Cyber Security Baseline for Consumer Internet of Things Device

Xiaomi AIoT Security Laboratory

December 2020

Contents

Preface

In recent years, with the development of technology and consumers' increasing demand for consumption upgrade, the market of Internet of Things (IoT) devices has also begun to flourish. In terms of the number of products, the quantity of IoT devices is expected to exceed 30 billion in 2020 and more than 75 billion in 2025 according to the research report of Statistics; from the perspective of market value, the IoT market space equals 17 billion dollars in 2019 while expected to exceed 81 billion dollars in 2025 according to the research report of the IoT analystic; from the data generated by IoT devices, the amount of data generated by IoT devices will grow at a compound annual growth rate of 28.7% from 2018 to 2025, reach 79.4 ZB by 2025 as expected according to IDC's research report.

The rapid acceleration of the IoT market means that IoT devices will quickly enter people's daily lives and be widely used in various scenarios. Different from traditional home appliances, consumer IoT devices, which actually are the extension of the Internet in users' various life scenarios, are able to sense objects, transmit and process information intelligently. Therefore, while facilitating people's lives, a large amount of users' personal information and even sensitive information will be collected, transmitted, stored and processed inevitably. Domestic and Global regulatory agencies as well as international standardization organizations have consecutively issued security and privacy laws, regulations and standards for IoT products. At the same time, consumers have increasingly higher requirements for security and privacy protection capabilities of IoT devices.

In case of the rapid increase of the amount of IoT devices, with the sharp increase of the data generated and the ever-increasing attention of supervision, standards organizations and users on security and privacy protection, how companies should ensure the security and privacy of IoT devices, so that users, regulators, and partners can be at ease and trust their IoT products, ensuring product development under compliance, has become the primary problem for companies to solve. However, there is no consumer IoT terminal devices security baseline guides that can be publicly queried and implemented in China, which can help companies improve the security and protection capabilities of their IoT terminal devices.

Xiaomi AIoT Security Laboratory has developed this baseline guide based on years of experience in security testing and compliance with domestic and global laws, regulations and standards. This guide is intended to help domestic IoT companies to deal with the above challenges by providing an open, convenient, and practical knowledge base. When designing and developing consumer IoT terminal products, companies can use this guide to avoid some basic security and privacy protection risks, so as to quickly improve the security and privacy protection capabilities of their products.

Scope

This baseline mainly describes the security baseline requirements for consumer IoT terminal devices.

Normative References

The following documents are necessary for the application of this document. For dated references, only the dated version applies to this document. For undated references, the latest version (including all amendments) applies to this document.

ETSI EN303645

Cyber Security for Consumer Internet of Things: Baseline Requirements

GB/T35273-2020

Information security technology—Personal information security specification

01

Hardware Security

1.1 Physical Debug Interface

- **Debug interface should be disabled by default**

Consumer IoT devices (hereinafter referred to as “devices”) should disable debug interface such as UART, JTAG, SWD and etc. at the factory by default. If the debug interface is necessary for dealing with after-sales problems or for other reasons, connecting with jump wire or special operations (such as special buttons combination, private USB Dongle serial connection, powering on device under tilting status and etc.) based on device sensors or other abilities should be done before the interface being enabled in order to reduce unnecessary physical debug interface exposure and information transmission.

- **Debug interface silk screen on PCB of the devices should be removed**

Debug interface **silk screen** (such as obvious TX, RX) on PCB board of IoT terminal devices is advised to be removed to prevent reverse engineering.

- **The information transmission at the device debug interface should follow the principle of data minimization**

After the debug interface is enabled for certain specific needs, the information input of the debug interface should not be enabled by default, and only log output that does not contain sensitive user and device information is allowed (sensitive information includes sensitive security parameters such as key, token and personal sensitive information such as password, Wi-Fi password, etc. More details about personal sensitive information can be looked up in Appendix A). If the complete data flow log is needed, such information should be displayed in a masking (eg:

password:********)

1.2 Local Data Storage

- **Encryption of sensitive information**

Device should encrypt the sensitive information stored in storage chips (flash, nand, emmc, etc.). The encryption scheme can be implemented by integrating an security chip or operating system partition encryption.

- **The device should enable chip Read Out Protection**

The device should enable chip Read Out Protection mechanism to prevent the firmware from being read through the debug interface.

1.3 Communication Link Data Transmission

The communication data itself transmitted in the hardware communication link (IIC/SPI) should be encrypted by the device. It's advisable to use a security chip to ensure the security of the encryption key, supplemented by the mechanism of mixing up real data with fake data before transmission in order to prevent attackers from stealing communication key and chip instructions by sniffing through the hardware channel.

Figure 1-Security Chip

1.4 Secure Boot

The device chip should support secure boot (Secure Boot's principle is shown as the figure). During startup , Firmware (uboot, kernel, rootfs) or Flash key partitions should do validity verification before loading to ensure only by passing the security verification can the system boot normally.

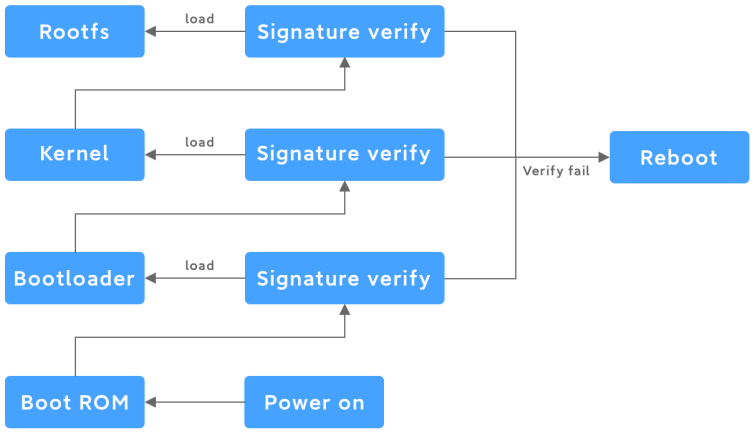

Figure 2-Secure boot

1.5 Boot Exception

When the device Bootloader / U-Boot throw exceptions, the operating system should automatically reboot which can avoid exposing the operating system bootloader console interface in order to prevent attackers from entering the console interface by inserting errors to get the permission to forge with the boot parameters of the device, thereby controlling the device.

1.6 MCU IAP Update Mechanism

MCU of some power control devices has the capability to support IAP of Type-C CC pin. Such MCU should have a secure update mechanism, such as packaging a private USB Dongle to ensure data encryption or signature verification for the MCU's IAP process to avoid the risk of device power supply and consumption caused by the forgeing of MCU firmware logic.

1.7 Protection against EMP Attack

High-security level devices should be equipped with an electromagnetic shield outside the chip to prevent device from EMP Attack, which may cause device logic or operation exception, thereby leading to device crash or circuit burn (such as a EMP Attack to a door lock device can lead to unauthorized unlocking).

