.3 Tests

163 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:	There are no physical platform resources identified in the second element that are explicitly denied. All physical platform resources are denied by default and must be explicitly granted access to the VM by the administrator. CPU and Memory are explicitly granted to the VM when the administrator creates or modifies a VM. Physical NIC access is configured by explicitly assigning a virtual NIC (vNIC) profile to a physical NIC. Then assigning the vNIC profile to the VM.

164 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 all of the physical platform resources below and confirm that the Guest VM is able to successfully access the physical platform resources. Repeat for all claimed physical platform resources. Physical platform resources: • Network Adapter (Physical NIC) • CPU • Memory
Findings: 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
Using the Guest VM from Test 1 with all physical platform resources configured, remove access to one physical platform resource at a time and confirm that the Guest VM is unable to successfully access the resource.
Findings: 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.

Findings:	The TOE does not claim a robust control interface. When a VM is created the devices are either allowed or not allowed to the VM. There are no other modes of access.

- 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.

Findings:	The TOE does not explicitly deny access to certain physical resources.

- 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.

Findings:	The TOE does not explicitly allow access to certain physical resources.

2.3.3 FDP_RIP_EXT.1 Residual Information in Memory

2.3.3.1 TSS

165 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 the TOE's Linux kernel clears memory prior to allocation to a Guest VM. The process is performed as part of standard kernel memory management when allocating memory to a userspace process. 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

166 **TD139:** The evaluator shall ensure that the TSS documents the conditions under which physical disk storage is not cleared prior to allocation to a Guest VM. The evaluator shall also ensure that the TSS documents the metadata used in its virtual disk files.

Findings:	Section 6.3.4 of the [ST] claims 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

167 The evaluator shall perform the following test:

- **TD0139:** 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 non-zero 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
Create file on the host in the logical volume that is 300GB. Release the file, but do not clear Create a virtual disk of 300GB and verify that all non-metadata in the file is set to zero.
Findings: PASS

2.3.5 FDP_VMS_EXT.1 VM Separation

2.3.5.1 TSS

168 The evaluator shall examine the TSS to verify that it documents all inter-VM communications mechanisms (as defined above), including how the mechanisms are configured, how they are invoked, and how they are disabled.

Findings:	The [ST] claims, in section 6.3.5, that Guest VMs can only 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.

169 *(This TSS AA was found buried in a test AA.)* The evaluator must ensure that the ST includes the following statement attesting that there are no other ways for data to be transferred between VMs other than those listed in FDP_VMS_EXT.1.1:

> *A Guest VM cannot access the data of another Guest VM, or transfer data to another Guest VM other than through the mechanisms described in FDP_VMS_EXT.1.1 when expressly enabled by an authorized Administrator. There are no design or implementation flaws that permit the above mechanisms to be bypassed or defeated, or for data to be transferred through undocumented mechanisms. This claim does not apply to covert channels or architectural side-channels.*

Findings:	This attestation is found in section 6.3.5 of the [ST].

2.3.5.2 Tests

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

a) Create two 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.

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

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

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

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

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

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

172 FMT_MSA_EXT.1.1 is met if communication is unsuccessful in step (b). FMT_MSA_EXT.1.2 is met if communication is successful in step (d). Additionally,

FMT_MSA_EXT.1 requires that the evaluator verifies that the TSS documents the inter-VM communications mechanisms as described above.

High-Level Test Description
The TOE supports inter-VM communications through virtual networking. Create a Guest VM (VM1) and configure a virtual network used for inter-VM communications. Create a Guest VM (VM2), by default the VM2 is not configured to connect to the virtual network for inter-VM communications. Test that the VM1 and VM2 are unable to communicate. Connect VM2 to the inter-VM virtual network. Confirm that VM1 and VM2 can successfully communicate. Disconnect the virtual network from VM1 and VM2 and confirm that VM1 and VM2 can no longer communicate.
Findings: PASS

2.3.6 FDP_VNC_EXT.1 Virtual Networking Components

2.3.6.1 TSS

173 The evaluator must ensure that the TSS … describes how to create virtualized networks and connect VMs to each other and to physical networks.

Findings:	The TSS in section 6.3.6 of the [ST] provides a pointer to a guidance document which describes how to configure virtualized networks and connect VMs to one another and to physical networks.

174 *(This TSS AA was found buried in a test AA.)* The evaluator must ensure that the ST includes the following statement attesting that virtual network traffic is visible only to VMs configured to be on that virtual network:

> *"Traffic traversing a virtual network is visible only to Guest VMs that are configured by an Administrator to be members of that virtual network. There are no design or implementation flaws that permit the virtual networking configuration to be bypassed or defeated, or for data to be transferred through undocumented mechanisms. This claim does not apply to covert channels or architectural side-channels."*

Findings:	This attestation is found in section 6.3.6 of the [ST].

2.3.6.2 Guidance Documentation

175 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:	[VMM_GUIDE] / 5.2 and Appendix A – Explanation of Settings in the New Network Interface and Edit Network Interface Windows, provides instructions on configuring Virtual NICs for Guest VMs.

[ADMIN] / 2.3.1 Customizing vNIC Profiles for Virtual Machines, includes instructions on creating virtualized networks and connecting VMs to each other.

[ADMIN] / 2.3.2 Attaching and Configuring a Logical Network to a Host Network Interface, describes how to attach a virtualized network to physical networks.

The evaluator reviewed the operational guidance and confirmed it describes how to create virtualized networks and connect VMs to each other and to physical networks.

2.3.6.3 Tests

176 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 an Authorized Administrator configure a Guest VM to connect to a virtual network component. Confirm that the network is successfully connected. As an unprivileged user, attempt to configure a Guest VM to connect to a virtual network component. The operation should fail.
Findings: PASS

177 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 an Authorized Administrator configure a Guest VM to connect to a physical network component. Confirm that the network is successfully connected. As an unprivileged user, attempt to configure a Guest VM to connect to a physical network component. The operation should fail.
Findings: PASS

2.4 Identification and Authentication (FIA)

2.4.1 FIA_AFL_EXT.1/SSH Authentication Failure Handling

2.4.1.1 Tests

This information replaced by TD0432.

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

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

    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.

    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.

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

        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).

        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.

High-Level Test Description
1. Attempt to authenticate 2 times using a bad password credential. Then attempt to authenticate using a good credential. Authentication is successful. 2. Attempt to authenticate 3 times using a bad password credential. Then attempt to authenticate using a good credential. Authentication fails. 3. Wait 4 mins and 30 secs and attempt to authenticate to the TOE with good credentials. Authentication fails. 4. Wait 5 mins and 10 secs and attempt to authenticate to the TOE with good credentials. Authentication succeeds.
Findings: PASS

2.4.2 FIA_AFL_EXT.1/OLVM Authentication Failure Handling

2.4.2.1 Tests

This information replaced by TD0432.

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

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

    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.

    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.

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

        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).

        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.

High-Level Test Description
1. Attempt to authenticate 2 times using a bad password credential. Then attempt to authenticate using a good credential. Authentication is successful. 2. Attempt to authenticate 3 times using a bad password credential. Then attempt to authenticate using a good credential. Authentication fails. 3. Wait 4 mins and 30 secs and attempt to authenticate to the TOE with good credentials. Authentication fails. 4. Wait 5 mins and 10 secs and attempt to authenticate to the TOE with good credentials. Authentication succeeds.
Findings: PASS

2.4.3 FIA_UAU.5 Multiple Authentication Mechanisms

2.4.3.1 TSS

180 *(This AA found buried in a test AA.)* If 'username and PIN that releases an asymmetric key' is selected, the evaluator will examine the TSS for guidance on supported protected storage…

Findings:	The [ST] in section 5.3.4 does not claim use of a username and PIN. Thus, no information is found in the TSS.

