LG App Monitoring Security Solution V1.0 for webOS TV Security Target V1.7

Document History

Version	Description of Change	Date
1.0	Initial Version	2022.02.21
1.1	Modified Terms and definition, threats and Security Objectives.	2022.03.21
1.2	Modified the document according to the internal review comments.	2022.04.27
1.3	Specific definition of Native App	2022.05.10
1.4	Clarification of physical of the TOE according to TOE user	2022.05.30
1.5	Add Terms and Definition items	2022.05.31
1.6	Modified TOE Overview	2022.06.08
1.7	Some document form changes	2022.06.21

Contents

1 Introduction of Security Target

1.1 Security Target Reference

This section provides information to refer to the Security Target (ST) as in the following Table. The ST is identified by the ST Title and the ST Version.

Security Target Title	LG App Monitoring Security Solution V1.0 for webOS TV Security Target
Security Target Version	V1.7
Publication Date	06/21/22
Authors	LG Electronics, Inc.
CC Identification	Common Criteria for Information Technology Security (CC Version 3.1 Revision 5)

1.2 TOE Reference

This section provides information to refer to the TOE as in the following Table.

TOE	LG App Monitoring Security Solution V1.0 for webOS TV
Detail Version	WO4S21.r01
Developer	LG Electronics, Inc.

1.3 TOE Overview

This section describes information about its purpose and key security functions of the software that is provided to use apps securely based on the LG Smart TV based on webOS 6.0(hereinafter "webOS").

LG App Monitoring Security Solution V1.0 for webOS TV(hereinafter "TOE") is a Smart TV Security Solution that provides security functions in the form of software by being embedded on "LG Smart TV based on webOS"(hereinafter "Smart TV"). The TOE is used to make a blocking request to webOS when an unauthorized Native Apps are executed to prevent operating Smart TV from performing unintended services offered by Native Apps. Unintended service is a service not provided by authorized Native App included in webOS TV. Whitelist used to verify unauthorized Native App is created by Smart TV Developer. The TOE user is a Smart TV Developer. Smart TV Developer uses TOE in the webOS TV development stage. The scenarios in which the TOE is used are as follows. First, the Smart TV Developer physically includes 'AppMonitoringService-1.0.WO4S21.r01' of the TOE(refer to the first TOE component in Table 3) in the webOS TV so that the webOS TV provides the Native App tamper detection function. In addition, the Smart TV Developer creates the Native App tamper detection rule using 'makeWhitelistRule-1.0.WO4S21.r01' of the TOE (refer to the second TOE component in Table 3). [1] [2] webOS is an LG-owned, Linux-based, Smart TV operating system that is set up to allow control and access of Smart TV's more advanced features and connected devices through a graphical user interface(GUI). webOS TV is a web-centric platform based on webOS and specialized for LG Smart TV.

The TOE provides major security features for the secure operation of Smart TV with Native App Integrity monitoring by comparing the hash value of Native App's ELF, Executable or text file with the Whitelist Rule. Whitelist Rule contain the hash values of Native Apps that are the list of Native

[1] Smart TV Users can use 'Native App Integrity Monitoring' provided by TOE.

[2] The second TOE component in table 3 that creates tamper detection rules for Native App is not physically included in webOS TV.

Apps installed on Smart TV that are allowed to execute.

The major security features are offered by the TOE as follows:

- Native App Integrity Monitoring

  The TOE performs tamper detection based on the Whitelist Rule. The Whitelist Rule contains a list of hash values that are allowed to be executable Native Apps on Smart TV. When executing the Native App, webOS requests the TOE to verify whether the Native App has been tampered with. The TOE requests the webOS(OpenSSL) to generate a hash value for the Native App's ELF, Executable or text file and compares it with the Whitelist Rule. If the comparison results match, the TOE requests the webOS(LSM) to allow the execution of the Native App. If the hash value doesn't match, the TOE requests webOS(LSM) to block the execution of the Native App and generates log comprising information on the Native App that is blocked from execution.

The TOE is delivered to the Smart TV Developers in the form of a software, and is not in charge of all kinds of security functions provided in Smart TV. The TOE provides only the security function defined in the above.

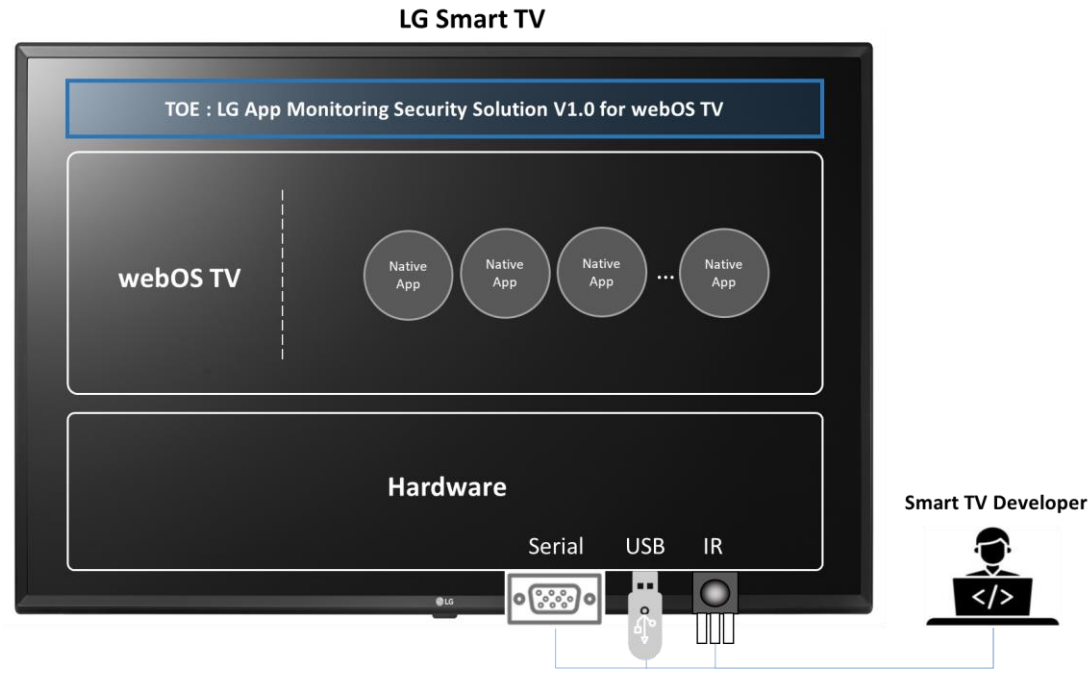

Figure 1. Operational environment for Smart TV Developer of the TOE

Figure 1. shows the operational environment of the TOE. And this environment comprises the hardware in Table 1 and the software in Table 2. The software comprises operating system(webOS 6.0) that the TOE executed on and Native Apps that are integrity monitored by the TOE. Figure 1. is an environment for Smart TV Developers, and Smart TV Users are operated with no serial provided.