1.8 Prevent Physical Dismantle

Outdoor devices (public area: such as outside the gate) should have a protection or warning mechanism when they are violently removed or dismantled. For example, use unusual fixing screws to fix the device or be able to sound an alarm when detect to be dismantled. (such as a doorbell)

1.9 Smart Door Lock Unlocking

Smart door lock NFC unlocking mechanism should apply CPU

card as the unlocking card, and ID card or M1 cards which are easily to be sniffed, cracked and copied shouldn't be used.

Type	Advantages	Disadvantage	Recommendation
CPU Card	High security. Large user space, fast reading speed	Slightly more expensive	Recommended
ID Card	Cheap price	Easy to be copied, low security	Should not be used
M1 Card	Support read and write	Easily sniffed, cracked and copied, slightly more expensive	Should not be used

02

General Communication Security

2.1 Privileged Function Interface

The privileged function or interfaces of the device that can directly access the device system (such as factory OTA, undisclosed function interfaces, debug backdoors, etc.) should be disabled by default. If they are necessary for certain purpose, authentication mechanism should be implemented.

2.2 Hard-Coded Key

The devices should not hard-code the key used for transmission encryption or authentication in program code, and measures like unique-password-per-device or generation from PSK should be applied to generate keys.

2.3 Communication Channel Encryption

The devices should encrypt the communication channel with other devices or applications, and destroy the session key in time at the end of the session. Refer to Chapters 3, 4, 6 for encryption schemes for different communication methods.

2.4 Communication Authentication

Two-way authentication should be carried out before data transmission during device communication to verify whether the real identities of both ends are valid as well as to check whether the control permission matched the identity, in order to prevent unauthorized control.

2.5 Anti-replay

The device communication should apply rolling code or counter mechanism. Only when the request operation count is greater than the device count, can the device execute the operation instruction, which aims to prevent devices from unauthorized control by capturing and replaying the request. For the anti-replay schemes under different communication protocols, please refer to the content of the following communication protocol security chapters.

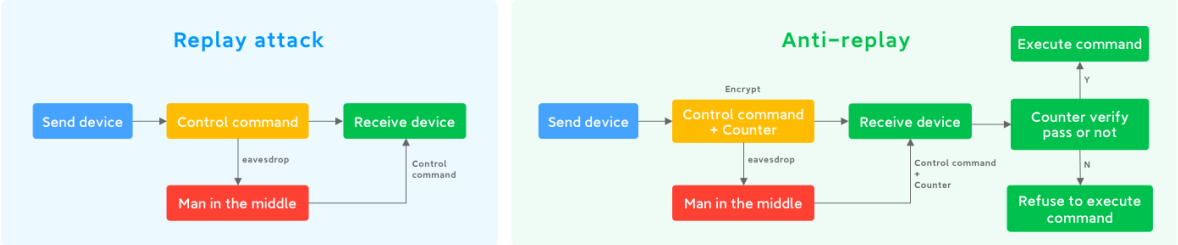

Figure 3-Replay attack and anti-replay

03

Ethernet Communication Security

The content of this chapter (Chapter 3) is applicable to devices using RJ45 and WLAN network access methods.

3.1 Data Transmission Encryption

The device should apply encrypted transmission protocols to encrypt communication while additional encryption of sensitive information must be performed during data transmission (see 13.1 encryption algorithm for key algorithm and requirements of length). The devices should adopt TLS (1.2+) protocol. Which aims to avoid the risk of information leakage or forgeing due to the use of plaintext transmission protocols such as MQTT and HTTP.

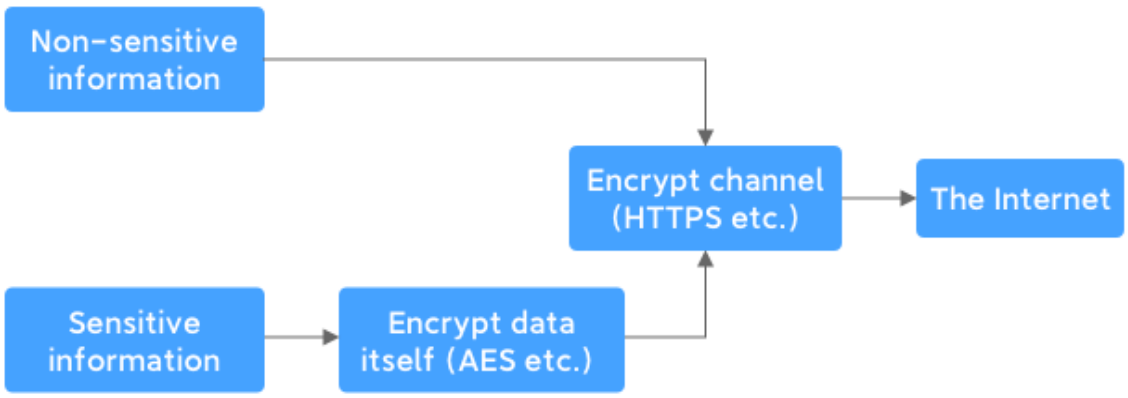

Figure 4-Data transmission encryption

3.2 HTTPS Certificate Verification

Strict certificate verification should be performed when the device applies HTTPS protocol, device should not ignore the verification.

- **Linux System**

The devices should strictly verify the validity of the server certificate. Using parameters to skip certificate and ignoring certificate verification errors are forbidden.

Tools/Lib	Recommendations for using parameters
curl	Shouldn't use -k parameter
wget	Shouldn't use --no-check-certificate parameter
libcurl	Should set CURLOPT_SSL_VERIFYPEER and CURLOPT_SSL_VERIFYHOST as True

- **Android System**

Server and client certificates should be strictly verified when devices applications apply SSL encrypted communication services. The devices should not trust any certificates, should not ignore abnormal events (such as empty return or null) ; If SSLx509 TrustManager is needed to be customized with the checkServerTrusted method rewritten, the server certificate verification must be strictly determined in the method to prevent the communication content from being hijacked, which will lead to communication data leakage or forgeing.

3.3 Device Service Port

Only the IoT SDK control service ports which are necessary

for device data interaction and control can be enabled.

3.4 Wi-Fi Access Point Password

The function of connecting the device using the Wi-Fi Direct access point to the control application should be strictly evaluated. If such function is necessary, the Wi-Fi Direct access point password shouldn't be fixed or empty, which should follow unique-password-per-device principle or generate password with real random number function every time you use it, then display the password on the screen (devices with screen) or set and store it in advance by the user during binding (devices without screen) to prevent WiFi from unauthorized access.

04

BLE Communication Security

4.1 BLE SoC SDK Version

Devices should regularly check and upgrade the BLE SoC SDK to the latest official version, rather than using such SDK with security issues (such as protocol stack vulnerability SweynTooth[1]) .

4.2 BLE Pairing

BLE devices with physical buttons should perform the bind confirm via physical buttons or enable bind window to avoid the risk of repeated binding or being bound by others without users consent.

4.3 Validity Verification of Control Instructions

For devices that support BLE, the session key should be negotiated every time when users log in and it should be used to encrypt transmission control instructions between the device and the control applications.