2.4.3.2 Tests

181 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.

Findings:	The evaluator demonstrated successful authentication with a known username and password for SSH and the WebGUI (OLVM) during the testing for FIA_AFL_EXT.1/SSH and FIA_AFL_EXT.1/OLVM.

- 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.

Findings:	The evaluator demonstrated unsuccessful authentication with a known username and incorrect password for SSH and the WebGUI (OLVM) during the testing for FIA_AFL_EXT.1/SSH and FIA_AFL_EXT.1/OLVM.

182 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.

183 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.

184 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:	These test cases are conducted as part of the FCS_SSHS_EXT.1 test cases.

2.4.4 FIA_UIA_EXT.1 Administrator Identification and Authentication

2.4.4.1 TSS

185 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".

Findings:	Section 6.4.5 of the [ST] provides information on the logon process for each login method as well as describing what constitutes a successful logon.

2.4.4.2 Guidance Documentation

186 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, etc.) to logging in are described.

Findings:	[OS_GUIDE] section 10 describes how to create user accounts and setting user account passwords. [OVIRT] section 1.1 describes creating user roles and accounts. [OS_GUIDE] section 5.3, provides instructions on creating a public key pair and installing the public key on the TOE to be used for SSH authentication. [OLVM_START] Section 3, describe the Browser and Client requirements to access the VM Portal. [OS_GUIDE] section 6 Configuring TLS, provides instructions on generating and

	signing x509 certificates to be used in the VM Portal (GUI) TLS connection. [OVIRT] Appendix D: Replacing the oVirt Engine CA Certificate, provides instructions to configure the TOE to use administrator's organization's third-party certificate to identify the TOE. The evaluator examined the operational user guidance and determined that any necessary preparatory steps to logging in were described.

187 For each supported the login method, the evaluator shall ensure the operational guidance provides clear instructions for successfully logging on.

Findings:	[OLVM_START] section 3 – Logging in, provides clear instructions for successfully logging on to the TOE's Administrative Portal. [AGD] section 3.2.4 provides instructions for successfully logging on the SSH using public key authentication and password-based authentication.

188 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:	[ST] section 6.4.5 states, "Administrators must be successfully authenticated before being granted access to any TOE functionality." The evaluator has examined the operational guidance and has not identified any services that are provided before authenticating to the TOE.

2.5 Security Management (FMT)

2.5.1 FMT_MSA_EXT.1 Default Data Sharing Configuration

189 This requirement is met if FDP_VMS_EXT.1 is met.

Findings:	FDP_VMS_EXT.1 has been found to have been met.

2.5.2 FMT_SMO_EXT.1 Separation of Management and Operational Networks

2.5.2.1 TSS

190 The evaluator shall examine the TSS to verify that it describes how management and operational networks may be separated.

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

2.5.2.2 Guidance Documentation

191 The evaluator shall examine the operational guidance to verify that it details how to configure the VS with separate Management and Operational Networks.

Findings:	[AGD] / 3.1.5 states, “Separate Management and Operational Networks can be configured by Attaching and Configuring a Logical Network to a Host Network Interface.” [ADMIN] / 2.3.2 states, “You can change the settings of physical host network interfaces, move the management network from one physical host network interface to another, and assign logical networks to physical host network interfaces.” The evaluator examined the operational guidance and confirmed that it details how to configure the VS with separate Management and Operational Networks.

2.5.2.3 Tests

192 The evaluator shall configure the management network as documented. If separation is cryptographic or logical, then the evaluator shall capture packets on the management network. If Guest network traffic is detected, the requirement is not met.

High-Level Test Description
The management network is connected to the Physical NIC of the TOE. The Guest networks are on a virtual network. Create a virtual network and connect 2 Guest VMs (VM1 and VM2). Send a constant stream of packets between VM1 and VM2. Using a packet capture tool, monitor all traffic on the management network to ensure no traffic from the Guest network is detected on the management interface.
Findings: PASS

2.5.3 FMT_MOF_EXT.1 Management of Security Functions Behavior

2.5.3.1 TSS

193 The evaluator shall examine the TSS … to ensure that it describes which security management functions require Administrator privilege and the actions associated with each management function.

Findings:	Section 6.7.1 of the [ST] indicates that the reader should review the table in section 5.3.5 of the [ST]. The table in section 5.3.5 of the [ST] is a representation of the required administrative functions needed to be provided by the TOE. The table has a column labelled “Administrator”. The TSS in section 6.7.1 of the [ST] indicates that any row marked with an ‘X’ or an ‘O’ is provided to that role. Therefore, the TSS requirement is met.

2.5.3.2 Guidance Documentation

194 The evaluator shall examine the … Operational Guidance to ensure that it describes which security management functions require Administrator privilege and the actions associated with each management function.

Findings:	[OVIRT] / 1.1.2 – Table 3 describes that the Administrator role (SuperUser) “Has full permissions across all objects and levels, can manage all objects across all data centers.” [OVIRT] / 1.1.2 – Table 1 describes that the User role has the following privileges “Can log in to the VM Portal, use assigned virtual machines and pools, view virtual machine state and details.”

	The privileges described above are consistent with the management functions identified in [ST] / Table 12. All of the management functions identified in [ST] / Table 12 were examined as part of the Guidance Documentation assurance activities for FAU_GEN.1. The evaluator confirmed that the operational guidance describes the actions associated with each management function.

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

Findings:	All of the management functions identified in [ST] / Table 12 were examined as part of the Guidance Documentation assurance activities for FAU_GEN.1. The evaluator confirmed that the operational guidance describes how the Administrator and/or User are able to perform each management function that the ST claims the TOE supports.

196 The evaluator shall verify for each claimed management function that the Operational Guidance is sufficiently detailed to allow the function to be performed and that the function can be performed by the role(s) that are authorized to do so.

Findings:	All of the management functions identified in [ST] / Table 12 were examined as part of the Guidance Documentation assurance activities for FAU_GEN.1. The evaluator determined that the operational guidance is sufficiently detailed to allow the claimed management function to be performed and the function can be performed by the role(s) that are authorized to do so.

2.5.3.3 Tests

197 The evaluator shall verify that for each management function and role specified in Table 1, the defined role is able to perform all mandatory functions as well as all optional or selection-based functions claimed in the ST.

High-Level Test Description
Exercise all management functions claimed in the [ST] FMT_MOF_EXT.1 not previously tested in other SFRs. Observe that the role is able to perform all mandatory, optional or selection-based functions claimed in the ST.
Findings: PASS

198 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
Log into the TOE as an unprivileged user and attempt to exercise all management functions that are denied to that role. The TOE should deny the unprivileged user from accessing the denied management functions.

High-Level Test Description
Findings: 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

199 **TD0206:** The evaluator shall connect a device to a VM, then using a device driver running in the guest, scan the VM's processor I/O ports to ensure that the device's ports are present. (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 I/O ports are no longer present.

High-Level Test Description
Connect a HDA Sound Device to a Guest VM. On the Guest VM scan the I/O ports to ensure the device is present. Disconnect the HDA Sound Device from the Guest VM and re-run the I/O ports scan. The device's I/O ports should no longer be present.
Findings: PASS

2.6.2 FPT_EEM_EXT.1 Execution Environment Mitigations

2.6.2.1 TSS

200 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 TOE uses ASLR and stack buffer overflow protection to mitigate against execution-based vulnerabilities. This corresponds to what is specified in FPT_EEM_EXT.1.1. The TOE makes use of DEP via processor-specified functionality.

