The EXFILES project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 883156

Auditing Closed Source Trusted Applications for Qualcomm Secure Execution Environment (QSEE)

Hector Marco - Fernando Vañó

CYBER INTELLIGENCE

DeepSec 2022

EX FILES

Presentation Outline

- Who we are
- Motivation
  - Mobile Devices
  - ARM TrustZone
  - Security of Trusted Applications (TAs)
- Qualcomm Secure Execution Environment (QSEE)
  - QSEE Overview
  - QSEE OS Versions & TA Loading
- Auditing Trusted Applications
  - TA Debugger
  - TA Fuzzer
- Results and Conclusions

0. Who we are

Dr. Hector Marco

- Founder of Cyber Intelligence S.L.
- Working in Cybersecurity > 15 years

Dr. Fernando Vañó

- Software team leader
- Specialized in smartphone security

Cyber Intelligence S.L.

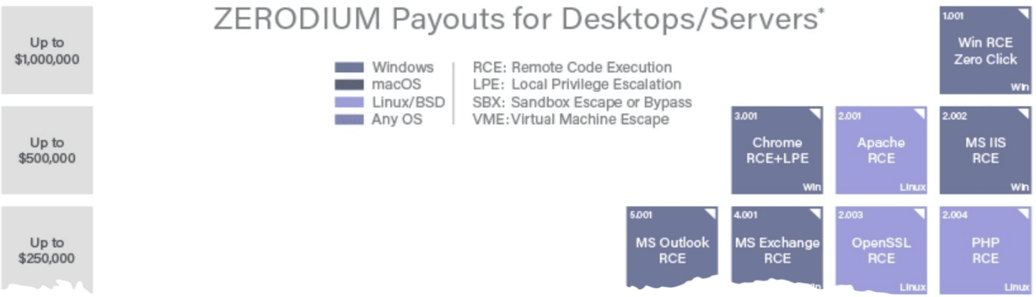
- Company based in Spain.
- Specialized in software and hardware security.
- `https://cyberintel.es`
- `security@cyberintel.es`

1. Motivation

1.1 Mobile Devices

- Computing power has been increasing
- Usage of mobile devices is heavily widespread
- Sensitive and valuable information
  - Interesting for attackers

“The evolution of the desk”
by The Harvard Innovation Labs

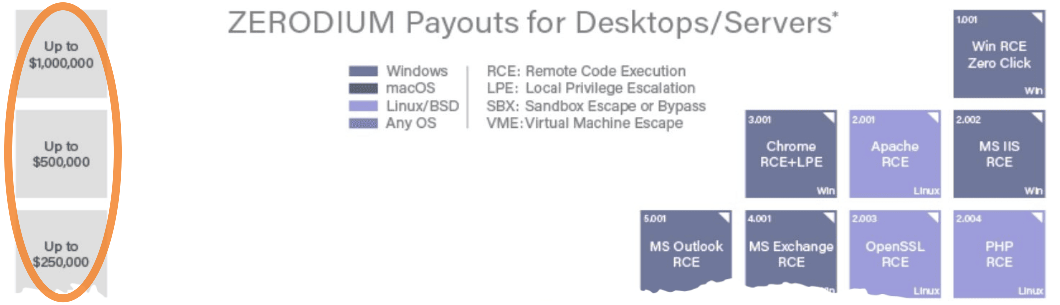


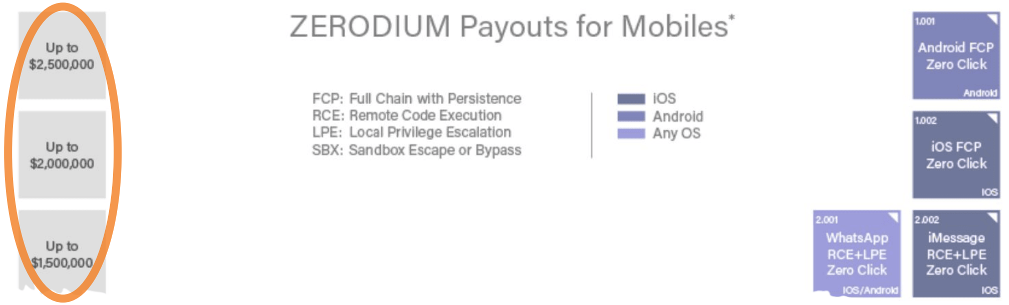
1.1 Mobile Devices

- Computing power has been increasing
- Usage of mobile devices is heavily widespread
- Sensitive and valuable information
  - Interesting for attackers

“The evolution of the desk”
by The Harvard Innovation Labs

$1M vs $2.5M Payouts in Zerodium





1.1 Mobile Devices

What are the main factors for those payouts ?

- ALL our live is in our smartphones.
- A full compromise in a smartphone should be more difficult than in a desktop/server.
- Getting `root` in `Android` does not mean attackers are done!
- Smartphones have 2 worlds: <u>Normal and Secure</u>
- We will focus on Android but iOS provides similar security features.

1.1 Mobile Devices

Normal world

Graphical Interface

- Apps, libs, Kernel.
- Attackers with `root` permissions can not access sensitive information.
- This information in handled in the secure world.

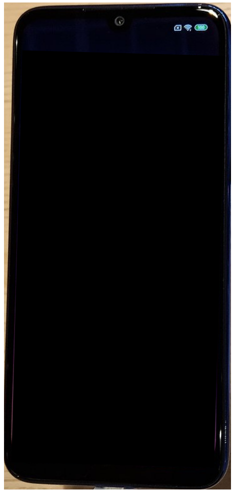
1.1 Mobile Devices

Normal world

Graphical Interface

- Apps, libs, Kernel.
- Attackers with `root` permissions can not access sensitive information.
- This information in handled in the secure world.

Secure world



NO Graphical Interface

- Apps, libs, kernel.
- No `libc` or `Linux`.
- Running in the “backgroud”.
- The real target of the attackers.

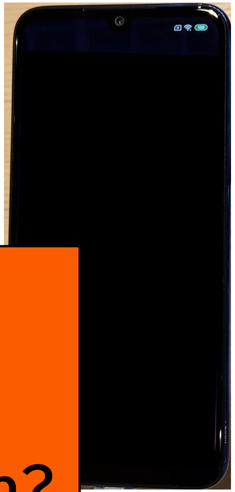
1.1 Mobile Devices

Normal world

Graphical Interface

- Apps, libs, Kernel.
- Attackers with `root` permissions can not access sensitive information.
- This information in handled in the secure world.

Secure world



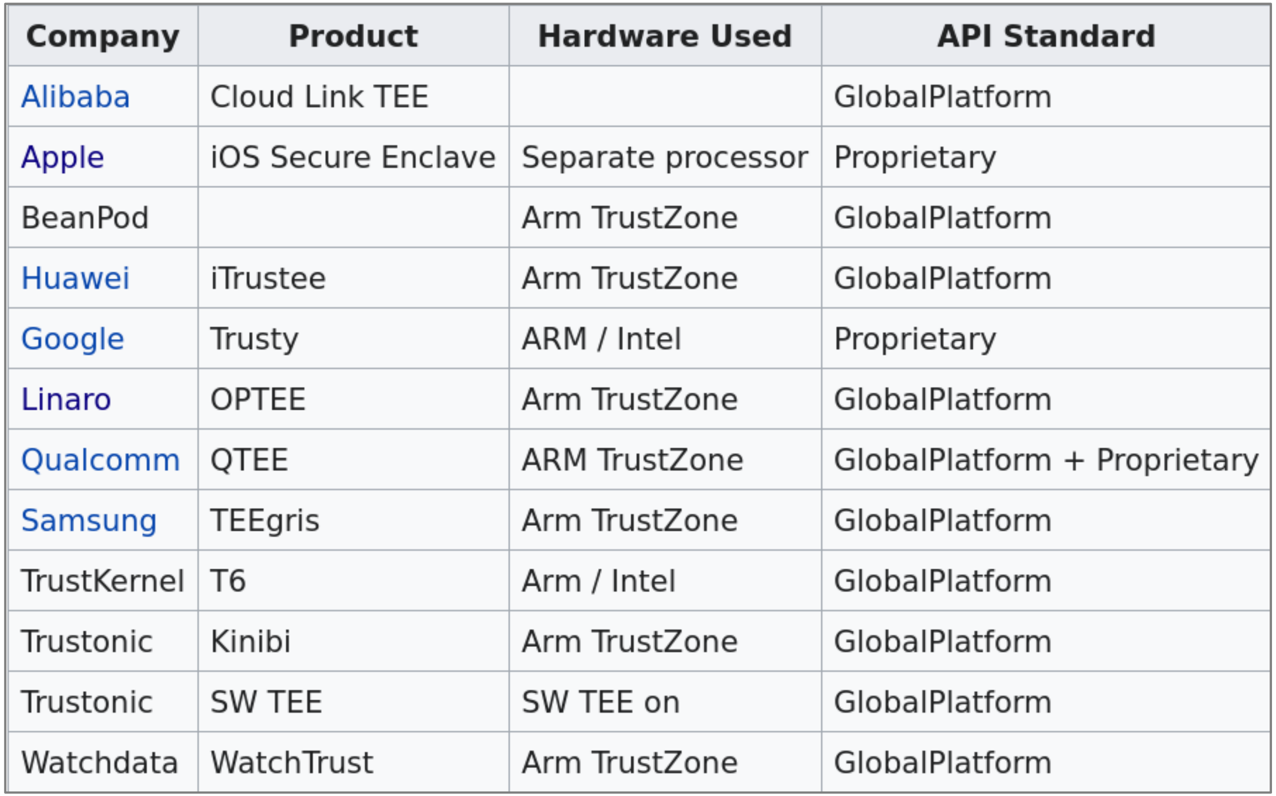
NO Graphical Interface

- Apps
- No l
- Runn
- The real target of the attackers.