4.4 BLE Advertising of Sensor Devices

Some devices can advertise the data collected by sensors through BLE, which will be analyzed through BLE gateways for devices linkage or perform as control instructions, thereby affect other devices. The BLE advertising of such devices should apply a

secure communication protocol as well as encrypt the dvertising data with the beaconkey generated during binding process.

4.5 BLE Protocol Version

BLE devices (such as mouse and speakers) shouldn't apply BLE protocol version less than 4.2 when performing BLE link layer encryption to prevent the device from leaking the BLE Long Term Keys (LTK), which may lead to risks of privacy leakage or device forgery.

4.6 BLE Control Instructions Authentication

The devices should detect the BLE binding status of the control application before using the OTA routine instructions provided by the BLE chip manufacturer. Which is intended to prevent the device from denial of service (such as chip reboot, switching work space, enabling DFU upgrade mode, etc.) due to the logic that may support unauthorized instructions to control the device in the routine.

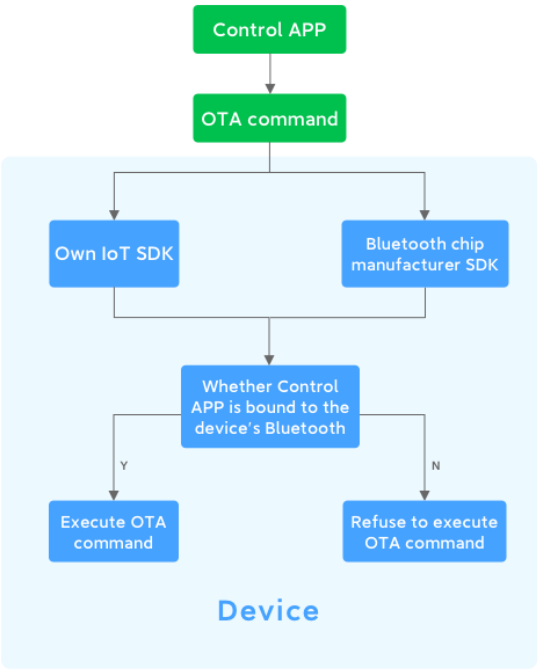

Figure 5-BLE control command authentication

4.7 BLE Advertising Anti-tracking Mechanism

The device should apply a random BLE MAC address and encrypt the BLE advertising content; if there's need to advertise identifiable information through the beacon, the device BLE MAC address should change regularly to prevent that others may analyze the advertising content by deploying enough detection devices and track the movement of the device.

4.8 BLE Sensitive Information Communication

When the devices communicate with the control application (Android) through BLE, the sensitive information content itself should be encrypted at the application layer to prevent that all applications on the phone can access the data transmitted between the two devices when user pair the phone with other smart devices for data transmission through BLE.

For the risk description, please refer to the development document[2] when Android 4.3 introduced BLE:

In contrast to Classic Bluetooth, Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption. This allows Android apps to communicate with BLE devices that have stricter power requirements, such as proximity sensors, heart rate monitors, and fitness devices.

Caution: When a user pairs their device with another device using BLE, the data that's communicated between the two devices is accessible to **all** apps on the user's device.

For this reason, if your app captures sensitive data, you should implement app-layer security to protect the privacy of that data.

05

Device Zigbee Communication Security

5.1 Default TCLK

Devices that support the Zigbee 1.2 protocol stack should not use the default TCLK (Trust Center Link Key) of the Zigbee Alliance: 5A6967426565416C6C6961 6E63653039 (ZigBeeAlliance09). The coordinator and nodes should be pre-built with non-default TCLK before leaving the factory.

Devices that support Zigbee3.0 protocol stack should apply official install code scheme[3] .

5.2 Anti-replay

The device should enable the Zigbee protocol frame counter to have the capability to protect itself against replay attacks.

```
Open method: set nwkAllFresh to TRUE
```

5.3 Insecure Rejoin

When the device applies Rejoin function, apsUseInsecureJoin should be set to False (the default value is true) to prevent Zigbee device which is not meeting requirement in 5.1 from being controlled by rejoin attack[4] using default TCLK to decrypt the Network Key even if the join window is not opened

06

Devices RF Communication Security

6.1 Anti-replay

The device should apply rolling code to communicate to avoid replay and forgery. Please refer to industry mature solutions[5] such as keeloq, DST40, Hitag2, etc.

6.2 RF Communication Packet Sequence Number

The radio frequency communication data packet of the device should use four bytes as sequence number variable space. A short sequence number is not recommended. This requirement is to avoid the risk that the communication may be brute-forced in a short time.

6.3 Hard-coded Communication Key

The radio frequency communication key of the device should be generated by paring and exchange of the transmitter and the receiver without preset in the code.

6.4 Communication Frequency

The device should apply the frequency hopping mechanism for communication to prevent channel congestion caused by the fixed radio frequency communication, resulting in abnormal communication of the device.

07

General Device System Security

7.1 Integrity and validity of Firmware Upgrade Package

Before device firmware upgrade (OTA remote or local upgrade), the firmware package should be hashed to get the firmware package integrity digest, then be signed and verified by public and private key measures (refer to Nordic routine[6]). The integrity and validity should be confirmed before upgrading to prevent the firmware being forgeed or replaced.

The integrity hash of the firmware upgrade package should use a secure hash algorithm (see 13.2 Hash Algorithm for more details). The Integrity credential should be encrypted before transmission in the channel between the device and the server.

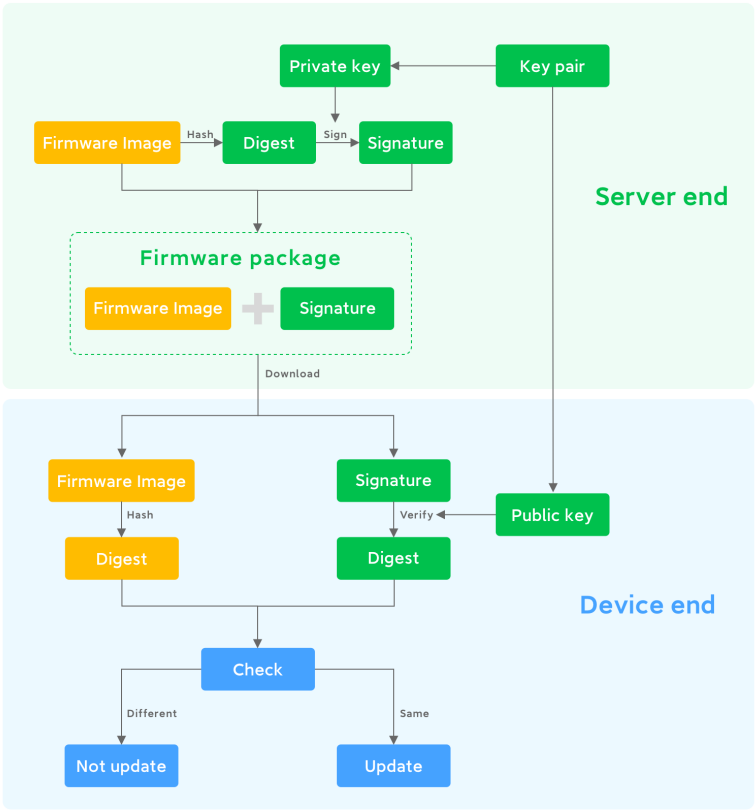

Figure 6–Validity and integrity verification of firmware upgrade package

7.2 Firmware Downgrade

When the device perform update through the OTA function, the device should reject the old firmware update version to prevent the re-exposure or use of some historical BUG or security risks. For special needs (such as testing, maintenance, etc.), the firmware can be flashed through SD card or wire flashing after checking the signature.

