App security guideline

for third party developer

Samsung Electronics Co., Ltd.

Contents

1. Overview - Smart TV App Security

Security is becoming an important issue with the increase of various smart devices. In order to protect data from users and businesses, Samsung Smart TVs are enhancing security in several layers, from hardware to software. As Samsung Smart TV apps are also software driven by Samsung Smart TV, the security needs to be taken into account.

Samsung smart TV app can store important information such as code and key values and personal information of the user, which is an important resource that must be protected. These resources can be leaked due to a variety of reasons, such as a simple mistake by a developer or hacking by an attacker. In order to safeguard this, Samsung smart TV app needs to be developed according to Secure by Design. In particular, the personal information of the user should comply with the policy related to the personal information for each country.

2. Secure by Design

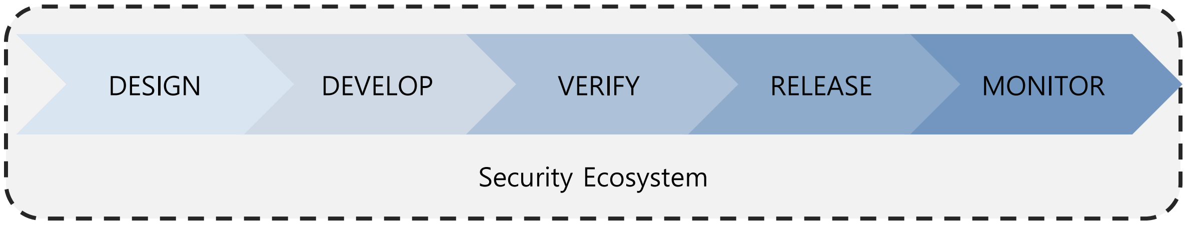
All software within the smart TV developed by Samsung are based on the Secure Development Lifecycle (SDL) model, and development step is divided into analysis, design, implementation, and testing, so vulnerability should be removed by performing a security review at each step. From the same point of view, apps operating on Samsung Smart TV should maintain the same security level. For this, we recommend that you consider security in the app development phase by referring to the following step-by-step security review.

- ✓ Security in the analysis/design phase: You should identify important information that is stored and transferred and ensure that the information is handled safely.
  - If you receive user input, you should review that you do not require more information than you need, and there is no issue with the input format.
  - You must identify the important information to be used and ensure that the information is displayed on vulnerable areas in the flow of the program.
  - In particular, when transmitting important information outside the device, you need to ensure that it communicates with the specified server through a secured channel.
  - At the time of designing, you must first define important information that needs to be protected and design it in a proper manner to protect it.
- ✓ Security in the implementation phase: It must be implemented in compliance with security rules to prevent information in the software from being leaked through known vulnerabilities.
  - Important information obtained in the design phase should be stored by applying security techniques such as encryption and make sure that it does not exist in plain text within the program.
  - Establish secure coding rules for each language and proceed with development accordingly.
  - You must use only the minimum permissions required and notify the user of the permissions you use.

  - You should make sure that the security channel is properly set on the network, and the latest version of the technology is applied.
  - If you use encryption algorithms, you must use them securely using verified standard algorithms where vulnerabilities are not reported.
- ✓ Security in Test phase: Security checks must be performed before deployment to prevent security issues and maintain security through maintenance after deployment.
  - Before deployment, it is necessary to verify that there is no issue with analysis, design, and implementation when actually operated through simulated hacking, packet checking, etc.
  - After deployment, if a new vulnerability is found or a modification occurs in the security check, it must be patched and applied to all users as soon as possible.

3. Security Review Process

In order to maintain the security of the app ecosystem, Samsung is performing security checks on the submitted apps.



Samsung checks the risk or misuse cases that may occur due to the submitted app, and if there is an issue, the deployment process can be stopped and the app submitter can be advised to fix it.

4. App Security Guidelines

This section provides basic security guidelines to consider in the development of smart TV apps. For a safe and reliable app running environment, we recommend that you proceed with the following points in the development phase.

