IBM AIX 7.2 Technology Level 5 Service Pack 3 Standard Edition Operating System Security Target

Version:	**1.3**
Status:	**Released**
Last Update:	**2022-06-21**

Trademarks

Legal Notice

Revision History

Revision	Date	Author(s)	Changes to Previous Revision
1.3	2022-06-21	Scott Chapman	Final.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Security Target Identification

Title:	IBM AIX 7.2 Technology Level 5 Service Pack 3 Standard Edition Operating System Security Target
Version:	1.3
Status:	Released
Date:	2022-06-21
Sponsor:	IBM Corporation
Developer:	IBM Corporation
Certification Body:	BSI
Certification ID:	BSI-DSZ-CC-1165
Keywords:	AIX, OS, operating system

1.2 TOE Identification

The TOE is the IBM AIX Version 7.2 Technology Level 5 (TL5) Service Pack 3 (SP3) Standard Edition (SE) operating system.

1.3 TOE Type

The TOE type is a general purpose operating system.

1.4 TOE Overview

The TOE is the AIX 7.2 TL5 SP3 (a.k.a. 07.02.05.03) SE operating system. The TOE was evaluated running in an IBM PowerVM Virtual I/O System (VIOS) version 3.1.1 virtual environment on an IBM Power System E950 rack mounted server with an IBM POWER9 SMT8 core processor and system firmware FW950.

The TOE conforms to the [OSPPv4.2.1] requirements. In addition, the TOE also conforms to the [SSHEPv1.0] requirements included in this ST.

1.4.1 Required non-TOE hardware and software

The TOE requires the following hardware and software components to operate in the evaluated configuration. The following hardware and other software components are part of the operational environment (OE).

- IBM Power System E950 rack mounted server with an IBM POWER9 SMT8 core processor and system firmware FW950
- IBM PowerVM Virtual I/O System (VIOS) version 3.1.1
- Remote administration:
    - Secure Shell (SSH) version 2 (v2) client
    - Remote system (to host the SSH v2 client)

1.4.2 Intended method of use

The TOE is intended to be used inside of an organization in a non-hostile environment, located in a protected environment (e.g., server room) where only trusted administrators have access to the physical computer.

1.4.3 Major security features

The TOE supports the following major security features.

- Auditing (FAU)
- Cryptographic support (FCS)
- User data protection (FDP)
- Identification and authentication (FIA)
- Security management (FMT)
- Protection of the TOE Security Functionality (TSF) (FPT)
- TOE access (FTA)
- Trusted path/channels (FTP)

1.5 TOE Description

1.5.1 Introduction

The AIX operating system is a general purpose, multi-user, multi-tasking operating system. It is compliant with major international standards for UNIX systems, such as the Portable Operating System Interface (POSIX) standards, X/Open Portability Guide (XPG) 4, and Spec 1170. It provides a platform for a variety of applications in the governmental and commercial environment. AIX is available on a broad range of computer systems from IBM, ranging from departmental servers to multi-processor enterprise servers, and is capable of running in a Logical Partition (LPAR) environment. Note that the TOE was only evaluated using the non-TOE hardware and software defined in section 1.4.1.

For user interaction, the TOE supports the following user interfaces.

- Local console—for local access
- SSH—for remote access

The TOE contains an SSH server allowing users and administrators to connect remotely and an SSH client allowing users and administrators to SSH to another system. SSH is implemented using OpenSSH. OpenSSH uses OpenSSL as its cryptographic module. The TOE also includes a Transport Layer Security (TLS) client for protected communications to an endpoint. TLS is implemented using OpenSSL.

1.5.2 Security functionality

This section introduces the TOE security functionality used to comply with [OSPPv4.2.1]☝ and [SSHEPv1.0]☝. It assumes the reader has an understanding of the concepts required by these two documents.

1.5.2.1 Auditing

The TOE contains an audit mechanism that generates local audit records and stores them in local audit files.

1.5.2.2 Cryptographic support

This section introduces several cryptographic algorithm acronyms.

Acronym	Definition
AES	Advanced Encryption System
CBC	Cipher Block Chaining
CTR	Counter
DRBG	Deterministic Random Bit Generator
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
HMAC	Hash-based (a.k.a. keyed-hash) Message Authentication Code
MAC	Message Authentication Code
RSA	Rivest–Shamir–Adleman
SHA	Secure Hash Algorithm

The TOE includes an SSH client and server using OpenSSH 8.1p1. SSH supports both password-based and public key-based authentication. It also uses keystores to store protected keys. OpenSSH uses OpenSSL for its cryptographic module. The OpenSSL cryptographic module resides in user space. Both OpenSSH and OpenSSL are 32-bit implementations.

The TOE also includes TLS client support using the 32-bit OpenSSL that resides in user space. In the evaluated configuration, the TOE's OpenSSL uses what is commonly known as the FIPS object module. This module has **not** been [FIPS140-2] nor [FIPS140-3] validated on the TOE.

The TOE supports two trusted update mechanisms: cumulative updates and interim fixes (ifixes). (See section 1.5.2.6 for more details.) For cumulative updates, the TOE's **suma** command checks for available cumulative updates and downloads cumulative updates. The **suma** command uses the IBM Java 8.6.30 TLS 1.2 client implementation and the IBMJCEPlusFIPS cryptographic provider for implementing its HTTPS connection. IBMJCEPlusFIPS uses the 64-bit IBM Crypto for C (ICC) cryptographic module. The ICC cryptographic module resides in user space. The TOE's **installp** command performs the installation of the cumulative updates. It uses the 32-bit OpenSSL cryptographic module to verify the signatures on the cumulative updates.

For ifixes, the TOE checks for available ifixes and downloads ifixes over an HTTPS connection. Once downloaded, the TOE verifies each ifix's signature and installs the verified ifixes. This process uses the **openssl s_client** command and 32-bit OpenSSL cryptographic module in user space to implement the TLS 1.2 client protocol for HTTPS and to verify the ifix package signatures.

The TOE includes an Encrypted File System (EFS) feature for encrypting sensitive data stored on non-volatile storage. EFS allows individual files to be selectively encrypted using AES-CBC. It requires users and groups to have individual RSA key pairs. EFS auto-generates a unique AES key for each file and protects the AES key by encrypting it with the user and group public keys that have access to the file and storing the encrypted result in the meta data of the file system. It uses keystores to store protected keys. EFS uses two IBM CryptoLite for C (CLiC) cryptographic modules: one in user space and one in kernel space. The CLiC modules contain both 32-bit and 64-bit implementations of the algorithms. Kernel space uses the 64-bit implementations and user space uses the 32-bit implementations.

In addition, the TOE's boot integrity code uses the CLiC cryptographic module in kernel space.

The above modules support the following TOE security features used to meet the requirements defined in [OSPPv4.2.1]☝ and [SSHEPv1.0]☝.

- CLiC kernel space version 4.10.1020
  - Boot integrity
  - EFS
- CLiC user space version 4.10.1020
  - EFS
- ICC version 8.6.0.0 (user space)
  - Trusted update (for HTTPS/TLS of cumulative update info)
- OpenSSL version 1.0.2u (user space) (a.k.a. openssl-fips-20.16.102.2103)
  - SSH
  - TLS
  - Trusted update (for HTTPS/TLS of ifix update info and signature verification of all updates)

The evaluated configuration uses the following algorithms from the CLiC kernel space cryptographic module.

- AES-CBC
- Hash_DRBG
- RSA key establishment, signature verification
- SHA message digests

The evaluated configuration uses the following algorithms from the CLiC user space cryptographic module.

- AES-CBC
- Hash_DRBG
- RSA key generation, key establishment
- SHA message digests

The evaluated configuration uses the following algorithms from the ICC cryptographic module.

- AES-GCM
- Hash_DRBG
- HMAC message authentication codes
- RSA key establishment, signature generation/verification

- SHA message digests

The evaluated configuration uses the following algorithms from the OpenSSL cryptographic module.

- AES-CBC, AES-CTR, AES-GCM
- CTR_DRBG(AES)
- ECDSA key generation/verification, key establishment, signature generation/verification
- HMAC message authentication codes
- RSA key generation, key establishment, signature generation/verification
- SHA message digests

The TOE also performs key destruction of private keys and key material in volatile and non-volatile memory under certain conditions.

The CLiC and OpenSSL cryptographic modules use a platform-based ring-oscillator noise source contained in the POWER9 SMT8 core processor for obtaining entropy data. The ICC cryptographic module uses a software-based noise source built into the module for obtaining entropy data.

1.5.2.3 User data protection

The TOE's Journaled File System version 2 (JFS2) supports the standard UNIX permission bit access control mechanism to protect both TSF and user data in named file system objects such as files and directories.

1.5.2.4 Identification and authentication

The TOE supports authentication failure handling. The authentication failure handling includes an administrator-configurable range of unsuccessful authentication attempts for username/password-based accounts that, when reached, locks the user's account.

The TOE supports multiple authentication mechanisms, specifically username/password-based and key-based authentication. The local console supports username/password-based authentication. The SSH server supports remote administration using username/password-based and key-based authentication. The SSH client also supports both username/password-based and key-based authentication.

The TOE's TLS client includes support for certificate-based authentication of a TLS connection. This includes X.509v3 certificate validation of TLS certificates. The validation includes both Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP) revocation status checking. The OpenSSL TLS uses CRLs and the IBM Java TLS (used by the **suma** command) uses OCSP.

1.5.2.5 Security management

The TOE supports management of its security features by administrators. These management features include the following.

- Session timeout
- Auditing:
  - Audit storage capacity configuration
  - Audit event/rule configuration
- Password composition configuration
- Firewall configuration

Users can modify their session timeout values.

1.5.2.6 Protection of the TSF

The TOE uses the access control mechanism described in section 1.5.2.3 to prohibit unprivileged users from modifying TSF functions such as the kernel, device drivers, security audit logs, etc. It also prohibits unprivileged users from reading security-relevant data such as security audit logs and system-wide credential repositories.

The TOE performs address space layout randomization (ASLR) for all applications unless the application is marked as an exception to ASLR. An administrator can selectively except applications from ASLR.

The TOE implements a Stack Execution Disable (SED) protection mechanism to aid in stopping stack buffer overflow attacks. This mechanism prevents processes from executing code residing on the process' stack.

The TOE uses digital signatures to validate its bootchain as the TOE boots (a.k.a. boot integrity). The validation includes the bootloader, kernel, device drivers, and kernel extensions.

The TOE performs digital signature validation of both OS updates and application updates (a.k.a. trusted update). This mechanism allows an administrator to validate the signatures of the updates prior to installing them.

The TOE supports two trusted update mechanisms.

- Cumulative updates
- Interim fixes (ifixes)

Cumulative updates, such as technology level updates and service packs, are provided at scheduled times throughout each year. Ifixes are provided in between cumulative updates making critical fixes available as soon as possible. Both update mechanisms can include security updates.

The TOE prevents allocation of memory regions with both write and execute permissions in the code, data, heap, and stack segments.

1.5.2.7 TOE access

The TOE displays an administrator-configurable advisory warning message before a user session is established.

1.5.2.8 Trusted path/channel

The TOE supports a TLS client protocol implementation via OpenSSL that allows applications to connect to TLS servers.

The **suma** command uses the IBM Java TLS client implementation to connect to the IBM fix server over HTTPS. The ifix commands use the OpenSSL TLS client implementation to connect to the IBM fix server over HTTPS.

The TOE supports a general purpose OpenSSH client allowing users to connect to other remote SSH servers via a secure connection. In this case, the TOE acts as an SSH client. It also supports inbound remote administration connections over SSH using OpenSSH. In this case, the TOE acts as an SSH server.

1.5.3 TOE boundaries

1.5.3.1 Physical

The TOE consists of the AIX operating system and guidance documents. The IBM ordering number is the following.

- IBM AIX Standard Edition, Program Number 5765-G98, VRM[1] 07.02.05.03

The TOE is packaged in Licensed Product Packages (LPPs)/File Sets. The evaluated configuration contains the following LPP names.

Table 1: List of LPPs/File Sets

LPP name	Description
bos	AIX base operating system
devices	AIX supported devices
sysmgt	System management tools

In addition, the TOE requires the following AIX packages to be installed. These packages are included as part of the AIX installation image, but must be independently installed and configured.

- OpenSSH Client Software
- OpenSSH Server Software
- openssl-fips-20.16.102.2103

The above operating system, OpenSSH, and OpenSSL packages are combined into a single ISO image downloadable from IBM.

In addition, the TOE requires the following AIX ifixes to be downloaded and installed.

Ifix name	Location/SHA2-256
CCEMGR_fix	https://aix.software.ibm.com/aix/efixes/cc/
	21917e86ea568d2b28dc87c4cf2b5e6727567cd51d6249b12dfa66faa415947f
CCECCSUMA1_fix	https://aix.software.ibm.com/aix/efixes/cc/
	7ab9a64cd98d0b8160e6fc0b67b0cd72c9779315b9ad293659e9a98dfa5c33e8
InstTU_fix	https://aix.software.ibm.com/aix/efixes/cc/
	a2606ac587237cdc60ab30ebc6793e94c607f69f8bdf0d4910b4d9749d79d414
TD0325_fix	https://aix.software.ibm.com/aix/efixes/cc/
	cbbdff7aabec93b022872a4e57c9cf683ac806b1e0bcbf0dbbc1fa4fd9f32a5b
efs_fix	https://aix.software.ibm.com/aix/efixes/security/
	206fa67aae93f1a1df78b287e50a3d972092998a80b5320b1dfe3e4c00651cf3

[1] VRM—Version/Release/Modification

Ifix name	Location/SHA2-256
lscore_fix	https://aix.software.ibm.com/aix/efixes/security/
	c4f97465829e8a25cccf3260057383626635b5922f181a9450e6a5fbbf3558b9
mount_fix	https://aix.software.ibm.com/aix/efixes/security/
	438f2e0d1bddedbb4983e962538664b8f0c14899233b859c9dd2e66228484835
openssh_fix14	https://aix.software.ibm.com/aix/efixes/security/
	23a32151be35c6322d80d4a3633effff92ed3bab8b45cc21f3494c5d92cf7678
audit_fix	https://aix.software.ibm.com/aix/efixes/security/
	0c2ef6fcc0f743e0a1a0ab71a9451f1ca592c663eb434c6af5219bea98cd1aeb
kernel_fix3	https://aix.software.ibm.com/aix/efixes/security/
	b217d346b5a6312ec15911ca9cbf64a5da098edbf8b47644273d71736c12de18
java_feb2022_fix	https://aix.software.ibm.com/aix/efixes/security/
	68f682d919024ab8eb1db5ab4fdcd1037709605424a55df5b7980baa9ff003e7

The TOE includes the following guidance document.

- IBM AIX 7.2 Technology Level 5 Service Pack 3 Common Criteria Administration Guidance Version 1.3

The above guidance document is available from the IBM Support site at the following link.

https://supportcontent.ibm.com/support/pages/common-criteria-administration-guidance-aix-7253

The TOE includes the following additional guidance documents.

- AIX Version 7.2 Base Operating System (BOS) Runtime Services
- AIX Version 7.2 Commands
- AIX Version 7.2 Files Reference
- AIX Version 7.2 Security

These additional guidance documents are available as a tar bundle from the IBM AIX download site at the following link.

https://aix.software.ibm.com/aix/efixes/cc/

1.5.3.2 Logical

The TOE is the AIX operating system executing inside an LPAR. The LPAR is in the operational environment. The TOE includes the security features described in section 1.5.2.