7.3 High-risk Network Services

The device should disable FTP, SSH, Telnet, HTTP, ADB and other high-risk management services or information data services by default.

7.4 OTA Upgrade Instructions

The device should only execute OTA upgrade instruction from authorized sources in the cloud. The LAN upgrade capability should be disabled to prevent attackers use LAN instruction to OTA upgrade malicious firmware.

08

Embedded Device Linux System Security

8.1 Address Space Layout Randomization

The device applying embedded Linux system should enable the address space layout randomization protection measures to prevent buffer overflow.

```
Open method: add kernel.randomize_va_space = 2 (Kernel >=2.6.12) in /etc/sysctl.conf
Parameter 2 means that in addition to the library and the stack, the heap is also
randomized protection, but it should be noted that ASLR is not responsible for the
random protection of the code side and the data side. This work needs to be implemented
by the compiler PIE, see: 11.3
```

8.2 Bootloader Boot

Bootloader of the device should not reserve interruption time before the system booting with Delay set to 0 in order to prevent that attackers may enter SHELL interface and gain control of the device by modifying the boot parameters.

8.3 Serial Port

The device should not bind system SHELL to a serial debug interface such as UART to prevent the device from being controlled through wiring. If there are special needs, requirement in 8.4 should be followed.

8.4 System Default User Password

Embedded Linux system default users (such as root, admin, etc.) need to set strong passwords and ensure unique-password-per-device. All devices with the same password or empty password are strictly forbidden.

```
Strong password: The password should be 10-14 digits and must contain four types
of characters: letters, uppercase letters, symbols, and numbers
```

8.5 Root File System Permission

The root file system of the embedded Linux of the device (such as /etc/ and other directories containing boot items) should be set to read-only permission to avoid being forgeed by malicious attackers.

8.6 Externally Stored Programs and Scripts

The embedded Linux operating system of the device should not run programs or scripts in external storage (SD card, USB flash disk, network storage, etc.) by default. If there are special needs, public and private key signature verification should be performed to prevent the system from being implanted with malicious software or scripts.

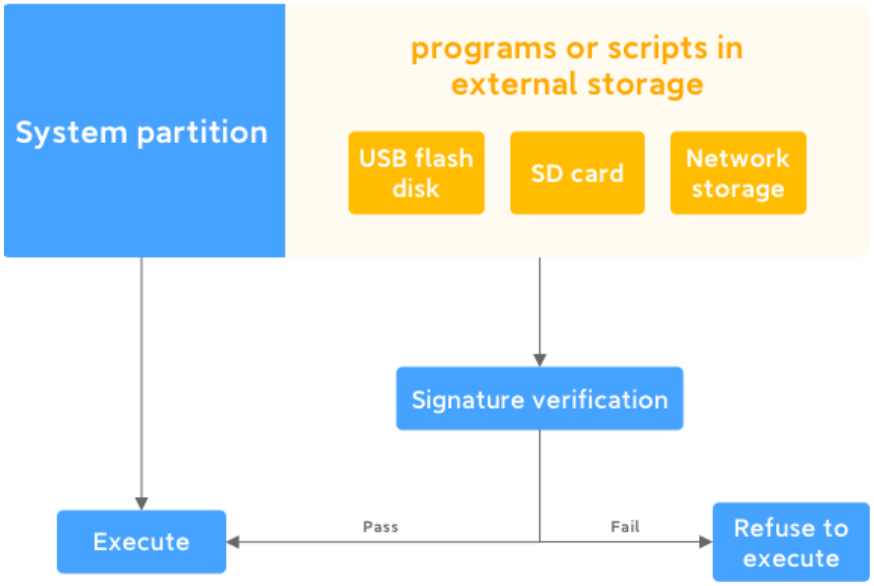

Figure 7-External program and script signature verification

09

Android Application Security

9.1 Socket Port Request

When the application enables the Socket network port and function, it should verify the authentication of the port request and the validity of the communication data to prevent that other local applications or remote attackers connect with the ports through the network and call functions maliciously.

9.2 Private Directory Permission

The files and folders in the private directory of the application shouldn't grant read, write and execute permission to other users (for example, the permission should be set as "-rw-rw---", not "-rw-rw-rw-") to prevent forgeing of the program logic or leakage of users' privacy information due to related permissions.

9.3 Local Information Storage

Application should not store sensitive information in plaintext in local files of the device, including text files, binary files, XML files like ShredPreference and database like WebView to prevent malicious files in APP data directory from reading and leaking sensitive information stored locally.

9.4 External Executable File

Files that need to be dynamically loaded from the outside should not be stored in such a location like SD card that writable

by any process, but should be placed in a private directory and verified the signature at the same time. If there's need to store files in external storage such as SD card for special reasons, the application should verify the signature before loading the file to prevent the external executable from being forgeed.

9.5 Files Decompression

When the application uses ZipEntry.getName() to decompress zip files, the parent directory string (../) should be filtered, and only the compressed package that contains no (../) special characters in the file name is allowed to be decompressed to prevent that the decompressed directory jump and overwrite key files in other directories, leading to the execution of any external code.

The application should perform digital signature verification on important Zip compressed files and only the compressed files that pass the verification are allowed to be decompressed.

9.6 XML Configuration

The application should close the application backup configuration and debug configuration to prevent malicious attackers who can reach the user device from launching the device USB debug function in a short time and use the ADB debug tool to steal application backup data or forge with the application logic without device ROOT permission, resulting in leakage of users' privacy and even property loss.

```
Configuration method:
Application backup: specify android:allowBackup="false" in the AndroidManifest.xml file.
Adjustable: keep the default configuration of debugable in AndroidManifest.xml: android:debugable="false"
```

9.7 Anti-Reverse Engineering

- **The application should enable protection against decompilation**

The application should refuse to execute su command or search for su file and stop after throwing exceptions when getting such command to prevent itself from being decompiled on a ROOT device.

- **The application should apply code obfuscation**

The application should use Proguard or other software hardening scheme to obfuscate the Java code in the application APK to prevent the core code from being stolen after the application is decompiled.

9.8 Application Integrity

Application should verify the integrity of the dex and apk package to prevent APP files from being forgeed which will lead to piracy, in-app purchase cracking, embedded advertising and malicious code.