2.6.3 FPT_HAS_EXT.1 Hardware Assists

2.6.3.1 TSS

201 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:	In section 6.6.3 of the [ST], the TOE uses Intel's VT-x to reduce or eliminate the need for binary translation. In addition, the TOE uses Intel's Extended Page Tables to

	reduce or eliminate the need to maintain shadow page tables. These mechanisms are the same as those indicated in section 5.3.7 of the [ST].

2.6.4 FPT_HCL_EXT.1 Hypercall Controls

2.6.4.1 TSS

202 **TD0250:** The evaluator shall examine the TSS or operational guidance to ensure it documents all Hypercall functions at the level necessary for the evaluator to disable the functions and run tests 1 and 2, below. Documentation must include, for each function, how to call the function, function parameters and legal values, configuration settings for enabling/disabling the function, and conditions under which the function can be disabled. The TSS must also specify those functions that cannot be disabled. While there is no expectation that the evaluator will need to examine source code in order to accomplish this Assurance Activity, the evaluator must ensure that there are no obvious or publicly known Hypercall functions missing from the TSS.

Findings:	In section 6.6.4 of the [ST], the hypercalls are documented. Hypercalls are enabled by default and cannot be disabled. The [ST] section 6.6.4 states, "The parameters and legal values for the above hypercalls are documented at the following location: https://www.kernel.org/doc/html/latest/virt/kvm/x86/hypercalls.html Where not explicitly implied, the legal values are within the bounds of the data type." The evaluator reviewed the source code for the Linux KVM kernel and found that the set of hypercalls described in the TSS match the set offered in the KVM header file: https://oss.oracle.com/ol7/SRPMS-updates/kernel-uek-5.4.17-2036.103.3.1.el7uek.src.rpm. The header file in question is include/uapi/linux/kvm_para.h. It contains several other hypercalls, but they are specific to MIPS or PowerPC architectures. The [ST] only claims to run on Intel x86 architectures.

2.6.4.2 Guidance Documentation

203 The evaluator shall examine the operational guidance to ensure it contains instructions for how to configure interface functions per FPT_HCL_EXT.1.2.

Findings:	[AGD] / 3.1.4 states, "The TOE functions for self-protection from hardware assists and hypervisor calls are enabled by default and do not require configuration." The [AGD] is consistent with the assignment operation in FPT_HCL_EXT.1.3 and has been reviewed by the TRRT.

2.6.4.3 Tests

204 The evaluator shall perform the following tests:

- For each configurable function that meets FPT_HCL_EXT.1.2, the evaluator shall follow the operational guidance to enable the function. The evaluator shall then attempt to call each function from within the VM. If the call is allowed, then the test succeeds.

High-Level Test Description
Hypercalls are enabled by default and cannot be disabled. Using a test framework, instantiate guest VMs which issue hypercalls to the host. Verify that the hypercall interface succeeds and that the VM host receives the hypercall request. Verify within the guest VM that the hypercall request is permitted. The TOE supports the following hypercalls: a) KVM_HC_VAPIC_POLL_IRQ - Trigger guest exit so that the host can check for pending interrupts on re-entry. b) KVM_HC_KICK_CPU - Hypercall used to wakeup a vCPU from halt (HLT) state. c) KVM_HC_CLOCK_PAIRING - Hypercall used to synchronize host and guest clocks. d) KVM_HC_SEND_IPI - Send Inter-processor Interrupt (IPIs) to multiple vCPUs.
Findings: PASS

- **TD0250:** For each configurable function that meets FPT_HCL_EXT.1.2, the evaluator shall configure the TSF to disable the function. The evaluator shall then attempt to call the function from within the VM. If the call is blocked, then the test succeeds.

Notes:	None of the hypercalls are configurable in this TOE. All calls are tested as part of test 1 above.

2.6.5 FPT_RDM_EXT.1 Removable Devices and Media

2.6.5.1 TSS

205 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:	The evaluator consulted the original SFR in the [PP] for FPT_RDM_EXT.1 as the TSS request was not entirely clear. Based on the review of the SFR and its application note, the evaluator is satisfied that the TSS in section 6.6.5 of the [ST] contains the information necessary. The AA is effectively asking the TSS to reiterate the supported devices and removable media defined in FPT_RDM_EXT.1.1 (and augmented in the SFR's application note) and the claimed actions in FPT_RDM_EXT.1.2. Based on this, the TSS has information consistent with the claims from the SFR in section 5.3.7 of the [ST]. In summary, the TOE administrator explicitly configures access to removable media devices and therefore the removable media itself.

2.6.5.2 Guidance Documentation

206 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:	[VMM_GUIDE] section 6.10.1 - Adding a Host Device to a Virtual Machine, provides instructions on configuring the following physical platform resources: - USB [VMM_GUIDE] section Appendix A - Virtual Machine Boot Options Settings Explained, provides details on configuring the virtual CD/DVD ROM. [VMM_GUIDE] section 7.10.2 - Using sysprep to initialize a virtual machine, provides instructions on configuring the virtual Floppy Disk for Windows based VM installations. The evaluator examined the operational environment and confirmed it documents how an administrator configures the behavior of each media or device.

2.6.5.3 Tests

207 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
Create two Client VMs (VM1 and VM2) in different information domains. Connect media or device to VM1. Disconnect the media or device from VM1 and connect it to VM2. The media or devices claimed: • Virtual CD-ROM • Virtual Floppy Drive • USB Storage Drive
Findings: PASS

2.6.6 FPT_TUD_EXT.1 Trusted Updates to the Virtualization System

2.6.6.1 TSS

208 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.

Findings:	According to section 6.6.6 of the [ST], “An Oracle PGP Public Key is used to verify the RPM during installation. The public key is installed on the system at the time of installation. The TOE leverages 2048 bit RSA digital signature mechanism for signing and verification of packages/updates.”

	The TSS claims that the digital signature is checked at the time of package installation. If the signature verification is successful, then the RPM package is installed. Otherwise, it fails the installation. The administrator must download the RPM from the Oracle download center.

209 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], updates are verified using 2048-bit RSA digital signatures that use SHA2-256. The TSS claims that the digital signature is checked at the time of package installation. If the signature verification is successful, then the RPM package is installed. Otherwise, it fails the installation. The administrator must download the RPM from the Oracle download center.

210 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 TOE uses a PGP public key for digital signature verification and not a certificate-based mechanism.

2.6.6.2 Guidance Documentation

211 *(The following guidance AA was found buried in a TSS requirement and has been edited inline for clarity.)* If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, … the evaluator … ensures that the TSS (or administrator guidance) describes how the certificates are installed/updated/selected, if necessary.

Findings:	The TOE uses a PGP public key for digital signature verification and not a certificate-based mechanism.

2.6.6.3 Tests

212 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
Verify the currently installed version of the Virtualization System (VS)

High-Level Test Description
Perform an update to the VS with a legitimate update. Verify the currently installed version of the VS matches the update.
Findings: 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
Verify the currently installed version of the Virtualization System (VS) Attempt to install all forms of illegitimate updates. The TOE rejects all attempts to install the illegitimate updates.
Findings: PASS

2.6.7 FPT_VDP_EXT.1 Virtual Device Parameters

2.6.7.1 TSS

This information replaced by TD0443.

213 The evaluator shall examine the TSS to ensure it lists all virtual devices accessible by the guest OS.

Findings:	In section 6.6.7 of the [ST], the ST author outlines the list of virtual devices which are supported by the TOE. These include vCPUs, a variety of bus architectures, virtual mouse and keyboards, emulated storage devices, emulated IDE devices, emulated floppy disk device, emulated sound devices, emulated graphics cards, emulated network devices and emulated watchdog devices.