1.5.3.3 Evaluated configuration

The following configuration specifics apply to the evaluated configuration of the TOE.

- The TOE must be in "root enabled mode."
- The TOE must have SED protection enabled system-wide (i.e., **sedmgr all**).

- The TOE must have Secure Boot policy *2* enabled.
- The TOE must have ASLR enabled for all applications (aslr=2).
- The TOE must have the **installp** command's *digital signature policy* set to *medium* after the initial installation of the TOE.
- The OPENSSL_FIPS environment variable must be set to 1.
- EFS file encryption algorithm must be configured to use either AES_128_CBC or AES_256_CBC for file encryption.[2]
- EFS keystore key pair algorithm for both users and groups must be configured to use either RSA_2048 or RSA_4096 for keystore key pair generation.[3]
- OpenSSH must be configured to support the SSH requirements in this document.
- All AIX ifixes listed in section 1.5.3.1 must be installed.

[2] The PP does not support AES-CBC-192; therefore, the AES_192_CBC EFS setting must not be used in the evaluated configuration.

[3] The PP does not support RSA key pairs smaller than 2048-bit; therefore, the RSA_1024 (RSA 1024-bit) EFS setting must not be used in the evaluated configuration.

2 CC Conformance Claim

This Security Target is CC Part 2 extended and CC Part 3 extended.

This Security Target claims conformance to the following Protection Profiles and PP packages:

- [OSPPv4.2.1]: Protection Profile for General Purpose Operating Systems. Version 4.2.1 as of 2019-04-22; exact conformance.
- [SSHEPv1.0]: Extended Package for Secure Shell (SSH). Version 1.0 as of 2016-02-19; exact conformance.

Common Criteria [CC] version 3.1 revision 5 is the basis for this conformance claim.

2.1 Protection Profile Tailoring and Additions

2.1.1 Protection Profile for General Purpose Operating Systems ([OSPPv4.2.1])

Table 2 contains the NIAP Technical Decisions (TDs) for this protection profile at the time of the evaluation and a statement of applicability to the evaluation.

Table 2: NIAP TDs for OSPPv4.2.1

NIAP TD	TD description	Applic-able?	Non-applicability rationale	TD reference
TD0600	Conformance claim sections updated to allow for MOD_VPNC_V2.3	No	PP-Module for VPN Client, Version 2.3 is not included in this ST.	[CCEVS-TD0600]
TD0578	SHA-1 is no longer mandatory	Yes		[CCEVS-TD0578]
TD0525	Updates to Certificate Revocation (FIA_X509_EXT.1)	Yes		[CCEVS-TD0525]
TD0501	Cryptographic selections and updates for OS PP	Yes		[CCEVS-TD0501]
TD0496	GPOS PP adds allow-with statement for VPN Client V2.1	No	PP-Module for VPN Client, Version 2.1 is not included in this ST.	[CCEVS-TD0496]
TD0493	X.509v3 certificates when using digital signatures for Boot Integrity	Yes		[CCEVS-TD0493]
TD0463	Clarification for FPT_TUD_EXT	Yes		[CCEVS-TD0463]
TD0441	Updated TLS Ciphersuites for OS PP	Yes		[CCEVS-TD0441]
TD0386	Platform-Provided Verification of Update	Yes		[CCEVS-TD0386]
TD0365	FCS_CKM_EXT.4 selections	Yes		[CCEVS-TD0365]

2.1.2 Extended Package for Secure Shell (SSH) ([SSHEPv1.0])

Table 3 contains the NIAP Technical Decisions (TDs) for this extended package at the time of the evaluation and a statement of applicability to the evaluation.

Table 3: NIAP TDs for SSHEPv1.0

NIAP TD	TD description	Applic-able?	Non-applicability rationale	TD reference
TD0598	Expanded AES Modes in FCS_COP for App PP	No	Not using App PP	[CCEVS-TD0598]
TD0446	Missing selections for SSH	Yes		[CCEVS-TD0446]
TD0420	Conflict in FCS_SSHC_EXT.1.1 and FCS_SSHS_EXT.1.1	Yes		[CCEVS-TD0420]
TD0332	Support for RSA SHA2 host keys	Yes		[CCEVS-TD0332]
TD0331	SSH Rekey Testing	Yes		[CCEVS-TD0331]
TD0240	FCS_COP.1.1(1) Platform provided crypto for encryption/decryption	Yes		[CCEVS-TD0240]

3 Security Problem Definition

3.1 Threat Environment

3.1.1 Threats countered by the TOE

T.NETWORK_ATTACK

An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may engage in communications with applications and services running on or part of the OS with the intent of compromise. Engagement may consist of altering existing legitimate communications.

T.NETWORK_EAVESDROP

An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between applications and services that are running on or part of the OS.

T.LOCAL_ATTACK

An attacker may compromise applications running on the OS. The compromised application may provide maliciously formatted input to the OS through a variety of channels including unprivileged system calls and messaging via the file system.

T.LIMITED_PHYSICAL_ACCESS

An attacker may attempt to access data on the OS while having a limited amount of time with the physical device.

3.2 Assumptions

3.2.1 Intended usage of the TOE

A.PLATFORM

The OS relies upon a trustworthy computing platform for its execution. This underlying platform is out of scope of this PP.

A.PROPER_USER

The user of the OS is not willfully negligent or hostile, and uses the software in compliance with the applied enterprise security policy. At the same time, malicious software could act *as* the user, so requirements which confine malicious subjects are still in scope.

A.PROPER_ADMIN

The administrator of the OS is not careless, willfully negligent or hostile, and administers the OS within compliance of the applied enterprise security policy.

4 Security Objectives

4.1 Objectives for the TOE

O.ACCOUNTABILITY

Conformant OSes ensure that information exists that allows administrators to discover unintentional issues with the configuration and operation of the operating system and discover its cause. Gathering event information and immediately transmitting it to another system can also enable incident response in the event of system compromise.

O.INTEGRITY

Conformant OSes ensure the integrity of their update packages. OSes are seldom if ever shipped without errors, and the ability to deploy patches and updates with integrity is critical to enterprise network security. Conformant OSes provide execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems.

O.MANAGEMENT

To facilitate management by users and the enterprise, conformant OSes provide consistent and supported interfaces for their security-relevant configuration and maintenance. This includes the deployment of applications and application updates through the use of platform-supported deployment mechanisms and formats, as well as providing mechanisms for configuration and application execution control.

O.PROTECTED_STORAGE

To address the issue of loss of confidentiality of credentials in the event of loss of physical control of the storage medium, conformant OSes provide data-at-rest protection for credentials. Conformant OSes also provide access controls which allow users to keep their files private from other users of the same system.

O.PROTECTED_COMMS

To address both passive (eavesdropping) and active (packet modification) network attack threats, conformant OSes provide mechanisms to create trusted channels for CSP and sensitive data. Both CSP and sensitive data should not be exposed outside of the platform.

4.2 Objectives for the Operational Environment

OE.PLATFORM

The OS relies on being installed on trusted hardware.

OE.PROPER_USER

The user of the OS is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy. Standard user accounts are provisioned in accordance with the least privilege model. Users requiring higher levels of access should have a separate account dedicated for that use.

OE.PROPER_ADMIN

The administrator of the OS is not careless, willfully negligent or hostile, and administers the OS within compliance of the applied enterprise security policy.

4.3 Security Objectives Rationale

4.3.1 Coverage

The following table provides a mapping of security objectives to threats, policies, and assumptions showing that each objective maps to at least one threat, policy, or assumption and vice versa.

Table 4: Mapping of security objectives to threats, policies, and assumptions

	O.ACCOUNTABILITY	O.INTEGRITY	O.MANAGEMENT	O.PROTECTED_STORAGE	O.PROTECTED_COMMS	OE.PLATFORM	OE.PROPER_USER	OE.PROPER_ADMIN
T.NETWORK_ATTACK	✓	✓	✓		✓			
T.NETWORK_EAVESDROP			✓		✓			
T.LOCAL_ATTACK	✓	✓						
T.LIMITED_PHYSICAL_ACCESS				✓				
A.PLATFORM						✓		
A.PROPER_USER							✓	
A.PROPER_ADMIN								✓

4.3.2 Sufficiency

The following rationale provides justification that the security objectives are suitable to counter each individual threat and that each security objective tracing back to a threat, when achieved, actually contributes to the removal, diminishing or mitigation of that threat.

Table 5: Sufficiency of objectives countering threats

Threat	Rationale for security objectives
T.NETWORK_ATTACK	The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides for integrity of transmitted data. The threat T.NETWORK_ATTACK is countered by O.INTEGRITY as this provides for integrity of software that is installed onto the system from the network. The threat T.NETWORK_ATTACK is countered by O.MANAGEMENT as this provides for the ability to configure the OS to defend against network attack.

Threat	Rationale for security objectives
	The threat T.NETWORK_ATTACK is countered by O.ACCOUNTABILITY as this provides a mechanism for the OS to report behavior that may indicate a network attack has occurred.
T.NETWORK_EAVESDROP	The threat T.NETWORK_EAVESDROP is countered by O.PROTECTED_COMMS as this provides for confidentiality of transmitted data. The threat T.NETWORK_EAVESDROP is countered by O.MANAGEMENT as this provides for the ability to configure the OS to protect the confidentiality of its transmitted data.
T.LOCAL_ATTACK	The objective O.INTEGRITY protects against the use of mechanisms that weaken the TOE with regard to attack by other software on the platform. The objective O.ACCOUNTABILITY protects against local attacks by providing a mechanism to report behavior that may indicate a local attack is occurring or has occurred.
T.LIMITED_PHYSICAL_ACCESS	The objective O.PROTECTED_STORAGE protects against unauthorized attempts to access physical storage used by the TOE.

The following rationale provides justification that the security objectives for the environment are suitable to cover each individual assumption, that each security objective for the environment that traces back to an assumption about the environment of use of the TOE, when achieved, actually contributes to the environment achieving consistency with the assumption, and that if all security objectives for the environment that trace back to an assumption are achieved, the intended usage is supported.

Table 6: Sufficiency of objectives holding assumptions

Assumption	Rationale for security objectives
A.PLATFORM	The operational environment objective OE.PLATFORM is realized through A.PLATFORM.
A.PROPER_USER	The operational environment objective OE.PROPER_USER is realized through A.PROPER_USER.
A.PROPER_ADMIN	The operational environment objective OE.PROPER_ADMIN is realized through A.PROPER_ADMIN.

5 Extended Components Definition

Extended components definitions (ECDs) are provided by the PP and EP to which this ST conforms.

6 Security Requirements

6.1 TOE Security Functional Requirements

The table below summarizes the SFRs for the TOE and the operations performed on the components according to CC part 1. Operations in the SFRs use the following convention:

- Iterations (Iter.) are identified by appending a suffix to the original SFR.
- Refinements (Ref.) added to the text are shown in *italic text*, deletions are shown as ~~strikethrough text~~.
- Assignments (Ass.) are shown in **bold text**.
- Selections (Sel.) are shown in **bold text**.

Table 7: SFRs for the TOE

Security functional class	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
FAU - Security audit	FAU_GEN.1 Audit Data Generation (Refined)		OSPPv4.2.1	No	No	Yes	Yes
FCS - Cryptographic support	FCS_CKM.1 Cryptographic Key Generation (Refined)		OSPPv4.2.1	No	No	No	Yes
	FCS_CKM.2 Cryptographic Key Establishment (Refined)		OSPPv4.2.1	No	No	No	Yes
	FCS_CKM_EXT.4 Cryptographic Key Destruction		OSPPv4.2.1	No	No	No	Yes
	FCS_COP.1(1) Cryptographic Operation - Encryption/Decryption (Refined)	FCS_COP.1	OSPPv4.2.1	Yes	No	No	Yes
	FCS_COP.1(2) Cryptographic Operation - Hashing (Refined)	FCS_COP.1	OSPPv4.2.1	Yes	No	No	Yes
	FCS_COP.1(3) Cryptographic Operation - Signing (Refined)	FCS_COP.1	OSPPv4.2.1	Yes	No	No	Yes
	FCS_COP.1(4) Cryptographic Operation - Keyed-Hash Message Authentication (Refined)	FCS_COP.1	OSPPv4.2.1	Yes	No	Yes	Yes
	FCS_COP.1(5) Cryptographic Operation - Encryption/Decryption (Refined)	FCS_COP.1	SSHEPv1.0	Yes	No	No	Yes
	FCS_RBG_EXT.1 Random Bit Generation		OSPPv4.2.1	No	No	No	Yes
	FCS_SSH_EXT.1 SSH Protocol		SSHEPv1.0	No	No	No	Yes

Security functional class	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	FCS_SSHC_EXT.1 SSH Protocol - Client		SSHEPv1.0	No	No	Yes	Yes
	FCS_SSHS_EXT.1 SSH Protocol - Server		SSHEPv1.0	No	No	Yes	Yes
	FCS_STO_EXT.1 Storage of Sensitive Data		OSPPv4.2.1	No	No	No	No
	FCS_TLSC_EXT.1 TLS Client Protocol		OSPPv4.2.1	No	Yes	No	Yes
	FCS_TLSC_EXT.2 TLS Client Protocol		OSPPv4.2.1	No	No	No	Yes
FDP - User data protection	FDP_ACF_EXT.1 Access Controls for Protecting User Data		OSPPv4.2.1	No	No	No	No
FIA - Identification and authentication	FIA_AFL.1 Authentication failure handling (Refined)		OSPPv4.2.1	No	No	Yes	Yes
	FIA_UAU.5 Multiple Authentication Mechanisms (Refined)		OSPPv4.2.1	No	No	Yes	Yes
	FIA_X509_EXT.1 X.509 Certificate Validation		OSPPv4.2.1	No	No	No	Yes
	FIA_X509_EXT.2 X.509 Certificate Authentication		OSPPv4.2.1	No	No	No	Yes
FMT - Security management	FMT_MOF_EXT.1 Management of security functions behavior		OSPPv4.2.1	No	No	No	No
	FMT_SMF_EXT.1 Specification of Management Functions		OSPPv4.2.1	No	No	Yes	Yes
FPT - Protection of the TSF	FPT_ACF_EXT.1 Access controls		OSPPv4.2.1	No	No	Yes	No
	FPT_ASLR_EXT.1 Address Space Layout Randomization		OSPPv4.2.1	No	No	Yes	Yes
	FPT_SBOP_EXT.1 Stack Buffer Overflow Protection		OSPPv4.2.1	No	No	No	Yes
	FPT_TST_EXT.1 Boot Integrity		OSPPv4.2.1	No	No	Yes	Yes
	FPT_TUD_EXT.1 Trusted Update		OSPPv4.2.1	No	No	No	Yes
	FPT_TUD_EXT.2 Trusted Update for Application Software		OSPPv4.2.1	No	No	No	No
	FPT_W^X_EXT.1 Write XOR Execute Memory Pages		OSPPv4.2.1	No	No	Yes	No

Security functional class	Security functional requirement	Base security functional component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
FTA - TOE access	FTA_TAB.1 Default TOE access banners		OSPPv4.2.1	No	No	No	No
FTP - Trusted path/channels	FTP_ITC_EXT.1 Trusted channel communication		OSPPv4.2.1	No	No	Yes	Yes
	FTP_TRP.1 Trusted Path		OSPPv4.2.1	No	No	No	Yes

6.1.1 Security audit (FAU)

6.1.1.1 Audit Data Generation (Refined) (FAU_GEN.1)

FAU_GEN.1.1 The OS shall be able to generate an audit record of the following auditable events:

a. Start-up and shut-down of the audit functions;

b. All auditable events for the not specified level of audit; and

c.
- Authentication events (Success/Failure);
- Use of privileged/special rights events (Successful and unsuccessful security, audit, and configuration changes);
- Privilege or role escalation events (Success/Failure);
-
  - **File and object events (Successful and unsuccessful attempts to create, access, delete, modify, modify permissions)**
  - **User and Group management events (Successful and unsuccessful add, delete, modify, disable, enable, and credential change)**
  - **Audit and log data access events (Success/Failure)**
  - **System reboot, restart, and shutdown events (Success/Failure)**
  - **Administrator or root-level access events (Success/Failure)**

FAU_GEN.1.2 The OS shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, **no other audit relevant information**.