4.1. Data Protection

Three key factors for data protection are confidentiality, integrity, and availability. If an app sends or stores sensitive information, the app must encrypt data stored on these devices and protect it from attackers.
It is very important to protect sensitive data such as user credentials or personal information in app security. If the mechanism of the operating system is not used correctly, sensitive data can be unintentionally exposed.

Definition of sensitive data

- Personally Identifiable Information that can be exploited for identity theft: Ex : Resident registration number, social security number, credit card number, bank account number, health information, etc.
- Sensitive data that can lead to loss of honor and loss of money if leaked
- All data that must be protected for legal or compliance reasons.

#	Description
4.1.1	Sensitive data, such as passwords or PIN data, should not be exposed through the user interface.
4.1.2	The key values used by the app must be hardcoded or not stored in plain text.
4.1.3	Sensitive data should not be stored in an app container or external storage.
4.1.4	Sensitive data should not be recorded in the application log.
4.1.5	Sensitive data should not be shared with third parties unless it is necessary in the architecture.
4.1.6	Keyboard cache must be disabled from the text input that processes sensitive data.
4.1.7	Sensitive data should not be exposed even during internal communication.
4.1.8	You should ensure that the data stored in the client-side storage (Ex: HTML5 local storage, session store, IndexedDB, regular cookie, or flash cookie) does not contain sensitive data.

4.1.9	Make sure that you have provided clear T&C for the collection and use of the provided personal information and that you have provided selective consent to the use of that data before you use it.

Refer :

- European Union General Data Protection Regulation (GDPR) overview
- European Union Data Protection Supervisor - Internet Privacy Engineering Network
- App Development Privacy Guide

4.2. Authentication

If there is a feature to log-in to the remote service by the user, it must be configured through security design. Even when most of the logic is operating on a remote service, the device must also meet security requirements on how to manage user accounts and sessions.

#	Description
4.2.1	If the app provides remote services to the user, user name and password authentication must be performed from the remote service.
4.2.2	If you use Status Storage Session Management, the remote service must authenticate the client request using the randomly generated session identifier without sending the user's credentials.
4.2.3	If using stateless token-based authentication, the remote services must provide signed tokens using security algorithms.
4.2.4	When a user logs out, the remote service must end the existing session.

Refer

4.3. Access Control

An app can access a resource only if it has access to it.

#	Description
4.3.1	App must require only the minimum access required.
4.3.2	App must use the privilege that match the permissions and specify the Privileges used.

4.3.3	When accessing user data, make sure that the principle of minimum access privilege requirement is followed. Apps must have access to APIs, data files, URLs, controllers, directories, services, and other resources with minimal access required.
4.3.4	You should verify and process all input from external resources and users. This should include data received through the UI, a user-defined URL, inter-process communication (IPC), etc.
4.3.5	If an app uses a completely unprotected custom URL, you should not export sensitive information.
4.3.6	Important data or APIs must be protected from user access other than data owners.

Refer :

- OWASP Cheat Sheet: Access Control

4.4. Communications

When the network is used, the app should not display the transmitted/received content using a secured channel.

#	Description
4.4.1	Data must be encrypted on the network using TLS(Transport Layer Security). Security channels must be used consistently throughout the app.
4.4.2	The setting of the security channel must be configured to protect information safely.
4.4.3	The data being transmitted must be protected from being snatched/taken over in the middle. (ex. Defence against Man In The Middle attack)

Refer :

- OWASP – TLS Cheat Sheet

4.5. Input Validation

You must defend the command insertion attack through validating the validity of input value. Input value validation should be considered at all stages of development.