Non-TOE hardware and software required by the TOE are as followed.

Table 1. Non-TOE hardware required for Smart TV Developer by the TOE

Category		**Minimum Specifications**
Hardware	CPU	ARM architecture (Cortex A53 Quad) or higher
	DDR Memory	2.0 GB or higher
	Flash Memory	15 MB or higher (Capacity required for TOE installation)
	USB	USB 2.0 X 1
	Serial	RS-232C X 1
	IR Receiver	IR Receiver Module X 1

※ After Smart TV development is completed, Serial is not provided to Smart TV User.

Table 2. Non-TOE software required by the TOE

Software	**Description**
webOS 6.0	Operating system based on Linux Kernel 4.4.84
OpenSSL 1.1.1g	Generates the required hash value when verifying the integrity of the Native app and creating a Whitelist Rule file

※ LS2, LSM, SAL and OpenSSL are included in the operating system(webOS 6.0)

The TOE is a Smart TV Security Solution that provides security functions in the form of software by being embedded on Smart TV. The TOE operates on the Smart TV which is developed based on webOS 6.0 and shall be installed and executed in the Smart TV. The TOE requests the webOS(LSM)

to allow Native Apps that are not tampered(ELF, Executable or text file hash value of Native App is not tampered) and requests the webOS(LSM) to deny Native Apps that are tampered(ELF, Executable or text file hash value of Native App is tampered). The subject that actually allows and blocks execution of Native App is webOS(LSM).

In order for the Smart TV Developer to operate the TOE, the Serial(RS-232), USB and IR Receiver must be supported in the TOE operating environment. Smart TV Developer can execute commands for the TOE and Native Apps installation, create the Whitelist Rule file, and review the TOE reference information. The Smart TV Developer saves the distributed the TOE installation file and Native App installation file (including binaries) to the USB in order to install it on the Smart TV and delivers it to the Smart TV. The Smart TV Developer can power on/off the Smart TV through the IR Receiver and start or stop the webOS including systemd in which the TOE execution daemon is registered.

1.4 TOE Description

1.4.1 Physical Scope of the TOE

The physical scope of the TOE includes software and developer guidance.

The TOE is delivered to Smart TV Developers through a distribution site where only the Smart TV Developers of LG Electronics can access including developer guide. The TOE is installed on webOS TV. The physical scope of the TOE includes only the software that is in charge of Native App Integrity Monitoring security function, however the scope does not include the other security functions in Smart TV.

Figure 2. shows the physical scope of the TOE.

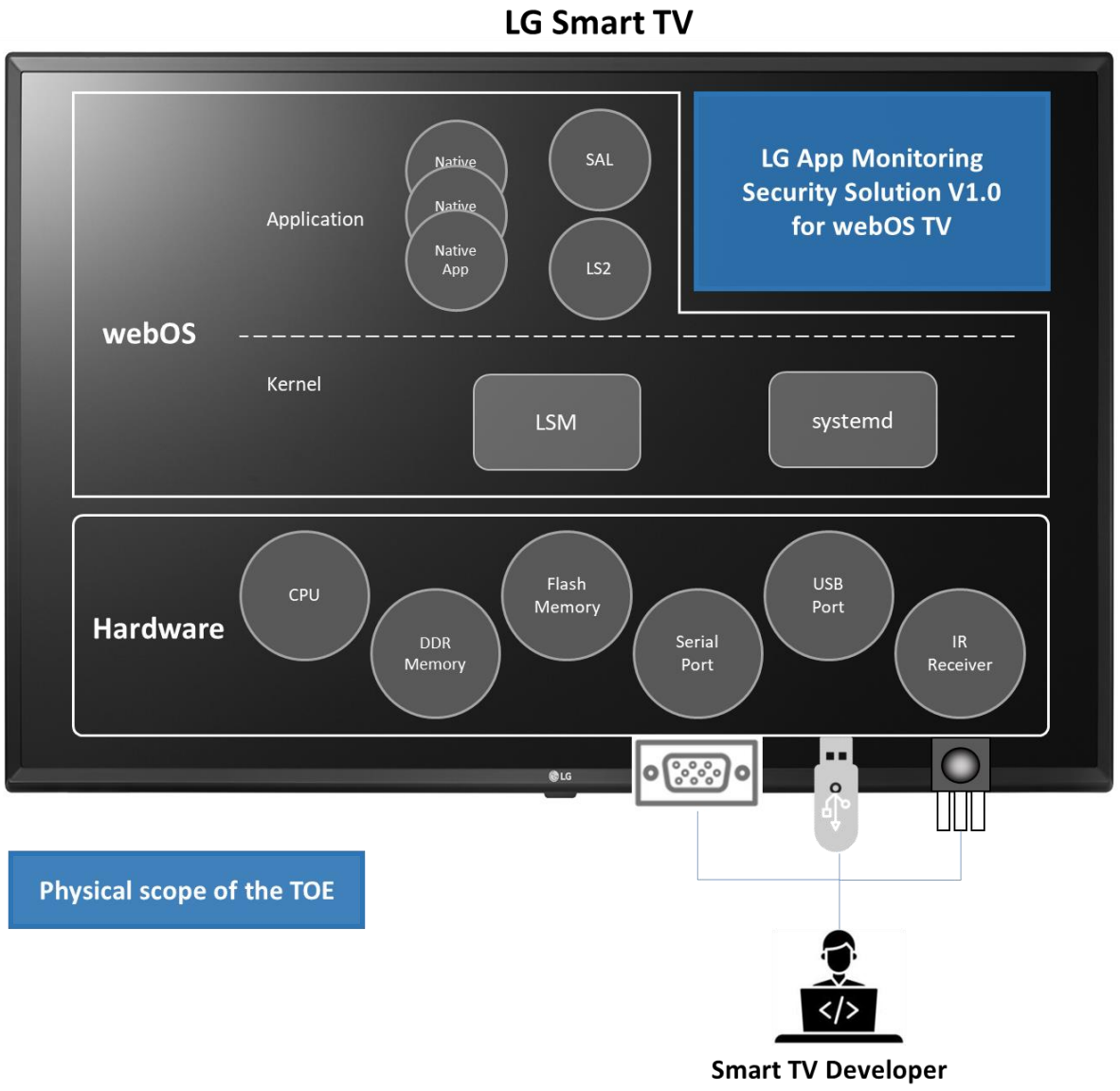

Figure 2. Physical scope of the TOE

Table 3. is the physical scope of the TOE component.