TSS Link: *TSS for FAU_GEN.1*

6.1.2 Cryptographic support (FCS)

6.1.2.1 Cryptographic Key Generation (Refined) (FCS_CKM.1)

FCS_CKM.1.1 The OS shall generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm[4]

- **RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3,**
- **ECC schemes using "NIST curves" P-256, P-384 and P-521 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4**

TSS Link: *TSS for FCS_CKM.1*

6.1.2.2 Cryptographic Key Establishment (Refined) (FCS_CKM.2)

FCS_CKM.2.1 The OS shall implement functionality to perform cryptographic key establishment in accordance with a specified cryptographic key establishment method:[5]

- **RSA-based key establishment schemes that meets the following: RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 8017, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2",**
- **Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"**

TSS Link: *TSS for FCS_CKM.2*

6.1.2.3 Cryptographic Key Destruction (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1 The OS shall destroy cryptographic keys and key material in accordance with a specified cryptographic key destruction method[6]

- **For volatile memory, the destruction shall be executed by a**
  - **single overwrite consisting of zeroes,**
  - **removal of power to the memory**
- **For non-volatile memory that consists of**
  - **the invocation of an interface provided by the underlying platform that**
    - **logically addresses the storage location of the key and performs a single overwrite consisting of zeroes,**

FCS_CKM_EXT.4.2 The OS shall destroy all keys and key material when no longer needed.

TSS Link: *TSS for FCS_CKM_EXT.4*

[4] [CCEVS-TD0501]☝ was applied to FCS_CKM.1.
[5] [CCEVS-TD0501]☝ was applied to FCS_CKM.2.
[6] [CCEVS-TD0365]☝ was applied to FCS_CKM_EXT.4.

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

FCS_COP.1.1(1) The OS shall perform encryption/decryption services for data in accordance with a specified cryptographic algorithm

- **AES-CBC (as defined in NIST SP 800-38A)**

and

- **AES-GCM (as defined in NIST SP 800-38D)**

and cryptographic key sizes **128-bit, 256-bit**.

TSS Link: *TSS for FCS_COP.1(1)*

6.1.2.5 Cryptographic Operation - Hashing (Refined) (FCS_COP.1(2))

FCS_COP.1.1(2) The OS shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm **SHA-1, SHA-256, SHA-384, SHA-512** and message digest sizes **160 bits, 256 bits, 384 bits, 512 bits** that meet the following: FIPS Pub 180-4.[7]

TSS Link: *TSS for FCS_COP.1(2)*

6.1.2.6 Cryptographic Operation - Signing (Refined) (FCS_COP.1(3))

FCS_COP.1.1(3) The OS shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm

- **RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4**
- **ECDSA schemes using "NIST curves" P-256, P-384 and P-521 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.**

TSS Link: *TSS for FCS_COP.1(3)*

6.1.2.7 Cryptographic Operation - Keyed-Hash Message Authentication (Refined) (FCS_COP.1(4))

FCS_COP.1.1(4) The OS shall perform keyed-hash message authentication services in accordance with a specified cryptographic algorithm **SHA-1, SHA-256, SHA-384, SHA-512** with key sizes **256 bits** and message digest sizes **160 bits, 256 bits, 384 bits, 512 bits** that meet the following: FIPS Pub 198-1 The Keyed-Hash Message Authentication Code and FIPS Pub 180-4 Secure Hash Standard.

TSS Link: *TSS for FCS_COP.1(4)*

[7] [CCEVS-TD0578]☝ was applied to FCS_COP.1(2).

6.1.2.8 Cryptographic Operation - Encryption/Decryption (Refined) (FCS_COP.1(5))

FCS_COP.1.1(5) The SSH software shall **perform** encryption/decryption services for data in accordance with a specified cryptographic algorithm AES-CTR (as defined in NIST SP 800-38A) mode and cryptographic key sizes **128-bit, 256-bit**.[8]

TSS Link: *TSS for FCS_COP.1(5)*

6.1.2.9 Random Bit Generation (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1 The OS shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using **Hash_DRBG (any), CTR_DRBG (AES)**.

FCS_RBG_EXT.1.2 The deterministic RBG used by the OS shall be seeded by an entropy source that accumulates entropy from a **software-based noise source, platform-based noise source** with a minimum of **256 bits** of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

Application Note: *For the German Scheme, the CTR_DRBG and Hash_DRBG algorithms from FCS_RBG_EXT.1 are shown translated as* [AIS20]/[AIS31] *compliant FCS_RNG.1 SFRs in this application note for equivalency purposes only. FCS_RNG.1(OpenSSL) is for OpenSSL which uses a platform-based noise source. FCS_RNG.1(CLiC) is for CLiC which uses a platform-based noise source. FCS_RNG.1(ICC) is for ICC which uses a software-based noise source.*

FCS_RNG.1(OpenSSL)
FCS_RNG.1.1(OpenSSL): The TSF shall provide a deterministic random number generator conforming to SP800-90A CTR_DRBG with an AES-256 core using a derivation function without prediction resistance that implements:

a) *DRG.2.1: If initialized with a random seed using a platform-based noise source, the internal state of the RNG shall have a minentropy of 256 bits.*
b) *DRG.2.2: The DRNG provides forward secrecy.*
c) *DRG.2.3: The DRNG provides backward secrecy.*

FCS_RNG.1.2(OpenSSL): The TSF shall provide random numbers that meet:

a) *DRG.2.4: The RNG initialized with a random seed that is equal in size as the generated random number, every time a random number is obtained generates output for which 2**19 strings of bit length 128 are mutually different with probability of greater than 1-2**-10.*
b) *DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.*

FCS_RNG.1(CLiC)
FCS_RNG.1.1(CLiC): The TSF shall provide a deterministic random number generator conforming to SP800-90A Hash_DRBG with a SHA2-256 or SHA2-512 core using a derivation function without prediction resistance that implements:

a) *DRG.2.1: If initialized with a random seed using a platform-based noise source, the internal state of the RNG shall have a minentropy of 256 bits.*
b) *DRG.2.2: The DRNG provides forward secrecy.*
c) *DRG.2.3: The DRNG provides backward secrecy.*

FCS_RNG.1.2(CLiC): The TSF shall provide random numbers that meet:

a) *DRG.2.4: The RNG initialized with a random seed that is equal in size as the generated random number, every time a random number is obtained generates output for which 2**19 strings of bit length 128 are mutually different with probability of greater than 1-2**-10.*
b) *DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.*

*FCS_RNG.1(ICC) (Note: This cryptographic module's RNG shall only be used with the TOE's **suma** command and for no other purposes.)*

[8] [CCEVS-TD0240] was applied to FCS_COP.1(5).

<u>FCS_RNG.1.1(ICC):</u> The TSF shall provide a deterministic random number generator conforming to SP800-90A Hash_DRBG with a SHA2-256 core using a derivation function without prediction resistance that implements:

a) *DRG.2.1: If initialized with a random seed using a software-based noise source, the internal state of the RNG shall have a minentropy of 256 bits.*
b) *DRG.2.2: The DRNG provides forward secrecy.*
c) *DRG.2.3: The DRNG provides backward secrecy.*

<u>FCS_RNG.1.2(ICC):</u> The TSF shall provide random numbers that meet:

a) *DRG.2.4: The RNG initialized with a random seed that is equal in size as the generated random number, every time a random number is obtained generates output for which 2**19 strings of bit length 128 are mutually different with probability of greater than 1-2**-10.*
b) *DRG.2.5: Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.*

TSS Link: *TSS for FCS_RBG_EXT.1*

6.1.2.10 SSH Protocol (FCS_SSH_EXT.1)

FCS_SSH_EXT.1.1 The SSH software shall implement the SSH protocol that complies with RFCs 4251, 4252, 4253, 4254 and **5656, 6668** as a **client, server**.

TSS Link: *TSS for FCS_SSH_EXT.1*

6.1.2.11 SSH Protocol - Client (FCS_SSHC_EXT.1)

FCS_SSHC_EXT.1.1 The SSH client shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, and **password-based**.

FCS_SSHC_EXT.1.2 The SSH client shall ensure that, as described in RFC 4253, packets greater than **262,144 (0x40000)** bytes in an SSH transport connection are dropped.

FCS_SSHC_EXT.1.3 The SSH software shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: aes128-ctr, aes256-ctr, **aes128-cbc, aes256-cbc, aes128-gcm@openssh.com, aes256-gcm@openssh.com**.[9]

FCS_SSHC_EXT.1.4 The SSH client shall ensure that the SSH transport implementation uses **rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256** and **ecdsa-sha2-nistp384** as its public key algorithm(s) and rejects all other public key algorithms.[10]

FCS_SSHC_EXT.1.5 The SSH client shall ensure that the SSH transport implementation uses **hmac-sha1, hmac-sha2-256, hmac-sha2-512** and **implicit** as its MAC algorithm(s) and rejects all other MAC algorithm(s).[11]

FCS_SSHC_EXT.1.6 The SSH client shall ensure that **ecdh-sha2-nistp256** and **ecdh-sha2-nistp384, ecdh-sha2-nistp521** are the only allowed key exchange methods used for the SSH protocol.

FCS_SSHC_EXT.1.7 The SSH server shall ensure that the SSH connection be rekeyed after **no more than 1 Gigabyte of data has been transmitted, no more than 1 hour** using that key.

[9] [CCEVS-TD0446]☝ was applied to FCS_SSHC_EXT.1.
[10] [CCEVS-TD0332]☝ was applied to FCS_SSHC_EXT.1.
[11] [CCEVS-TD0446]☝ was applied to FCS_SSHC_EXT.1.

FCS_SSHC_EXT.1.8 The SSH client shall ensure that the SSH client authenticates the identity of the SSH server using a local database associating each host name with its corresponding public key or **no other methods** as described in RFC 4251 section 4.1.

TSS Link: *TSS for FCS_SSHC_EXT.1*

6.1.2.12 SSH Protocol - Server (FCS_SSHS_EXT.1)

FCS_SSHS_EXT.1.1 The SSH server shall ensure that the SSH protocol implementation supports the following authentication methods as described in RFC 4252: public key-based, and **password-based**.

FCS_SSHS_EXT.1.2 The SSH server shall ensure that, as described in RFC 4253, packets greater than **262,144 (0x40000)** bytes in an SSH transport connection are dropped.

FCS_SSHS_EXT.1.3 The SSH server shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: aes128-ctr, aes256-ctr, **aes128-cbc, aes256-cbc, aes128-gcm@openssh.com, aes256-gcm@openssh.com**.[12]

FCS_SSHS_EXT.1.4 The SSH server shall ensure that the SSH transport implementation uses **rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256** and **ecdsa-sha2-nistp384** as its public key algorithm(s) and rejects all other public key algorithms.[13]

FCS_SSHS_EXT.1.5 The SSH server shall ensure that the SSH transport implementation uses **hmac-sha1, hmac-sha2-256, hmac-sha2-512** and **implicit** as its MAC algorithm(s) and rejects all other MAC algorithm(s).[14]

FCS_SSHS_EXT.1.6 The SSH server shall ensure that **ecdh-sha2-nistp256** and **ecdh-sha2-nistp384, ecdh-sha2-nistp521** are the only allowed key exchange methods used for the SSH protocol.

FCS_SSHS_EXT.1.7 The SSH server shall ensure that the SSH connection be rekeyed after **no more than 1 Gigabyte of data has been transmitted, no more than 1 hour** using that key.

TSS Link: *TSS for FCS_SSHS_EXT.1*

6.1.2.13 Storage of Sensitive Data (FCS_STO_EXT.1)

FCS_STO_EXT.1.1 The OS shall implement functionality to encrypt sensitive data stored in non-volatile storage and provide interfaces to applications to invoke this functionality.

TSS Link: *TSS for FCS_STO_EXT.1*

[12] [CCEVS-TD0446] was applied to FCS_SSHS_EXT.1.
[13] [CCEVS-TD0332] was applied to FCS_SSHS_EXT.1.
[14] [CCEVS-TD0446] was applied to FCS_SSHS_EXT.1.

6.1.2.14 TLS Client Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1 The OS shall implement TLS 1.2 (RFC 5246) supporting the following cipher suites:[15]

suma command:

- **TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,**
- **TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288**

OpenSSL:

- **TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,**
- **TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,**
- **TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,**
- **TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,**
- **TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,**
- **TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,**
- **TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,**
- **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,**
- **TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,**
- **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,**
- **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,**
- **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,**
- **TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,**
- **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289**

FCS_TLSC_EXT.1.2 The OS shall verify that the presented identifier matches the reference identifier according to RFC 6125.

FCS_TLSC_EXT.1.3 The OS shall only establish a trusted channel if the peer certificate is valid.

TSS Link: *TSS for FCS_TLSC_EXT.1*

6.1.2.15 TLS Client Protocol (FCS_TLSC_EXT.2)

FCS_TLSC_EXT.2.1 The OS shall present the Supported Groups Extension in the Client Hello with the following supported groups: **secp256r1, secp384r1, secp521r1**.

TSS Link: *TSS for FCS_TLSC_EXT.2*

[15] [CCEVS-TD0441]☝ was applied to FCS_TLSC_EXT.1.

6.1.3 User data protection (FDP)

6.1.3.1 Access Controls for Protecting User Data (FDP_ACF_EXT.1)

FDP_ACF_EXT.1.1 The OS shall implement access controls which can prohibit unprivileged users from accessing files and directories owned by other users.

TSS Link: *TSS for FDP_ACF_EXT.1*

6.1.4 Identification and authentication (FIA)

6.1.4.1 Authentication failure handling (Refined) (FIA_AFL.1)

FIA_AFL.1.1 The OS shall detect when **an administrator configurable positive integer within 1 to 255** unsuccessful authentication attempts occur related to events with **authentication based on user name and password**.

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts for an account has been met, the OS shall: **Account Lockout**.

TSS Link: *TSS for FIA_AFL.1*

6.1.4.2 Multiple Authentication Mechanisms (Refined) (FIA_UAU.5)

FIA_UAU.5.1 The OS shall provide the following authentication mechanisms

- **authentication based on user name and password**
- **for use in SSH only, SSH public key-based authentication as specified by the EP for Secure Shell**

to support user authentication.