214 The TSS, or a separate proprietary document, must also document all virtual device interfaces at the level of I/O ports -- including port number(s) (absolute or relative to a base), port name, and a description of legal input values. The documentation must be sufficient to enable the evaluator to effectively run the tests in FPT_DVD_EXT.1.

Findings:	The vendor provided a separate proprietary document [VIRT_DEV]. The document details all virtual devices at the level of I/O ports, including port numbers, port name and a description of legal input values. The evaluator determined the documentation to be sufficient to perform tests in FPT_DVD_EXT.1.

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

Findings:	The evaluator compared the list of supported virtual devices with the proprietary document [VIRT_DEV] and determined that there are no obvious or publicly known virtual I/O ports missing from the TSS.

216 The evaluator ensures that the ST includes the following statement attesting that parameters passed from a Guest VM to virtual device interfaces are thoroughly validated, that all values outside the legal values specified in the TSS are rejected, and that any data passed to the virtual device interfaces is unable to degrade or disrupt the functioning of other VMs, the VMM, or the Platform:

> *Parameters passed from Guest VMs to virtual device interfaces are thoroughly validated and all illegal values (as specified in the TSS) are rejected. Additionally, parameters passed from Guest VMs to virtual device interfaces are not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. Thorough testing and architectural design reviews have been conducted to ensure the accuracy of these claims, and there are no known design or implementation flaws that bypass or defeat the security of the virtual device interfaces.*

Findings:	The attestation can be found in section 6.6.7 of the [ST].

2.6.8 FPT_VIV_EXT.1 VMM Isolation from VMs

2.6.8.1 TSS

217 The evaluator ensures that the ST includes the following statement attesting that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform:

> *"Software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform. There are no design or implementation flaws that bypass or defeat VM isolation."*

Findings:	The attestation can be found in section 6.6.8 of the [ST].

2.7 TOE Access (FTA)

2.7.1 FTA_TAB.1 TOE Access Banner

2.7.1.1 Tests

218 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 logging can occur.

High-Level Test Description
Configure the TOE to display an access banner on both the SSH CLI and the OLVM interfaces.

High-Level Test Description
Connect to the SSH CLI and OLVM interfaces and confirm the access banner is displayed prior to being able to authenticate to the TOE.
Findings: PASS

2.8 Trusted path/channels (FTP)

2.8.1 FTP_ITC_EXT.1 Trusted Channel Communications

2.8.1.1 TSS

219 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/or remote users used for the channel as well as the protocol that is used for each.

Findings:	The TSS in section 6.9.1 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 5.3.9 of the [ST]. The TSS indicates that SSH is used for syslog (remote entity); SSH is used for the CLI (administrative access); and HTTPS/TLS is used to implement the OLVM web interface (administrative access).

2.8.1.2 Tests

220 The evaluator will configure the TOE to communicate with each external IT entity and/or 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.

Findings:	Testing to ensure conformance with the protocols identified in the selection were performed in the SFRs associated with the protocols claimed. FCS_TLSS_EXT.1 tested the trusted channel between the TOE and the remote administrator over TLS. FCS_SSHS_EXT.1 tested the trusted channel between the TOE and the remote administrator over SSH. FCS_SSHC_EXT.1 tested the trusted channel between the TOE and the remote syslog server over SSH.

2.8.2 FTP_UIF_EXT.1 User Interface: I/O Focus

2.8.2.1 TSS

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

Findings:	Section 6.9.3 of the [ST] states that the TOE supports keyboard and pointer input devices. These are standard devices used in non-virtualized environments as well.

2.8.2.2 Guidance Documentation

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

Findings:	[AGD] Section 3.2.4 states, “VMs are assigned a unique name when they are created. This name is displayed to users of the VM in the title bar of the Remote Viewer window in which the VM is running.”

2.8.2.3 Tests

223 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
Open a terminal window in two VMs (VM1 and VM2). Select VM1 where it is indicated to have the input focus and type into the terminal window. Ensure that the second VM2 without the input focus does not register any keyboard presses. Select VM2 and type into the terminal window. Ensure that no keyboard presses are registered by VM1. On both VMs (VM1 and VM2) run a program to monitor mouse movement and button presses. Select VM1 and move the mouse around the screen and click the buttons. Ensure that no mouse input was detected on VM2. Select VM2 and move the mouse around the screen and click the buttons. Ensure that no mouse input was detected on VM1.
Findings: PASS

2.8.3 FTP_UIF_EXT.2 User Interface: Identification of VM

2.8.3.1 TSS

224 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:	Section 6.9.4 of the [ST] indicates that the TOE implements unique names for identifying VMs. This name is displayed to users in the Remote Viewer window in which the VM is running. According to section 6.9.4 of the [ST], the name is “unique” which strongly implies that conflicts are prevented due to the uniqueness property enforced by the TOE.

2.8.3.2 Tests

225 The evaluator shall perform the following test:

226 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 multiple VMs with the same identifier.
Findings: PASS

3 Evaluation Activities for Optional Requirements

227 No optional requirements have been selected by this evaluation.

4 Evaluation Activities for Selection-Based Requirements

4.1.1 FCS_HTTPS_EXT.1 HTTPS Protocol

4.1.1.1 TSS

228 The evaluator shall check the TSS to ensure that it is clear on how HTTPS uses TLS to establish an administrative session, focusing on any client authentication required by the TLS protocol vs. security administrator authentication which may be done at a different level of the processing stack.

Findings:	Section 6.2.10 provides the necessary information. Only username/password-based authentication is claimed.

4.1.1.2 Tests

229 Testing for this activity is done as part of the TLS testing; this may result in additional testing if the TLS tests are done at the TLS protocol level.

NOTE:	Please refer to testing assurance activities in FCS_TLSS_EXT.1 for details.

4.1.2 FCS_TLSS_EXT.1 TLS Server Protocol

4.1.2.1 FCS_TLSS_EXT.1.1 TSS

230 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

Findings:	Section 6.2.11 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_TLSS_EXT.1.1 Guidance Documentation

231 The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

Findings:	[AGD] section 3.2.2 provides instructions for configuration so the TLS conforms to the claimed ciphersuites. “To configure the TLS cipher suites, add the following parameter to the /etc/httpd/conf.d/ssl.conf file:

SSLCipherSuite AES128-SHA: DHE-RSA-AES128-SHA256: DHE-RSA-AES256-SHA256”

4.1.2.3 FCS_TLSS_EXT.1.1 Tests

232 Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites 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 ciphersuite 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 ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using the claimed ciphersuites.
Findings: PASS

233 Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server’s ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using an unsupported ciphersuite. Then connect to the TOE using TLS_NULL_WITH_NULL_NULL.
Findings: PASS

234 Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using a supported ciphersuite. The test tool will, at the appropriate time, send back a Client Key Exchange message that does not match the expected key exchange algorithm. For RSA key exchanges, the test tool will send back an ECDHE key exchange. For ECDHE and DHE key exchanges, the test tool will send back an RSA key exchange.
Findings: PASS

235 Test 4: The evaluator shall perform the following modifications to the traffic:

- Modify at a byte in the client’s nonce in the Client Hello handshake message, and verify that the server rejects the client’s Certificate Verify handshake message (if using mutual authentication) or that the server denies the client’s Finished handshake message.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using a supported ciphersuite. The test tool will, modify the client's none in the Client Hello handshake message.
Findings: PASS

- **TD0431:** (conditional): If an ECDHE or DHE cipher suite is selected, modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using a supported DHE ciphersuite. The test tool will, modify the DHE public key in the Client's Key Exchange handshake message. Verify that the TOE denies the client's Finished handshake message.
Findings: PASS

- Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE and modify the first payload byte in the Client Finished message.
Findings: PASS

- **TD0431:** [conditional] After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection. This test is not required for applications with a TLS implementation that does not support session IDs.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE and capture the session ID sent back from the server. At the end of this initial handshake, reorder the ChangeCipherSpec and Finished messages so that the connection does not complete. Secondly, reconnect to the TOE and sent the previously captured session ID in the hopes that we can avoid the remainder of the handshake. Verify the TOE does not permit this.
Findings: PASS

- Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE and send a garbled message from the client after the client has issued the ChangeCipherSpec message.
Findings: PASS

4.1.2.4 FCS_TLSS_EXT.1.2 TSS

Replaced with TD0431.