Table 3. Physical scope of the TOE component

TOE Component	Description	Distribution Form	Distribution Method
AppMonitoringService-1.0.WO4S21.r01 (AppMonitoringService-1.0.WO4S21.r01.tar)	Binary and setting files that perform Native App tamper detection	S/W	(Distributed in a file format through a distribution site where only the Smart TV Developers of LG Electronics can access.
makeWhitelistRule-1.0.WO4S21.r01 (makeWhitelistRule-1.0.WO4S21.r01)	Binary to create Whitelist Rule		
LG App Monitoring Security Solution V1.0 for webOS TV Developer guide V1.3 (LG App Monitoring Security Solution V1.0 for webOS TV Developer guide V1.3.pdf)	Developer Guide	Electronic document File (PDF)	

1.4.2 Logical Scope of the TOE

The logical scope of the TOE comprises the security function(Native App Integrity Monitoring function) that are offered by the software included in the physical scope of the TOE.

In order to perform Native App Integrity Monitoring function provided by the TOE accurately, the functions offered by webOS(LSM, LS2, SAL and OpenSSL) should be supported. webOS(LSM) sends the information of the Native App to the TOE to determine whether the Native App is tampered with after changing the executed Native App to a running standby state. webOS(LS2) is responsible for delivering the commands (review the TOE reference information) entered by the Smart TV Developer using Serial to the TOE. webOS(SAL) is responsible for receiving audit logs generated by the TOE and storing them in filesystem. webOS(OpenSSL) receives the request from the TOE and generates a hash value that verifies the integrity of the native app and creates a whitelisting rules

file. When a Power on signal is received via the IR Receiver, systemd is responsible for automatically executing the TOE.

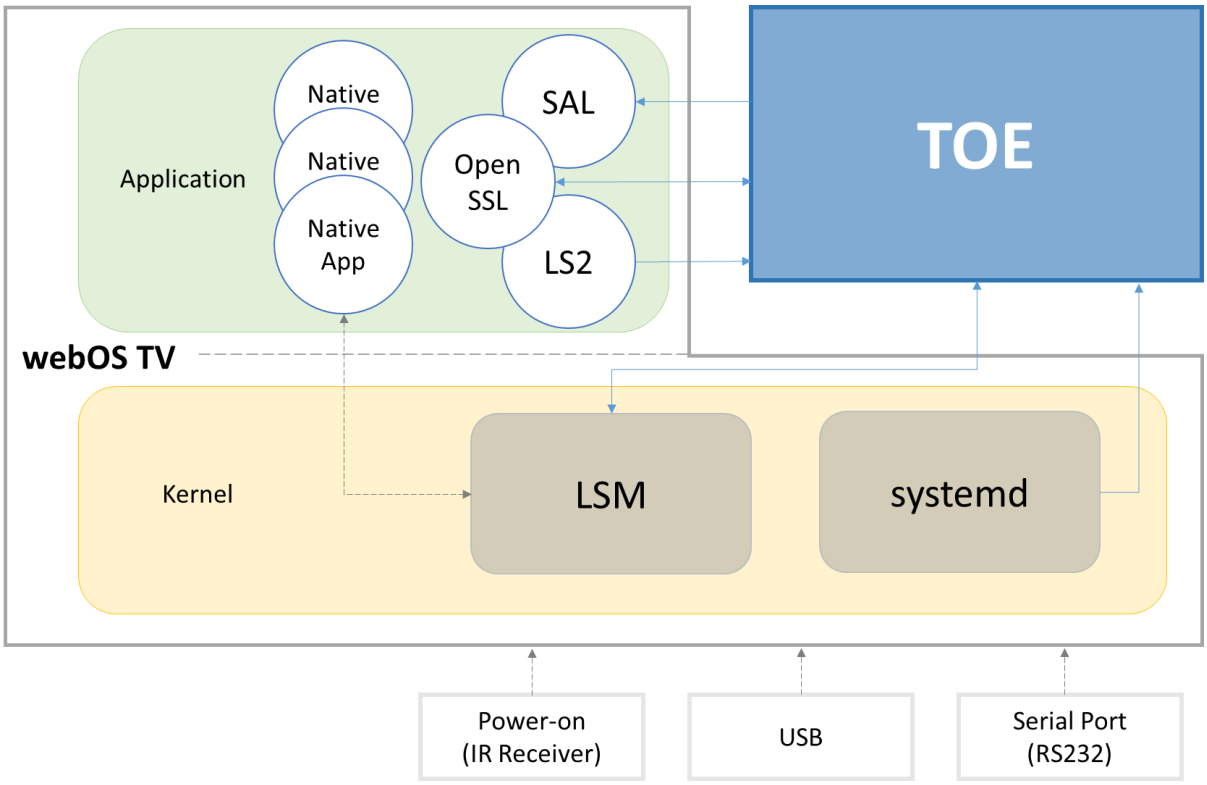

Figure 3. Logical Scope of the TOE

The detailed security functions included in the logical scope of are as follows:

Native App Integrity Monitoring

This function provides Integrity Monitoring for Native App execution. Based on Native App Tamper Detection, this function allows or denies executions of Native Apps. The TOE creates a Whitelist Rule by using webOS(OpenSSL) to generate hash values for ELF, Executable or text files of All Native Apps installed on Smart TV at the time of initial installation of the TOE or installation of the new Native App. The Whitelist Rule includes a list of hash values that are executable Native Apps on Smart TV. When the Native App installed on the Smart TV is executed, webOS(LSM) transmits the Native App information to the TOE and requests integrity verification. The TOE generates a hash value for ELF, Executable or text files of Native Apps using webOS(OpenSSL) and compares them

with the Whitelist Rule. If the comparison results match, the TOE requests the webOS(LSM) to allow the execution of the Native App. If the hash value doesn't match, the TOE requests webOS(LSM) to block the execution of the Native App.

The TOE provides Smart TV Developer with the ability to review the TOE information. LS2 is responsible for delivering the command(review the TOE reference information) entered by the Smart TV Developer using Serial to the TOE. When the Smart TV Developer reviews the TOE information, the TOE returns version information and detailed information.

The TOE can generate security-related logs and write logs to the filesystem using SAL. The logs record a Timestamp, which is dependent on the time of the webOS operating system. This logs comprise information on the Native App that is blocked from execution. Logs generated by the TOE are delivered to webOS(SAL) and stored in the filesystem. In addition, the TOE generates a log of the results of Whitelist Rule file creation and overwrite.

1.5 Conventions

This section describes the conventions used to denote Common Criteria (CC) operations on security functional components and to distinguish text with special meaning. The notation, formatting, and conventions used in this ST are largely consistent with those used in the CC. Four presentation choices are discussed here.

Refinement

The refinement operation is used to add detail to a requirement, and, thus, further restricts a requirement. Refinement of security requirements is denoted by **bold text**.