FIA_UAU.5.2 The OS shall authenticate any user's claimed identity according to the

- **Username/password (local console and SSH): The TOE locally verifies the password hash matches the stored password hash associated with the provided username.**
- **SSH public key (SSH): The TOE verifies the signature can be verified using a public key associated with the provided username.**

TSS Link: *TSS for FIA_UAU.5*

6.1.4.3 X.509 Certificate Validation (FIA_X509_EXT.1)

FIA_X509_EXT.1.1 The OS shall implement functionality to validate certificates in accordance with the following rules:[16]

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- The OS shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field
- The OS shall validate the revocation status of the certificate using **OCSP as specified in RFC 6960, CRL as specified in RFC 5759**.
- The OS shall validate the extendedKeyUsage field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
  - S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the EKU field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing Purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.
  - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the EKU field. (conditional)

FIA_X509_EXT.1.2 The OS shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

TSS Link: *TSS for FIA_X509_EXT.1*

6.1.4.4 X.509 Certificate Authentication (FIA_X509_EXT.2)

FIA_X509_EXT.2.1 The OS shall use X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and **no other protocols** connections.

TSS Link: *TSS for FIA_X509_EXT.2*

[16] [CCEVS-TD0525]☝ was applied to FIA_X509_EXT.1.

6.1.5 Security management (FMT)

6.1.5.1 Management of security functions behavior (FMT_MOF_EXT.1)

FMT_MOF_EXT.1.1 The OS shall restrict the ability to perform the function indicated in the "Administrator" column in FMT_SMF_EXT.1.1 to the administrator.

TSS Link: *TSS for FMT_MOF_EXT.1*

6.1.5.2 Specification of Management Functions (FMT_SMF_EXT.1)

FMT_SMF_EXT.1.1 The OS shall be capable of performing the following management functions:

Table 8: Management functions

Management Function	**Administrator**	**User**
Enable/disable **session timeout**	X	X
Configure **session** inactivity timeout	X	X
Configure local audit storage capacity	X	
Configure minimum password length	X	
Configure minimum number of special characters in password	X	
Configure minimum number of numeric characters in password	X	
Configure minimum number of uppercase characters in password	X	
Configure minimum number of lowercase characters in password	X	
Configure lockout policy for unsuccessful authentication attempts through **timeouts between attempts, limiting number of attempts during a time period**		
Configure host-based firewall	X	
Configure name/address of directory server with which to bind		
Configure name/address of remote management server from which to receive management settings		
Configure name/address of audit/logging server to which to send audit/logging records		
Configure audit rules	X	
Configure name/address of network time server		
Enable/disable automatic software update		
Configure WiFi interface		
Enable/disable Bluetooth interface		
Enable/disable **no other external interfaces**		

Management Function	Administrator	User
no other management functions		

Application Note: *X—Supported by the specified role. Grey—Not supported by the specified role.*

TSS Link: *TSS for FMT_SMF_EXT.1*

6.1.6 Protection of the TSF (FPT)

6.1.6.1 Access controls (FPT_ACF_EXT.1)

FPT_ACF_EXT.1.1 The OS shall implement access controls which prohibit unprivileged users from modifying:

- Kernel and its drivers/modules
- Security audit logs
- Shared libraries
- System executables
- System configuration files
- **no other objects**.

FPT_ACF_EXT.1.2 The OS shall implement access controls which prohibit unprivileged users from reading:

- Security audit logs
- System-wide credential repositories
- **no other objects**.

TSS Link: *TSS for FPT_ACF_EXT.1*

6.1.6.2 Address Space Layout Randomization (FPT_ASLR_EXT.1)

FPT_ASLR_EXT.1.1 The OS shall always randomize process address space memory locations with **at least 16** bits of entropy except for **explicit exceptions: none**.

TSS Link: *TSS for FPT_ASLR_EXT.1*

6.1.6.3 Stack Buffer Overflow Protection (FPT_SBOP_EXT.1)

FPT_SBOP_EXT.1.1 The OS shall **employ stack-based buffer overflow protections**.

TSS Link: *TSS for FPT_SBOP_EXT.1*

6.1.6.4 Boot Integrity (FPT_TST_EXT.1)

FPT_TST_EXT.1.1 The OS shall verify the integrity of the bootchain up through the OS kernel and

- **device drivers (including boot device drivers)**

prior to its execution through the use of **a digital signature using a hardware-protected asymmetric key**.[17]

TSS Link: *TSS for FPT_TST_EXT.1*

[17] [CCEVS-TD0493]☝ was applied to FPT_TST_EXT.1.

6.1.6.5 Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1 The OS shall provide the ability to check for updates to the OS software itself and shall use a digital signature scheme specified in FCS_COP.1(3) to validate the authenticity of the response.[18]

FPT_TUD_EXT.1.2 The OS shall **cryptographically verify** updates to itself using a digital signature prior to installation using schemes specified in FCS_COP.1(3).[19]

TSS Link: *TSS for FPT_TUD_EXT.1*

6.1.6.6 Trusted Update for Application Software (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1 The OS shall provide the ability to check for updates to application software and shall use a digital signature scheme specified in FCS_COP.1(3) to validate the authenticity of the response.[20]

FPT_TUD_EXT.2.2 The OS shall cryptographically verify the integrity of updates to applications using a digital signature specified by FCS_COP.1(3) prior to installation.

TSS Link: *TSS for FPT_TUD_EXT.2*

6.1.6.7 Write XOR Execute Memory Pages (FPT_W^X_EXT.1)

FPT_W^X_EXT.1.1 The OS shall prevent allocation of any memory region with both write and execute permissions except for **memory mapped (a.k.a. mmap) segments**.

TSS Link: *TSS for FPT_W^X_EXT.1*

6.1.7 TOE access (FTA)

6.1.7.1 Default TOE access banners (FTA_TAB.1)

FTA_TAB.1.1 Before establishing a user session, the OS shall display an advisory warning message regarding unauthorized use of the OS.

TSS Link: *TSS for FTA_TAB.1*

6.1.8 Trusted path/channels (FTP)

6.1.8.1 Trusted channel communication (FTP_ITC_EXT.1)

FTP_ITC_EXT.1.1 The OS shall use

- **TLS as conforming to FCS_TLSC_EXT.1**
- **SSH as conforming to the EP for Secure Shell**

to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: **application-initiated TLS, remote user connections to SSH servers** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

18 [CCEVS-TD0463]☝ was applied to FPT_TUD_EXT.1.
19 [CCEVS-TD0386]☝ was applied to FPT_TUD_EXT.1.
20 [CCEVS-TD0463]☝ was applied to FPT_TUD_EXT.2.

TSS Link: *TSS for FTP_ITC_EXT.1*

6.1.8.2 Trusted Path (FTP_TRP.1)

FTP_TRP.1.1 The OS shall provide a communication path between itself and **remote** users that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from modification, disclosure.

FTP_TRP.1.2 The OS shall permit **remote users** to initiate communication via the trusted path.

FTP_TRP.1.3 The OS shall require use of the trusted path for all remote administrative actions.

TSS Link: *TSS for FTP_TRP.1*

6.2 Security Functional Requirements Rationale

A mapping of the Security Functional Requirements to the Security Objectives for the TOE are provided in [OSPPv4.2.1]☝ section 4.1, which also provides rationale for how the SFRs satisfy each objective. [OSPPv4.2.1]☝ Appendix D provides rationale for implicitly satisfied requirements for missing SFR dependencies.

In addition, FCS_SSH_EXT.1, FCS_SSHC_EXT.1, and FCS_SSHS_EXT.1 from [SSHEPv1.0]☝ address O.PROTECTED_COMMS. These SFRs define the ability of the TOE to use the SSH protocol as a method of enforcing protected communications.

Also, FCS_COP.1(5) maps to FCS_COP.1(1) from [SSHEPv1.0]☝ and addresses O.PROTECTED_COMMS. This SFR defines additional cryptographic operations used by SSH to protect communications.

6.3 Security Assurance Requirements

The security assurance requirements (SARs) for the TOE are defined in the OSPPV4.2.1 protection profile.

The following table shows the SARs, and the operations performed on the components according to CC part 3: iteration (Iter.), refinement (Ref.), assignment (Ass.) and selection (Sel.).

Table 9: SARs

Security assurance class	Security assurance requirement	Source	Operations			
			Iter.	Ref.	Ass.	Sel.
ASE Security Target evaluation	ASE_CCL.1 Conformance claims	OSPPV4.2.1	No	No	No	No
	ASE_ECD.1 Extended components definition	OSPPV4.2.1	No	No	No	No
	ASE_INT.1 ST introduction	OSPPV4.2.1	No	No	No	No
	ASE_OBJ.2 Security objectives	OSPPV4.2.1	No	No	No	No
	ASE_REQ.2 Derived security requirements	OSPPV4.2.1	No	No	No	No
	ASE_SPD.1 Security problem definition	OSPPV4.2.1	No	No	No	No
	ASE_TSS.1 TOE summary specification	OSPPV4.2.1	No	No	No	No

Security assurance class	Security assurance requirement	Source	Operations			
			Iter.	Ref.	Ass.	Sel.
ADV Development	ADV_FSP.1 Basic functional specification	OSPPV4.2.1	No	No	No	No
AGD Guidance documents	AGD_OPE.1 Operational user guidance	OSPPV4.2.1	No	No	No	No
	AGD_PRE.1 Preparative procedures	OSPPV4.2.1	No	No	No	No
ALC Life-cycle support	ALC_CMC.1 Labelling of the TOE	OSPPV4.2.1	No	No	No	No
	ALC_CMS.1 TOE CM coverage	OSPPV4.2.1	No	No	No	No
	ALC_TSU_EXT.1	OSPPV4.2.1	No	No	No	No
ATE Tests	ATE_IND.1 Independent testing - conformance	OSPPV4.2.1	No	No	No	No
AVA Vulnerability assessment	AVA_VAN.1 Vulnerability survey	OSPPV4.2.1	No	No	No	No

6.4 Security Assurance Requirements Rationale

The Security Assurance Requirements were chosen because they are required by the [OSPPv4.2.1]☝, and the ST claims exact conformance to the [OSPPv4.2.1]☝.

7 TOE Summary Specification

7.1 TSS Security Assurance Evaluation Activity

7.1.1 Timely security updates (ALC_TSU_EXT.1)

TSS Evaluation Activity:

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

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

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

The IBM Product Security Incident Response Team (PSIRT) is the centralized process through which IBM customers, security researchers, industry groups, government organizations, or vendors report potential IBM security vulnerabilities.

A global team manages the receipt, investigation and internal coordination of security vulnerability information related to all IBM products, offerings and websites. This team then coordinates with each individual IBM product and solution team across the world to investigate, and if needed, identify the appropriate response plan. Maintaining communication between all involved parties, both internal and external, is a key component of IBM's vulnerability response process.

IBM handles potential vulnerabilities using the flow shown in Figure 1 below.

AIX customers can report vulnerabilities of the product through the normal support channel. This is done over an HTTPS-protected channel. Non-customers, such as 3rd party security researchers, can submit vulnerability reports through the HTTPS-protected URL https://hackerone.com/IBM.

Additionally, AIX registers the open source packages used in the product with a tool provided by the global team. The global team monitors the security vulnerabilities reported on the open source packages of interest and alerts the respective product team for analysis and further processing.

IBM's response time depends the severity of the vulnerability.

- Severity 1: 30 days
- Severity 2: 90 days
- Severity 3: 180 days

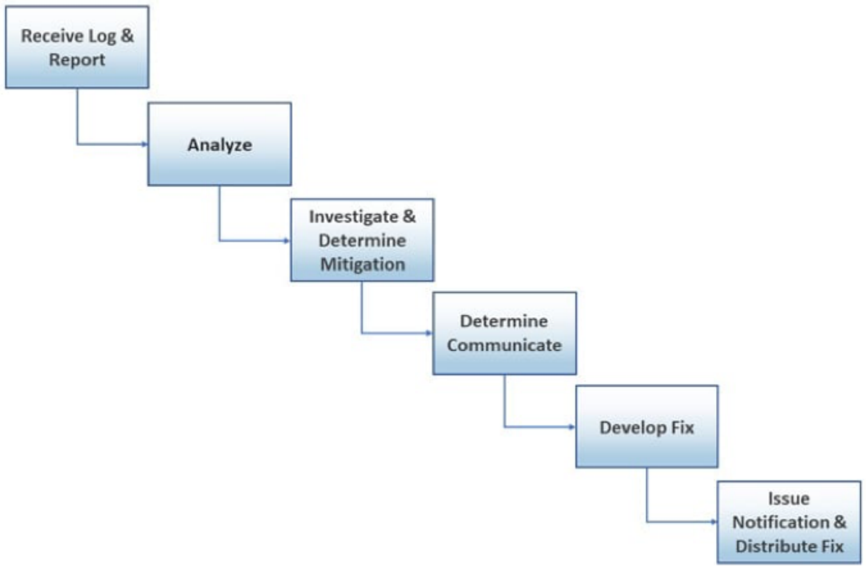
Receive Log & Report
Analyze
Investigate & Determine Mitigation
Determine Communicate
Develop Fix
Issue Notification & Distribute Fix

Figure 1: Potential vulnerability handling flow

7.2 TOE Security Functionality

7.2.1 FAU_GEN.1 Audit Data Generation (Refined)

PP: OSPPv4.2.1
SFR Link: FAU_GEN.1

TSS Evaluation Activity:

None.

TSS:

The TOE generates audit records for the following auditable events.

- Startup and shutdown events
- Authentication events (Success/Failure)
- Use of privileged/special rights events (Successful and unsuccessful security, audit, and configuration changes)
- Privilege or role escalation events (Success/Failure)
- File and object events (Successful and unsuccessful attempts to create, access, delete, modify, and modify permissions)
- User and Group management events (Successful and unsuccessful add, delete, modify, disable, enable, and credential change)
- Audit and log data access events (Success/Failure)
- System reboot, restart, and shutdown events (Success/Failure)
- Administrator or root-level access events (Success/Failure)

Each audit record includes the following information: date and time of the event, type of event, subject identity (if applicable), and outcome (success or failure) of the event.

7.2.2 FCS_CKM.1 Cryptographic Key Generation (Refined)

PP: OSPPv4.2.1
SFR Link: FCS_CKM.1

TSS Evaluation Activity: [21]

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

TSS:

Table 10 shows the cryptographic modules, algorithms, key sizes, and standards supported by the TOE in the evaluated configuration along with the TOE functionality that invokes the algorithms (a.k.a. usage). ("n/a" means not applicable.)

Table 10: Asymmetric key generation

Module	Algorithm	Capabilities	Standard	Usage
CLiC kernel space (64-bit)	n/a	n/a	n/a	n/a
CLiC user space (32-bit)	RSA KeyGen	Key lengths: 2048, 4096 (bits)	[FIPS186-4]	EFS (generate user and group key pairs)
ICC (64-bit)	n/a	n/a	n/a	n/a
OpenSSL (32-bit)	RSA KeyGen	Key lengths: 2048, 3072, 4096 (bits)	[FIPS186-4]	SSH (authentication)
	ECDSA KeyGen/KeyVer	Curves: P-256, P-384, P-521	[FIPS186-4]	SSH, TLS

7.2.3 FCS_CKM.2 Cryptographic Key Establishment (Refined)

PP: OSPPv4.2.1
SFR Link: FCS_CKM.2

[21] [CCEVS-TD0501] was applied to the TSS Evaluation Activity for FCS_CKM.1.

TSS Evaluation Activity: [22]

The evaluator will 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 will examine the TSS to verify that it identifies the usage for each scheme.

SP800-56B Key Establishment Schemes
The evaluator will verify that the TSS describes whether the OS acts as a sender, a recipient, or both for RSA-based key establishment schemes.