9.9 Installation Package Signature

The application should be signed with the digital certificate of the subordinate party before being released. It should not be

signed with the certificate of the third-party to prevent that the application cannot be overwritten and installed when the old signature doesn't match the new one , in case the cooperation with the third-party developer is terminated and the signature of the application need to switch to subordinate party, which will make the application subordinate lose the initiative in version management.

9.10 Android Component

- **Android Component Permission Control**

The four major components of the Android application can be set to be export or not through the export properties. Non-exported components are only allowed to be accessed within the application, while the exported components can be accessed by any components of any program.

- The four major components of the application's export attribute should be actively and clearly set to export or not, in order to express the intention of whether the component can be accessed by the outside clearly. The default export rules should not be used to prevent components that are not originally intended to be accessed by the outside from being accidentally set to accessible by the outside when developers are not familiar with the default export rules for different versions of Android and different component types.

- Components which only used internally by the application

should be forced to not be exported to prevent unnecessary external access.

- For components that need to be open to other applications with the same signature, only the components with function requirements can be set to export, and custom permission should be used whose level needs to be set as signature[7] to prevent the components' interface from being used maliciously.

```
Setting method:
Set the export attribute "android:exported" in AndroidManifest.xml to not
export ("false") or export ("true")
```

- **Activity Anti-hijacking**

Applications should take the following two protection measures to prevent the application startup page Activity from being hijacked as a phishing tool:

- Interface Switching

The application should pop up a warning message when the Activity interface is switched to the backstage to prevent the hijacker from attaching its page to client.

- Process Stack Protection

The application should protect the process stack accordingly, and should not allow other process to be placed on the client.

- **Content Provider Access**

When the file access API of the ContentProvider component is called externally, the absolute path of the file corresponding to the incoming target file URL parameter should be filtered and

judged, and only the access requests with the allowed access path can be executed to prevent attackers from reading readable files in any directory without authorization by passing in the URL containing the parent directory, causing data leakage.

```
Filtering method: Obtain the absolute path of the file to be accessed (via File.getCanonicalPath)
method, and determine whether the path starts with the directory path allowed to be accessed.
If not, the accessed file is illegal and does not provide subsequent business functions.
```

- Intent

  - Empty Intent of implicit Intent should not be used when using PendingIntent

Applications should not apply empty Intents to construct PendingIntents, and the Intents to construct PendingIntents must be set with ComponentName or action to prevent the Intent from being hijacked causing information leakage.

  - The application should strictly filter the Intent object and URL when receiving the URL passed in by the Intent

The URL passed in the application by Intent should be strictly filtered and the corresponding filtering rules should be set for the Intent object to avoid intent-based attacks caused by the improper processing of the Intent Scheme URL by the application. At the same time, it can also prevent the incoming Intent from directly starting system components, allowing attacker to bypass some signatures and start components directly or start components that haven't been exported.

- **WebView Certificate Authentication**

When a certificate authentication error[8] occurs during WebView component loads a webpage, the webpage should be stopped loading while the certificate error should not be ignored to prevent privacy leakage due to man-in-the-middle attacks.

```
Method: If the onReceivedSslError method is overloaded, the handler.cancel() method
should be used in onReceivedSslError to stop loading the problem page
Note: Do not call handler.proceed() in the overloaded onReceivedSslError method to
ignore the certificate verification and continue to load the web page
```

- **WebView Cross-domain Access**

Cross-domain access to local file in WebView should be disabled by the application to prevent application cloning attacks or remote leakage of local data.

```
Closing method: setAllowFileAccess(false), setAllowFileAccessFromFileURLs(false),
setAllowUniversalAccessFromFileURLs(false), and filter and judge the final access
file URL passed by the Intent (see 9.10 Content Provider access for details)
```

- **WebView jsbridge Call**

The application should strictly limit the schema to https, and use a whitelist to limit the domains that WebView can access. At the same time, the absolute path of the final access file should be filtered and judged (see 9.10 ContentProvider access for details) to prevent JSAPI from allowing any webpages to execute without authentication, leading to user privacy leakage or remote code execution attacks.

- **WebView Password Storage**

It's not allowed to store password during application using WebView in order to prevent the username and password input in WebView by users from being stored in databases/webview.db of the application's data directory in plaintext which may lead to personal sensitive information leakage.

```
Method: Use the WebView.setSavePassword(false) method
```

9.11 Anti-Screen Recording

The application should use the mark characters to obscure the users' privacy information, or monitor the process list to determine whether the screencap (screen recording) process is running. If there exists screencap running, application should prompt user of the risk to prevent user's sensitive information from being eavesdropped by malicious programs.

9.12Memory Data Protection

Application should encrypt the memory data or restrict attackers from injecting, hooking, and anti-debugging by detecting root permissions to prevent attackers from accessing the application memory space and exposing the accounts, password or other sensitive information in memory.

10

General Coding Security

10.1 Third-Party Software / Library

The third-party software/library used by the device should be the latest version. If the latest version cannot be used for special requirements, the version used should go through security undergo. If there is clear security vulnerability, they should be updated to the latest version in time which the loophole is fixed.

FYI: check https://www.cvedetails.com/product-list.php for Software / Library loophole list.

10.2 Random Number Generation Function

Device system should use true random algorithms to generate random numbers required for authentication or encryption.

Type	Generation Method	Recommendation
True / Strong Pseudorandom	/dev/urandom Entropy random ability supported by the chip	Recommendated
Pseudorandomn	srand()、rand() with time(0) as the seed	Not Recommendated

10.3 String or Memory Manipulation Function

The system and applications should use secure string or memory manipulation functions to prevent the use of strings or memory functions that do not perform boundary checks from being used by attackers to carry out buffer overflow attacks.

```
Safety advice:
Do not use strcpy, use strncpy;
Do not use strcat, use strncat;
Do not use vsprint, use vsnprintf;
Do not use sprintf/vsprintf, use snprintf()
```

Note: the unsafe usage of the safe function can still lead to security problems

For example, strncpy(dest, src, size), it's necessary to specify the size before each copy according to the size of the dest buffer. Do not use approach like"the dest fixed length and sizelen(src)".In this case, the data length obtained by dest is equal to the length of src, and strncopy function is equivalent to the strcpy function.

Example code is as follow:

```
【False code example】
dest = malloc(20);
src = *param1;
size = len(src);
strncpy(dest, src, size);
```

In the above code example, dest applies for a size of 20, while src is a parameter passed in from outside, which means it is controllable by the attacker. The size is equal to the length of src, which means the strncpy is equivalent to srecpy.

```
【Correct code example 1】
dest = malloc(50);
strncpy(dest, src, 40) // At this time, the buffer size of dest must be greater than size
【Correct code example 2】
size = len(src);
dest = malloc(size + 1);
strncpy(dest, src, size); // This ensures that the capacity of dest must be greater than src
```

10.4 Format String Function Parameters

System and applications using format string functions [9]should not use external controllable variables as parameters to prevent format string vulnerabilities from causing serious harms.

```
For example:
In sprintf(char *string, char *format [,argument,...]), the format parameter is
not allowed to use external controllable variables.
```

10.5 Code Repository Management

Device-related code repository should not be uploaded to public code repositories such as Github and Gitee or other public, semi-public services such as Baidu Netdisk without permission to prevent source code leakage.