#	Description

4.5.1	Input values must process the data based on type and content, applicable laws, regulations and other policy compliance, and define how to handle it.
4.5.2	You must ensure that input validation is performed on a trusted service layer.
4.5.3	You need to check whether it protects against parameter attacks such as mass parameter allocation attacks or unsafe parameter allocation.
4.5.4	All possible input values (e.g. HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc.) must be checked using validation (ex. whitelist).
4.5.5	You should check whether the values entered are in the correct form in well-defined schemas, including allowed characters, lengths, and patterns.
4.5.6	The URL redirection and forward should display a warning that only whitelist targets are allowed or that you are connecting with potentially untrusted content.
4.5.7	Make sure you use memory safety strings, secure memory copy, and pointer calculation to detect or prevent stacks, buffers, or heap overflows.
4.5.8	In order to prevent integer overflow, you need to make sure that sign, range, and input validation techniques are used.

Refer :

- XML External Entity (XXE) Prevention Cheat Sheet
- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems

4.6. Password Management

In case of app with different user password, security settings are required for them.

#	Description
4.6.1	You must ensure that the password does not contain spaces and cut/copy is not performed.
4.6.2	In the Password Change feature, you should check that the user's current password and new password are required.
4.6.3	It is recommended to provide a password strength meter so that users can set a stronger password. It is also recommended to provide rules that limit allowed character types (Uppercase letter, numeric, special characters).
4.6.4	You should check that it is recommended to change your user password within the right due date.
4.6.5	Do not store the user password in the app's properties or settings file in plain text or recoverable form.

4.6.6	Passwords must be stored, transferred, and compared in a hashed state using a standard hash function.
4.6.7	To prevent random attacks, you should use the login limit(number of login) or CAPTCHA.
4.6.8	Default password should not be generated.
4.6.9	Make sure you do not show the key information, like passwords in the log.

Refer :

- CWE-804: Guessable CAPTCHA
- CWE-836: Use of Password Hash Instead of Password for Authentication
- CWE-257: Storing Passwords in a Recoverable Format
- CWE-261: Weak Encoding for Password
- CWE-263: Password Aging with Long Expiration

4.7. Session Manager

A session is a technique for controlling and maintaining the status of a user or device interacting with one user in a web-based app. A session has a unique value for each user and cannot guess or share that value.

#	Description
4.7.1	You should check that the session token is not exposed/displayed in the app's URL parameter or error message.
4.7.2	Make sure the app generates a new session token from user authentication.
4.7.3	You should check that the session token is stored using properly secured cookies or security methods.
4.7.4	You should check that a session token is generated using a standard encryption algorithm.
4.7.5	Make sure the session is not reused by verifying that the session token is invalid when logout and session expires.

Refer :

- OWASP Session Management Cheat Sheet
- Algorithms, key size and parameters report 2014

4.8. Error Handling

The purpose of error handling is to allow applications to provide security events related to monitoring, status check, and increase in permission, and not just creating logs.

#	Description
4.8.1	You must ensure that common error handling formats and access method are used.
4.8.2	You must make sure exception handling is used on the code base to explain expected and unexpected error conditions.
4.8.3	You must ensure that other error handlers that can prepare all unprocessed exceptions are defined.
4.8.4	In case of an error, you must make sure that the message shown to the user does not contain application-related technical or sensitive information. We recommend using separate error codes for error support..

Refer :

4.9. Release

Check the following before releasing the app.

#	Description
4.9.1	App must be signed and distributed with a valid certificate, and the private key must be properly protected.
4.9.2	Debugging code and developer support code (test code, back door, hidden settings, etc.) must be removed. Deployed apps should not output or record detailed errors or debugging messages.
4.9.3	Libraries and frameworks etc. used by apps should be checked for known vulnerabilities.
4.9.4	The equipment used for release must be able to respond to external threats (viruses, hacking, etc.).
4.9.5	It should be built in release mode. A separate debug message should not be left from the app.
4.9.6	If you include binary, debug information should be removed.

4.9.7	If a vulnerability occurs after release, you should update the app as soon as possible and always keep the latest version.

Refer :

5. References

- https://owasp.org/www-project-web-security-testing-guide/
- https://www.owasp.org/index.php/OWASP_Secure_Software_Development_Lifecycle_Project
- https://www.microsoft.com/en-us/securityengineering/sdl
- https://cwe.mitre.org/data/index.html