The evaluator will ensure that the TSS describes how the OS handles decryption errors. In accordance with NIST Special Publication 800-56B, the OS must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations.

TSS:

Table 11 shows the cryptographic modules, algorithms, key sizes, and standards supported by the TOE in the evaluated configuration along with the TOE functionality that invokes the algorithms (a.k.a. usage). No SP800-56B key establishment schemes are claimed. ("n/a" means not applicable.)

Table 11: Key establishment

Module	Algorithm	Capabilities	Standard	Usage
CLiC kernel space (64-bit)	RSA Key Establishment following RSAES-PKCS1-v1_5	Key lengths: 2048, 4096 (bits)	[RFC8017]	EFS (key pairs are generated in user space and used in kernel space.)
CLiC user space (32-bit)	RSA Key Establishment following RSAES-PKCS1-v1_5	Key lengths: 2048, 4096 (bits)	[RFC8017]	EFS (key pairs are generated and used in user space.)
ICC (64-bit)	RSA Key Establishment following RSAES-PKCS1-v1_5	Key lengths: 2048, 3072, 4096 (bits)	[RFC8017]	TLS (**suma**)
OpenSSL (32-bit)	RSA Key Establishment following RSAES-PKCS1-v1_5	Key lengths: 2048, 3072, 4096 (bits)	[RFC8017]	SSH, TLS
	Elliptic Curve-based Key Establishment (KAS-ECC-SSC)	Curves: P-256, P-384, P-521	[SP800-56A-Rev3]	SSH, TLS

7.2.4 FCS_CKM_EXT.4 Cryptographic Key Destruction

PP: OSPPv4.2.1
SFR Link: FCS_CKM_EXT.4

[22] [CCEVS-TD0501] was applied to the TSS Evaluation Activity for FCS_CKM.2.

TSS Evaluation Activity: [23]

The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator will check to ensure the TSS lists each type of key that is stored in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator will verify that the pattern does not contain any CSPs.

The evaluator will check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

If the selection "destruction of all key encrypting keys protecting target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived" is included the evaluator shall examine the TOE's keychain in the TSS and identify each instance when a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM_EXT.4.1 The evaluator shall verify that all of the keys capable of decrypting the target key are not able to be derived to reestablish the keychain after their destruction.

TSS:

7.2.4.1 EFS key destruction

For EFS keys, Table 12 shows the volatile memory key destruction instances. The "Intro methods(s)" column describes how the key is introduced into volatile memory. The "Destruct method" column describes how the key is destroyed in volatile memory. For a description of EFS and its associated keys, see section 7.2.14.

For non-volatile memory, the life of a key in the EFS keystores is indefinite and, thus, the keys are never deleted. User and group keys remain on the system even after the user or group accounts are deleted to prevent loss of data (i.e., in case there are files remaining on the system that only those keys can decrypt). A script exists to manually overwrite a keystore with all zeros once the keystore is deemed unnecessary. User and group keystores can be manually destroyed by an administrator using this script. EFS uses file system APIs to store and retrieve keys in the keystores.

Table 12: EFS key destruction in volatile memory

Program	Key	Volatile memory	
		Intro method(s)	**Destruct method**
Login programs (i.e., terminal, SSH)	EFS RSA private keys (plaintext)	1) From user keystores 2) From group keystores 3) From EFS administrator's keystore	Removal of power to memory

[23] [CCEVS-TD0365] was applied to the TSS Evaluation Activity for FCS_CKM_EXT.4.

Program	**Key**	**Volatile memory**	
		Intro method(s)	**Destruct method**
	EFS user *access keys* (plaintext)	1) From user's *admin keystore*	Removal of power to memory
	EFS group *access keys* (plaintext)	1) From user keystores	Removal of power to memory
efsenable command (EFS)	EFS user *access key* (derived)	From user *admin keystore*	Removal of power to memory
	EFS RSA private key (plaintext)	1) From user keystore 2) From group keystore 3) Generated using DRBG	Removal of power to memory
	EFS group *access keys* (plaintext)	1) From user keystore 2) From group *admin keystore* 3) Generated using DRBG	Removal of power to memory
efskeymgr command (EFS)	EFS user *access key* (derived)	From user *admin keystore*	Removal of power to memory
	EFS RSA private key (plaintext)	1) From user keystore 2) From group keystore 3) Generated using DRBG	Removal of power to memory
	EFS group *access keys* (plaintext)	1) From user keystore 2) From group *admin keystore* 3) Generated using DRBG	Removal of power to memory
mkuser commands	EFS user and group *access keys* (plaintext)	1) Generated using DRBG 2) From user keystore 3) From *admin keystore*	Removal of power to memory
mkgroup commands	EFS user *access key* (plaintext)	From user keystore	Removal of power to memory
	EFS group *access key* (plaintext)	1) Generated using DRBG 2) From user keystore 3) From *admin keystore*	Removal of power to memory
passwd command	EFS user and group *access keys* (plaintext)	1) From user keystores 2) From admin keystores	Removal of power to memory
	EFS RSA private key (plaintext)	From user keystore	Removal of power to memory
Kernel	EFS user and group RSA private keys (plaintext)	From user space	Removal of power to memory

Program	**Key**	**Volatile memory**	
		Intro method(s)	**Destruct method**
	EFS *file keys*	1) Generated using DRBG 2) From EFS meta data	Removal of power to memory

7.2.4.2 SSH key destruction

For SSH keys, Table 13 shows the volatile memory key destruction instances. The "Intro methods(s)" column describes how the key is introduced into volatile memory. The "Destruct method" column describes how the key is destroyed in volatile memory.

For non-volatile memory, the life of a key pair in an SSH keystore is indefinite. A script exists to manually overwrite a keystore with all zeros once the keystore is deemed unnecessary. SSH uses OpenSSL APIs to store and retrieve keys in the keystore.

Table 13: SSH key destruction in volatile memory

Function	**Key**	**Volatile memory**	
		Intro method(s)	**Destruct method**
SSH (OpenSSH)	Private key	From keystore	Single overwrite of zeros
	Ephemeral ECDH private keys	Derived during handshake	Removal of power to memory
	Ephemeral MAC keys	Derived during handshake	Single overwrite of zeros
	Ephemeral symmetric session keys	Derived during handshake	Single overwrite of zeros

7.2.4.3 OpenSSL TLS key destruction

For TLS keys, Table 14 shows the volatile memory key destruction instances. The "Intro methods(s)" column describes how the key is introduced into volatile memory. The "Destruct method" column describes how the key is destroyed in volatile memory.

The ifix commands use the OpenSSL implementation of the TLS client (**openssl s_client**) to connect to the IBM fix server. The ifix commands do not use TLS mutual authentication. The commands retrieve CA public certificates from a keystore using OpenSSL APIs. Because the connection does not perform mutual authentication, the ifix commands' non-volatile memory keystore does not contain private keys, only public keys; thus, the keystore never needs to be destroyed. Destroying this keystore may cause the ifix commands to malfunction.

A script exists to manually overwrite a keystore with all zeros if the keystore is deemed unnecessary.

Table 14: OpenSSL TLS key destruction in volatile memory

Function	Key	Volatile memory	
		Intro method(s)	**Destruct method**
TLS (OpenSSL)	Ephemeral ECDH private keys	Derived during handshake	Removal of power to memory
	Ephemeral MAC keys	Derived during handshake	Single overwrite of zeros
	Ephemeral symmetric session keys	Derived during handshake	Single overwrite of zeros

7.2.4.4 IBM Java TLS key destruction

For TLS keys, Table 15 shows the volatile memory key destruction instances. The "Intro methods(s)" column describes how the key is introduced into volatile memory. The "Destruct method" column describes how the key is destroyed in volatile memory.

The **suma** command uses the IBM Java implementation of the TLS client to connect to the IBM fix server. This connection does not use mutual authentication. The command retrieves CA public certificates from a keystore using IBM Java APIs. Because the connection does not perform mutual authentication, the **suma** command's non-volatile memory keystore does not contain private keys, only public keys; thus, the keystore never needs to be destroyed. Destroying this keystore may cause the **suma** command to malfunction.

Table 15: IBM Java TLS key destruction in volatile memory

Function	Key	Volatile memory	
		Intro method(s)	**Destruct method**
TLS (**suma** command)	Ephemeral MAC keys	Derived during handshake	Removal of power to memory
	Ephemeral symmetric session keys	Derived during handshake	Removal of power to memory

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

PP: OSPPv4.2.1
SFR Link: FCS_COP.1(1)

TSS Evaluation Activity:

None.

TSS:

Table 16 shows the cryptographic modules, algorithms, key sizes, and standards supported by the TOE in the evaluated configuration along with the TOE functionality that invokes the algorithms (a.k.a. usage).

Table 16: Symmetric encryption/decryption

Module	Algorithm	Capabilities	Standard	Usage
CLiC kernel space (64-bit)	AES-CBC encrypt/decrypt	Key lengths: 128, 256 (bits)	[SP800-38A]	EFS
CLiC user space (32-bit)	AES-CBC encrypt/decrypt	Key lengths: 256 (bits)	[SP800-38A]	EFS keystore
ICC (64-bit)	AES-GCM encrypt/decrypt	Key lengths: 128, 256 (bits) Tag length: 128 (bits) IV length: 96 (bits)	[SP800-38D]	TLS (**suma**)
OpenSSL (32-bit)	AES-CBC encrypt/decrypt	Key lengths: 128, 256 (bits)	[SP800-38A]	SSH, TLS
	AES-GCM encrypt/decrypt	Key lengths: 128, 256 (bits) Tag length: 128 (bits) IV length: 96 (bits)	[SP800-38D]	SSH, TLS

7.2.6 FCS_COP.1(2) Cryptographic Operation - Hashing (Refined)

PP: OSPPv4.2.1
SFR Link: FCS_COP.1(2)

TSS Evaluation Activity:

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

TSS:

Table 17 shows the cryptographic modules, algorithms, digest sizes, and standards supported by the TOE in the evaluated configuration along with the TOE functionality that invokes the algorithms (a.k.a. usage).

Table 17: Hash algorithms

Module	Algorithm	Digest size	Standard	Usage
CLiC kernel space (64-bit)	SHA2-256, SHA2-512	256-bit, 512-bit (byte-oriented)	[FIPS180-4]	Boot integrity (for signature verification)
	SHA2-512	512-bit (byte-oriented)	[FIPS180-4]	EFS (for Hash_DRBG)
CLiC user space (32-bit)	SHA2-256	256-bit (byte-oriented)	[FIPS180-4]	EFS (for Hash_DRBG)
ICC (64-bit)	SHA-1, SHA2-256, SHA2-384, SHA2-512	160-bit, 256-bit, 384-bit, 512-bit (byte-oriented)	[FIPS180-4]	TLS (**suma**)

Module	Algorithm	Digest size	Standard	Usage
OpenSSL (32-bit)	SHA-1, SHA2-256, SHA2-384, SHA2-512	160-bit, 256-bit, 384-bit, 512-bit (byte-oriented)	[FIPS180-4]☝	SSH, TLS, and Trusted update[24] (for signature verification; for SSH and TLS signature generation, HMACs, DRBGs, and other protocol needs)

7.2.7 FCS_COP.1(3) Cryptographic Operation - Signing (Refined)

PP: OSPPv4.2.1
SFR Link: FCS_COP.1(3)

TSS Evaluation Activity:

None.

TSS:

Table 18 shows the cryptographic modules, algorithms, key sizes, and standards supported by the TOE in the evaluated configuration along with the TOE functionality that invokes the algorithms (a.k.a. usage). ("n/a" means not applicable.)

Table 18: Signature generation/verification

Module	Algorithm	Capabilities	Standard	Usage
CLiC kernel space (64-bit)	RSA signature verification RSASSA-PKCS1-v1_5	Key lengths: 2048, 4096 (bits) with hash algorithms: SHA2-256, SHA2-512	[FIPS186-4]☝	Boot integrity
CLiC user space (32-bit)	n/a	n/a	n/a	n/a
ICC (64-bit)	RSA signature generation RSASSA-PKCS1-v1_5	Key lengths: 2048, 3072, 4096 (bits) each with hash algorithms: SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	TLS (**suma**)
	RSA signature verification RSASSA-PKCS1-v1_5	Key lengths: 2048, 3072, 4096 (bits) each with hash algorithms: SHA-1, SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	TLS (**suma**)

[24] Trusted update only uses SHA2-256.

Module	Algorithm	Capabilities	Standard	Usage
	RSA signature generation RSASSA-PSS	Key lengths: 2048, 3072, 4096 (bits) each with hash algorithms: SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	TLS (**suma**)
	RSA signature verification RSASSA-PSS	Key lengths: 2048, 3072, 4096 (bits) each with hash algorithms: SHA-1, SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	TLS (**suma**)
OpenSSL (32-bit)	RSA signature generation RSASSA-PKCS1-v1_5	Key lengths: 2048, 3072, 4096 (bits) each with hash algorithms: SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	SSH, TLS
	RSA signature verification RSASSA-PKCS1-v1_5	Key lengths: 2048, 3072, 4096 (bits) each with hash algorithms: SHA-1, SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	SSH, TLS, Trusted update[25]
	ECDSA signature generation	Curves: P-256, P-384, P-521 each with hash algorithms: SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	SSH, TLS
	ECDSA signature verification	Curves: P-256, P-384, P-521 each with hash algorithms: SHA-1, SHA2-256, SHA2-384, SHA2-512	[FIPS186-4]☝	SSH, TLS

7.2.8 FCS_COP.1(4) Cryptographic Operation - Keyed-Hash Message Authentication (Refined)

PP: OSPPv4.2.1
SFR Link: FCS_COP.1(4)

[25] Trusted update only uses RSA 2048-bit with SHA2-256 for package signature verification.

TSS Evaluation Activity:

None.

TSS:

Table 19 shows the cryptographic modules, algorithms, digest sizes, and standards supported by the TOE in the evaluated configuration along with the TOE functionality that invokes the algorithms (a.k.a. usage). ("n/a" means not applicable.)

Table 19: Keyed-hash message authentication

Module	Algorithm	Digest size	Standard	Usage
CLiC kernel space (64-bit)	n/a	n/a	n/a	n/a
CLiC user space (32-bit)	n/a	n/a	n/a	n/a
ICC (64-bit)	HMAC-SHA2-256	256 bits	[FIPS198-1]☝	TLS (**suma**)
	HMAC-SHA2-384	384 bits	[FIPS198-1]☝	TLS (**suma**)
OpenSSL (32-bit)	HMAC-SHA-1	160 bits	[FIPS198-1]☝	SSH, TLS
	HMAC-SHA2-256	256 bits	[FIPS198-1]☝	SSH, TLS
	HMAC-SHA2-384	384 bits	[FIPS198-1]☝	TLS
	HMAC-SHA2-512	512 bits	[FIPS198-1]☝	SSH

7.2.9 FCS_COP.1(5) Cryptographic Operation - Encryption/Decryption (Refined)

PP: SSHEPv1.0
SFR Link: FCS_COP.1(5)

TSS Assurance Activity: [26]

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.

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.

TSS:

Table 20 shows the cryptographic modules, algorithms, key sizes, and standards supported by the TOE in the evaluated configuration along with the TOE functionality that invokes the algorithms (a.k.a. usage). ("n/a" means not applicable.)