11

Linux Application Coding Security

11.1 Stack Cookie Overflow Protection

When compiling the Linux program code, the CANARY stack overflow protection option of the Linux program should be enabled.

```
Open method: add gcc compilation parameter -fstack-protector-all
```

11.2 Executable Space Protection

When compiling the Linux program code, the Executable Space Protection option of the Linux program should be enabled.

```
Open method: add gcc compilation parameter -z noexecstack
```

11.3 Address Space Layout Randomization (ASLR)

The application should enable the PIE application Address Space Layout Randomization (ASLR) protection option.

```
Method: gcc compile parameters -pie -fPIE (pay attention to
upper and lower case), and need system ASLR support, see: 8.1
```

11.4 Linux Program Code Compilation

When compiling the Linux program code, strip function should be applied to drop the debug symbol table to increase the difficulty of reverse analysis and reduce the size of the program.

11.5 System Call function Parameters

When applying system call functions in program(such as system, popen, exec, etc.), external controllable parameters should be filtered to prevent command injection.

```
method:
Perform strict character filtering on system call function parameters, such as
$/()/`/:/|/&/<>/" and other characters with special shell meanings;
Limit the parameter to a pair of single quotes, and then filter or escape all
subsequent incoming single quotes;
Execute parameterized execution of the specified process using execl class
functions. The first parameter of execl is not recommended to be a shell program
```

12

Business Function Logic Security

12.1 Device Binding Status

If the unbound device is not bound after being powered on for 30 minutes, the binding status should keep shut off before it being repowered to avoid the risk that the device always accepts binding requests which may lead to malicious binding.

12.2 Binding Confirmation

When the device is being bound, the user should be asked to confirm of binding on device (OOB) when the product capability permits. Vehicles and high security level devices (such as doorbells, door locks, outdoor monitoring) should be mandatory to require OOB to prevent the device from being maliciously bound.

12.3 Anti-rebinding

For devices with reset capability, they should be reset before accepting the binding request again to prevent repeated binding or malicious binding and control by other users.

12.4 Strong Binding Relationship

Outdoor devices, high security level devices should establish a strong binding relationship between the device and the account in the cloud, that is, the cloud records the binding relationship between the device ID and user ID.

When the device is bound, the cloud record should be verified first, and only the binding request which the device doesn't have a binding record in the cloud is allowed, and the binding record stored in cloud shouldn't be deleted by device reset. Only after the device owner (the user ID with the binding record in the cloud) unbinds on the control application can the binding relationship be cleared, in order to prevent the device from being controlled by unauthorized binding after reset.

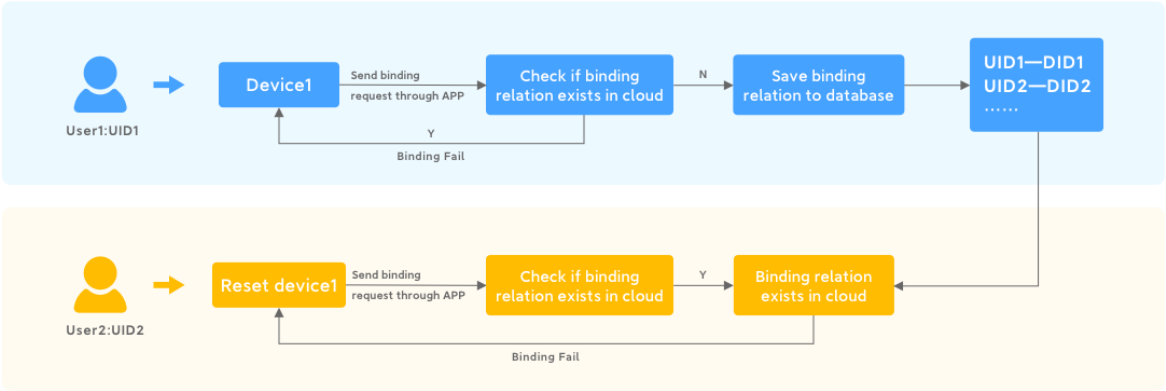

Figure 8-Strong binding

12.5 Restore Factory Settings

After the device is restored to factory settings, all user data and settings in the device (such as user usage records, settings, NFC door cards, eSIM card records, etc.) should be cleared completely.

12.6 Wi-Fi Access Point Usage

The Wi-Fi access point established by the device should only allow communication with the device itself, and should not

access the device's external network (such as the home network or the Internet) to prevent attackers from penetrating the home network through the device's Wi-Fi.

12.7 Identity Authentication Logic

The code logic of the device's identity authentication should be correctly judged, and the credential should be verified to be completely correct to pass the authentication, so as to prevent skipping the authentication judgement by empty authentication and entering the control logic directly.

```
Examples of logical vulnerabilities:
Put the authentication detection logic in the judgment of whether the authentication
credential is empty, and the exception handling that the authentication credential is
empty is not implemented, such as:
if (token !==''){
     if( checkauth( token) == FALSE) {
         exit("Auth fail");
     }
}
```

13

Data Security

13.1 Encryption Algorithms

The device should apply a secure encryption algorithm and the key length should satisfy the minimum requirements of the corresponding algorithm (see the table below).

Insecure encryption algorithms such as DES, TDES, RC4, etc. should not be used.

Algorithm	Key Length (bit)
AES	128 / 192 / 256
ECDSA / ECDH	256/384/512
RSA	2048 /3072/ 4096
DSA	Prime P: 2048 /3072/4096
	Prime Q: 256/384/512
SHA	256/384/512
SM4	128

Note:

1. NoPadding shouldn't be used when applying RSA algorithm encryption.

2.When IvParameterSpec is initialized, constant vector shouldn't be used.

3.Avoid using ECB mode when choosing encryption mode.

13.2 Hash Algorithm

The device should use a secure hash function with a length of at least 256 bits, such as SHA-2 (256/384/512), or SHA-3 in the future, etc.

The compromised hash functions are not suitable for use, such as MD5, SHA-1.

13.3 Multiple Keys

Some devices have multiple key agreement logic, that is, the device derives the exchange key through built-in information, which is used to obtain the control instruction key through encrypted communication with the server when the device is bound. The subsequent device control instruction uses the acquired control instruction key to perform encryption. The exchange key should only be used to obtain the control instruction key during the binding process, and should not be used to encrypt subsequent control commands.