Is this world audited enough?

1.1 Mobile Devices

- Challenge when auditing: Different O.S providing the TEE environment:

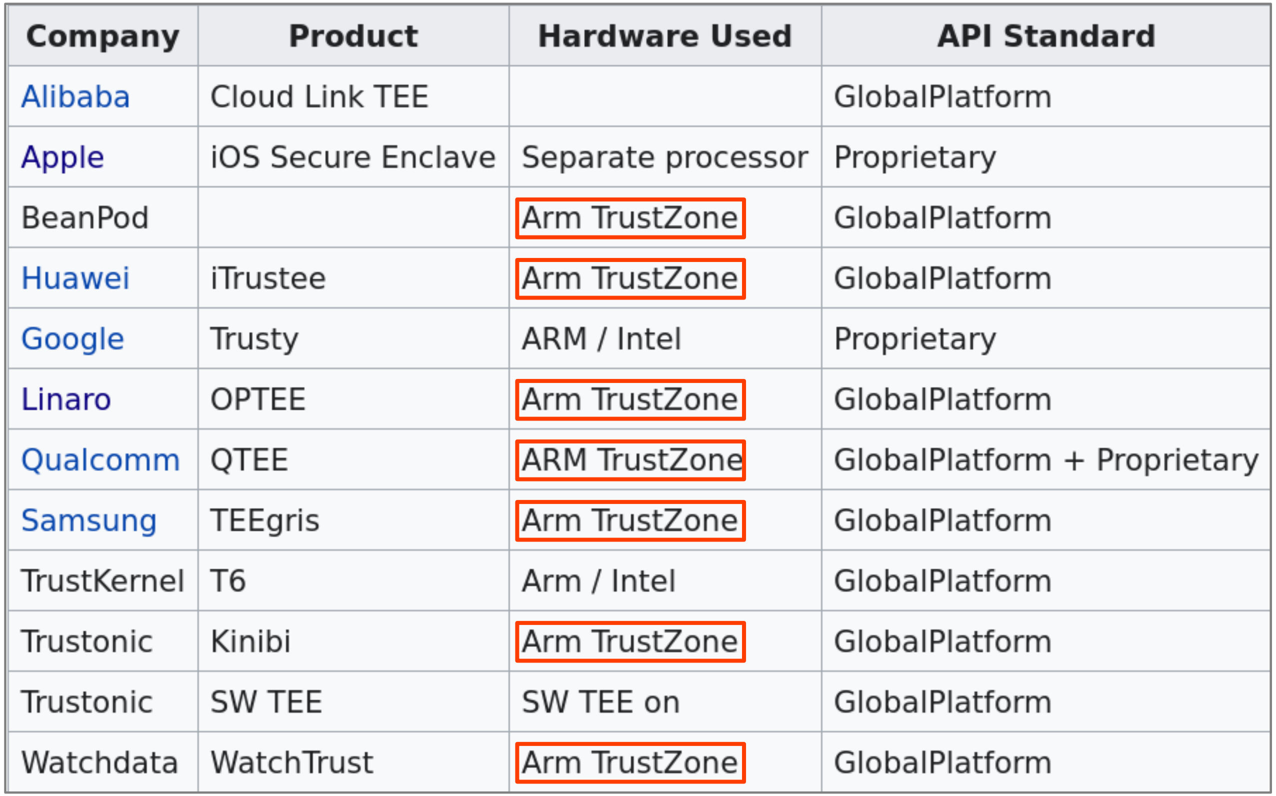


Company	Product	Hardware Used	API Standard
Alibaba	Cloud Link TEE		GlobalPlatform
Apple	iOS Secure Enclave	Separate processor	Proprietary
BeanPod		Arm TrustZone	GlobalPlatform
Huawei	iTrustee	Arm TrustZone	GlobalPlatform
Google	Trusty	ARM / Intel	Proprietary
Linaro	OPTEE	Arm TrustZone	GlobalPlatform
Qualcomm	QTEE	ARM TrustZone	GlobalPlatform + Proprietary
Samsung	TEEgris	Arm TrustZone	GlobalPlatform
TrustKernel	T6	Arm / Intel	GlobalPlatform
Trustonic	Kinibi	Arm TrustZone	GlobalPlatform
Trustonic	SW TEE	SW TEE on	GlobalPlatform
Watchdata	WatchTrust	Arm TrustZone	GlobalPlatform

TEE Operating Systems

1.1 Mobile Devices

- Challenge when auditing: Different O.S providing the TEE environment:

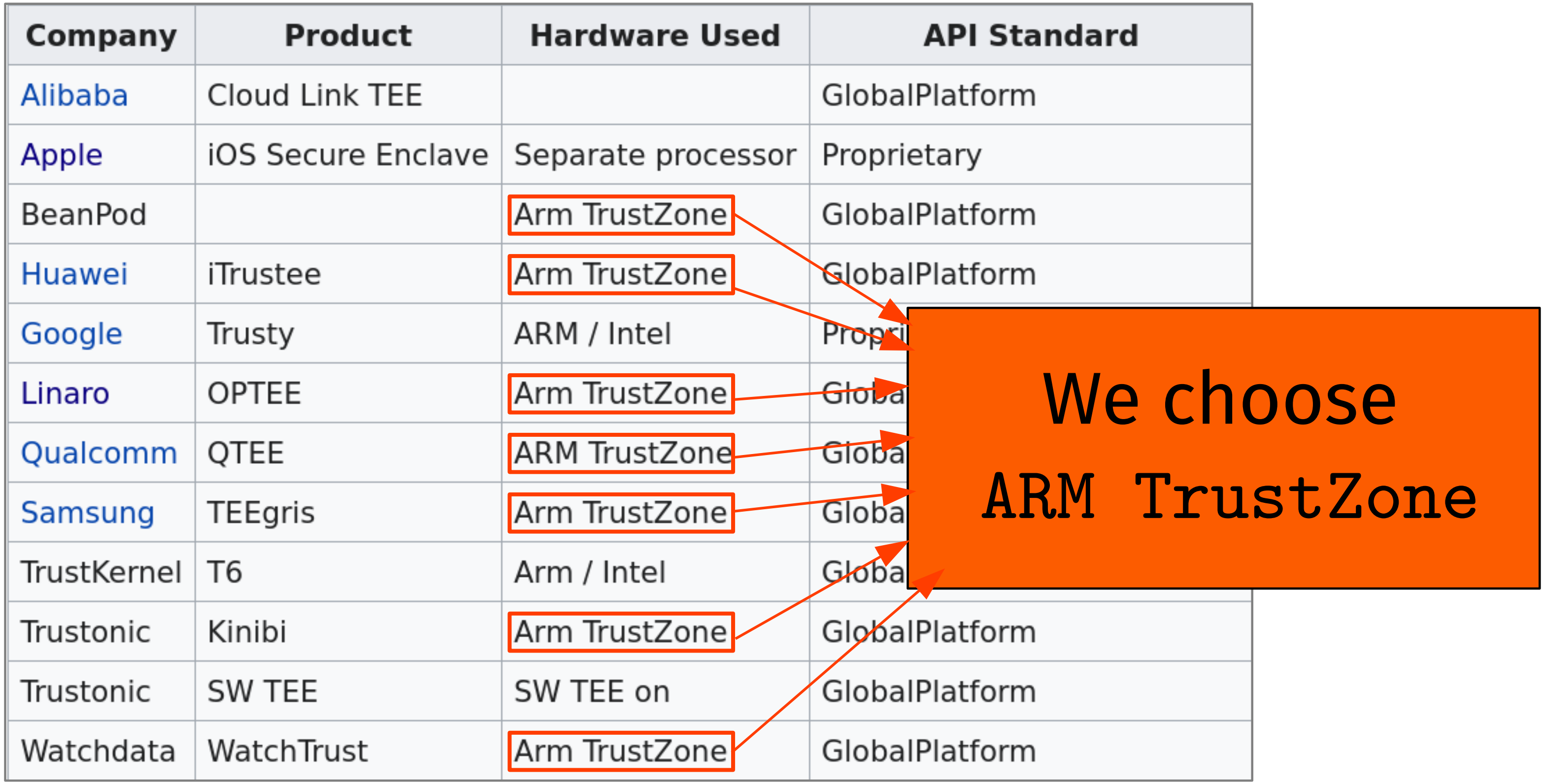


Company	Product	Hardware Used	API Standard
Alibaba	Cloud Link TEE		GlobalPlatform
Apple	iOS Secure Enclave	Separate processor	Proprietary
BeanPod		Arm TrustZone	GlobalPlatform
Huawei	iTrustee	Arm TrustZone	GlobalPlatform
Google	Trusty	ARM / Intel	Proprietary
Linaro	OPTEE	Arm TrustZone	GlobalPlatform
Qualcomm	QTEE	ARM TrustZone	GlobalPlatform + Proprietary
Samsung	TEEgris	Arm TrustZone	GlobalPlatform
TrustKernel	T6	Arm / Intel	GlobalPlatform
Trustonic	Kinibi	Arm TrustZone	GlobalPlatform
Trustonic	SW TEE	SW TEE on	GlobalPlatform
Watchdata	WatchTrust	Arm TrustZone	GlobalPlatform