[26] [CCEVS-TD0240]☝ was applied to the TSS Assurance Activity for FCS_COP.1(5).

Table 20: AES-CTR encryption/decryption

Module	Algorithm	Key size	Standard	Usage
CLiC kernel space (64-bit)	n/a	n/a	n/a	n/a
CLiC user space (32-bit)	n/a	n/a	n/a	n/a
ICC (64-bit)	n/a	n/a	n/a	n/a
OpenSSL (32-bit)	AES-CTR encrypt/decrypt	Key lengths: 128, 256 (bits)	[SP800-38A]	SSH

The CTR mode counter is a 128-bit value output from the SSH key exchange, so it is guaranteed to be unique. The counter is incremented by 1 for each block that is encrypted. SSH rekeys at least every 512MB of data transmitted for each key, so only a maximum of 2^25 counter values could be used, ensuring the counter does not wrap.

7.2.10 FCS_RBG_EXT.1 Random Bit Generation

PP: OSPPv4.2.1
SFR Link: FCS_RBG_EXT.1

TSS Evaluation Activity:

None.

TSS:

Table 21 shows the TOE's DRBGs on a per cryptographic module basis.

Table 21: Random bit generation

Module	Algorithm	Mode	Standard	Noise source	Usage
CLiC kernel space (64-bit)	Hash_DRBG	SHA2-512	SP800-90A	Platform-based	EFS
CLiC user space (32-bit)	Hash_DRBG	SHA2-256	SP800-90A	Platform-based	EFS
ICC (64-bit)	Hash_DRBG	SHA2-256	SP800-90A	Software-based	TLS (**suma**)
OpenSSL (32-bit)	CTR_DRBG	AES-256	SP800-90A	Platform-based	SSH, TLS

The CLiC kernel space cryptographic module uses the 64-bit Hash_DRBG(SHA2-512) when generating key material. The CLiC user space cryptographic module uses the 32-bit Hash_DRBG(SHA2-256) when generating key material. The ICC cryptographic module uses the Hash_DRBG(SHA2-256) when generating key material. The OpenSSL cryptographic module uses the CTR_DRBG(AES-256) for generating TLS key material. OpenSSH also uses CTR_DRBG(AES-256) from OpenSSL for generating SSH key material.

For the CLiC kernel space, CLiC user space, and OpenSSL cryptographic modules, the DRBGs are seeded using a platform-based noise source with at least 256 bits of entropy. The platform-based noise source uses hardware ring oscillators built into the POWER9 SMT8 core processor. The processor provides the Deliver A Random Number (**darn**) instruction for accessing the entropy of its ring oscillators. Both raw and conditioned entropy are available through the **darn** instruction.

The CLiC user space and OpenSSL cryptographic modules use the */dev/random* device to obtain entropy for their DRBGs. The */dev/random* device returns data from the **darn** instruction to fulfill requests. The CLiC kernel space cryptographic module uses an internal kernel function to obtain entropy which also returns data from the **darn** instruction.

For the ICC cryptographic module, the DRBG is seeded using a software-based noise source with at least 256 bits of entropy. The noise source is built into the ICC cryptographic module and has its own internal interface for accessing entropy data.

7.2.11 FCS_SSH_EXT.1 SSH Protocol

PP: SSHEPv1.0
SFR Link: FCS_SSH_EXT.1

TSS Assurance Activity:

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

TSS:

The TOE uses OpenSSH 8.1p1 for its SSH client and SSH server. OpenSSH corresponds to RFCs 4251, 4252, 4253, 4254, 5656 (elliptic curve support), and 6668 (hmac-sha2-256 and hmac-sha2-512 support). OpenSSH also implements the aes128-gcm@openssh.com and aes256-gcm@openssh.com transport encryption algorithms from [OpenSshSpec]☝ section 1.6.

7.2.12 FCS_SSHC_EXT.1 SSH Protocol - Client

PP: SSHEPv1.0
SFR Link: FCS_SSHC_EXT.1

TSS Assurance Activity:

FCS_SSHC_EXT.1.1: 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 are also allowed.

FCS_SSHC_EXT.1.2: The evaluator will check that the TSS describes how "large packets" in terms of RFC 4253 are detected and handled.

FCS_SSHC_EXT.1.3: 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.

FCS_SSHC_EXT.1.4: 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.

FCS_SSHC_EXT.1.5: 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.

FCS_SSHC_EXT.1.6: 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.

TSS:

The TOE uses OpenSSH for its SSH client. The SSH client supports both public key-based authentication and password-based authentication. The TOE's public key-based authentication supports the following public key algorithms and rejects all other public key algorithms.

- rsa-sha2-256
- rsa-sha2-512
- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384

In the evaluated configuration, the TOE is configured to reject packets greater than 262,144 (0x40000) bytes.

The SSH client supports the following transport encryption algorithms and rejects all other encryption algorithms.

- aes128-ctr
- aes256-ctr
- aes128-cbc
- aes256-cbc
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com

The SSH client supports the following message authentication code (MAC) algorithms and rejects all other MAC algorithms.

- hmac-sha1
- hmac-sha2-256
- hmac-sha2-512
- Implicit:
  - aes128-gcm@openssh.com
  - aes256-gcm@openssh.com

The SSH client supports the following key exchange methods which are the only allowed key exchange methods.

- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The SSH client supports the following rekey methods. The OpenSSH **RekeyLimit** value must be configured as follows.

- no more than 512MB of data has been transmitted
- no more than 1 hour

Limiting the client size to 512MB limits the total combined transfer size to 1GB.

The SSH client authenticates the identity of the SSH server using a local database that associates each host name with its corresponding public key as described in RFC 4251 section 4.1.

The following optional characteristics are supported by OpenSSH.

- [RFC4252]☝ password-based authentication
- [RFC4253]☝ protocol version comment string
- [RFC4253]☝ zlib compression
- [RFC6668]☝ hmac-sha2-512

7.2.13 FCS_SSHS_EXT.1 SSH Protocol - Server

PP: SSHEPv1.0
SFR Link: FCS_SSHS_EXT.1

TSS Assurance Activity:

FCS_SSHS_EXT.1.1: 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 are also allowed.

FCS_SSHS_EXT.1.2: The evaluator will check that the TSS describes how "large packets" in terms of RFC 4253 are detected and handled.

FCS_SSHS_EXT.1.3: 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.

FCS_SSHS_EXT.1.4: 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.

FCS_SSHS_EXT.1.5: 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.

FCS_SSHS_EXT.1.6: 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.

TSS:

The TOE uses OpenSSH for its SSH server. The SSH server supports both public key-based authentication and password-based authentication. The TOE's public key-based authentication supports the following public key algorithms and rejects all other public key algorithms.

- rsa-sha2-256
- rsa-sha2-512
- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384

In the evaluated configuration, the TOE is configured to reject packets greater than 262,144 (0x40000) bytes.

The SSH server supports the following transport encryption algorithms.

- aes128-ctr
- aes256-ctr

- aes128-cbc
- aes256-cbc
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com

The SSH server supports the following MAC algorithms and rejects all other MAC algorithms.

- hmac-sha1
- hmac-sha2-256
- hmac-sha2-512
- Implicit:
    - aes128-gcm@openssh.com
    - aes256-gcm@openssh.com

The SSH server supports the following key exchange methods which are the only allowed key exchange methods.

- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The SSH server supports the following rekey methods. The OpenSSH **RekeyLimit** value must be configured as follows.

- no more than 512MB of data has been transmitted
- no more than 1 hour

Limiting the server size to 512MB limits the total combined transfer size to 1GB.

The following optional characteristics are supported by OpenSSH.

- [RFC4252]☝ password-based authentication
- [RFC4253]☝ protocol version comment string
- [RFC4253]☝ zlib compression
- [RFC6668]☝ hmac-sha2-512

7.2.14 FCS_STO_EXT.1 Storage of Sensitive Data

PP: OSPPv4.2.1
SFR Link: FCS_STO_EXT.1

TSS Evaluation Activity:

The evaluator will check the TSS to ensure that it lists all persistent sensitive data for which the OS provides a storage capability. For each of these items, the evaluator will confirm that the TSS lists for what purpose it can be used, and how it is stored. The evaluator will confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1(1).

TSS:

7.2.14.1 EFS

The TOE's JFS2 file system supports an Encrypted File System (EFS) feature for encrypting sensitive data stored in non-volatile storage. EFS performs file-based encryption of individual files instead of block-based encryption of an entire file system. This allows individual files to be selectively encrypted by users and applications while other files remain unencrypted. EFS also allows for new files in a directory to be encrypted by default via a directory encryption inheritance mechanism. When inheritance is set on a directory, EFS automatically encrypts new files created in that directory.

EFS requires each user and group wishing to encrypt and decrypt files to have their own asymmetric key pair and keystore. Key pairs and keystores can be generated using the **efskeymgr** command. Each user and group, that wishes to interact with encrypted files, has their own private PKCS #12 keystore in the */var/efs/* directory for storing their key pair. These keystores are only readable (and writable) by the root user; thus, a user cannot directly read their own keystore. Private keys stored in these keystores are encrypted using one of the following methods.

- AES-CBC-256 and a 256-bit password-based key (e.g., the user's login password) derived using PBKDF2 (user keystores)
- AES-CBC-256 and a 256-bit TOE-generated *access key* (user and group keystores)
- Public key from an asymmetric key pair (user and group keystores)

Thus, no secrets are stored in plaintext in the keystores (i.e., in non-volatile storage). User keystores are most commonly encrypted using either a password or a TOE-generated *access key*. Group keystores are encrypted using a TOE-generated *access key*.

TOE-generated *access keys* are used for user keystores, instead of passwords, when the user wants an administrator to be able to recover the user's keystore keys when the user forgets their password. The TOE calls this **admin mode**. In **admin mode**, the TOE generates a user *access key*, wraps it with a special EFS administrator's public key that is created when EFS is enabled on a file system, adds the wrapped key to a separate keystore called an *admin keystore*, and uses this *access key* to encrypt secret keys in the user's keystore. The *admin keystore* is stored in the same directory as the user's keystore. This allows an administrator to decrypt the user's *access key* using the EFS administrator's private key and access the user's keystore. When the user's password is used to encrypt the user's keystore (called **guard mode**) and the user forgets their password, an administrator is unable to recover the user's private key from the keystore.

The evaluated configuration supports the following asymmetric key pair algorithms and key sizes.

- RSA_2048
- RSA_4096

Asymmetric key pairs and *access keys* are generated using the Hash_DRBG(SHA2-512) algorithm which is seeded from the blocking entropy pool (e.g., the */dev/random* device).

In order for a user to have group access to an encrypted file, the user keystores contain group *access keys* for the groups to which the user is a member. This allows EFS to load and use group asymmetric keys (from group keystores) of groups associated with that user.

To initially encrypt a file, the kernel generates a random symmetric file encryption key, called the *file key*. A *file key* is generated using the Hash_DRBG(SHA2-512) algorithm which is seeded using the kernel's internal equivalent of the */dev/random* device. The symmetric encryption algorithms supported in the evaluated configuration are the following.

- AES_128_CBC
- AES_256_CBC

The key size of the *file key* matches the algorithm used to encrypt the file (e.g., a 256-bit key is generated for AES_256_CBC). The *file key* is never exported from the kernel.

Once generated, the *file key* is then wrapped with the public key of the user creating the file and separately with the public key of the group, and then both stored as part of the file's meta data in the file system. For other users and groups to be able to decrypt the file, the *file key* must be separately wrapped with their public keys and stored in the file's meta data along with the creator's wrapped *file key*. (The **efsmgr** command, which can enable encryption on a file, allows the invoker to list the users and groups that can decrypt the file on the command line.)

When EFS is used, it is typically enabled on an existing file system, but existing files in that file system will not be automatically encrypted by the enablement. To encrypt a existing unencrypted file, the user must use the **efsmgr** command, specify the file name, the users and groups allowed to decrypt the file (assuming they have an asymmetric key pair), and have standard access rights to the file. This command can also disable encryption on a file.

The encryption and decryption happen transparently to the user's processes that have cryptographic access. The transparency is implemented as follows. The login programs (e.g., terminal, SSH) and/or the **efskeymgr** command obtain access to the user's keystore (e.g., prompt the user for their password), then push the user's key pair into the kernel and associate the key pair with the process. If valid group *access keys* are found in the user's keystore, the corresponding group keystores are also opened and the group's key pair automatically pushed into the kernel. Child processes automatically inherit indirect use of these keys. Thus, when a user's process accesses an encrypted file, the kernel has the user's set of keys stored in memory in order to decrypt the file's *file key*.

By default, the EFS feature is not enabled when the TOE is installed. An administrator can enable EFS using the **efsenable** command.

The default asymmetric key pair algorithm and symmetric encryption algorithm are set when EFS is enabled. The **efsenable** command accepts an asymmetric algorithm value via the **-k** *keystore_algo* command line parameter. The command accepts a symmetric algorithm value via the **-f** *cipher* command line parameter. This sets the default values for these algorithms as well as defines the asymmetric algorithm used by the *admin keystore*.

The **mkuser** command can create a user keystore and generate a user *access key*. The **passwd** command can create a user keystore and change the user's keystore password. The **mkgroup** command can create a group keystore and generate a group *access key*. The **efsenable** command can create a user keystore, a group keystore (for the *security* group), and generate the user and group *access keys* to these keystores.

The kernel uses the CLiC kernel space cryptographic module for the EFS-related algorithms mentioned above. The commands use the CLiC user space cryptographic module for the EFS-related algorithms mentioned above.

For APIs, AIX provides the **efs_closeKS** subroutine used to disassociate a process from all open keystores. The subroutine is useful after a call to the **fork** subroutine in order for a child process to disassociate any EFS keys inherited from the parent process.

7.2.14.2 OpenSSL keystores

The TOE uses OpenSSL keystores to securely store sensitive data. The OpenSSL keystores protect sensitive cryptographic data such as asymmetric private keys used for SSH communication. The keystores use AES-CBC-128 or AES-CBC-256 to protect the data.

7.2.15 FCS_TLSC_EXT.1 TLS Client Protocol

PP: OSPPv4.2.1
SFR Link: FCS_TLSC_EXT.1

TSS Evaluation Activity:

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

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

TSS:

The TOE's **suma** command, which uses IBM Java TLS to connect to the IBM fix server, supports the following TLS 1.2 cipher suites.

- TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288

The TOE's ifix commands, which use OpenSSL TLS client (**openssl s_client**) to connect to the IBM fix server, supports the following TLS 1.2 cipher suites.

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,
- TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,
- TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,
- TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,
- TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

The TOE establishes the reference identifier by parsing the Domain Name System (DNS) Name or Internet Protocol (IP) address for the configured Session Initiation Protocol (SIP) server. The reference identifier is matched against the Subject Alternative Name (SAN), if present. If the SAN is not present, the referenced identifier is matched against the Common Name (CN). The TOE supports wildcards in the DNS name of the server certificate. The TOE does not support Uniform Resource Identifier (URI) reference identifiers, Service (SRV) record reference identifiers, or certificate pinning.

7.2.16 FCS_TLSC_EXT.2 TLS Client Protocol

PP: OSPPv4.2.1
SFR Link: FCS_TLSC_EXT.2

TSS Evaluation Activity:

The evaluator will verify that TSS describes support for the Supported Groups Extension and whether the required behavior is performed by default or may be configured.

