riscure

# Premium Content Delivery on Android: how to guard an open platform

Delivering premium content on a consumer device in a flexible and secure manner requires considerable investment in the software stack. Meanwhile, Android has become the operating system of choice for many embedded devices, both stationary and portable due to its broad hardware support, relative ease of customization and application development. Android also brings new risks that are different from the traditional STB security scenarios. Defending an Android implementation of a secure media path also requires a unique set of methods. This whitepaper provides a high-level overview of Android security risks for the content protection and mitigations required to increase its robustness against common attacks.

## Android's Security Risks Overview: Multiple ways to repurpose the device

The Android platform gives the user a variety of options to interface with the device. Even though this gives the user more freedom, it also gives the user the ability to repurpose the device. Additionally, it widens the attack surface immensely, compared to traditional STBs. Since Android is open source, everything an attacker needs to know can be found online. Additionally, Android is supported by a vast community with extensive knowledge of the platform.

Before running the Android OS, a secure boot implementation has to be in place that authenticates each boot stage up to and including the Android OS. Any vulnerability in one of these boot stages can compromise all the following boot stages. The secure boot stage must be developed with security in mind and needs to implement secure coding practices to prevent logical errors. It should also include mechanisms to defend against fault injection and side channel analysis if those mechanisms are not sufficiently provided by the hardware. Android provides a feature called dm-verity to support the secure boot process. However, there are tools available to bypass the secure boot of the Android OS and the following stages, e.g., TWRP. At Riscure we are helping our customers to mitigate this risk by ensuring a desired level of security during the earlier boot stages.

Updates are rolled out to enhance the product, but also to resolve security vulnerabilities. An anti-rollback system prevents an attacker from flashing older firmware on the device. The older firmware could have publicly known vulnerabilities that can be exploited. The anti-rollback system can use specific hardware, e.g., OTP, to prevent these downgrade attacks. An example is the TrustZone downgrade attack, which is run in conjunction with Android most of the times.

When running the Android environment, a user might repurpose the device by installing, e.g. a Netflix application. It will cause the user not to have to purchase the premium content of the provider, and hence diminishing the revenue. It could also involve other applications, not specific to video content, e.g., gaming purposes.

Rooting an Android could be as easy as installing an application and pressing a big button. Even though some devices might not be listed as supported, the underlying platform is not very different. With some minor tweaking, these services can be used to root an unsupported device as well. Known publicly available services are Kingroot and Magisk.

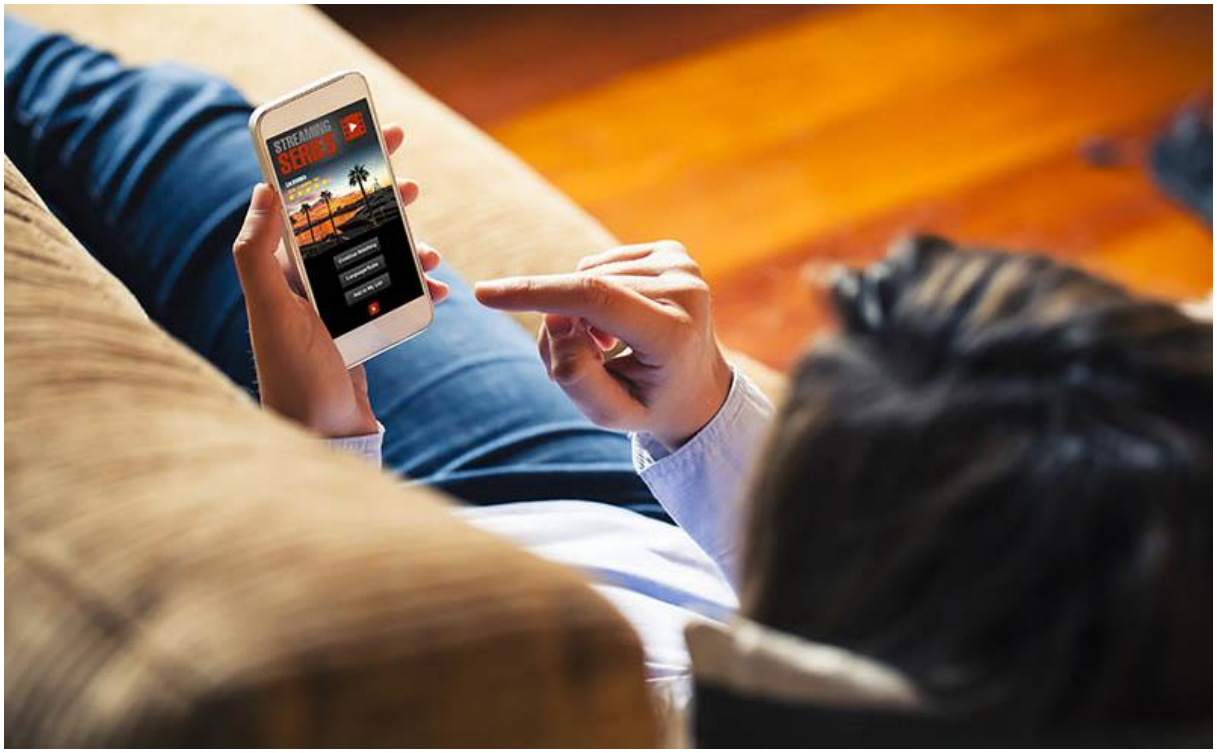
Android comes with more malware already existing, which is a different scenario compared to traditional STBs, where malware had to be customized for every type of an STB. Malware makes it possible for an attacker to repurpose the device by making it part of a botnet and making it part of the denial of service attacks towards other entities. An example of this kind of malware is Sockbot.