Selection

The selection operation is used to select one or more options provided by the CC in stating a requirement. Selections are denoted by *underlined italicized text*.

Assignment

The assignment operation is used to assign a specific value to an unspecified parameter such as the length of a password. Showing the value in square brackets [assignment_value] indicates an assignment.

Iteration

Iteration is used when a component is repeated with varying operations. The result of iteration is marked with an iteration number in parenthesis following the component identifier, i.e., denoted as (iteration No.)

Application Note

Application Notes that introduced by the ST author have the detailed additional description.

1.6 Terms and Definition

App

An app refers to apps that can be installed on a Smart TV and perform various functions.

LSM

It refers to Linux Security Module (LSM), and this function is included in webOS. It helps the TOE by hooking an execution of Native App, sending information of the Native App to the TOE, and blocking an execution of tampered Native App.

LS2

LS2 means APIs that webOS service daemons provide based on the Luna-Bus which is a bus system used in webOS. The Luna Bus and APIs are implemented based on the Luna-Service2 library, and are referred to as LS2.

Native App

Native Apps are implemented based on C/C++ optimized for webOS using webOS NDK(Native Development Kit). Native App runs by directly calling the services provided by the kernel and UI framework, so the execution speed is fast and stable. Native Apps contain executable file types (ELF, Executable, text).

SAL

It refers to Security Audit Log, and this function is included in webOS. It receives audit log messages from the TOE, and it stores the audit log in the file system.

Smart TV Developer

Smart TV Developers refer to the developers who implement the services in the Smart TV which interoperate with the TOE during development of the Smart TV's software image. Smart TV Developer performs installation, setting and testing.

Smart TV User

A Smart TV User is a person who can use the Native Service authorized by the Smart TV Developer after development of the smart TV.

systemd

The systemd is an init system that bootstraps userspace instead of the init processor and finally manages all processes.

Tamper Detection

It refers that something can detect the integrity is compromised or not.

Threat agent

Unauthorized users who threaten to access, change, or delete assets illegally

TSF

TOE Security Functionality

web-centric platform

A web app refers to application software that can be used in a web browser and refers to a platform on which the web app can be run.

webOS

webOS is an LG-owned, Linux-based, Smart TV operating system that is set up to allow control and access of Smart TV's more advanced features and connected devices through a graphical user interface(GUI). webOS includes LSM, LS2, SAL and OpenSSL functions and includes Native Apps essential for webOS.

webOS TV

webOS TV is a web-centric platform based on webOS and specialized for LG Smart TV. TOE is installed and operated on webOS TV.

Whitelist Rule

Whitelist Rule indicates a hash value of Native App. The Whitelist Rules are stored in a file called "whitelist.rule", which is located in file system.

2 Conformance Claims

This chapter describes how the Security Target conforms to the Common Criteria, Protection Profile and Package.

2.1 CC Conformance Claim

This Security Target conforms to the following Common Criteria.:

Common Criteria Identification

- Common Criteria for information Technology Security Evaluation, Part1: Introduction and general model, April 2017, Version 3.1 Revision 5, CCMB-2017-04-001
- Common Criteria for Information Technology Security Evaluation, Part2: Security Functional components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-002
- Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-003

Common Criteria Conformance

- Common Criteria for Information Technology Security Evaluation, Part 2 conformant
- Common Criteria for Information Technology Security Evaluation, Part 3 conformant.

2.2 Protection Profiles Claim

This Security Target does not claim conformance to PP.

2.3 Packages Claim

This Security Target claims conformance to assurance package EAL2.

2.4 Conformance Rationale

Since this Security Target does not declare that it conforms to other Protection Profiles, the conformance rationale is not provided.

3 Security Problem Definition

This chapter defines the threats, OSPs (Organizational Security Policies) and assumptions which are intended to be addressed by the TOE and its operational environment.

The assets covered in the Smart TV are as follows.

Assets in the webOS Smart TV

- Native App

Native Apps are implemented based on C/C++ optimized for webOS using webOS NDK(Native Development Kit). Native App runs by directly calling the services provided by the kernel and UI framework, so the execution speed is fast and stable. Native Apps contain executable file types (ELF, Executable, text).

3.1 Threats

The threat agents are IT entities and users that adversely act on the assets through unauthorized access or using unusual methods, and may cause the following threats: The threat agents to the TOE have Basic attack potential of expertise, resources, and motivation.

T.UnauthorizedNativeAppExecution

Threat agents can attempt malicious actions such as tampering with Native Apps.

3.2 Organizational Security Policies

The following describes the organizational security policies, which will be performed by the TOE, the TOE operational environment, or both.

P.AuditLog

Security-related events should be recorded to trace responsibility for security-related actions.

P.SecurityManagement

Smart TV Developers can review reference information or create Whitelist Rules for safe use of Native App Integrity Monitoring functions.

3.3 Assumptions

The following describes the assumptions that are made on the operational environment to provide security functionality.

A.TrustedDeveloper

The TOE Developer does not have any malicious purposes, has been properly trained for use of the TOE, and performs its obligations adequately in accordance with the developer guidance. In addition, Smart TV Developers who develop services in smart TVs that interoperate with the TOE should not implement to include any malicious behaviors intentionally in the LG Smart TV services

A.webOSFunctionSupport

The TOE uses webOS 6.0 function. webOS 6.0 support LS2, SAL and LSM. LS2 is responsible for delivering the commands (review the TOE reference information) entered by the Smart TV Developer. SAL is the ability to store security logs in the filesystem. LSM is a function that hooks information (Executed Native App ELF, Executable or text file path and PID) of Native App when Native App is executed. In addition, it receives and executes requests to allow and block Native App execution through the TOE.

4 Security Objectives

This Security Target classifies security objectives into 2 groups: security objectives for the TOE and security objectives for the operational environment. The security objectives for the TOE are those that are directly handled by the TOE, and the security objectives for the operational environment are those that must be addressed through the technical and procedural measures which are supported by the operational environment for the TOE to provide security functionality.

4.1 Security Objectives for the TOE

The following are security objectives that should be directly handled by the TOE.

O.NativeAppIntegrityMonitoring

When a Native App is running, the TOE monitors the integrity of executed Native App to make a request to webOS to block unauthorized Native App.

O.AuditLog

The TOE accurately generates security-related events to trace responsibility for security-related actions.

O.SecurityManagement

TOE provides Smart TV Developers with the ability to review the TOE identification information. And the TOE provides a function to create a Whitelist Rule file.

4.2 Security Objectives for the Operation Environment

This section describes security objectives which should be addressed by the nontechnical/procedural measures that are supported by the operational environment for the TOE to accurately provide security functionality.