TSS:

The TOE's OpenSSL TLS client presents the Supported Groups Extension in the Client Hello with the following groups: secp256r1, secp384r1, and secp521r1. The TOE supports these groups by default along with other groups.

The TOE's IBM Java TLS client does not support elliptic curves in the evaluated configuration.

7.2.17 FDP_ACF_EXT.1 Access Controls for Protecting User Data

PP: OSPPv4.2.1
SFR Link: FDP_ACF_EXT.1

TSS Evaluation Activity:

The evaluator will confirm that the TSS comprehensively describes the access control policy enforced by the OS. The description must include the rules by which accesses to particular files and directories are determined for particular users. The evaluator will inspect the TSS to ensure that it describes the access control rules in such detail that given any possible scenario between a user and a file governed by the OS the access control decision is unambiguous.

TSS:

The TOE supports the follow discretionary access control (DAC) mechanism.

- Standard UNIX permission bits (a.k.a mode bits)

Subjects (i.e., processes) contain a real user ID (RUID), an effective user ID (EUID), a real group ID (RGID), an effective group ID (EGID), and a supplemental groups list (i.e., a user may be a member of multiple groups). DAC decisions are based on the EUID, EGID, or the supplemental groups list. The user ID (UID) of 0 (a.k.a. root) is considered an administrative UID. When a process' EUID is 0, the process can bypass much of the permission bit enforcement.

7.2.17.1 Permission bits

The TOE uses standard UNIX permission bits to provide one form of DAC for named file system objects. There are three sets of three bits that define access for three categories of users: the owning user (a.k.a. owner), users in the owning group (a.k.a. group), and other users (a.k.a. others, world). The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write (w), and one for execute/search (x). Each subject's access to an object is defined by some combination of these bits. For example on a file object:

- rwx symbolizing read/write/execute
- r-x symbolizing read/execute
- r-- symbolizing read
- --- symbolizing null

When a process attempts to reference an object protected only by permission bits, the TOE determines the access as follows.

- If the object's owning UID matches the process' EUID and the object's user permission bits match the process' requested access, then the TOE grants the requested access to the process. No further access checks are performed.
- If the object's owning GID matches the process' EGID or one of the process' supplemental groups and the object's group permission bits match the process' requested access, then the TOE grants the requested access to the process. No further access checks are performed.
- If the process is neither the owner nor a member of an appropriate group and the world permission bits allow the process' requested access, then the TOE grants the requested access to the process. No further access checks are performed.
- If the process' EUID is 0 and the process is attempting to access the object for read and/or write, then the process is permitted access regardless of the permission bit settings.
- If the process' EUID is 0 and the process is attempting to execute the object (or search a directory), then the process is permitted to execute the object only if at least one of the execution bits is set in the permission bits.

Read-only file systems are a special case. If a file system is mounted as read-only, then all write requests to objects on the file system are denied.

7.2.17.2 Files and directories

7.2.17.2.1 Ordinary files

Ordinary files support the concept of execution. Execute access is required to execute the file as a program or script. When an executable file has the set-user-ID or set-group-ID flags set and the file owner or file group is not the same as the process' current EUID or EGID, the executing program changes the process' EUID and/or EGID to match the set-user-ID and/or set-group-ID values. Otherwise, the attributes remain unchanged. AIX does not support setuid or setgid scripts with this mechanism.

7.2.17.2.2 Directories

The execute access for directories governs the ability to traverse the directory as part of a pathname. A process must have execute access in order to traverse the directory during pathname resolution. Directories may not be written directly, but only by creating, renaming, and removing (unlinking) objects within them. These operations are considered writes for the purpose of the DAC policy.

7.2.17.3 DAC defaults

The default access control on newly created file system objects is determined by the permissions associated with the directory where the object was created, the EUID, EGID, **umask** value of the process that created the object, and the specific permissions requested by the process creating the object. The **umask** value is a permission bits bitmask value controllable by a user. Initial access permissions on a file system object are those specified by the creating process bitwise ANDed with the one's complement of the **umask** value. For example, if a program specified initial permissions of 0664 (read/write for owner, read/write for group, and read for world) but the **umask** value were set to 0027 (prevent write for group or world, prevent all permissions for world), then the initial file permissions would be set to 0640 (or 0664 bitwise-AND 0750).

7.2.18 FIA_AFL.1 Authentication failure handling (Refined)

PP: OSPPv4.2.1
SFR Link: FIA_AFL.1

TSS Evaluation Activity:

None.

TSS:

The TOE detects unsuccessful authentication attempts in an administrator-configurable range from 1 to 255 for user name and password-based authentication attempts. When the unsuccessful authentication amount is met, the TOE locks the user's account. An administrator must unlock the account.

7.2.19 FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

PP: OSPPv4.2.1
SFR Link: FIA_UAU.5

TSS Evaluation Activity:

FIA_UAU.5.1: If user name 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 OS can interface.

FIA_UAU.5.2: The evaluator will ensure that the TSS describes each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication.

TSS:

For the local console, the TOE supports username/password authentication. For SSH, the TOE supports both username/password authentication and public key-based authentication.

For username/password authentication, the TOE requires the user to be defined in both the */etc/passwd* file and the */etc/security/passwd* file. The */etc/passwd* file contains all usernames as well as user ID and other user information. The user's salted hashed passwords are maintained in the */etc/security/passwd* file which is only accessible by an administrator. When a user enters their password, the password is hashed and compared to the user's hashed password in the */etc/security/passwd* file. If the hashed passwords match, the authentication mechanism allows the authentication process to continue.

In the evaluated configuration, OpenSSH server supports a key-based authentication. When a user logs in, instead of providing a password, the user's SSH client sends a message signed with the user's private key to the OpenSSH server. If the OpenSSH server can verify the signature using a public key located in the user’s *authorized_keys* file where the OpenSSH server resides, the OpenSSH server considers the user authenticated.

7.2.20 FIA_X509_EXT.1 X.509 Certificate Validation

PP: OSPPv4.2.1
SFR Link: FIA_X509_EXT.1

TSS Evaluation Activity:

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

TSS:

The TOE supports the use of X.509v3 certificates as defined by [RFC5280] to support authentication for TLS connections.

The TOE validates the X.509 certificates using the certificate path validation algorithm defined by [RFC5280], which is summarized as follows.

- Check public key algorithm and parameters
- Check current date/time validity period
- If IBM Java TLS: Check revocation status using OCSP as per [RFC6960]
- If OpenSSL TLS: Check revocation status using CRL as per [RFC5759]
- Check that the issuer name matches the subject name
- Process certificate extensions

When the certificate being validated is for a TLS server, the TOE ensures the extendedKeyUsage extension contains the Server Authentication purpose. The TOE ensures all CA certificates contain the basic constraints extension and that the CA=TRUE flag is set. The TOE certificate validation algorithm ensures that the certificate path terminates in a trusted root CA.

Only the TLS client protocol is claimed by this ST; therefore, TLS client certificates are not validated by the TOE. Enrollment over secure transport (EST) is not supported.

7.2.21 FIA_X509_EXT.2 X.509 Certificate Authentication

PP: OSPPv4.2.1
SFR Link: FIA_X509_EXT.2

TSS Evaluation Activity:

None.

TSS:

The TOE supports the use of X.509v3 certificates as defined by [RFC5280] to support authentication for TLS connections.

7.2.22 FMT_MOF_EXT.1 Management of security functions behavior

PP: OSPPv4.2.1
SFR Link: FMT_MOF_EXT.1

TSS Evaluation Activity:

The evaluator will verify that the TSS describes those management functions that are restricted to Administrators, including how the user is prevented from performing those functions, or not able to use any interfaces that allow access to that function.

TSS:

The TOE restricts all management activities listed in FMT_SMF_EXT.1 to administrators only. Non-administrative users are prevented from performing these functions through the use of the access control mechanisms defined in FDP_ACF_EXT.1. In addition, administrative accounts are protected by passwords.

7.2.23 FMT_SMF_EXT.1 Specification of Management Functions

PP: OSPPv4.2.1
SFR Link: FMT_SMF_EXT.1

TSS Evaluation Activity:

None.

TSS:

For an SSH session, an administrator can enable and disable the TOE's SSH session timeout feature and configure the session inactivity timeout. This can be configured in the */etc/ssh/sshd_config* file.

For a local console session, any user that has access to the local console and a valid account can enable and disable the TOE's local console session timeout feature and configure the session inactivity timeout. The default values can be configured by an administrator in the */etc/profile* file. Each user can override the default value in their own *$HOME/.profile* file.

An administrator can configure the local audit storage capacity via the */etc/security/audit/config* file using the **binsize** and **freespace** attributes.

An administrator can configure the following password composition attributes in the */etc/security/user* file.

- **minlen**—Minimum password length
- **minspecialchar**—Minimum number of special characters in a password
- **mindigit**—Minimum number of numeric characters in a password
- **minupperalpha**—Minimum number of uppercase characters in a password
- **minloweralpha**—Minimum number of lowercase characters in a password

An administrator can configure a TCP/IP filter (i.e., firewall) using the following commands.

- **chfilt**—Changes existing filter rules
- **genfilt**—Adds a filter rule to the table; Also use to create new filters
- **lsfilt**—Lists filter rules present in the table.
- **mkfilt**—Activate or deactivate the filter rules in the table, enable or disable logging for filters, and change the default rules
- **rmfilt**—Removes existing filter rules

An administrator can configure the set of audited events (i.e., audit rules) audited by the TOE.

7.2.24 FPT_ACF_EXT.1 Access controls

PP: OSPPv4.2.1
SFR Link: FPT_ACF_EXT.1

TSS Evaluation Activity:

The evaluator will confirm that the TSS specifies the locations of kernel drivers/modules, security audit logs, shared libraries, system executables, and system configuration files. Every file does not need to be individually identified, but the system's conventions for storing and protecting such files must be specified.

TSS:

The TOE uses the access control described in FDP_ACF_EXT.1 to prohibit unprivileged users from modifying the following.

- Kernel and its drivers/modules in:
  - */dev/*
  - */unix/*
  - */usr/lib/*
- Security audit logs in:
  - */audit/*
- Shared libraries in:
  - */usr/lib/*
- System executables in:
  - */usr/bin/*
  - */usr/sbin/*
- System configuration files in:
  - */etc/*
  - */etc/security/*
  - */etc/security/audit/*

The TOE uses the access control described in FDP_ACF_EXT.1 to prohibit unprivileged users from reading the following.

- Security audit logs in:
  - */audit/*
- System-wide credential repositories in:
  - */etc/security/*
  - */var/efs/*

7.2.25 FPT_ASLR_EXT.1 Address Space Layout Randomization

PP: OSPPv4.2.1
SFR Link: FPT_ASLR_EXT.1

TSS Evaluation Activity:

None.

TSS:

In the evaluated configuration, ASLR is enabled for all applications (no exceptions) using option setting 2. For 32-bit applications, the execution code segment has 16 bits of randomization and the stack has 24 bits of randomization. For 64-bit applications, the execution code segment has 29 bits of randomization and the stack has 32 bits of randomization. The **vmo** command supports the enabling and disabling of ASLR.

7.2.26 FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

PP: OSPPv4.2.1
SFR Link: FPT_SBOP_EXT.1

TSS Evaluation Activity:

For stack-based OSes, the evaluator will determine that the TSS contains a description of stack-based buffer overflow protections used by the OS. These are referred to by a variety of terms, such as stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner.

For OSes that store parameters/variables separately from control flow values, the evaluator will verify that the TSS describes what data structures control values, parameters, and variables are stored. The evaluator will also ensure that the TSS includes a description of the safeguards that ensure parameters and variables do not intermix with control flow values.

TSS:

The TOE contains a Stack Execution Disable (SED) protection mechanism. SED ensures that code residing on the stack of selected processes cannot be executed by the processes. This mechanism prevents an attacker from adding and executing code on a process' stack via a buffer overflow attack.

SED is configured by an administrator with the **sedmgr** command. Each executable's header contains two flags *select* and *exempt* that inform the SED mechanism to enable SED protection on the executable when SED is only enabled for a limited set of executables (e.g., **sedmgr select** or **setgidfiles**) or to exempt the executable from using SED protection when SED is enabled across all executables (i.e., **sedmgr all**). If a process is configured to deny execution of code on its stack and the process attempts to execute code on its stack, the system will generate an exception and terminate the process. This helps prevent buffer overflow attacks by not allowing attackers to execute arbitrary code on the stack of an executable.

In the evaluated configuration, the TOE is configured with the **sedmgr**'s system-wide **all** flag. This flag enables SED on all executables with no exceptions.

7.2.27 FPT_TST_EXT.1 Boot Integrity

PP: OSPPv4.2.1
SFR Link: FPT_TST_EXT.1

TSS Evaluation Activity:

The evaluator will verify that the TSS section of the ST includes a comprehensive description of the boot procedures, including a description of the entire bootchain, for the TSF. The evaluator will ensure that the OS cryptographically verifies each piece of software it loads in the bootchain to include bootloaders and the kernel. Software loaded for execution directly by the platform (e.g. first-stage bootloaders) is out of scope. For each additional category of executable code verified before execution, the evaluator will verify that the description in the TSS describes how that software is cryptographically verified.

The evaluator will verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

TSS:

The TOE boots in a PowerVM virtual environment using the AIX Secure Boot feature. The AIX Secure Boot feature extends the chain of trust to the AIX LPAR by digitally signing the following AIX code.

- OS boot loader
- Kernel
- Device drivers, including boot device drivers

The hardware (operational environment) contains a burned-in IBM public key used to validate the signature of the virtualization software (PowerVM). PowerVM (operational environment) contains an IBM public key used to validate the signature of the partition firmware (PFW). The PFW (operational environment) contains an IBM public key used to validate the AIX bootloader's signature. The AIX bootloader (TOE) contains an IBM public key used to validate the AIX kernel's signature. The AIX kernel (TOE) contains an IBM public key used to validate the AIX device driver signatures.

The TOE supports multiple Secure Boot policies. Secure Boot policy *2* is used in the evaluated configuration. This policy aborts the boot operation if a signature verification fails.

The AIX bootloader uses the CLiC kernel space cryptographic module to validate the AIX kernel's digital signature. The AIX kernel uses the CLiC kernel space cryptographic module to validate the device driver signatures. These signatures use either RSA 2048-bit or RSA 4096-bit key sizes with the SHA2-256 hash algorithm.

7.2.28 FPT_TUD_EXT.1 Trusted Update

PP: OSPPv4.2.1
SFR Link: FPT_TUD_EXT.1

TSS Evaluation Activity:

None.

TSS:

7.2.28.1 Cumulative updates

The TOE's **suma** command provides the ability to check for and download OS and application cumulative update packages. Once downloaded, the TOE's **installp** command verifies the update package signatures and installs the verified updates.

The **suma** command checks the IBM fix server for updates over an HTTPS client connection. The HTTPS connection uses TLS version 1.2 from IBM Java and the ICC cryptographic module to protect the communication channel. The TLS signature verification algorithm is conformant to FCS_COP.1(3).

The OS and application cumulative update packages are digitally signed using RSA 2048-bit keys and SHA2-256 hashes. The **installp** command verifies each package's signature prior to installing the package based on the *digital signature policy* active at the time of execution. The TOE's **chsignpolicy** command allows an administrator to set the *digital signature policy*.

For the evaluated configuration, the *digital signature policy* is set to *medium* which will cause the **installp** command to verify the signature of the update and install the update if the signature is successfully verified. If the signature verification fails, the command prompts whether to continue with the installation or terminate the installation.