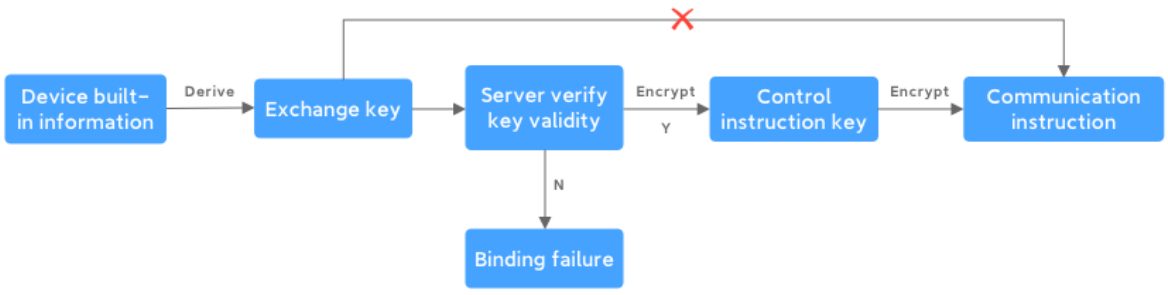

Figure 9-Multiple keys

13.4 Telemetry Upload

Telemetry printed by devices, systems and applications should not upload any plaintext or encrypted sensitive information (such as Wi-Fi SSID, password, mobile phone IMEI, geographic location and other sensitive information). If it needs to be reported for use according to evaluation, the role and storage method should be clearly stated in the privacy policy.

13.5 Cross-border Network Requests

The device or control applications should identify the country or region of the current user, and send network requests based on the privacy compliance requirements of different countries or regions to prevent privacy risks caused by cross-border data transmission.

Terms and Definitions

The following terms and definitions apply to this document

1 Consumer IoT Device

Network connected devices (referred to as “devices” in the article), consumer IoT terminal devices in this specification refer to wireless(Wi-Fi, BLE/Mesh, Bluetooth classic, Zigbee, NFC, RF), wired (RJ45) connectivity and network organization capability, or any smart terminal product with firmware logic update capabilities.

Note 1: Consumer IoT devices are also commonly used in commercial environments. These devices are still classified as consumer IoT devices.

Note 2: Consumer IoT devices are usually available for purchase by consumers in retail environments. Consumer IoT devices can also be commissioned and/or professionally installed.

2 User

Natural person or organization

3 Critical security parameter

critical security parameter: security-related secret information whose disclosure or modification can compromise the security of a security module.

EXAMPLE: secret cryptographic keys, authentication values such as passwords, PINs, private components of certificates.

4 Public security parameter

security related public information whose modification can

compromise the security of a security module

EXAMPLE 1: A public key to verify the authenticity/integrity of software updates.

EXAMPLE 2: Public components of certificates.

5 Sensitive security parameters

critical security parameters and public security parameters

6 Personal data

any information relating to an identified or identifiable natural person

7 Sensitive information

Sensitive information is a collective term for sensitive security parameters and personal information.

8 Debug interface

physical interface used by the manufacturer to communicate with the device during development or to perform triage of issues with the device and that is not used as part of the consumer-facing functionality

EXAMPLE: Test points, UART, SWD, JTAG.

9 Logical interface

software implementation that utilizes a network interface to communicate over the network via channels or ports

10 Network interface

physical interface that can be used to access the functionality of consumer IoT via a network

11 Physical interface

physical port or air interface (such as radio, audio or optical) used to communicate with the device at the physical layer

EXAMPLE: Radios, ethernet ports, serial interfaces such as USB, and those used for debugging.

12 Security update

software update that addresses security vulnerabilities either discovered by or reported to the manufacturer

NOTE: Software updates can be purely security updates if the severity of the vulnerability requires a higher priority fix.

13 Telemetry

data from a device that can provide information to help the manufacturer identify issues or information related to device usage

EXAMPLE: A consumer IoT device reports software malfunctions to the manufacturer enabling them to identify and remedy the cause.

14 Authentication mechanism

method used to prove the authenticity of an entity

NOTE: An "entity" can be either a user or machine.

EXAMPLE: An authentication mechanism can be the requesting of a password, scanning a QR code, or use of a biometric fingerprint scanner.

15 Flash

Flash is a kind of a memory chip, and the data in it can be modified through a specific program. FLASH often means Flash Memory in the fields of electronics and semiconductors, that is,

"flash memory", the full name is Flash EEPROM Memory.

16 Bootloader

In an embedded operating system, BootLoader runs before the operating system kernel runs. It can initialize hardware devices and establish a memory map to bring the system's software and hardware environment to a suitable state, so that it is ready for the final call to the operating system kernel.

17 U-boot

U-Boot is a boot loader mainly used for embedded systems. It can support a variety of different computer system structures, including PPC, ARM, AVR32, MIPS, x86, 68k, Nios and MicroBlaze.

18 Kernel

Embedded kernel is an abstraction layer between embedded hardware and software. It is called "kernel" in Linux terminology and can also be called "core". The main modules (or components) of the Linux kernel are divided into the following parts: storage management, CPU and process management, file system, device management and drivers, network communication, and system initialization (boot), system calls, etc.

19 Beacon

The BLE Beacon is a dvertising protocol based on the BLE protocol.

20 ASLR

Address Randomization (ASLR) is a security protection

technology against buffer overflow. By randomizing the layout of linear areas such as heap, stack, and shared library mapping, it increases the difficulty for attackers to predict the destination address, which can further prevent overflow attacks.

21 Identification Card

The full name of ID card is Identification Card. It is a non-writable proximity card with a fixed number. It mainly includes EM format of Taiwan SYRIS and HIDMOTOROLA of the United States. The ID card and other types. like the magnetic card, only uses the "card number". Except for the card number, there is no confidentiality function inside the card, and the "card number" is public and exposed. So ID card is "inductive magnetic card".

22 M1 card

M1 chip card refers to the abbreviation of the chip produced by NXP, a subsidiary of Philips. The full name is NXP Mifare1 series. Two types including S50 and S70 are commonly used. Common ones are card type and keychain type.

23 Routine

A routine is a collection of functional interfaces or services provided externally by a certain system. For example, the APIs and services of the operating system are routines.

Abbreviation

The following abbreviations apply to this document.

API	Application Programming Interface
IoT	Internet of Things
IP	Internet Protocol
JTAG	Joint Test Action Group
OTA	Over The Air
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
SSH	Secure Shell
MAC	Media Access Control
TLS	Transport Layer Security
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
FTP	File Transfer Protocol
UID	User ID
DID	Device ID
SWD	Serial Wire Debug
NAND	Not And
EMMC	Embedded Multi Media Card
MQTT	Message Queuing Telemetry Transport
TCLK	Trust Center Link Key
ASLR	Address Space Layout Randomization·
NX Stack	No Execute Stack
PIE	Position Independent Executables
OOB	Out of Band
XML	eXtensible Markup Language
AES	Advanced Encryption Standard
DES	Data Encryption Standard
RSA	An asymmetric encryption algorithm named after the first letter of the last name of the three co-authors of the algorithm
TX	transmit
RX	Receive
PSK	Pre-Shared Key
ADB	Android Debug Bridge
RF	Radio Freqency
JSAPI	JavaScript Application Programming Interface

Appendix

Personal information

Personal information refers to various information recorded electronically or in other ways that can identify a specific natural person alone or in combination with other information or reflect the activities of a natural person.