TEE Operating Systems

1.1 Mobile Devices

- Challenge when auditing: Different O.S providing the TEE environment:



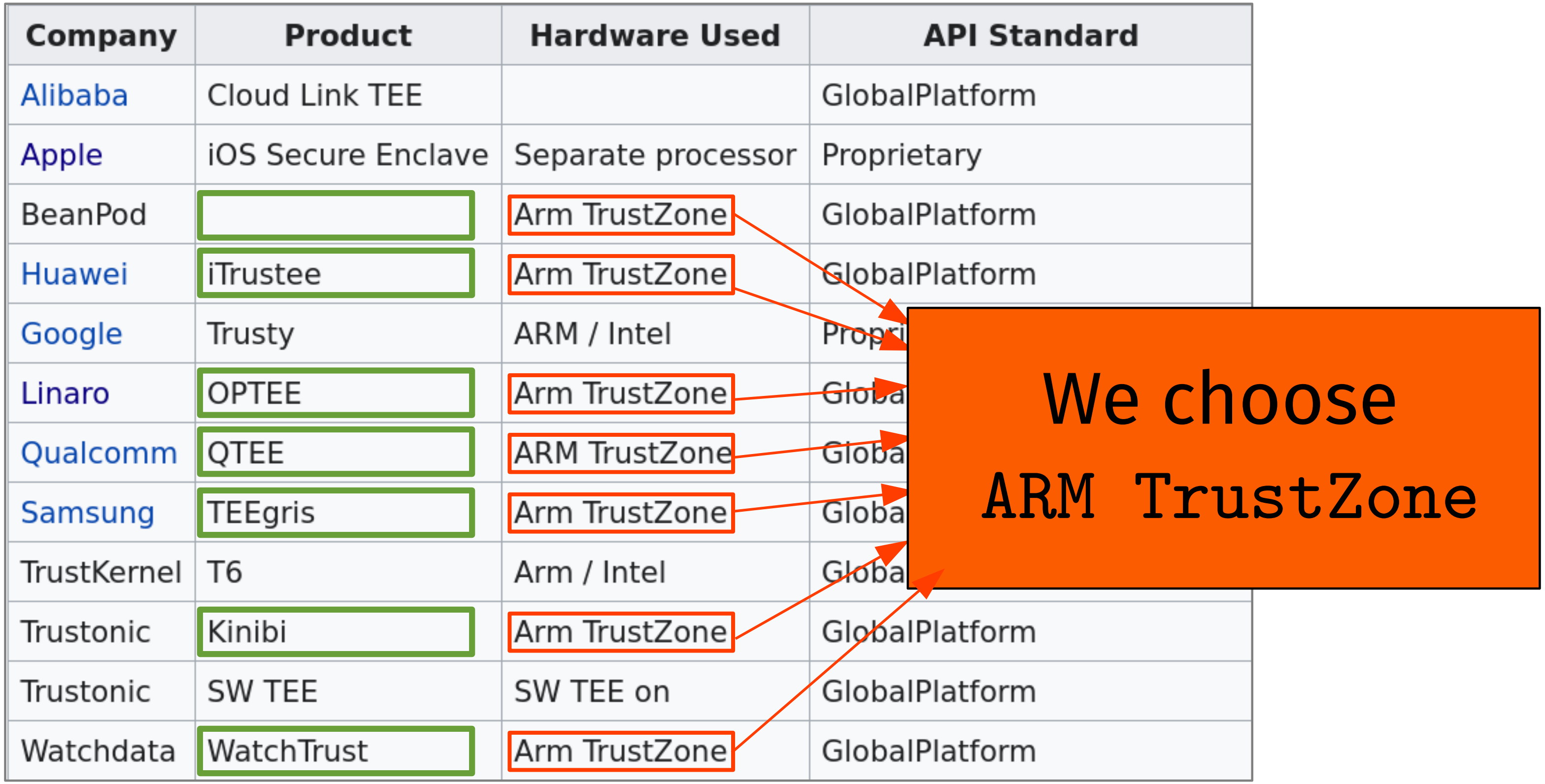
Company	Product	Hardware Used	API Standard
Alibaba	Cloud Link TEE		GlobalPlatform
Apple	iOS Secure Enclave	Separate processor	Proprietary
BeanPod		Arm TrustZone	GlobalPlatform
Huawei	iTrustee	Arm TrustZone	GlobalPlatform
Google	Trusty	ARM / Intel	Propri
Linaro	OPTEE	Arm TrustZone	Globa
Qualcomm	QTEE	ARM TrustZone	Globa
Samsung	TEEgris	Arm TrustZone	Globa
TrustKernel	T6	Arm / Intel	Globa
Trustonic	Kinibi	Arm TrustZone	GlobalPlatform
Trustonic	SW TEE	SW TEE on	GlobalPlatform
Watchdata	WatchTrust	Arm TrustZone	GlobalPlatform

We choose ARM TrustZone

TEE Operating Systems

1.1 Mobile Devices

- Challenge when auditing: Different O.S providing the TEE environment:



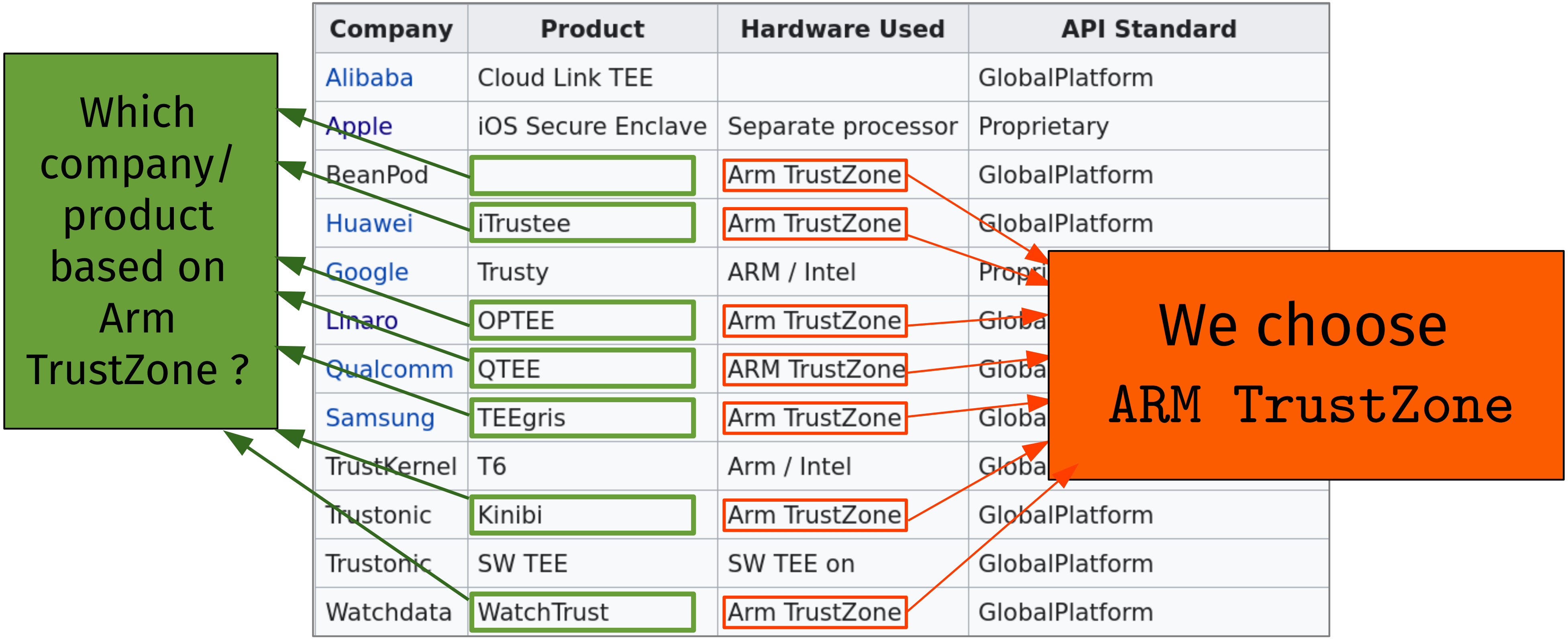
Company	Product	Hardware Used	API Standard
Alibaba	Cloud Link TEE		GlobalPlatform
Apple	iOS Secure Enclave	Separate processor	Proprietary
BeanPod		Arm TrustZone	GlobalPlatform
Huawei	iTrustee	Arm TrustZone	GlobalPlatform
Google	Trusty	ARM / Intel	Propri
Linaro	OPTEE	Arm TrustZone	Globa
Qualcomm	QTEE	ARM TrustZone	Globa
Samsung	TEEgris	Arm TrustZone	Globa
TrustKernel	T6	Arm / Intel	Globa
Trustonic	Kinibi	Arm TrustZone	GlobalPlatform
Trustonic	SW TEE	SW TEE on	GlobalPlatform
Watchdata	WatchTrust	Arm TrustZone	GlobalPlatform

We choose ARM TrustZone

TEE Operating Systems

1.1 Mobile Devices

- Challenge when auditing: Different O.S providing the TEE environment:



Which company/ product based on Arm TrustZone ?