The **installp** command uses the OpenSSL cryptographic module's RSA signature verification algorithm conformant to FCS_COP.1(3).

7.2.28.2 Ifix updates

For ifixes, the TOE's **emgr_check_ifixes** command checks for available ifixes on the IBM fix server over an HTTPS connection. The TOE's **emgr_download_ifix** command downloads ifixes from the IBM fix server over an HTTPS connection. In both cases, these commands use TLS 1.2 via the **openssl s_client** command to protect the connection. Once downloaded, an administrator uses the TOE's **emgr_sec_patch** command which controls the signature verification steps of each ifix package and the installation of the successfully verified ifix packages. The signature verification is performed using the **openssl** command. If the signature verification fails, the ifix package is not installed.

The TOE uses the user space 32-bit OpenSSL implementation for both the HTTPS' TLS client connections and the ifix package signature verification. The OpenSSL's signature verification algorithms are conformant to FCS_COP.1(3).

The OS and application ifix packages are digitally signed using RSA 2048-bit keys and SHA2-256 hashes.

7.2.29 FPT_TUD_EXT.2 Trusted Update for Application Software

PP: OSPPv4.2.1
SFR Link: FPT_TUD_EXT.2

TSS Evaluation Activity:

None.

TSS:

The TSS for FPT_TUD_EXT.1 includes the description of the application software trusted update process.

7.2.30 FPT_W^X_EXT.1 Write XOR Execute Memory Pages

PP: OSPPv4.2.1
SFR Link: FPT_W^X_EXT.1

TSS Evaluation Activity:

None.

TSS:

The TOE prevents allocation of any memory region with both write and execute permissions in the following segments.

- Code segments
- Data, heap, and stack segments (requires SED enabled)

The exception is memory mapped (a.k.a. **mmap**) memory segments.

7.2.31 FTA_TAB.1 Default TOE access banners

PP: OSPPv4.2.1
SFR Link: FTA_TAB.1

TSS Evaluation Activity:

None.

TSS:

The TOE displays an administrator-configurable advisory warning message before a user session is established.

7.2.32 FTP_ITC_EXT.1 Trusted channel communication

PP: OSPPv4.2.1
SFR Link: FTP_ITC_EXT.1

TSS Evaluation Activity:

None.

TSS:

The TOE supports a TLS client protocol implementation (via OpenSSL) allowing an application to initiate and protect communications to remote TLS IT entities. The **suma** command and the ifix commands use the TLS client protocol (via HTTPS) to connect to the IBM fix server.

The TOE supports an SSH client that allows users to initiate a protected connection between themselves and other remote SSH servers.

For both TLS and SSH, the TOE supports public/private key algorithms and certificate validation to provide assured identity of the endpoints.

7.2.33 FTP_TRP.1 Trusted Path

PP: OSPPv4.2.1
SFR Link: FTP_TRP.1

TSS Evaluation Activity:

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

TSS:

The TOE supports an SSH server for inbound remote administration. The remote user initiates the SSH connection using an SSH client. The TOE's SSH server identifies itself using public/private key algorithms. The user of the SSH client identifies itself by authenticating to the TOE. The TOE requires the use of SSH for all remote administrative actions.

8 Abbreviations, Terminology and References

8.1 Abbreviations

AES	Advanced Encryption Standard
AIX	Advanced Interactive eXecutive
API	Application Programming Interface
App	Application
ASLR	Address Space Layout Randomization
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CC	Common Criteria
CCEVS	Common Criteria Evaluation and Validation Scheme
CEM	Common Evaluation Methodology
CLiC	CryptoLite for C
CMC	Certificate Management over CMS
CMS	Cryptographic Message Syntax
CN	Common Name
CRL	Certificate Revocation List
CSP	Critical Security Parameter
CTR	Counter
DAC	Discretionary Access Control
DAR	Data At Rest

darn	Deliver A Random Number
DEP	Data Execution Prevention
DNS	Domain Name System
DRBG	Deterministic Random Bit Generation
DVD	Digital Versatile Disc or Digital Video Disc
ECC	Elliptic Curve Cryptography
ECD	Extended Components Definition
ECDSA	Elliptic Curve Digital Signature Algorithm
EFS	Encrypted File System
EGID	Effective GID
EP	Extended Package
EST	Enrollment over Secure Transport
EUID	Effective UID
FIPS	Federal Information Processing Standards
GB	Gigabyte
GCM	Galois/Counter Mode
GID	Group ID
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure

ICC	IBM Crypto for C
ID	Identity
Ifix	Interim fix
IP	Internet Protocol
IV	Initialization Vector
I/O	Input/Output
JFS2	Journaled File System version 2
KAS	Key Agreement Scheme
KEK	Key Encryption Key
KeyGen	Key Generation
KeyVer	Key Verification
LPAR	Logical Partition
LPP	Licensed Product Package
MAC	Message Authentication Code
MB	Megabyte
OCSP	Online Certificate Status Protocol
OE	Operating Environment
OS	Operating System
PFW	Partition Firmware
PII	Personally Identifiable Information

PIN	Personal Identification Number
PKCS	Public Key Cryptography Standards
POSIX	Portable Operating System Interface
POWER	Performance Optimization With Enhanced RISC
PP	Protection Profile
PSIRT	Product Security Incident Response Team
RBG	Random Bit Generator
RGID	Real GID
RISC	Reduced Instruction Set Computer
RSA	Rivest–Shamir–Adleman
RTOS	Real Time Operating System
RUID	Real UID
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SE	Standard Edition
SED	Stack Execution Disable
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
SMT8	Simultaneous MultiThreading 8-way

SP3	Service Pack 3
SRV	Service
SSC	Shared Secret Computation
SSH	Secure Shell
SSL	Secure Sockets Layer
ST	Security Target
TD	Technical Decision
TL5	Technology Level 5
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification
UID	User ID
URI	Uniform Resource Identifier
VIOS	Virtual I/O System
VM	Virtual Machine
VRM	Version/Release/Modification
XOR	Exclusive Or

XPG
: X/Open Portability Guide

8.2 Terminology

This section contains definitions of technical terms that are used with a meaning specific to this document. Terms defined in the [CC] are not reiterated here, unless stated otherwise.

Administrator
: An administrator is responsible for management activities, including setting policies that are applied by the enterprise on the operating system. This administrator could be acting remotely through a management server, from which the system receives configuration policies. An administrator can enforce settings on the system which cannot be overridden by non-administrator users.

API
: A specification of routines, data structures, object classes, and variables that allows an application to make use of services provided by another software component, such as a library. APIs are often provided for a set of libraries included with the platform.

app
: Software that runs on a platform and performs tasks on behalf of the user or owner of the platform, as well as its supporting documentation.

ASLR
: An anti-exploitation feature which loads memory mappings into unpredictable locations. ASLR makes it more difficult for an attacker to redirect control to code that they have introduced into the address space of a process.

CC
: Common Criteria for Information Technology Security Evaluation.

CEM
: Common Evaluation Methodology for Information Technology Security Evaluation.

Credential
: Data that establishes the identity of a user, e.g. a cryptographic key or password.

CSP
: Information that is either user or system defined and is used to operate a cryptographic module in processing encryption functions including cryptographic keys and authentication data, such as passwords, the disclosure or modification of which can compromise the security of a cryptographic module or the security of the information protected by the module.

DAR Protection
: Countermeasures that prevent attackers, even those with physical access, from extracting data from non-volatile storage. Common techniques include data encryption and wiping.

DEP
: An anti-exploitation feature of modern operating systems executing on modern computer hardware, which enforces a non-execute permission on pages of memory. DEP prevents pages of memory from containing both data and instructions, which makes it more difficult for an attacker to introduce and execute code.

Developer
: An entity that writes OS software. For the purposes of this document, vendors and developers are the same.

EP
: An implementation-independent set of security requirements for a specific subset of products described.

General Purpose Operating System
: A class of OSes designed to support a wide-variety of workloads consisting of many concurrent applications or services. Typical characteristics for OSes in this class include support for third-party applications, support for multiple users, and security separation between users and their respective resources. General Purpose Operating Systems also lack the real-time constraint that defines Real Time Operating Systems (RTOS). RTOSes typically power routers, switches, and embedded devices.

Host-based Firewall
: A software-based firewall implementation running on the OS for filtering inbound and outbound network traffic to and from processes running on the OS.

OS
: Software that manages physical and logical resources and provides services for applications. The terms *TOE* and *OS* are interchangeable in this document.

PII
: Any information about an individual maintained by an agency, including, but not limited to, education, financial transactions, medical history, and criminal or employment history and information which can be used to distinguish or trace an individual's identity, such as their name, social security number, date and place of birth, mother's maiden name, biometric records, etc., including any other personal information which is linked or linkable to an individual.

PP
: An implementation-independent set of security requirements for a category of products.

SAR
: A requirement to assure the security of the TOE.

Sensitive Data
: Sensitive data may include all user or enterprise data or may be specific application data such as PII, emails, messaging, documents, calendar items, and contacts. Sensitive data must minimally include credentials and keys. Sensitive data shall be identified in the OS's TSS by the ST author.

SFR
: A requirement for security enforcement by the TOE.

ST
: A set of implementation-dependent security requirements for a specific product.

TOE
: The product under evaluation.

TSF
: The security functionality of the product under evaluation.

TSS
: A description of how a TOE satisfies the SFRs in a ST.

User

A user is subject to configuration policies applied to the operating system by administrators. On some systems under certain configurations, a normal user can temporarily elevate privileges to that of an administrator. At that time, such a user should be considered an administrator.

8.3 References

AIS20 **Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren (Functionality Classes and Evaluation Methodology for Deterministic RNGs)**

Version	3
Date	2013-05-15
Location	https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_pdf.pdf?__blob=publicationFile

AIS31 **Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren (Functionality Classes and Evaluation Methodology for Physical RNGs)**

Version	3
Date	2013-05-15
Location	https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_pdf.pdf?__blob=publicationFile

CC **Common Criteria for Information Technology Security Evaluation**

Version	3.1R5
Date	April 2017
Location	http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
Location	http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf
Location	http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf

CCEVS-TD0240 **FCS_COP.1.1(1) Platform provided crypto for encryption/decryption**

Date	2017-11-27
Location	https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0240

CCEVS-TD0331 **SSH Rekey Testing**

Date	2018-06-01
Location	https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0331

CCEVS-TD0332 **Support for RSA SHA2 host keys**

Date	2018-06-08
Location	https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0332

CCEVS-TD0365 **FCS_CKM_EXT.4 selections**
Date 2018-10-12
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0365

CCEVS-TD0386 **Platform-Provided Verification of Update**
Date 2019-02-07
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0386

CCEVS-TD0420 **Conflict in FCS_SSHC_EXT.1.1 and FCS_SSHS_EXT.1.1**
Date 2019-05-10
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0420

CCEVS-TD0441 **Updated TLS Ciphersuites for OS PP**
Date 2019-08-21
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0441

CCEVS-TD0446 **Missing selections for SSH**
Date 2019-10-18
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0446

CCEVS-TD0463 **Clarification for FPT_TUD_EXT**
Date 2019-11-12
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0463

CCEVS-TD0493 **X.509v3 certificates when using digital signatures for Boot Integrity**
Date 2020-03-04
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0493

CCEVS-TD0496 **GPOS PP adds allow-with statement for VPN Client V2.1**
Date 2020-01-29
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0496

CCEVS-TD0501 **Cryptographic selections and updates for OS PP**
Date 2020-09-03
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0501

CCEVS-TD0525 **Updates to Certificate Revocation (FIA_X509_EXT.1)**
Date 2020-07-01
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0525

CCEVS-TD0578	**SHA-1 is no longer mandatory**	
	Date	2021-02-12
	Location	https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0578
CCEVS-TD0598	**Expanded AES Modes in FCS_COP for App PP**	
	Date	2021-08-03
	Location	https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0598
CCEVS-TD0600	**Conformance claim sections updated to allow for MOD_VPNC_V2.3**	
	Date	2021-08-10
	Location	https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?TD=0600
FIPS140-2	**Security Requirements for Cryptographic Modules**	
	Date	2002-12-03
	Location	https://csrc.nist.gov/publications/detail/fips/140/2/final
FIPS140-3	**Security Requirements for Cryptographic Modules**	
	Date	2019-03-22
	Location	https://csrc.nist.gov/publications/detail/fips/140/3/final
FIPS180-4	**Secure Hash Standard (SHS)**	
	Date	2015-08-04
	Location	https://csrc.nist.gov/publications/detail/fips/180/4/final
FIPS186-4	**Digital Signature Standard (DSS)**	
	Date	2013-07-19
	Location	https://csrc.nist.gov/publications/detail/fips/186/4/final
FIPS198-1	**The Keyed-Hash Message Authentication Code (HMAC)**	
	Date	2008-07-16
	Location	https://csrc.nist.gov/publications/detail/fips/198/1/final
OpenSshSpec	**OpenSSH Protocol Specification v1.31**	
	Date	None specified
	Location	https://cvsweb.openbsd.org/cgi-bin/cvsweb/src/usr.bin/ssh/PROTOCOL?rev=1.31
OSPPv4.2.1	**Protection Profile for General Purpose Operating Systems Version 4.2.1**	
	Version	4.2.1
	Date	2019-04-22
	Location	https://www.niap-ccevs.org/MMO/PP/PP_OS_V4.2.1.pdf
RFC4252	**The Secure Shell (SSH) Authentication Protocol**	
	Author(s)	T. Ylonen, C. Lonvick
	Date	2006-01-01
	Location	http://www.ietf.org/rfc/rfc4252.txt

RFC4253	**The Secure Shell (SSH) Transport Layer Protocol**	
	Author(s)	T. Ylonen, C. Lonvick
	Date	2006-01-01
	Location	http://www.ietf.org/rfc/rfc4253.txt
RFC5280	**Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**	
	Author(s)	D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk
	Date	2008-05-01
	Location	http://www.ietf.org/rfc/rfc5280.txt
RFC5759	**Suite B Certificate and Certificate Revocation List (CRL) Profile**	
	Author(s)	J. Solinas, L. Zieglar
	Date	2010-01-01
	Location	http://www.ietf.org/rfc/rfc5759.txt
RFC6668	**SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol**	
	Author(s)	D. Bider, M. Baushke
	Date	2012-07-01
	Location	http://www.ietf.org/rfc/rfc6668.txt
RFC6960	**X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP**	
	Author(s)	S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams
	Date	2013-06-01
	Location	http://www.ietf.org/rfc/rfc6960.txt
RFC8017	**PKCS #1: RSA Cryptography Specifications Version 2.2**	
	Author(s)	K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch
	Date	2016-11-01
	Location	http://www.ietf.org/rfc/rfc8017.txt
SP800-38A	**Recommendation for Block Cipher Modes of Operation: Methods and Techniques**	
	Date	2001-12-01
	Location	https://csrc.nist.gov/publications/detail/sp/800-38a/final
SP800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**	
	Date	2007-11-28
	Location	https://csrc.nist.gov/publications/detail/sp/800-38d/final
SP800-56A-Rev3	**Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography**	
	Date	2018-04-16
	Location	https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final

SSHEPv1.0	**Extended Package for Secure Shell (SSH) v1.0**	
	Version	1.0
	Date	2016-02-19
	Location	https://www.niap-ccevs.org/MMO/pp/pp_ssh_ep_v1.0.pdf