Table A.1 Examples of personal information	
Basic personal information	Name, date of birth, gender, ethnic group, nationality, family relation, address, personal phone number, email address, etc.
Personal identity information	ID card, military officer certificate, passport, driver's license, employee ID, pass, social security card, resident certificate, etc.
Personal biometric information	Personal gene, fingerprint, voice print, palm print, auricle, iris, and facial recognition features, etc.
Online identity information	A PI subject's account, IP address, and personal digital certificate.
Physiological and health information	Records generated in connection with medical treatment, such as pathological information, hospitalization records, physician's instructions, test reports, surgical and anesthesia records, nursing records, medication administration records, drug and food allergy, fertility information, medical history, diagnosis and treatment, family illness history, history of present illness, and history of infection, and personal health information such as weight, height, and lung capacity.
Personal education information	Personal occupation, position, work unit, educational background, academic degree, educational experience, work experience, training records, transcript, etc.
Personal property information	Bank account, authentication information (password), bank deposit information (including amount of funds, payment and collection records), real estate information, credit records, credit information, transaction and consumption records, bank statement, etc., and virtual property information such as virtual currency, virtual transaction and game CD Keys.
Personal communication information	Communications records and content, SMS, MMS, emails, data that describe personal communications (often referred to as metadata), etc.
Contact information	Contacts, friend list, list of chat groups, email address list, etc.
Personal web surfing record	Refers to records of a PI Subject's operations stored in the logs, including web browsing records, software use records, click records, and favorites.
Information of often used equipment	Refers to the information describing the general conditions of the equipment often used by an individual, including hardware serial number, equipment MAC address, list of software, and unique equipment identifier (e.g. IMEI/Android ID/IDFA/Open UDID/GUID, SIM card IMSI information).

Personal location information	Including records of whereabouts, precise location information, accommodation information, longitude and latitude.
Other information	Marriage history, religious preference, sexual orientation, undisclosed criminal records, etc.

[Source: Appendix A (Informative) Examples of personal information – GB/T35273–2020]

Personal sensitive information

Personal sensitive information refers to personal information that, once leaked, illegally provided, or misused, may endanger personal and property safety, and easily lead to personal reputation, physical and mental health damage, or discriminatory treatment.

Table B.1 Examples of sensitive personal information	
Personal property information	Bank account, authentication information (password), bank deposit information (including amount of funds, payment and collection records), real estate information, credit records, credit information, transaction and consumption records, bank statement, etc., and virtual property information such as virtual currency, virtual transaction and game CD Keys.
Physiological and health information	The records generated in connection with medical treatment, including pathological information, hospitalization records, physician's instructions, test reports, surgical and anesthesia records, nursing records, medicine administration records, drug and food allergy, fertility information, medical history, diagnosis and treatment, family illness history, history of present illness, history of infection.
Personal biometric information	Personal gene, fingerprint, voice print, palm print, auricle, iris, and facial recognition features, etc.
Personal identity information	ID card, military officer certificate, passport, driver's license, employee ID, social security card, resident certificate, etc.
Other information	Sexual orientation, marriage history, religious preference, undisclosed criminal records, communications records and content, contacts, friends list, list of chat groups, records of whereabouts, web browsing history, precise location information, accommodation information, etc.

[Source: Appendix B (Informative) Identification of personal sensitive information – GB/T35273–2020]

Reference

【1】 SweynTooth vulnerability:

- The affected protocol stack version of Nordic chip as GATT client is S110/S120/S130/S132 v2.0.0

https://infocenter.nordicsemi.com/pdf/in_119_v1.0.pdf?cp=3_1_3_1

- Dialog, Telink (link layer encryption), NXP, Cypress, Microchip, Texas Instruments, STMicroelectronics specific model version reference:

https://asset-group.github.io/disclosures/sweyntooth/

SweynTooth Affected Product Version List

SoC vendor	Soc model	SDK（<= below versions means vulnerable)
BLE Version 5.0/5.1		
Cypress (PSoC 6)	CYBLE-416045	2.1
Texas Instruments	CC2640R2	3.30.00.20
Telink	TLSR8258	3.4.0
STMicroelectronics	WB55	1.3.0
STMicroelectroncis	BlueNRG-2	3.1.0
Dialog	DA1469X*	10.0.6
Dialog	DA14585/6*	6.0.12.1020
BLE Version 4.2		
Cypress (PSoC 4)	CYBL11573	3.6
NXP	KW41Z	2.2.1
Dialog	DA14680	1.0.14.X
BLE Version 4.1		
Texas Instruments	CC2540	1.5.0
Dialog	DA14580	5.0.4
Microchip	ATSAMB11	6.2

- SweynTooth Vulnerability related information:

https://asset-group.github.io/disclosures/sweyntooth/#commento-login-box-container

【2】 Developer documentation when Android 4.3 introduced BLE:

https://developer.android.com/guide/topics/connectivity/bluetooth-le

【3】 Zigbee 3.0 official Install code program:

https://zigbeealliance.org/solution/zigbee/

【4】 Rejoin attack:

https://research.kudelskisecurity.com/2017/11/21/zigbee-security-basics-part-3/

https://www.nxp.com/docs/en/supporting-information/MAXSECZBNETART.pdf

Re-using link key: ZigBee allows link keys to be re-used for rejoining the network. Hence, it would make it possible for an attacker to copy a device's addressing credentials and spoof a network layer insecure rejoin using a separate device. This would result in the Trust Center passing the network key encrypted with the previously used link key to the cloned device. As a consequence, an attacker could gain complete access to the network key and hence the entire network.

【5】 Rolling code

The rolling code is used for device communication authentication to prevent replay attacks.

https://ww1.microchip.com/downloads/en/Appnotes/Atmel-2600-AVR411-Secure-Rolling-Code-Algorithm-for-Wireless-Link_Application-Note.pdf

- Industry mature plan:

Keeloq: https://blog.csdn.net/kangweijian/article/details/43491047

DST40: https://en.wikipedia.org/wiki/Digital_signature_transponder

Hitag2: https://blog.csdn.net/spenghui/article/details/71930428

【6】 Nordic OTA rutine

https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v15.3.0%2Fexamples_bootloader.html&cp=5_1_4_4

【7】 Android permission level

https://developer.android.com/guide/topics/permissions/overview?hl=en-us

【8】 Android Webview onReceivedSslError method:

https://developer.android.com/reference/android/webkit/WebViewClient#onReceivedSslError(android.webkit.WebView, %20android.webkit.SslErrorHandler, %20android.net.http.SslError)

【9】 Format string vulnerability

Take snprintf as an example, it will format the variable parameters (...) into a string according to format, and then copy them to str.

- If the length of the formatted string is less than size, copy the string to str and add a string terminator ('\0') after it;
- If the length of the formatted string >= size, only (size-1) characters in it are copied to str, and a string terminator ('\0') is added after it, and the return value is The length of the string to be written.

When the length of the string parameter received by the string manipulation function is too large, it will cause integer overflow and stack overflow. If an attacker embeds malicious code in the string, it may cause information leakage and data forgeing. Therefore, for scenarios where the return value of functions such as snprintf is used for size judgment, it is necessary to strictly check whether the value exceeds the range.