OE.TrustedDeveloper

The TOE developer does not have any malicious purposes, has been properly trained for use of the TOE, and performs its obligations adequately in accordance with the developer guidance. In addition, Smart TV Developers who develop services in smart TVs that interoperate with the TOE should not implement to include any malicious behaviors intentionally in the LG Smart TV services

OE.webOSFunctionSupport

The TOE must uses webOS 6.0 function. webOS 6.0 must support LS2, SAL and LSM. LS2 is responsible for delivering the commands (review the TOE reference information) entered by the Smart TV Developer. SAL is the ability to store security logs in the filesystem. LSM is a function that hooks information (Executed Native App ELF, Executable or text file path and PID) of Native App when Native App is executed. In addition, it receives and executes requests to allow and block Native App execution through the TOE.

OE.TimeStamp

The TOE must accurately generate security-related events using the reliable timestamp provided by the TOE operating environment.

4.3 Security Objectives Rationale

The security objectives rationale demonstrates that the specified security objectives are adequate, sufficient to address security problems, not too excessive, and must be required.

The security objectives rationale demonstrates the following:

- Each threat, organizational security policies, and assumption is addressed by at least one security objective.

- Each security objective addresses at least one threat, organizational security policies, or assumption.

Table 4. Security problems and Security Objectives

	TOE Security Objectives			**Security Objectives for Operational Environment**		
	O.NativeAppIntegrityMonitoring	O.AuditLog	O.SecurityManagement	OE.TrustedDeveloper	OE.webOSFunctionSupport	OE.TimeStamp
T.UnauthorizedNativeAppExecution	X					
P.AuditLog		X				X
P.SecurityManagement			X			
A.TrustedDeveloper				X		
A.webOSFunctionSupport					X	

4.3.1 The Security Objectives Rationale of the TOE

O.NativeAppIntegrityMonitoring

This security objective ensures that to block the execution of unauthorized Native App. Therefore, this security objective is necessary to addresses the threat T.UnauthorizedNativeAppExecution.

O.AuditLog

This security objective generates audit log accurately so that the TOE can trace responsibility for security-related events. Therefore, this security objective is necessary to carry out the organization's security policy P.AuditLog.

O.SecurityManagement

This security objective is necessary for the organization's security policy P. SecurityManagement, as the TOE provides Smart TV Developer with a means to safely manage the TOE.

4.3.2 The Security Objectives Rationale of the Operational Environment

OE.TrustedDeveloper

This security objective for operational environment supports the assumption A.TrustedDeveloper by executing the following. The TOE Developer does not have any malicious intentions, has been properly trained for use of the TOE, and performs its obligations adequately in accordance with the developer guidance. In addition, Smart TV Developers who develop services in LG smart TVs that interoperate with the TOE should not implement to include any malicious behaviors intentionally in the LG Smart TV services.

OE.webOSFunctionSupport

This security objective for the operating environment is necessary to support the assumption A.webOSFunctionSupport because it ensures that the TOE operates the security functions safely and reliably by supporting LS2, LSM and SAL functions in webOS 6.0 where the TOE is installed.

OE.TimeStamp

This security objective for the operating environment ensures that TOE accurately records security-related events using the trusted timestamp provided by the TOE operating environment, so it is necessary to execute the organization's security policy P.AuditLog.

5 Extended Components Definition

This Security Target does not include any extended components that are extended from the Common Criteria (CC) Part 2 or Part 3.

6 Security Requirements

This chapter describes security functional requirements and security assurance requirements which should be satisfied in the TOE.

6.1 Security Functional Requirements

The security functional requirements defined in this Security Target are based on the functional requirements in Part 2 of the Common Criteria.

Following table shows the security functional requirements that are used in this Security Target document.

Table 5. Security Functional Requirements

Security Functional Class	Security Functional Components	
Security Audit	FAU_GEN.1	Audit data generation
User Data Protection	FDP_SDI.2	Stored data integrity monitoring and action
Security Management	FMT_MTD.1	Management of TSF data
	FMT_SMF.1	Specification of Management Functions
	FMT_SMR.1	Security roles

6.1.1 Security audit

FAU_GEN.1 Audit data generation

Hierarchical to: No other components.

Dependencies: FPT_STM.1 Reliable time stamps

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

a) Start-up and shutdown of the audit functions;

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

c) [Request blocking of Native App execution, creating/overwriting of Whitelist Rule file].

FAU_GEN.1.2 The TSF 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 the 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, [None].

6.1.2 User data protection

FDP_SDI.2 Stored data integrity monitoring and action

Hierarchical to: FDP_SDI.1 Stored data integrity monitoring

Dependencies: No dependencies.

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for [Mismatching a hash value of executed Native App and Whitelist Rule] on all objects, based on the following attributes: [A hash value of executed Native App].

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall [Requests blocking the execution of tampered Native App to webOS(LSM)].

Application Note : Whitelist Rule refers to the hash value generated using webOS(OpenSSL)'s SHA256 algorithm) for the ELF, Executable or text files of Native Apps that are allowed to execute on Smart TV. A hash value of executed Native App refers to the hash value that webOS(OpenSSL) generates upon request by the TOE for the ELF, Executable or text file of that Native App at the time the Native App requests execution. The TOE delivers integrity verification results to webOS(LSM), and allowing or blocking Native Apps to be performed by webOS(LSM).

6.1.3 Security management

FMT_MTD.1 Management of TSF data

Hierarchical to: No other components.

Dependencies: FMT_SMR.1 Security roles

FMT_SMF.1 Specification of Management Functions

FMT_MTD.1.1 The TSF shall restrict the ability to *query, [create, overwrite]* the [TOE reference information, Whitelist Rule file] to [Authorized Roles in the following table].

Operation	TSF Data	Authorized Roles
create, overwrite	Whitelist Rule file	Smart TV Developer
query	TOE reference information	

Application Note : The TOE creates Whitelist Rule file at the time of the initial installation of the TOE and overwrites when the new Native App is installed.

FMT_SMF.1 Specification of Management Functions

Hierarchical to: No other components.

Dependencies: No dependencies.

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions: [query the TOE reference information, create/overwrite Whitelist Rule file].

FMT_SMR.1 Security roles

Hierarchical to: No other components.

Dependencies: FIA_UID.1 Timing of identification

FMT_SMR.1.1 The TSF shall maintain the roles [Smart TV Developer].

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

6.2 Security Assurance Requirements

Security assurance requirements (SAR) defined in this document consists of assurance component in Common Criteria for Information Technology Security Evaluation, Part 3. The Evaluation Assurance Levels (EALs) is EAL2. The following table shows the summary of the security assurance requirements.