Company	Product	Hardware Used	API Standard
Alibaba	Cloud Link TEE		GlobalPlatform
Apple	iOS Secure Enclave	Separate processor	Proprietary
BeanPod		Arm TrustZone	GlobalPlatform
Huawei	iTrustee	Arm TrustZone	GlobalPlatform
Google	Trusty	ARM / Intel	Propri
Linaro	OPTEE	Arm TrustZone	Globa
Qualcomm	QTEE	ARM TrustZone	Globa
Samsung	TEEgris	Arm TrustZone	Globa
TrustKernel	T6	Arm / Intel	Globa
Trustonic	Kinibi	Arm TrustZone	GlobalPlatform
Trustonic	SW TEE	SW TEE on	GlobalPlatform
Watchdata	WatchTrust	Arm TrustZone	GlobalPlatform

We choose ARM TrustZone

TEE Operating Systems

1.2 ARM TrustZone

TEEs based on TrustZone

- **Qualcomm's Secure Execution Environment (QSEE)**
  - Xiaomi, Motorola, LG, HTC, Sony, Google Nexus and Pixel series
- Trustronic's Kinibi
  - Old Samsung Galaxy devices
- TEEGRIS
  - Samsung devices since Galaxy S10
- HiSilicon's Trusted Core
  - Huawei
- Google's Trusty TEE
  - Newer Google Pixel series
- Linaro OP-TEE

1.2 ARM TrustZone

TEEs based on TrustZone

- **Qualcomm's Secure Execution Environment (QSEE)**
  - Xiaomi, Motorola, LG, HTC, Sony, Google Nexus and Pixel series
- Trustronic's Kinibi
  - Old Samsung Galaxy devices
- TEEGRIS
  - Samsung devices since Galaxy S10
- HiSilicon's Trusted Core
  - Huawei
- Google's Trusty TEE
  - Newer Google Pixel series
- Linaro OP-TEE

We choose

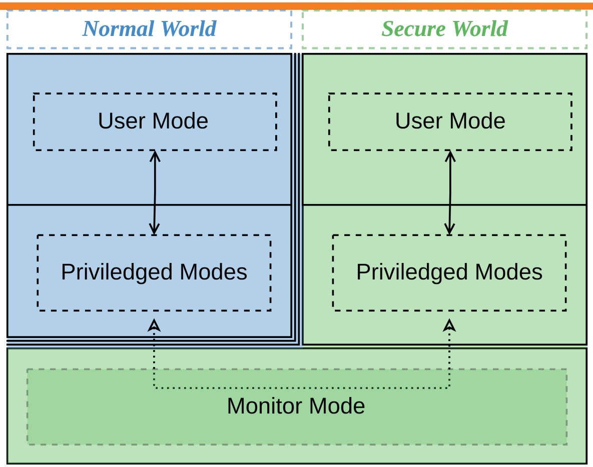
QSEE

1.2 ARM TrustZone

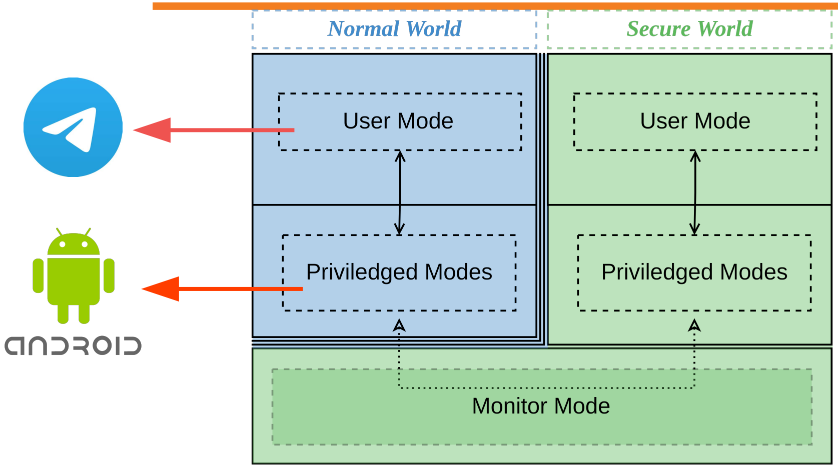
- We have defined what we are going to audit:
  - The secure world of smartphones
  - Based on Arm TustZone
  - Qualcomm’s Secure Execution Environment (QSEE)

- Let’s explore the Trusted Execution Environment (TEE)
  - What kind of applications are designed to run in the secure world?
  - How the normal world interacts with the secure world?
  - How can we load applications in the secure world from the normal one?