Malware can also repurpose the Android TV to make it part of a cryptocurrency mining network. An example is a Trojan called Loapi, and BadLepricon. In this case, an attacker gains the profit of the calculations performed on the Android TV. Additionally, the Android TVs need to have certain hardware specifications to be compliant (if desired by the manufacturer), making the hardware better than a regular router or STB, and thus more suitable for cryptocurrency mining. Mining cryptocurrency wears out the hardware, diminishing the lifetime of

the device and making equipment replacement necessary.

Since the Android TV can be connected to the internet, it is also vulnerable to remote attackers. Over the past years, there have been several remote execution vulnerabilities, which can be used to abuse and repurpose the device to do whatever functionality the attacker wants, including installing ransomware like SLocker. Each major update of Android is accompanied by a change log, which includes the kind of vulnerabilities that have been resolved and a CVE number belonging to that vulnerability. Attackers can use this information, in addition to diffing the patched firmware with the original firmware, to develop an attack to gain remote code execution on any device that has not yet received the patch.

Web browsers are widely used on Android TV devices, both as standalone apps (such as Chrome or Firefox) and as embedded browser frames in various applications. The complexity of the modern web technologies on the browser side opens up a wide variety of opportunities to the attacker, even if they do not result in a full compromise of the device. Malware embedded in a web page, for example in JavaScript code, can have several results, ranging from persistence in the browser for some minutes and stealing CPU time, to exploiting the entire system. The malware might also reside in malicious advertising, which is not always under the control of the provider of the content of the page, since it is part of an external advertisement framework.



## Defending your backend system

Android TVs give more freedom to the user, and hence also to an attacker. An attacker can misuse the device to attack the backend system. Hopefully, the backend system already authenticates each device it is communicating with. However, with Android TV it becomes easier to hijack the device and use that device to communicate with your backend system. Therefore, simple authentication of the device is not enough.

Remote attestation is a popular technology used to authenticate the hardware and software configuration of a potentially untrusted client device. Remote attestation could allow the backend system to detect changes on a client device. Depending on the exact security solution used for remote attestation, it could be possible to bypass this feature with some additional attacker effort. Therefore, the backend system should still be wary of information passed to it and enforce proper input sanitation.

The backend system will also be the gateway to multiple similar devices, belonging to other users. The backend system needs to protect against attacks that would use it to communicate with those devices. Gaining access to other devices could result in any of the repurposing attacks for those devices as well. An example attack was presented at CCC 32C3, in a presentation titled “Beyond your cable modem. How not to do DOCSIS networks”, where the backend system allowed for devices to interact with each other and did not protect login credentials to the other boxes.

## Denial-of-service attacks

Denial-of-service attacks prevent a user from watching the content. Hence the operator misses out on revenue. Additionally, the time of not being able to use the service might be obliged for reimbursement for the operator.

Android TVs are connected to the internet, making them vulnerable to remote attacks. Both successful and failed remote attacks could crash a device, effectively resulting in a denial of service attack. An example is an open network port running a vital service for operation of the device, which parses malicious data incorrectly and crashes.

There are also local denial-of-service attacks. A malicious application could result in a significant CPU usage, making the device slow and unresponsive. The HiddenMiner is such an application, which uses the CPU for mining cryptocurrency. Another local denial-of-service attack is consuming a lot of memory (either storage or working memory), such that no memory is left for the other legitimate applications.

There are transient and persistent denial-of-service attacks. Some of the previous could be transient, where a reboot of the device might resolve the issue. An example of a persistent denial-of-service attack is setting the version of the software to be used to the highest value. This version is stored in internal chip memory that can only be set once and cannot be reverted (OTP). Therefore, the device cannot roll-back to a previous version, and cannot boot using the current firmware versions. The manufacturer could choose to set the firmware versions to the highest possible to prevent this attack, but that comes with the downside that any vulnerability in that firmware cannot be reverted or fixed.