Table 6. Security Assurance Requirements

Assurance Class	Assurance Components
ADV: Development	ADV_ARC.1 Security architecture description
	ADV_FSP.2 Security-enforcing functional specification
	ADV_TDS.1 Basic design
AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.2 Use of a CM system
	ALC_CMS.2 Parts of the TOE CM coverage
	ALC_DEL.1 Delivery procedures
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims
	ASE_ECD.1 Extended components definition
	ASE_INT.1 ST introduction
	ASE_OBJ.2 Security objectives
	ASE_REQ.2 Derived security requirements
	ASE_SPD.1 Security problem definition
	ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_COV.1 Evidence of coverage
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.2 Vulnerability analysis

6.2.1 Development

ADV_ARC.1 Security architecture description

Dependencies: ADV_FSP.1 Basic functional specification

ADV_TDS.1 Basic design

Developer action elements:

ADV_ARC.1.1D The developer shall design and implement the TOE so that the security features of the TSF cannot be bypassed.

ADV_ARC.1.2D The developer shall design and implement the TSF so that it is able to protect itself from tampering by untrusted active entities.

ADV_ARC.1.3D The developer shall provide a security architecture description of the TSF.

Content and presentation elements:

ADV_ARC.1.1C The security architecture description shall be at a level of detail commensurate with the description of the SFR-enforcing abstractions described in the TOE design document.

ADV_ARC.1.2C The security architecture description shall describe the security domains maintained by the TSF consistently with the SFRs.

ADV_ARC.1.3C The security architecture description shall describe how the TSF initialisation process is secure.

ADV_ARC.1.4C The security architecture description shall demonstrate that the TSF protects itself from tampering.

ADV_ARC.1.5C The security architecture description shall demonstrate that the TSF prevents bypass of the SFR-enforcing functionality.

Evaluator action elements:

ADV_ARC.1.1E The evaluator shall confirm that the information provided meets all requirements

for content and presentation of evidence.

ADV_FSP.2 Security-enforcing functional specification

Dependencies: ADV_TDS.1 Basic design

Developer action elements:

ADV_FSP.2.1D The developer shall provide a functional specification.

ADV_FSP.2.2D The developer shall provide a tracing from the functional specification to the SFRs.

Content and presentation elements:

ADV_FSP.2.1C The functional specification shall completely represent the TSF.

ADV_FSP.2.2C The functional specification shall describe the purpose and method of use for all TSFI.

ADV_FSP.2.3C The functional specification shall identify and describe all parameters associated with each TSFI.

ADV_FSP.2.4C For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions associated with the TSFI.

ADV_FSP.2.5C For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting from processing associated with the SFR-enforcing actions.

ADV_FSP.2.6C The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Evaluator action elements:

ADV_FSP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.2.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

ADV_TDS.1 Basic design

Dependencies: ADV_FSP.2 Security-enforcing functional specification

Developer action elements:

ADV_TDS.1.1D The developer shall provide the design of the TOE.

ADV_TDS.1.2D The developer shall provide a mapping from the TSFI of the functional specification to the lowest level of decomposition available in the TOE design.

Content and presentation elements:

ADV_TDS.1.1C The design shall describe the structure of the TOE in terms of subsystems.

ADV_TDS.1.2C The design shall identify all subsystems of the TSF.

ADV_TDS.1.3C The design shall provide the behaviour summary of each SFR-supporting or SFR-non-interfering TSF subsystem.

ADV_TDS.1.4C The design shall summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.

ADV_TDS.1.5C The design shall provide a description of the interactions among SFR-enforcing subsystems of the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.

ADV_TDS.1.6C The mapping shall demonstrate that all TSFIs trace to the behavior described in the TOE design that they invoke.

Evaluator action elements:

ADV_TDS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_TDS.1.2E The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.

6.2.2 Guidance documents

AGD_OPE.1 Operational user guidance

Dependencies: ADV_FSP.1 Basic functional specification

Developer action elements:

AGD_OPE.1.1D The developer shall provide operational user guidance.

Content and presentation elements:

AGD_OPE.1.1C The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2C The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3C The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4C The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5C The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AGD_OPE.1.6C The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfil the security objectives for the operational

environment as described in the ST.

AGD_OPE.1.7C The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence

AGD_PRE.1 Preparative procedures

Dependencies: No dependencies.

Developer action elements:

AGD_PRE.1.1D The developer shall provide the TOE including its preparative procedures.

Content and presentation elements:

AGD_PRE.1.1C The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2C The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_PRE.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2E The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

6.2.3 Life-cycle support

ALC_CMC.2 Use of a CM system

Dependencies: ALC_CMS.1 TOE CM coverage

Developer action elements:

ALC_CMC.2.1D The developer shall provide the TOE and a reference for the TOE.

ALC_CMC.2.2D The developer shall provide the CM documentation.

ALC_CMC.2.3D The developer shall use a CM system.

Content and presentation elements:

ALC_CMC.2.1C The TOE shall be labelled with its unique reference.

ALC_CMC.2.2C The CM documentation shall describe the method used to uniquely identify the configuration items.

ALC_CMC.2.3C The CM system shall uniquely identify all configuration items.

Evaluator action elements:

ALC_CMC.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_CMS.2 Parts of the TOE CM coverage

Dependencies: No dependencies.

Developer action elements:

ALC_CMS.2.1D The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.2.1C The configuration list shall include the following: the TOE itself; the evaluation evidence required by the SARs; and the parts that comprise the TOE.

ALC_CMS.2.2C The configuration list shall uniquely identify the configuration items.

ALC_CMS.2.3C For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

Evaluator action elements:

ALC_CMS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DEL.1 Delivery procedures

Dependencies: No dependencies.

Developer action elements:

ALC_DEL.1.1D The developer shall document and provide procedures for delivery of the TOE or parts of it to the consumer.

ALC_DEL.1.2D The developer shall use the delivery procedures.

Content and presentation elements:

ALC_DEL.1.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to the consumer.

Evaluator action elements:

ALC_DEL.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.2.4 Security Target evaluation

ASE_INT.1 ST introduction

Dependencies: No dependencies.

Developer action elements:

ASE_INT.1.1D The developer shall provide an ST introduction.

Content and presentation elements:

ASE_INT.1.1C The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.

ASE_INT.1.2C The ST reference shall uniquely identify the ST.

ASE_INT.1.3C The TOE reference shall uniquely identify the TOE.

ASE_INT.1.4C The TOE overview shall summarise the usage and major security features of the TOE.

ASE_INT.1.5C The TOE overview shall identify the TOE type.

ASE_INT.1.6C The TOE overview shall identify any non-TOE hardware/software/firmware required by the TOE.

ASE_INT.1.7C The TOE description shall describe the physical scope of the TOE.

ASE_INT.1.8C The TOE description shall describe the logical scope of the TOE.