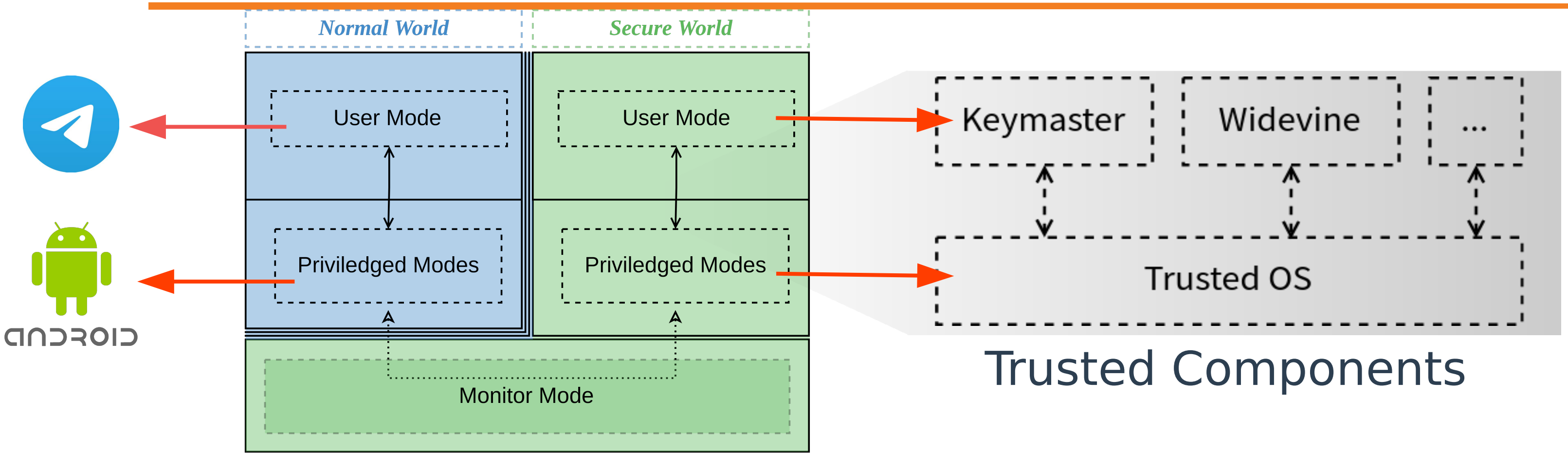
1.2 ARM TrustZone



1.2 ARM TrustZone

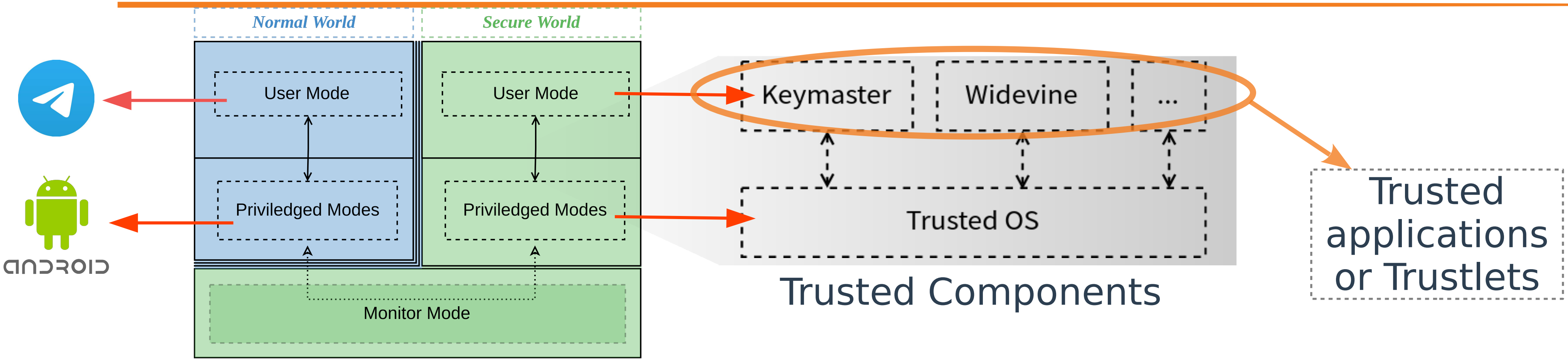


1.2 ARM TrustZone

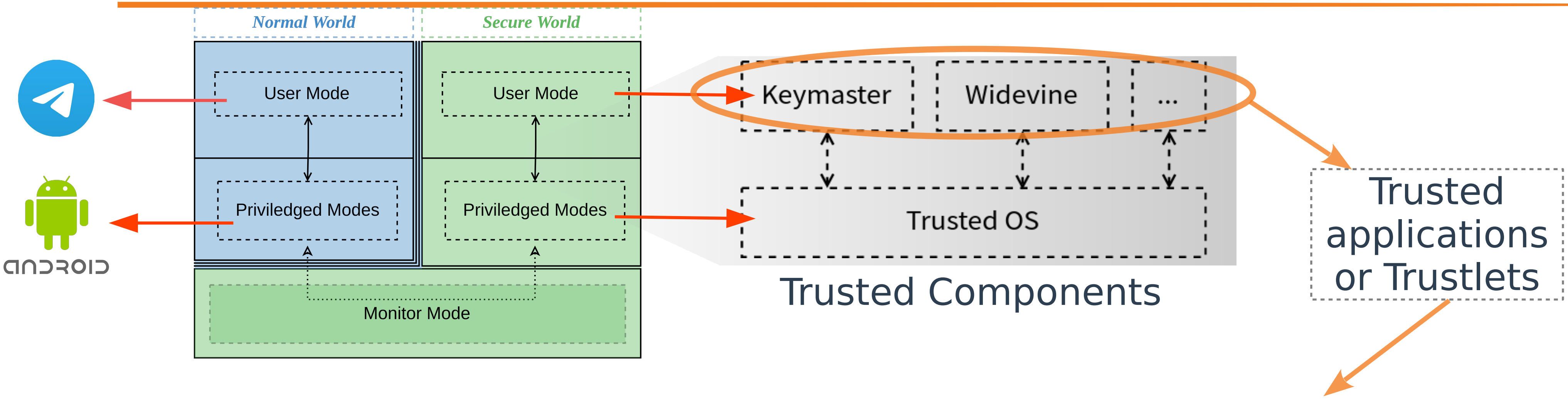


Trusted Components

1.2 ARM TrustZone



1.2 ARM TrustZone



Normal World

Secure World

User Mode

User Mode

Priviledged Modes

Priviledged Modes

Monitor Mode

Keymaster

Widevine

...

Trusted OS

Trusted Components

Trusted applications or Trustlets

TA Name	Description / Usage
Keymaster	Android Hardware-Backed Keystore
Widevine	Digital Rights Management (DRM)
PlayReady	
SecureFP	Fingerprint Sensor Services
Prov	Device Root Key (DRK) Provisioning

1.3 Security of Trusted Applications

Assumption
TEE is trusted

1.3 Security of Trusted Applications

Assumption

TEE is trusted

Challenges to verify the Assumption

- Closed-source OS and Apps
  - Security through obscurity
  - Requires considerable efforts to assess their security
- No publicly available emulators
  - Need to debug and audit
- Trusted OS and Applications **can also have vulnerabilities**

1.3 Security of Trusted Applications

Assumption

TEE is trusted

Challenges to verify the Assumption

- Closed-source OS and Apps
  - Security through obscurity
  - Requires considerable efforts to assess their security
- No publicly available emulators
  - Need to debug and audit
- Trusted OS and Applications **can also have vulnerabilities**

Examples of TA vulnerabilities:

- CVE-2014-9974
  - Keymaster TA not validating buffer lengths
- CVE-2015-6639
  - Privilege Escalation exploiting Widevine TA
- CVE-2015-9183
  - Integer overflow in TQS Trusted Application
- CVE-2016-0825
  - Widevine leaking data from secure storage
- CVE-2020-11221
  - Extraction of Trusted OS diagnostic info
- …

1.3 Security of Trusted Applications

→ **How to enforce the security of the TEE?**

- Formal verification
- Stronger isolation
- Finding bugs

→ **Goal: <u>Develop tools to assist in debugging and auditing TAs</u>**

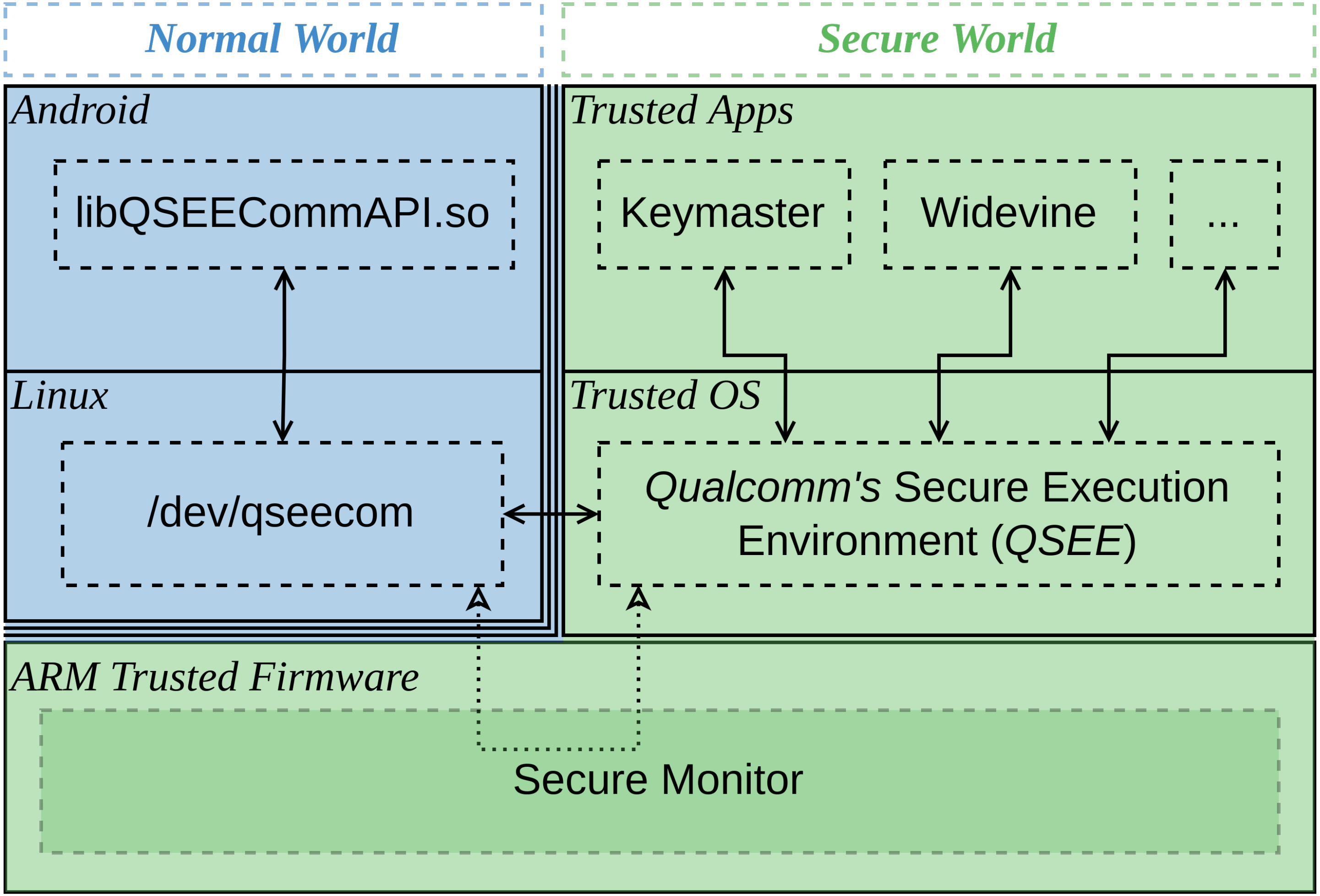
- Easier to understand the behaviour of a TA
- Find attack surfaces
- Be able to fuzz
- Propose fixes in cases where security issues are raised