236 The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

Findings:	Section 6.2.11 of the [ST] states that the TOE will only accept TLS 1.2 and reject all other protocol versions. This assertion matches the claims in section 5.3.2 of the [ST].

4.1.2.5 FCS_TLSS_EXT.1.2 Guidance Documentation

Replaced with TD0431.

237 The evaluator shall verify that any configuration necessary to meet the requirement are contained in the AGD guidance.

Findings:	[AGD] section 3.2.2 describes how to configure the TLS ciphersuites. By default the TLS configuration denies the TLS protocols selected in the ST. No Configuration required.

4.1.2.6 FCS_TLSS_EXT.1.2 Tests

Replaced with TD0431.

238 The evaluator shall send a Client Hello requesting a connection for all mandatory and selected protocol versions in the SFR (e.g. by enumeration of protocol versions in a test client) and verify that the server denies the connection for each attempt.

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE and attempt to negotiate SSL 2.0, SSL 3.0, TLS 1.0 and TLS 1.1 and confirm that the TOE denies the connection for each attempt.
Findings: PASS

4.1.2.7 FCS_TLSS_EXT.1.3 TSS

239 The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

Findings:	Section 6.2.11 of the [ST] indicates that the TOE is capable of both RSA key exchange using RSA 2048- and 3072-bit keys and using DHE key agreement with keys of size 2048 and 3072 bits.

4.1.2.8 FCS_TLSS_EXT.1.3 Guidance Documentation

240 The evaluator shall verify that any configuration necessary to meet the requirement is contained in the AGD guidance.

Findings:	[AGD] section 3.2.2 describes how to configure the TLS ciphersuites. The configured ciphersuites include the claimed key exchange methods.

4.1.2.9 FCS_TLSS_EXT.1.3 Tests

241 If the second selection includes any choice other than “no other”, the evaluator shall attempt a connection using an ECDHE ciphersuite and a configured curve and, using a packet analyzer, verify that the key agreement parameters in the Key Exchange message are the ones configured. (Determining that the size matches the expected size for the configured curve is sufficient.) The evaluator shall repeat this test for each supported NIST Elliptic Curve and each supported Diffie-Hellman key size.

High-Level Test Description
The TOE uses the same key size of the x509 RSA certificate key in the DHE Key Exchange. By default the TOE uses a x509 RSA certificate of key size 2048. Using a Lightship developed TLS client, connect to the TOE with a TLS_DHE-* ciphersuite and examine the Server Key Exchange message to confirm the DHE key size is 2048 bits. Run the sslscan tool to confirm that the DHE key size is consistent. Configure the TOE to use a x509 RSA certificate with key size 3072. Using a Lightship developed TLS client, connect to the TOE with a TLS_DHE-* ciphersuite and examine the Server Key Exchange message to confirm the DHE key size is 3072 bits. Run the sslscan tool to confirm that the DHE key size is consistent.
Findings: PASS

4.1.3 FIA_PMG_EXT.1 Password Management

4.1.3.1 Guidance Documentation

242 The evaluator shall examine the operational guidelines 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:	[AGD] section 3.2.5.1 provides instructions on configuring the TOE to require strong passwords and how to set the minimum password length for the SSH interface. [AGD] section 3.2.5.1 provides instruction on configuring the TOE to require strong passwords and how to set the minimum password length for the OLVM (WebGUI) interface.

4.1.3.2 Tests

243 The evaluator shall also perform the following tests. Note that one or more of these tests may be performed with a single test case.

- 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 minimum password length to 15 characters. Attempt to set a password less than the minimum length and show it is not accepted. Attempt to set passwords that fail to include characters from the out-of-the-box password complexity requirements and show they are not accepted. Attempt to set a password that meets the complexity and length requirements and show it is accepted. Show the password can be used on applicable management interfaces to log in successfully. Show that an admin with privileges can change another user's password and that the audit log reflects this capability.
Findings: PASS

4.1.4 FTP_TRP.1 Trusted Path

4.1.4.1 TSS

244 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 6.9.2 of the [ST] indicates that the TOE offers two methods of remote administration: via a web server for OLVM (protected by HTTPS/TLS) and over a CLI (protected by SSH). Both of these protocols are selected in section 5.3.2 of the [ST].

4.1.4.2 Guidance Documentation

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

Findings:	[OLVM] section 2.1.3 provides instructions for establishing a remote administrative session with the OLVM (WebGUI). "You log in to the Administration Portal using a web browser and the default admin@internal user. 1. Go to https://<manager-fqdn>/ovirt-engine. The Welcome page displays. 2. (Optional) Change the preferred language from the drop-down list on the Welcome page. You can view the Administration Portal in multiple languages. The default language is based on the locale of your web browser. 3. Click Administration Portal. The Login page displays. 4. Enter admin for the Username and the password you specified when you configured the Manager. 5. From the Profile list, select internal and click Log In." [AGD] section 3.2.4 provides instructions for establishing an administrative session over SSH. "The Oracle Linux CLI can be accessed over SSH … An SSH client should be used to connect. Once successful authentication is complete, the user will be provided with the command prompt."

4.1.4.3 Tests

246 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.

Notes:	The TOE claims SSH and HTTPS/TLS for its remote administration. The evaluator has tested SSH and HTTPS/TLS for remote administration as part of testing performed as part of FCS_SSHS_EXT.1 and FCS_TLSS_EXT.1. The evaluator ensured that the 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
Verify all open ports on the TOE to confirm there are no other methods to establish a remote administrative session.
Findings: 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 capture tool, examine the traffic of the administrative sessions to confirm that the channel data is not sent in plaintext.
Findings: 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 Lightship custom SSH tool, attempt to modify the channel data and confirm that it is detected by the TOE. The custom SSH tool will establish a connection, will mangle every 5th data packet (packets defined as a message type with type 94 (0x5e)) and dumps the encrypted buffer before and after the modification. HTTPS/TLS channel was tested as part of TLS protocol testing in FCS_TLSS_EXT.1.1- Test 4.
Findings: PASS

247 Further assurance activities are associated with the specific protocols.

4.1.5 FCS_COP.1(1)/SSH Cryptographic Operation - Encryption/Decryption (Refined)

4.1.5.1 TSS

248 **TD0240 -** The evaluator shall review the TSF of the base PP to verify consistency with the functionality that was claimed by the base PP to ensure that applicable dependencies are met.

Findings:	Section 6.2.4 of the [ST] was reviewed and found consistent with the selections in the associated SFR.

249 **TD0240 -** If perform encryption/decryption services is chosen, the evaluator shall verify that the TSS describes the counter mechanism including rationale that the counter values provided are unique.

Findings:	The ST in section 6.2.4 claims that the counter mechanism is implemented by the underlying platform.

4.1.5.2 Tests

250 **This whole section modified by TD0240**

AES-CTR Tests:

251 Test 1: Known Answer Tests (KATs)

252 There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, 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.

253 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.

254 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.

255 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.

256 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.

257 Test 2: Multi-Block Message Test

258 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.

259 Test 3: Monte-Carlo Test

260 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.