Evaluator action elements:

ASE_INT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_INT.1.2E The evaluator shall confirm that the TOE reference, the TOE overview, and the TOE description are consistent with each other.

ASE_CCL.1 Conformance claims

Dependencies: ASE_INT.1 ST introduction

ASE_ECD.1 Extended components definition

ASE_REQ.1 Stated security requirements

Developer action elements:

ASE_CCL.1.1D The developer shall provide a conformance claim.

ASE_CCL.1.2D The developer shall provide a conformance claim rationale.

Content and presentation elements:

ASE_CCL.1.1C The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.

ASE_CCL.1.2C The CC conformance claim shall describe the conformance of the ST to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.

ASE_CCL.1.3C The CC conformance claim shall describe the conformance of the ST to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.

ASE_CCL.1.4C The CC conformance claim shall be consistent with the extended components definition.

ASE_CCL.1.5C The conformance claim shall identify all PPs and security requirement packages to which the ST claims conformance.

ASE_CCL.1.6C The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.

ASE_CCL.1.7C The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.

ASE_CCL.1.8C The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.

ASE_CCL.1.9C The conformance claim rationale shall demonstrate that the statement of security

objectives is consistent with the statement of security objectives in the PPs for which conformance is being claimed.

ASE_CCL.1.10C The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.

Evaluator action elements:

ASE_CCL.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_SPD.1 Security problem definition

Dependencies: No dependencies.

Developer action elements:

ASE_SPD.1.1D The developer shall provide a security problem definition.

Content and presentation elements:

ASE_SPD.1.1C The security problem definition shall describe the threats.

ASE_SPD.1.2C All threats shall be described in terms of a threat agent, an asset, and an adverse action.

ASE_SPD.1.3C The security problem definition shall describe the OSPs.

ASE_SPD.1.4C The security problem definition shall describe the assumptions about the operational environment of the TOE.

Evaluator action elements:

ASE_SPD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_OBJ.2 Security objectives

Dependencies: ASE_SPD.1 Security problem definition

Developer action elements:

ASE_OBJ.2.1D The developer shall provide a statement of security objectives.

ASE_OBJ.2.2D The developer shall provide a security objectives rationale.

Content and presentation elements:

ASE_OBJ.2.1C The statement of security objectives shall describe the security objectives for the TOE and the security objectives for the operational environment.

ASE_OBJ.2.2C The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective.

ASE_OBJ.2.3C The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.

ASE_OBJ.2.4C The security objectives rationale shall demonstrate that the security objectives counter all threats.

ASE_OBJ.2.5C The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.

ASE_OBJ.2.6C The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Evaluator action elements:

ASE_OBJ.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_ECD.1 Extended components definition

Dependencies: No dependencies.

Developer action elements:

ASE_ECD.1.1D The developer shall provide a statement of security requirements.

ASE_ECD.1.2D The developer shall provide an extended components definition.

Content and presentation elements:

ASE_ECD.1.1C The statement of security requirements shall identify all extended security requirements.

ASE_ECD.1.2C The extended components definition shall define an extended component for each extended security requirement.

ASE_ECD.1.3C The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.

ASE_ECD.1.4C The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.

ASE_ECD.1.5C The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements can be demonstrated.

Evaluator action elements:

ASE_ECD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_ECD.1.2E The evaluator shall confirm that no extended component can be clearly expressed using existing components.

ASE_REQ.2 Derived security requirements

Dependencies: ASE_OBJ.2 Security objectives

ASE_ECD.1 Extended components definition

Developer action elements:

ASE_REQ.2.1D The developer shall provide a statement of security requirements.

ASE_REQ.2.2D The developer shall provide a security requirements rationale.

Content and presentation elements:

ASE_REQ.2.1C The statement of security requirements shall describe the SFRs and the SARs.

ASE_REQ.2.2C All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.

ASE_REQ.2.3C The statement of security requirements shall identify all operations on the security requirements.

ASE_REQ.2.4C All operations shall be performed correctly.

ASE_REQ.2.5C Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.

ASE_REQ.2.6C The security requirements rationale shall trace each SFR back to the security objectives for the TOE.

ASE_REQ.2.7C The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.

ASE_REQ.2.8C The security requirements rationale shall explain why the SARs were chosen.

ASE_REQ.2.9C The statement of security requirements shall be internally consistent.

Evaluator action elements:

ASE_REQ.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_TSS.1 TOE summary specification

Dependencies: ASE_INT.1 ST introduction

ASE_REQ.1 Stated security requirements

ADV_FSP.1 Basic functional specification

Developer action elements:

ASE_TSS.1.1D The developer shall provide a TOE summary specification.

Content and presentation elements:

ASE_TSS.1.1C The TOE summary specification shall describe how the TOE meets each SFR.

Evaluator action elements:

ASE_TSS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ASE_TSS.1.2E The evaluator shall confirm that the TOE summary specification is consistent with the TOE overview and the TOE description.

6.2.5 Tests

ATE_COV.1 Evidence of coverage

Dependencies: ADV_FSP.2 Security-enforcing functional specification

ATE_FUN.1 Functional testing

Developer action elements:

ATE_COV.1.1D The developer shall provide evidence of the test coverage.

Content and presentation elements:

ATE_COV.1.1C The evidence of the test coverage shall show the correspondence between the tests in the test documentation and the TSFIs in the functional specification.

Evaluator action elements:

ATE_COV.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_FUN.1 Functional testing

Dependencies: ATE_COV.1 Evidence of coverage

Developer action elements:

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

Content and presentation elements:

ATE_FUN.1.1C The test documentation shall consist of test plans, expected test results and actual test results.

ATE_FUN.1.2C The test plans shall identify the tests to be performed and describe the scenarios for performing each test. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.1.3C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.1.4C The actual test results shall be consistent with the expected test results.

Evaluator action elements:

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2 Independent testing - sample

Dependencies: ADV_FSP.2 Security-enforcing functional specification

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

ATE_COV.1 Evidence of coverage

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2.1D The developer shall provide the TOE for testing.

Content and presentation elements:

ATE_IND.2.1C The TOE shall be suitable for testing.

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Evaluator action elements:

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements

for content and presentation of evidence

ATE_IND.2.2E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

ATE_IND.2.3E The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified

6.2.6 Vulnerability assessment

AVA_VAN.2 Vulnerability analysis

Dependencies: ADV_ARC.1 Security architecture description

ADV_FSP.2 Security-enforcing functional specification

ADV_TDS.1 Basic design

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

Developer action elements:

AVA_VAN.2.1D The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN.2.1C The TOE shall be suitable for testing.

Evaluator action elements:

AVA_VAN.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.2.2E The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.2.3E The evaluator shall perform an independent vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design and security architecture description to identify potential vulnerabilities in the TOE.