2. Qualcomm Secure Execution Environment

2.1 QSEE Overview

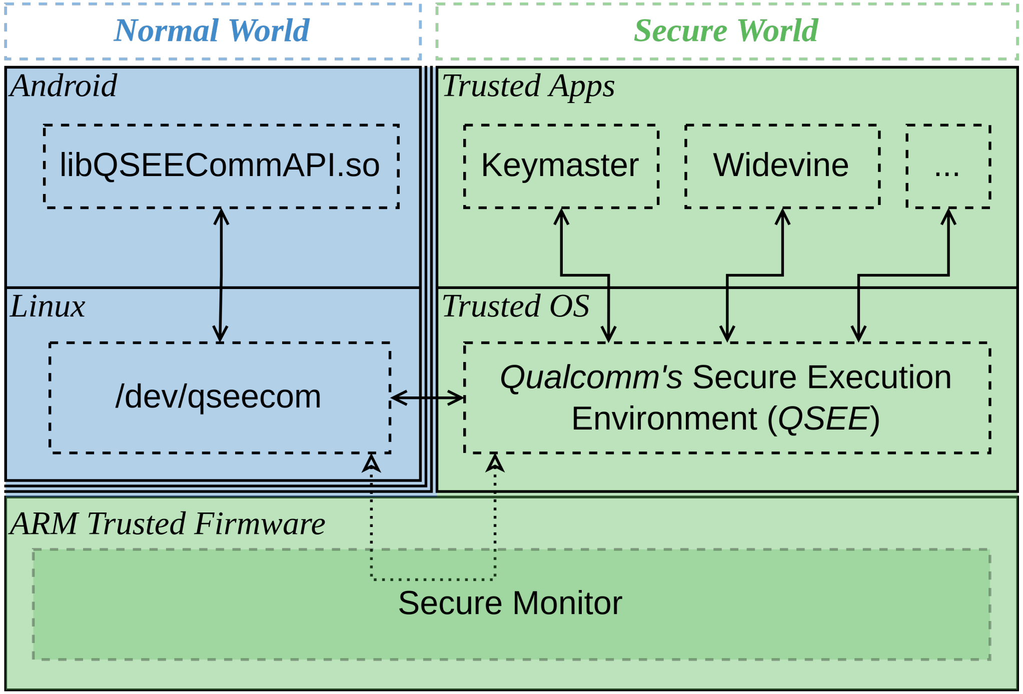
- Qualcomm’s commercial TrustZone-based TEE solution
- Also known as Qualcomm Trusted Execution Environment (QTEE)
- Closed-Source
- Limited public information about its internals
- Widely utilized by different mobile devices
  - Xiaomi, Motorola, LG, HTC, Sony, Google Nexus and Pixel series

2.1 QSEE Overview



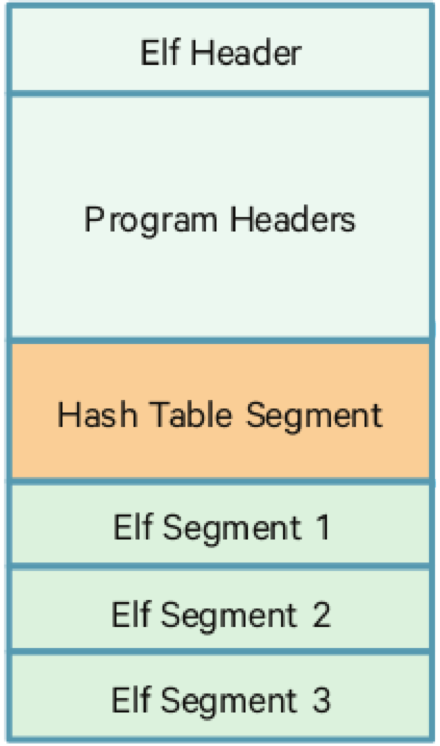
2.1 QSEE Overview

- The chip (SoC) now implements two execution contexts (NW | SW)
- A Secure monitor switches between the NW and SW
- SW offers services to NW
- Memory and IO separation between NW and SW
- Only signed Trustlets are allowed to be loaded in the SW



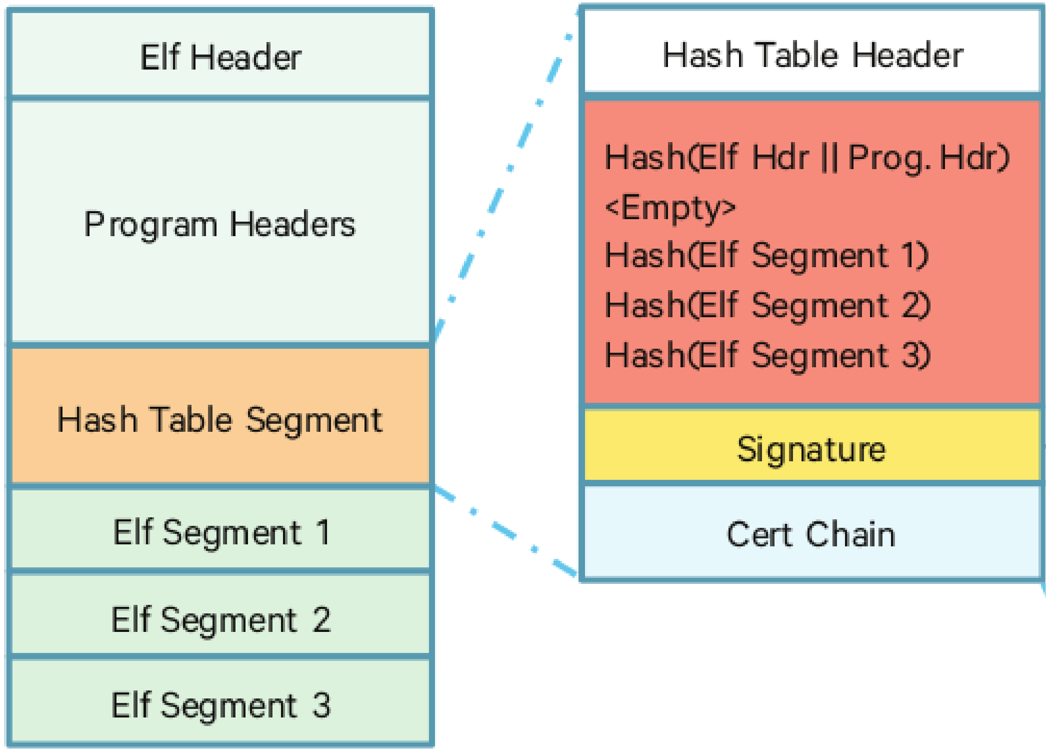
2.2 QSEE Signatures and OS Versions

- Signatures when loading a Trustlet:



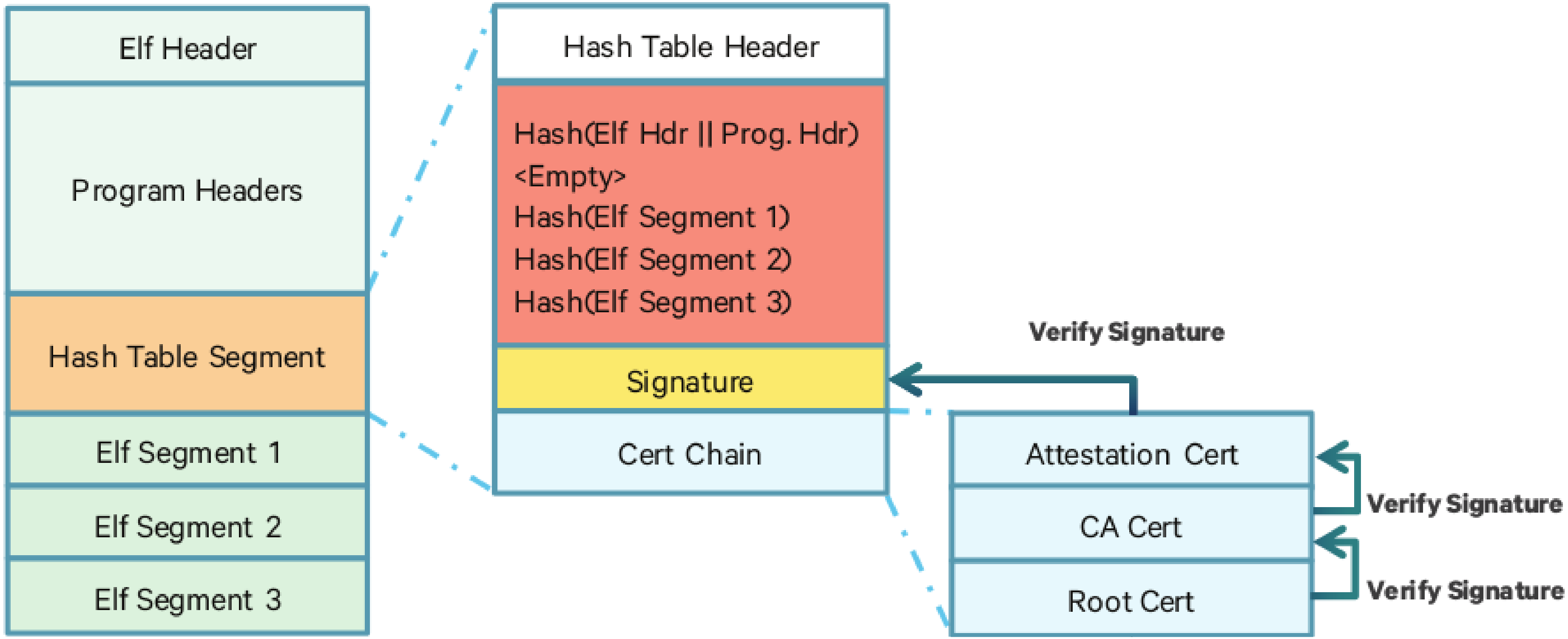
2.2 QSEE Signatures and OS Versions

- Signatures when loading a Trustlet:



2.2 QSEE Signatures and OS Versions

- Signatures when loading a Trustlet:

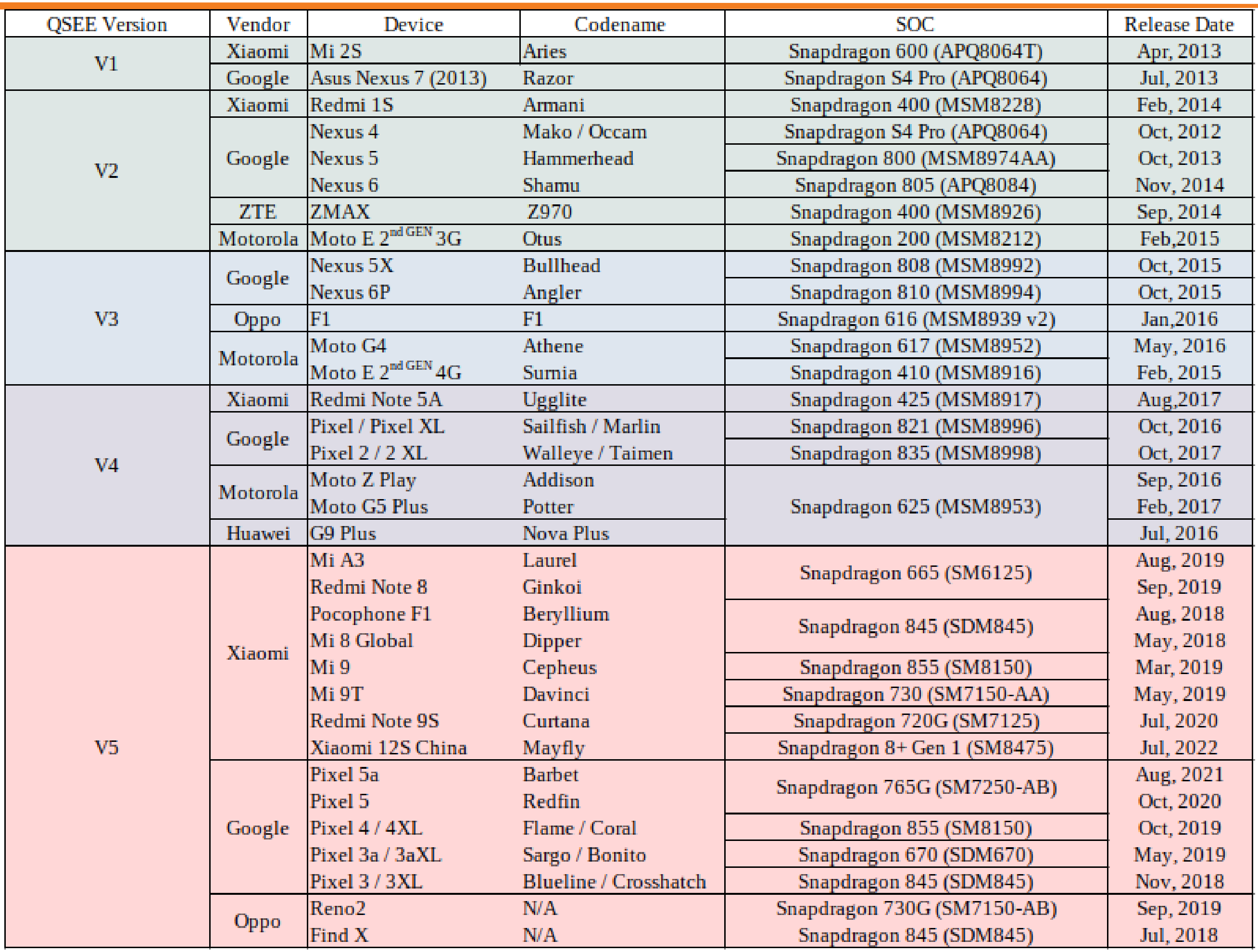


2.2 QSEE Signatures and OS Versions

- To be able to develop tools to emulate and fuzz Trustlets:
  - Load Trustlets as the secure kernel does
  - We can skip the signature verification
  - Simulate syscalls
  - Simulate hardware
  - Prepare stack, heap, etc.

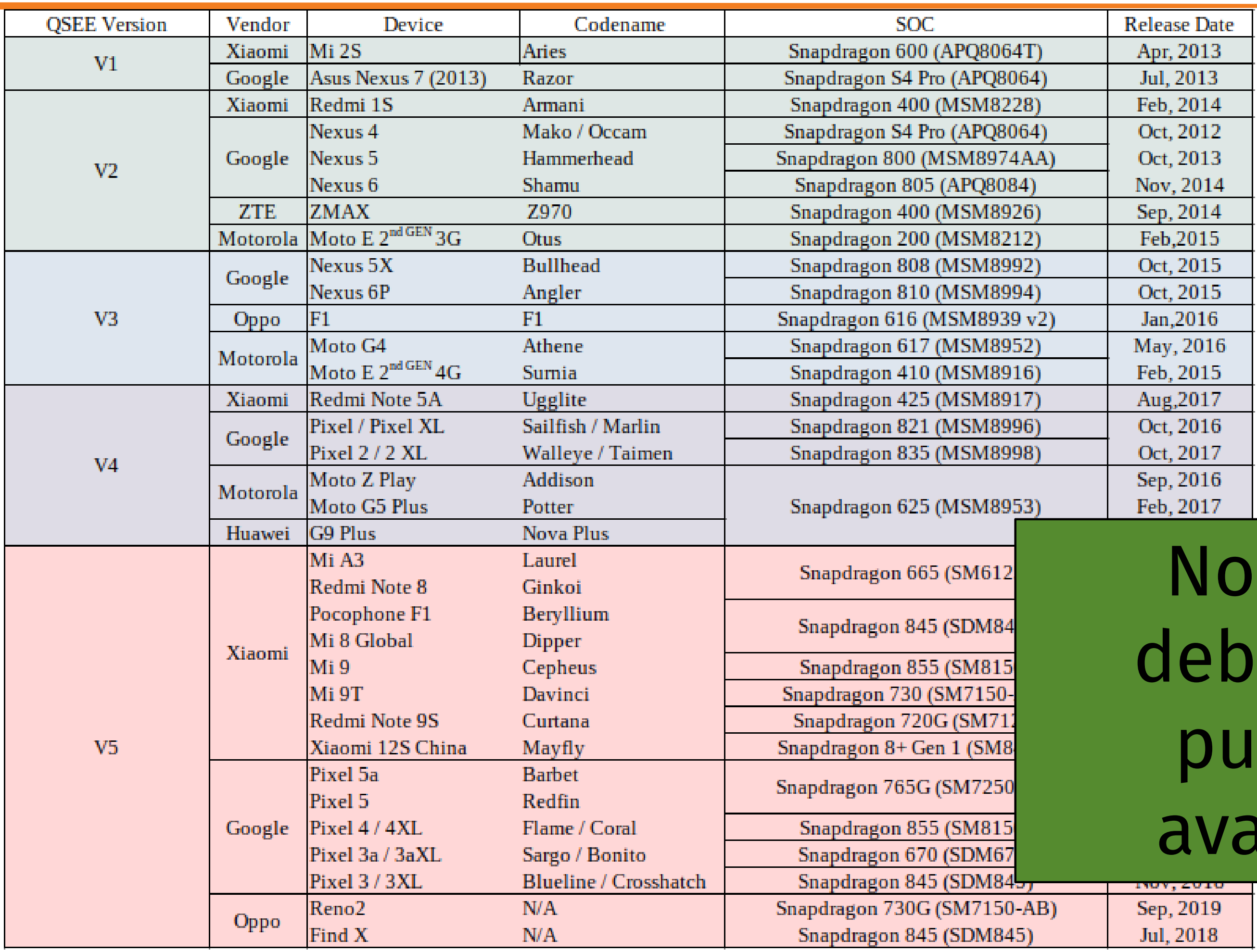
- How many different kernels versions should we simulate?
  - We did a significant research/reversing engineering effort
  - Found that only for QSEE there are many different kernel versions