In the worst case scenario, a distributed denial-of-service (DDoS) attack is performed against the provider itself or a specific part of its infrastructure. Such a scenario can lead to a malfunction of the update mechanism, and the only way to mitigate the attack will be the manual replacement of the STBs for many infected users.

Even though denial of service attacks are hard to mitigate, several countermeasures could be implemented to reduce the risk. A strictly configured firewall could mitigate part of the risk for a remote attack. Proper resource management can mitigate the risk for a local attack.

## User privacy violation / identity theft

Android TVs typically store a lot of data. This data may include usernames, passwords, email addresses, and up to credit card information. Android provides a secure storage mechanism that can be used for this, but this isn't always used by every application. This personal data can sometimes be easily recovered from the flash embedded in the device. Android doesn't enforce a common interface to securely store user data, except for some sensitive data, e.g., a fingerprint.

Most operators will provide functionality to purchase content on demand, by, e.g. paying with a credit card number or personal PIN. The operator needs to maintain that this information does not leak and needs to guarantee the confidentiality of the data. Personal data should be protected when it is not used, and when it is used. While processing personal data, it could be protected using anti-tampering and anti-debugging mechanisms. Obfuscation of code handling the data also makes it harder to recover the data. When not used, it should be stored encrypted, and only be decrypted when it needs to be used. Decryption can be performed using hardware cryptographic engines. When there is no such hardware available, white-box cryptography could be used.

## Content protection in Android: security foundations

Securing the premium content on an open platform is a challenge that can only be overcome with layered defenses in hardware and software.

### Trusted hardware and software

Content protection starts with a robust implementation of the secure boot process. Several CAS solutions leverage the secure media path to protect the content. Additionally, a trusted execution environment (TEE) is used to control this secure media path. The TEE relies on features implemented in hardware, e.g., memory separation and execution separation. There are several hardware solutions on the market, e.g., ARM TrustZone.

A TEE cannot exist without proper configuration. The configuration of the hardware features to support a TEE can be performed during boot (in the secure boot process) or runtime. Any misconfiguration could result in content recovery. An example of a misconfiguration is the configuration of the secure memory, such that the Rich execution environment (REE) can access (parts of) the content due to the content being written outside of the secure memory range because the range is too small.

**Content key recovery**

The content is encrypted using a key, which is embedded in the transmission. However, this key is protected by encrypting it with another key. To recover the key, a hardware feature could be used, which is then forwarded to the descrambler and not available to any other hardware peripheral. The hardware feature provides several countermeasures against logical attacks, but also fault injection and side-channel attacks.
Because the OS can run on several devices, there is no guarantee that each device will have the same hardware interface. Therefore, a software implementation called white-box cryptography (WBC) is used. WBC does not provide the same level of security that hardware can, but could provide enough security based on the mode of operation.

## Security evaluation: assurance for your solution

A software solution can never be perfect, and vulnerabilities will continue to exist. However, an indication of the assurance for a secure solution can be provided. Riscure is perfectly situated to provide such an assessment, as we are involved in the secure development process from the beginning (the making of the chip), all the way to the end (the final product).

Good security begins at the core, which is the hardware of the chip. Riscure performs security evaluations of multiple chips used in the content protection market, which are approved by the CA vendors. These chip evaluations start with vulnerability analysis, followed by a testing phase. It covers a range of possible attack methods, e.g., logical attacks, fault injection attacks, side-channel analysis, and physical attacks. Every security feature is analyzed in detail. The start of the secure boot process is verified by a source code review.
Riscure performs follow-up evaluations in addition to the chip hardware evaluation. The evaluation includes source code reviews of bootloaders residing in flash which are executed in a secure state. Additionally, a source code review of the TEE OS is verified.

These evaluations should give a good assurance indication of the chip and its embedded software. Such package could be delivered to the following parties, e.g., STB manufacturers. Hence, the overall security of the device is thoroughly evaluated. The STB manufacturer, operator, and any other third party combine this into a final product, which includes, e.g. the PCB layout and the graphical user interface presented to the user.

The final product can be evaluated by Riscure using two approaches, white-box, and black-box. The final product is evaluated by evaluating all the interfaces, uncovering potential vulnerabilities. All we need is the final product and documentation (depending on the test methodology). Riscure helps their customers by providing assurance related to the security mechanisms responsible for protecting the relevant assets within the device even when the attackers have root access.

**Riscure B.V.**
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90
www.riscure.com

**Riscure North America**
550 Kearny St., Suite 330
San Francisco, CA 94108 USA
Phone: +1 650 646 99 79
inforequest@riscure.com

**Riscure China**
Room 2030-31, No. 989, Changle Road, Shanghai 200031
China
Phone: +86 21 5117 5435
inforcn@riscure.com