AVA_VAN.2.4E The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6.3 Security Requirements Rationale

Security Requirements Rationale demonstrates that the described security requirements are suitable to satisfy security objectives and, as a result, appropriate to address security problems.

6.3.1 Security Functional Requirements Rationale

Rationale of security functional requirements demonstrates the followings. Each TOE security objective has at least one security functional requirement tracing to it. Each TOE security functional requirement traces back to at least one TOE security objectives.

Table 7. Summary of Mappings between Security Objectives and Security Functional

Security Objectives / Security Functional Requirements	O.NativeApp Integrity Monitoring	O.AuditLog	O.Security Management
FAU_GEN.1		X	
FDP_SDI.2	X		
FMT_MTD.1			X
FMT_SMF.1			X
FMT_SMR.1			X

FAU_GEN.1 Audit data generation

This component satisfies the TOE security objective O.AuditLog because the TOE generates audit log.

FDP_SDI.2 **Stored data integrity monitoring and action (Native App Integrity)**

This component satisfies the TOE security objective O.NativeAppIntegrityMonitoring because it monitors hash value of Native App and check integrity of the hash value whether it is tampered or not. If the hash value is tampered, then it blocks the execution of the Native App.

FMT_MTD.1 **Management of TSF data**

This component satisfies the TOE security objective O.SecurityManagement because the TOE provides review the Smart TV Developer with the ability to review the TOE reference information. Also, the TOE provides the ability to create Whitelist Rule file.

FMT_SMF.1 **Specification of Management Functions**

This component satisfies the TOE security objective O.SecurityManagement because the TOE provides review the Smart TV Developer with the ability to review the TOE reference information. Also, the TOE provides the ability to create Whitelist Rule file.

FMT_SMR.1 **Security roles**

This component satisfies the TOE security objective O.SecurityManagement because Smart TV Developer is responsible for managing security.

6.3.2 Security Assurance Requirements Rationale

The security assurance level of this Security Target was selected as EAL2 in consideration of the

value and threat level of the assets protected by the TOE.

EAL2 requires only the effort to submit design information and test results to the extent that developers are dealing with robust commercial methodologies.

EAL2 is applicable when all development records are not readily available and when developers or users need low to medium levels of independently assured security.

EAL2 provides assurance by analyzing the security functional requirements contained in the complete ST using the functional and interface specifications, documentation, and basic descriptions of the TOE structure to understand the security behavior. This analysis demonstrates the independent testing of the TSF, the proof of the tests performed by the developer based on the functional specification, the independent verification of the developer's sample of the test results, and the immunity from the attacker's penetration attack with the basic attack potential (Based on the provided functional specification, The TOE design, structural design, and documentation evidence).

EAL2 provides assurance through the proof of configuration management system and secure distribution procedures.

In addition, in order to respond to vulnerability analysis attacks, AVA_VAN.2, vulnerability analysis is requested, and the penetration test is performed by the evaluator to confirm that the potential vulnerability cannot be exploited in the TOE operational environment. The evaluator assumes basic attack potential and performs penetration testing.

6.3.3 Dependency Rationale

Rationale of dependency is demonstrated by dependency of security functional requirements and dependency of security assurance requirements.

The following table shows dependencies of security functional requirements.

Table 8. Dependencies on the TOE Security Functional Components

No.	Functional Components	Dependencies	Reference No.
1	FAU_GEN.1	FPT_STM.1 Reliable time stamps	Rationale (OE.TimeStamp)
2	FDP_SDI.2	N/A	N/A
3	FMT_MTD.1	FMT_SMR.1 Security roles FMT_SMF.1 Specification of Management Functions	4, 5
4	FMT_SMF.1	N/A	N/A
5	FMT_SMR.1	FIA_UID.1 Timing of identification	Rationale

FAU_GEN.1 has a dependency on FPT_STM.1 but the reliable time stamp is supported by OE.TimeStamp, so a dependency is satisfied.

FMT_SMR.1 has a dependency on FIA_UID.1 Smart TV embedded with security function of the TOE is generally the possession of the individual, and all the rights(right of usage, right of development) are given to individual users, and thus identification or authentication is not provided.

In addition, the dependency of each assurance package provided in the CC has already satisfied.

7 TOE Summary Specification

This section summarizes security functions provided by the TOE in terms of how they satisfy the security functional requirements. Following sections describe how each TOE security function satisfies the security functional requirements which are described in the section 6 Security Requirements.

7.1 Native App Integrity Monitoring

TOE performs Native App Integrity Monitoring for Native App execution based on Native App Tamper Detection, this function allows or denies executions of Native Apps.

The TOE loads Whitelist Rule into Smart TV memory and requests the webOS(LSM) to hook information about the executed Native Apps that are installed in webOS TV.

When the Native App is executed, the webOS(LSM) changes to the standby state and delivers information about the Native App to the TOE. The hooked information is the ELF, Executable or text file path of the executed Native App. The TOE uses webOS(Oepnssl)'s SHA256 encryption algorithm to generate hash values of the executed Native App ELF, Executable or text files. After that, it is compared with the hash value in the Whitelist Rule. If the comparison results match, the TOE requests the webOS(LSM) to allow the execution of the Native App. If the hash value doesn't match, the TOE requests webOS(LSM) to block the execution of the Native App. The target of Native App monitoring is all files whose file types include ELF, Executable and text. Typically, these types of files are Native App binary and script.

webOS(LSM) changes the Native App from the standby state to the running state when the TOE requests allow the execution of the Native App, and changes the Native App from the standby state to the destroy state when the TOE requests to block the execution of the Native App.

Related SFRs: FDP_SDI.2

The TOE provides the Smart TV Developer with the ability to review the TOE reference information. Also, the TOE provides the ability to create Whitelist Rule file. The Smart TV Developer can send the review the TOE reference information command to the TOE through webOS(LS2) When the Smart TV Developer reviews the TOE reference information, the TOE returns reference information(TOE Name and Version). Smart TV Developer can create Whitelist Rule file. The TOE creates the Whitelist Rule that is a hash of value generated using the webOS(OpenSSL)'s SHA256 algorithm by searching all Native Apps installed in webOS TV. The target of Native App monitoring is all files whose file types include ELF, Executable and text. Typically, these types of files are Native App binary and script.

Related SFRs: FMT_MTD.1, FMT_SMF.1, SMT_SMR.1

TOE can generate security-related logs. The generated log includes requesting the LSM to block the tampered Native App execution. The log records a Timestamp, which is dependent on the time of the webOS operating system. Also the result of creating a Whitelist Rule file is also created as a log by the TOE. Among security-related logs, block the tempered Native App execution logs are stored in the filesystem through webOS(SAL).

Related SFRs: FAU_GEN.1