261 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]

262 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. CAVP A1400, A1401 and A1402 have the appropriate certificates for the claimed platforms for this mode and key sizes: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?product=13879

	https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?product=13878 https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?product=13877

263 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, for each listening SSH socket connection on the TOE, the evaluator configures an SSH client to connect with an invalid cryptographic algorithm and key-size. The evaluator observes that the connection fails. Likewise, for initiated connection, the evaluator configures a listening SSH socket on the remote server that accepts only invalid cryptographic algorithms and keys. The evaluator observes that the connection fails.

High-Level Test Description
Using the Lightship SSH client attempt to connect to the TOE using an invalid cryptographic algorithm and key-size. TOE refuses to connect. Using the Lightship SSH server configured to only accept invalid cryptographic algorithm and key-size. Attempt to have the TOE connect to the SSH server and observe that the connection fails. The Lightship SSH client and server will be configured to present proposals with invalid cryptographic algorithms and valid key sizes and valid cryptographic algorithms with invalid key sizes. (aes128-gcm, aes256-gcm, aes192-cbc, aes192-ctr)
Findings: PASS

4.1.6 FCS_SSH_EXT.1 SSH Protocol

4.1.6.1 TSS

264 The evaluator will ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Findings:	The top-level SFR element selections are consistent with the selections made in the dependent components. The ST claims RFC 6668 which is consistent with the inclusion of HMAC-SHA2 algorithms in dependent components. Subsequently, the fact that the top-level SFR element claims SSH in a client and server capacity implies FCS_SSHS_EXT.1 and FCS_SSHC_EXT.1 are claimed (and they are).

4.1.7 FCS_SSHC_EXT.1 SSH Protocol – Client

4.1.7.1 FCS_SSHC_EXT.1.1

4.1.7.1.1 TSS

265 **TD0420 -** The evaluator will check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHC_EXT.1.4, and ensure that password-based authentication methods, if supported, are described.

Findings:	Section 6.2.12 of the [ST] specifies that the TOE supports only rsa-sha2-512 based authentication methods. This conforms to the selections in FCS_SSHC_EXT.1.4

	(host public key selections) and FCS_SSHC_EXT.1.1 (other authentication methods).

4.1.7.1.2 Tests

266 Test 1: The evaluator will, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection to an SSH server. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.

High-Level Test Description
Attempt to establish an SSH connection to the SSH server using a rsa-sha2-512 public key. Show that the connection succeeds.
Findings: PASS

267 **TD0420:** Test 2 [conditional]: Using the guidance documentation, the evaluator will configure the TOE to perform password-based authentication to an SSH server, and demonstrate that a user can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.

Notes:	The TOE does not claim password-based authentication to the audit server.

4.1.7.2 FCS_SSHC_EXT.1.2

4.1.7.2.1 TSS

268 The evaluator will check that the TSS describes how ‘large packets’ in terms of RFC 4253 are detected and handled.

Findings:	Section 6.2.12 of the [ST] specifies that SSH packets greater than 256KB are automatically dropped.

4.1.7.2.2 Tests

269 The evaluator will demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

High-Level Test Description
Send a packet from the SSH client to the TOE SSH server slightly smaller than the claimed maximum and show that the TOE accepts the packet. Send a packet from the SSH client to the TOE SSH server slightly larger than the defined maximum and show the TOE drops the packets and terminates the connection.
Findings: PASS

4.1.7.3 FCS_SSHC_EXT.1.3

4.1.7.3.1 TSS

270 The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator will check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.12 of the [ST] specifies that encryption algorithms aes128-ctr, aes256-ctr, aes128-cbc, and aes256-cbc are supported. This list is identical to the selections made in FCS_SSHC_EXT.1.3. No optional characteristics are claimed.

4.1.7.3.2 Guidance Documentation

271 The evaluator will also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the SSH client to conform to the claimed ciphers. “Ciphers aes-ctr-128, aes-ctr-256, aes-cbc-128, aes-cbc-256”

4.1.7.3.3 Tests

272 Test 1: The evaluator will establish an SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Connect using each of the claimed ciphersuites and show they are successful in turn.
Findings: PASS

273 Test 2: The evaluator will configure an SSH server to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator will attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

High-Level Test Description
Connect to the TOE over SSH using the 3des-cbc cipher and show it fails to successfully negotiate.
Findings: PASS

4.1.7.4 FCS_SSHC_EXT.1.4

4.1.7.4.1 TSS

274 The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator will check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.12 of the [ST] specifies that the TOE supports rsa-sha2-512 public key algorithms. This conforms to the selection in FCS_SSHC_EXT.1.4.

4.1.7.4.2 Guidance Documentation

275 The evaluator will also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the claimed the public key algorithms: **"**HostKeyAlgorithms rsa-sha2-512"

4.1.7.4.3 Tests

276 Test 1: The evaluator will establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Configure the SSH Server to present a rsa-sha2-512 host public key. Attempt to establish an SSH session from the TOE and show the TOE successfully negotiates the Host public key algorithm.
Findings: PASS

277 Test 2: The evaluator will configure an SSH server to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator will attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.

High-Level Test Description
Configure the SSH server to only allow ssh-dsa public key algorithm and no other public key algorithms. Show that the connection fails.
Findings: PASS

4.1.7.5 FCS_SSHC_EXT.1.5

4.1.7.5.1 TSS

278 **This entire section modified by TD0446**

279 The evaluator will check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

Findings:	Section 6.2.12 of the [ST] specifies that the TOE supports hmac-sha1, hmac-sha2-256, and hmac-sha2-512 data integrity algorithms. This conforms to the selections in FCS_SSHC_EXT.1.5.

4.1.7.5.2 Guidance Documentation

280 **This entire section modified by TD0446**

281 The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the “none” MAC algorithm is not allowed).

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the claimed the data integrity algorithms: **“**MACs hmac-sha1, hmac-sha2-256, hmac-sha2-512”

4.1.7.5.3 Tests

282 **This entire section modified by TD0446**

283 Test 1: The evaluator will establish a SSH connection using each of the integrity algorithms, except "implicit", specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Using each of the defined integrity algorithms (and a supported ciphersuite permitting its use), show that the algorithm is supported.
Findings: PASS

284 Test 2: The evaluator will configure an SSH server to only allow the “none” MAC algorithm. The evaluator will attempt to connect from the TOE to the SSH server and observe that the attempt fails.

Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

High-Level Test Description
Using the ‘none’ integrity algorithms (and a supported ciphersuite permitting its use), show that the algorithm is NOT supported.
Findings: PASS

285 Test 3: The evaluator will configure an SSH server to only allow the hmac- md5 MAC algorithm. The evaluator will attempt to connect from the TOE to the SSH server and observe that the attempt fails.

High-Level Test Description
Using the hmac-md5 integrity algorithms (and a supported ciphersuite permitting its use), show that the algorithm is NOT supported.
Findings: PASS

4.1.7.6 FCS_SSHC_EXT.1.6

4.1.7.6.1 TSS

286 The evaluator will check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

Findings:	Section 6.2.12 of the [ST] specifies that the TOE supports the diffie-hellman-group14-sha1 key exchange algorithm. This conforms to the selection in FCS_SSHC_EXT.1.6.

4.1.7.6.2 Guidance Documentation

287 The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the claimed the key exchange algorithms: **“**KexAlgorithms diffie-hellman-group14-sha1”

4.1.7.6.3 Tests

288 Test 1: The evaluator will configure an SSH server to permit all allowed key exchange methods. The evaluator will attempt to connect from the TOE to the SSH server using each allowed key exchange method, and observe that each attempt succeeds.

High-Level Test Description
Using each of the defined key exchange algorithms show that the algorithm is supported.
Findings: PASS

4.1.7.7 FCS_SSHC_EXT.1.7

4.1.7.7.1 Tests

289 **TD0331 -** Test 1: The evaluator will configure the TOE to create a log entry when a rekey occurs. The evaluator will connect to the TOE with an SSH client and cause a rekey to occur according to the selection(s) in the ST, and subsequently the evaluator uses available methods and tools to verify that rekeying occurs. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports auditing of rekey events.

High-Level Test Description
Show that when the TOE reaches its rekey limits, the TOE will perform a rekey operation and that the rekey action is capable of being logged.
Findings: PASS

4.1.7.8 FCS_SSHC_EXT.1.8

4.1.7.8.1 Tests

290 Test 1: The evaluator will delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator will initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the user to accept or deny the key before continuing the connection.

High-Level Test Description
Delete SSH server host keys for the SSH server. Initiate a connection from the TOE to the SSH server and confirm displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the user to accept or deny the key before continuing the connection.
Findings: PASS

291 Test 2: The evaluator will add an entry associating a host name with a public key into the TOE's local database. The evaluator will replace, on the corresponding SSH server, the server's host key with a different host key. The evaluator will initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords).

High-Level Test Description
Replace the previous SSH server with a new host key. Attempt to initiate a connection from the TOE to the SSH server using password-based authentication. Ensure the TOE rejects the connection and ensure that the password was not transmitted to the SSH server.
Findings: PASS

4.1.8 FCS_SSHS_EXT.1 SSH Protocol – Server

4.1.8.1 FCS_SSHS_EXT.1.1

4.1.8.1.1 TSS

292 **TD0420 -** The evaluator will check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHS_EXT.1.4, and ensure that password-based authentication methods, if supported, are described.

Findings:	Section 6.2.13 of the [ST] specifies that the TOE supports rsa-sha2-256 and rsa-sha2-512 public key client authentication and password based authentication methods. This conforms to the selection in FCS_SSHS_EXT.1.4.

4.1.8.1.2 Tests

293 Test 1: The evaluator will, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection from

an SSH client. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.

High-Level Test Description
Configure the user to be permitted to use the claimed public key algorithm. Load the public key half into the TOE. Log into the TOE with the correct user using the private key half and show it is successful. Repeat for each claimed public key algorithm.
Findings: PASS

294 Test 2: The evaluator shall choose one public key algorithm supported by the TOE. The evaluator shall generate a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

High-Level Test Description
Building on the previous test case, generate a new RSA key pair. Attempt to connect to the TOE with the new key pair and show that the authentication fails.
Findings: PASS

295 **TD0420 -** Test 3 [conditional]: Using the guidance documentation, the evaluator will configure the TOE to perform password-based authentication on a client, and demonstrate that a user can be successfully authenticated by the TOE using a password as an authenticator.

High-Level Test Description
Log into the TOE using a username and known-good password and show it is successful.
Findings: PASS

296 **TD0420 -** Test 4 [conditional]: The evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.

High-Level Test Description
Log into the TOE using a username and a bad password and show it is NOT successful.
Findings: PASS

4.1.8.2 FCS_SSHS_EXT.1.2

4.1.8.2.1 TSS

297 The evaluator will check that the TSS describes how ‘large packets’ in terms of RFC 4253 are detected and handled.

Findings:	Section 6.2.13 of the [ST] specifies that SSH packets greater than 256KB are automatically dropped.

4.1.8.2.2 Tests

298 The evaluator will demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

High-Level Test Description
Send a packet from the SSH client to the TOE SSH server slightly smaller than the claimed maximum and show that the TOE accepts the packet. Send a packet from the SSH client to the TOE SSH server slightly larger than the defined maximum and show the TOE drops the packets and terminates the connection.
Findings: PASS

4.1.8.3 FCS_SSHS_EXT.1.3

4.1.8.3.1 TSS

299 The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator will check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.13 of the [ST] specifies that encryption algorithms aes128-ctr, aes256-ctr, aes128-cbc, and aes256-cbc are supported. This list is identical to the selections made in FCS_SSHS_EXT.1.3.

4.1.8.3.2 Guidance Documentation

300 The evaluator will also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the SSH client to conform to the claimed ciphers. “Ciphers aes-ctr-128, aes-ctr-256, aes-cbc-128, aes-cbc-256”

4.1.8.3.3 Tests

301 Test 1: The evaluator will initiate an SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Connect using each of the claimed ciphersuites and show they are successful in turn.
Findings: PASS

302 Test 2: The evaluator will configure an SSH client to only propose the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator will attempt to establish an SSH connection from the client to the TOE server and observe that the connection is rejected.

High-Level Test Description
Connect to the TOE over SSH using the 3des-cbc cipher and show it fails to successfully negotiate.
Findings: PASS

4.1.8.4 FCS_SSHS_EXT.1.4

4.1.8.4.1 TSS

303 The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator will check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

Findings:	Section 6.2.13 of the [ST] specifies that the TOE supports ssh-rsa, rsa-sha2-256, rsa-sha2-512 public key algorithms. This conforms to the selection in FCS_SSHS_EXT.1.4. No optional characteristics are claimed.

4.1.8.4.2 Guidance Documentation

304 The evaluator will also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the claimed the public key algorithms: **“**HostKeyAlgorithms rsa-sha2-512,rsa-sha2-256,ssh-rsa” “PubkeyAcceptedKeyTypes rsa-sha2-512,rsa-sha2-256,ssh-rsa” Note: ssh-rsa must be added to PubKeyAcceptedKeyTypes, but is not advertised or accepted by the SSH server.

4.1.8.4.3 Tests

305 The Test 1: Using an appropriately configured client, the evaluator will establish an SSH connection using each of the public key algorithms specified by the requirement to authenticate. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Using SSH client, attempt to connect to the TOE SSH server only allowing each of the specified public key algorithms.
Findings: PASS

306 Test 2: The evaluator will configure an SSH client to propose only the ssh-dsa public key algorithm and no other public key algorithms. Using this client, the evaluator will attempt to establish an SSH connection to the TOE and observe that the connection is rejected.

High-Level Test Description
Using SSH client, attempt to connect to the TOE by proposing only the ssh-dsa public key algorithm. Show that the connection is rejected.
Findings: PASS

4.1.8.5 FCS_SSHS_EXT.1.5

4.1.8.5.1 TSS

307 **This entire section modified by TD0446**

308 The evaluator will check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

Findings:	Section 6.2.13 of the [ST] specifies that the TOE supports hmac-sha1, hmac-sha2-256, and hmac-sha2-512 data integrity algorithms. This conforms to the selections in FCS_SSHS_EXT.1.5.

4.1.8.5.2 Guidance Documentation

309 **This entire section modified by TD0446**

310 The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the “none” MAC algorithm is not allowed).

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the claimed the data integrity algorithms: **“**MACs hmac-sha1, hmac-sha2-256, hmac-sha2-512”

4.1.8.5.3 Tests

311 **This entire section modified by TD0446**

312 Test 1: Using an appropriately configured client, the evaluator will establish a SSH connection using each of the integrity algorithms, except "implicit", specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

High-Level Test Description
Using SSH client, log into the TOE using each of the claimed integrity algorithms in turn and show that the communication is successful. Review the negotiation line from the server to ensure that

High-Level Test Description
there are no additional integrity algorithms claimed by the implementation that differ from the ST or the PP requirements.
Findings: PASS

313 Test 2: The evaluator will configure an SSH client to only allow the “none” MAC algorithm. Using this client, the evaluator will attempt to connect to the TOE and observe that the attempt fails.

Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

High-Level Test Description
Using the ‘none’ integrity algorithms (and a supported ciphersuite permitting its use), show that the algorithm is NOT supported.
Findings: PASS

314 Test 3: The evaluator will configure an SSH client to only allow the hmac- md5 MAC algorithm. using this client, the evaluator will attempt to connect to the TOE and observe that the attempt fails.

High-Level Test Description
Using the hmac-md5 integrity algorithms (and a supported ciphersuite permitting its use), show that the algorithm is NOT supported.
Findings: PASS

4.1.8.6 FCS_SSHS_EXT.1.6

4.1.8.6.1 TSS

315 The evaluator will check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

Findings:	Section 6.2.13 of the [ST] specifies that the TOE supports the diffie-hellman-group14-sha1 key exchange algorithm. This conforms to the selection in FCS_SSHS_EXT.1.6.

4.1.8.6.2 Guidance Documentation

316 The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections to the TOE.

Findings:	[AGD] section 3.2.3.2 provides the following instruction to configure the claimed the key exchange algorithms: **“**KexAlgorithms diffie-hellman-group14-sha1”

4.1.8.6.3 Tests

317 Test 1: For each of the allowed key exchange methods, the evaluator will configure an SSH client to propose only it and attempt to connect to the TOE and observe that each attempt succeeds.

High-Level Test Description
Using SSH client, log into the TOE using diffie-hellman-group-14-sha1 key exchange algorithm and show that the communication is successful.
Findings: PASS

318 Test 2: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the SSH Server and observe that the attempt fails.

High-Level Test Description
Using SSH client, log into the TOE using diffie-hellman-group-1-sha1 key exchange algorithm and show that the communication is unsuccessful.
Findings: PASS

4.1.8.7 FCS_SSHS_EXT.1.7

4.1.8.7.1 Tests

319 **TD0331 -** Test 1: The evaluator will configure the TOE to create a log entry when a rekey occurs. The evaluator will connect to the TOE with an SSH client and cause a rekey to occur according to the selection(s) in the ST, and subsequently the evaluator uses available methods and tools to verify that rekeying occurs. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports auditing of rekey events.

High-Level Test Description
Using a custom SSH client, log into the TOE and push at least 2^28 packets of data to force rekeying. Show that the TOE rekeys before the 2^28 packets of data is reached.
Findings: PASS

5 TOE Security Assurance Requirements

5.1 Class ASE: Security Target Evaluation

320 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

321 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

322 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.

323 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.

324 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

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

326 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:	Instructions for configuring the password characteristics are identified in the Guidance Document assurance activities for FIA_PMG_EXT.1. [OS_GUIDE] / 8 contains instructions for configuring the number of allowed authentication attempt failures. [OVIRT] / 19.8.3 – Setting User Timeout, provides instructions on setting the lockout period times for inactivity. Instructions for configuring the notice and consent warning are identified in the Guidance Document assurance activities for FTA_TAB.1.

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

Findings:	Please refer to AGD_PRE.1.2 ETR work unit which satisfies this assurance activity.

328 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:	[AGD] / 2.3 provides instructions for querying the current version of the TOE software. “To verify that Virtualization Manager 4.3 is installed: [root@ovirt: ~]# yum list ovirt-engine”

- 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(2) 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 TOE does not use a hash or certificate-based signature verification. Per [ST] section 6.6.6, the TOE uses an Oracle PGP Public key for signature verification and is the Oracle public key is supplied with the TOE during initial installation.

- 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:	Per [AGD] section 2.2, all updates must be performed using the “yumlog” script. The updates will be retrieved from the Oracle repositories. [AGD] section 2.2.1 describes how to perform local updates by first downloading the update packages with “yumdownloader” and performing the update as described with the “yumlog” script.

- 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:	Per [AGD] section 2.2, all updates must be performed using the “yumlog” script to ensure required logs are maintained.

5.3.2 AGD_PRE.1 Preparative Procedures

329 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 evaluator checked the guidance provided and determined that it adequately addresses all platforms claimed for the TOE in the ST.

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

Findings:	Please refer to AGD_PRE.1.2 ETR work unit which satisfies this assurance activity.

5.4 Class ALC: Life-Cycle Support

331 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

332 **TD0230:** The evaluator shall check the ST to ensure that it contains an identifier(such as a product name/version number) that specifically identifies the version that meets

the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Findings:	Section 2.4 in the [ST] identifies the TOE as "the KVM & Virtualization Manager version 4.3.10.4-1.0.21 running on Oracle Linux 7.6 UEK 5". [AGD] Section 2 describes the steps to install the Oracle Linux 7.6 OS with the UEK 5 kernel, as well as how to KVM and Virtualization Manager 4.3.10.4-1.0.21.el7 which is consistent with the TOE identifier in the [ST]. The evaluator reviewed the versions of the TOE software installed on the system and confirmed that they are consistent with those in the [ST]. The evaluator reviewed the web site advertising the TOE and ensured that the information in the ST is sufficient to distinguish the product. In the website there is an explicit section for the TOE major/minor version identified in the [ST] – Oracle Linux Virtualization Manager (OLVM) v4.3. https://docs.oracle.com/en/virtualization/oracle-linux-virtualization-manager/

5.4.2 ALC_CMS.1 TOE CM Coverage

333 **TD0230:** 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(s) 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.

Findings:	By default Address Space Layout Randomization (ASLR) is enabled to randomize the positions of the stack, the virtual dynamic shared object (VDSO) page, shared memory regions, and the data segment. ASLR can held defeat certain types of buffer overflow attacks. The description of ASLR and configuration parameters are found in Section 4.17.1 at the following public-facing guidance: https://docs.oracle.com/en/operating-systems/oracle-linux/7/security/ol7-implement-sec.html#ol7-s5-syssec Oracle Linux Admin tips for hardening Oracle Linux describes the compiler setting to detect potential buffer overflows. https://www.oracle.com/technical-resources/articles/it-infrastructure/admin-tips-harden-oracle-linux.html "The gcc compiler features several buffer overflow protection features. Setting the FORTIFY_SOURCE option causes the compiler to issue a warning when it detects a defect such as a potential buffer overflow." Oracle Linux v7.6 Common Criteria Guidance also provides application developers with best practices for gcc compiler and linker options. https://www.oracle.com/a/ocom/docs/oracle-linux-v7.6-common-criteria-guidance-v1.7.pdf "Application developers should use the following compiler options as best practice

	when developing applications invoking the gcc compiler and linker. The stack-protector-strong flag has been developed to broaden the scope of the stack protection without extending it to every function in the program. -fstack-protector-strong --param=ssp-buffer-size=4 ASLR improves executable security in terms of memory randomization and access protection. -fpie -Wl,-pie”

334 **TD0230:** 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 evaluator reviewed the web site advertising the TOE and ensured that the information in the ST is sufficient to distinguish the product. In the website there is an explicit section for the TOE major/minor version identified in the [ST] – Oracle Linux Virtualization Manager (OLVM) v4.3. https://docs.oracle.com/en/virtualization/oracle-linux-virtualization-manager/

5.4.3 ALC_TSU_EXT.1 Timely Security Updates

335 This component requires the TOE developer, in conjunction with any other necessary parties, to provide information as to how the Virtualization System 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.

336 The evaluator shall confirm that the information *(below)* provided meets all requirements for content and presentation of evidence.

337 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”.

338 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”.

339 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 is emailing the “secalert_us@oracle.com”. The PGP key is published with the email address.

5.5 Class ATE: Tests

340 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 – Sample

341 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 Assurance 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.

342 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 Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

343 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.

344 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).

345 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

346 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.

347 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.

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

348 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. As part of this document, the evaluator documented their sources and reported any vulnerabilities found or residual vulnerabilities remaining in the product. Appropriate attack potential calculations were used where required.