2.2 QSEE Signatures and OS Versions



QSEE Version	Vendor	Device	Codename	SOC	Release Date
V1	Xiaomi	Mi 2S	Aries	Snapdragon 600 (APQ8064T)	Apr, 2013
	Google	Asus Nexus 7 (2013)	Razor	Snapdragon S4 Pro (APQ8064)	Jul, 2013
V2	Xiaomi	Redmi 1S	Armani	Snapdragon 400 (MSM8228)	Feb, 2014
	Google	Nexus 4	Mako / Occam	Snapdragon S4 Pro (APQ8064)	Oct, 2012
		Nexus 5	Hammerhead	Snapdragon 800 (MSM8974AA)	Oct, 2013
		Nexus 6	Shamu	Snapdragon 805 (APQ8084)	Nov, 2014
	ZTE	ZMAX	Z970	Snapdragon 400 (MSM8926)	Sep, 2014
	Motorola	Moto E $2^{\text{nd GEN}}$ 3G	Otus	Snapdragon 200 (MSM8212)	Feb,2015
V3	Google	Nexus 5X	Bullhead	Snapdragon 808 (MSM8992)	Oct, 2015
		Nexus 6P	Angler	Snapdragon 810 (MSM8994)	Oct, 2015
	Oppo	F1	F1	Snapdragon 616 (MSM8939 v2)	Jan,2016
	Motorola	Moto G4	Athene	Snapdragon 617 (MSM8952)	May, 2016
		Moto E $2^{\text{nd GEN}}$ 4G	Surnia	Snapdragon 410 (MSM8916)	Feb, 2015
V4	Xiaomi	Redmi Note 5A	Ugglite	Snapdragon 425 (MSM8917)	Aug,2017
	Google	Pixel / Pixel XL	Sailfish / Marlin	Snapdragon 821 (MSM8996)	Oct, 2016
		Pixel 2 / 2 XL	Walleye / Taimen	Snapdragon 835 (MSM8998)	Oct, 2017
	Motorola	Moto Z Play	Addison	Snapdragon 625 (MSM8953)	Sep, 2016
		Moto G5 Plus	Potter		Feb, 2017
	Huawei	G9 Plus	Nova Plus		Jul, 2016
V5	Xiaomi	Mi A3	Laurel	Snapdragon 665 (SM6125)	Aug, 2019
		Redmi Note 8	Ginkoi		Sep, 2019
		Pocophone F1	Beryllium	Snapdragon 845 (SDM845)	Aug, 2018
		Mi 8 Global	Dipper		May, 2018
		Mi 9	Cepheus	Snapdragon 855 (SM8150)	Mar, 2019
		Mi 9T	Davinci	Snapdragon 730 (SM7150-AA)	May, 2019
		Redmi Note 9S	Curtana	Snapdragon 720G (SM7125)	Jul, 2020
		Xiaomi 12S China	Mayfly	Snapdragon 8+ Gen 1 (SM8475)	Jul, 2022
	Google	Pixel 5a	Barbet	Snapdragon 765G (SM7250-AB)	Aug, 2021
		Pixel 5	Redfin		Oct, 2020
		Pixel 4 / 4XL	Flame / Coral	Snapdragon 855 (SM8150)	Oct, 2019
		Pixel 3a / 3aXL	Sargo / Bonito	Snapdragon 670 (SDM670)	May, 2019
		Pixel 3 / 3XL	Blueline / Crosshatch	Snapdragon 845 (SDM845)	Nov, 2018
	Oppo	Reno2	N/A	Snapdragon 730G (SM7150-AB)	Sep, 2019
		Find X	N/A	Snapdragon 845 (SDM845)	Jul, 2018

2.2 QSEE Signatures and OS Versions

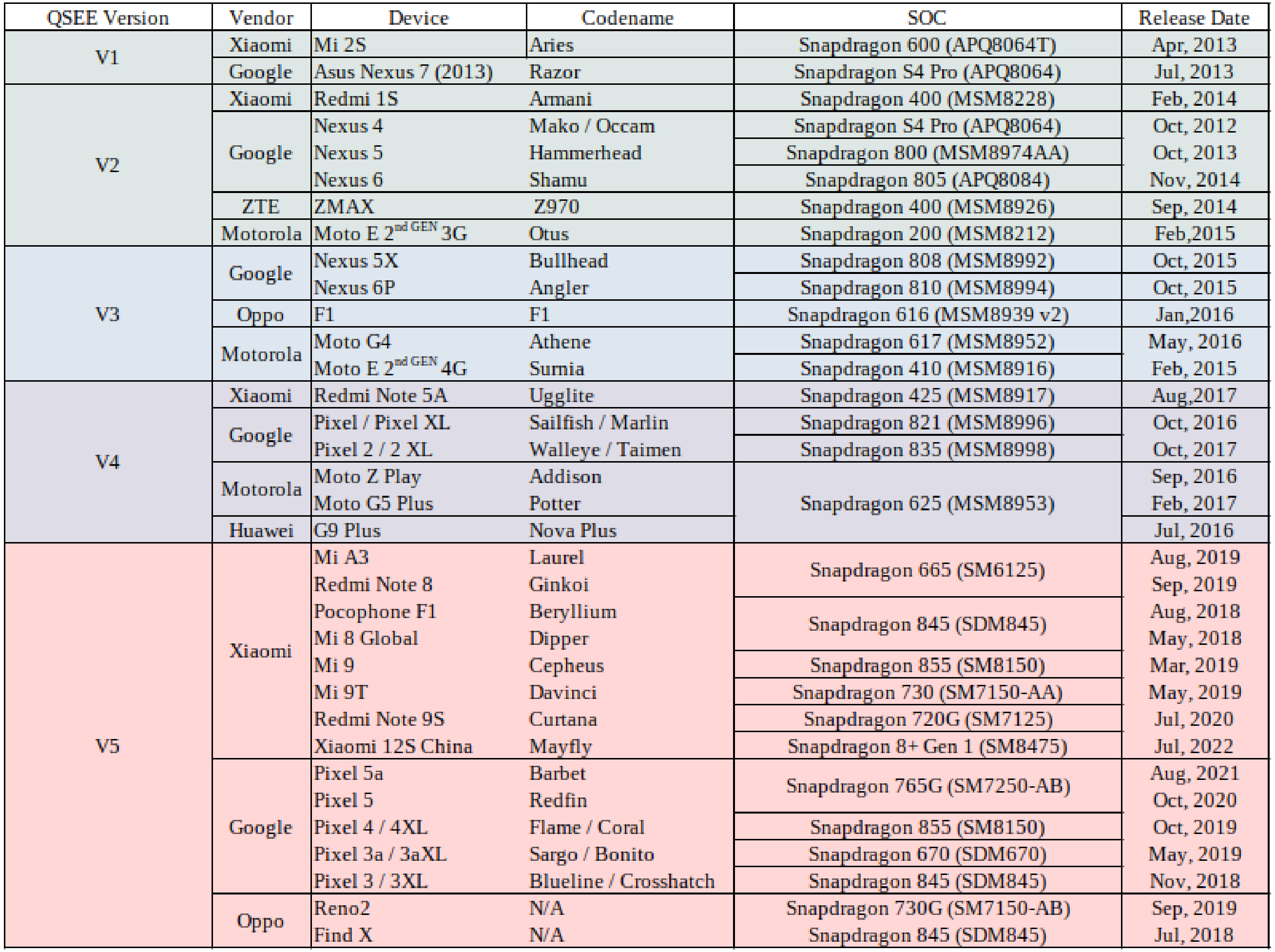


QSEE Version	Vendor	Device	Codename	SOC	Release Date
V1	Xiaomi	Mi 2S	Aries	Snapdragon 600 (APQ8064T)	Apr, 2013
	Google	Asus Nexus 7 (2013)	Razor	Snapdragon S4 Pro (APQ8064)	Jul, 2013
V2	Xiaomi	Redmi 1S	Armani	Snapdragon 400 (MSM8228)	Feb, 2014
	Google	Nexus 4	Mako / Occam	Snapdragon S4 Pro (APQ8064)	Oct, 2012
		Nexus 5	Hammerhead	Snapdragon 800 (MSM8974AA)	Oct, 2013
		Nexus 6	Shamu	Snapdragon 805 (APQ8084)	Nov, 2014
	ZTE	ZMAX	Z970	Snapdragon 400 (MSM8926)	Sep, 2014
	Motorola	Moto E 2$^{\text{nd GEN}}$ 3G	Otus	Snapdragon 200 (MSM8212)	Feb,2015
V3	Google	Nexus 5X	Bullhead	Snapdragon 808 (MSM8992)	Oct, 2015
		Nexus 6P	Angler	Snapdragon 810 (MSM8994)	Oct, 2015
	Oppo	F1	F1	Snapdragon 616 (MSM8939 v2)	Jan,2016
	Motorola	Moto G4	Athene	Snapdragon 617 (MSM8952)	May, 2016
		Moto E 2$^{\text{nd GEN}}$ 4G	Surnia	Snapdragon 410 (MSM8916)	Feb, 2015
V4	Xiaomi	Redmi Note 5A	Ugglite	Snapdragon 425 (MSM8917)	Aug,2017
	Google	Pixel / Pixel XL	Sailfish / Marlin	Snapdragon 821 (MSM8996)	Oct, 2016
		Pixel 2 / 2 XL	Walleye / Taimen	Snapdragon 835 (MSM8998)	Oct, 2017
	Motorola	Moto Z Play	Addison	Snapdragon 625 (MSM8953)	Sep, 2016
		Moto G5 Plus	Potter		Feb, 2017
	Huawei	G9 Plus	Nova Plus		
V5	Xiaomi	Mi A3	Laurel	Snapdragon 665 (SM612	
		Redmi Note 8	Ginkoi		
		Pocophone F1	Beryllium	Snapdragon 845 (SDM84	
		Mi 8 Global	Dipper		
		Mi 9	Cepheus	Snapdragon 855 (SM815	
		Mi 9T	Davinci	Snapdragon 730 (SM7150-	
		Redmi Note 9S	Curtana	Snapdragon 720G (SM71	
		Xiaomi 12S China	Mayfly	Snapdragon 8+ Gen 1 (SM8	
	Google	Pixel 5a	Barbet	Snapdragon 765G (SM7250	
		Pixel 5	Redfin		
		Pixel 4 / 4XL	Flame / Coral	Snapdragon 855 (SM815	
		Pixel 3a / 3aXL	Sargo / Bonito	Snapdragon 670 (SDM67	
		Pixel 3 / 3XL	Blueline / Crosshatch	Snapdragon 845 (SDM845)	Nov, 2018
	Oppo	Reno2	N/A	Snapdragon 730G (SM7150-AB)	Sep, 2019
		Find X	N/A	Snapdragon 845 (SDM845)	Jul, 2018

No QSEE debuggers publicly available

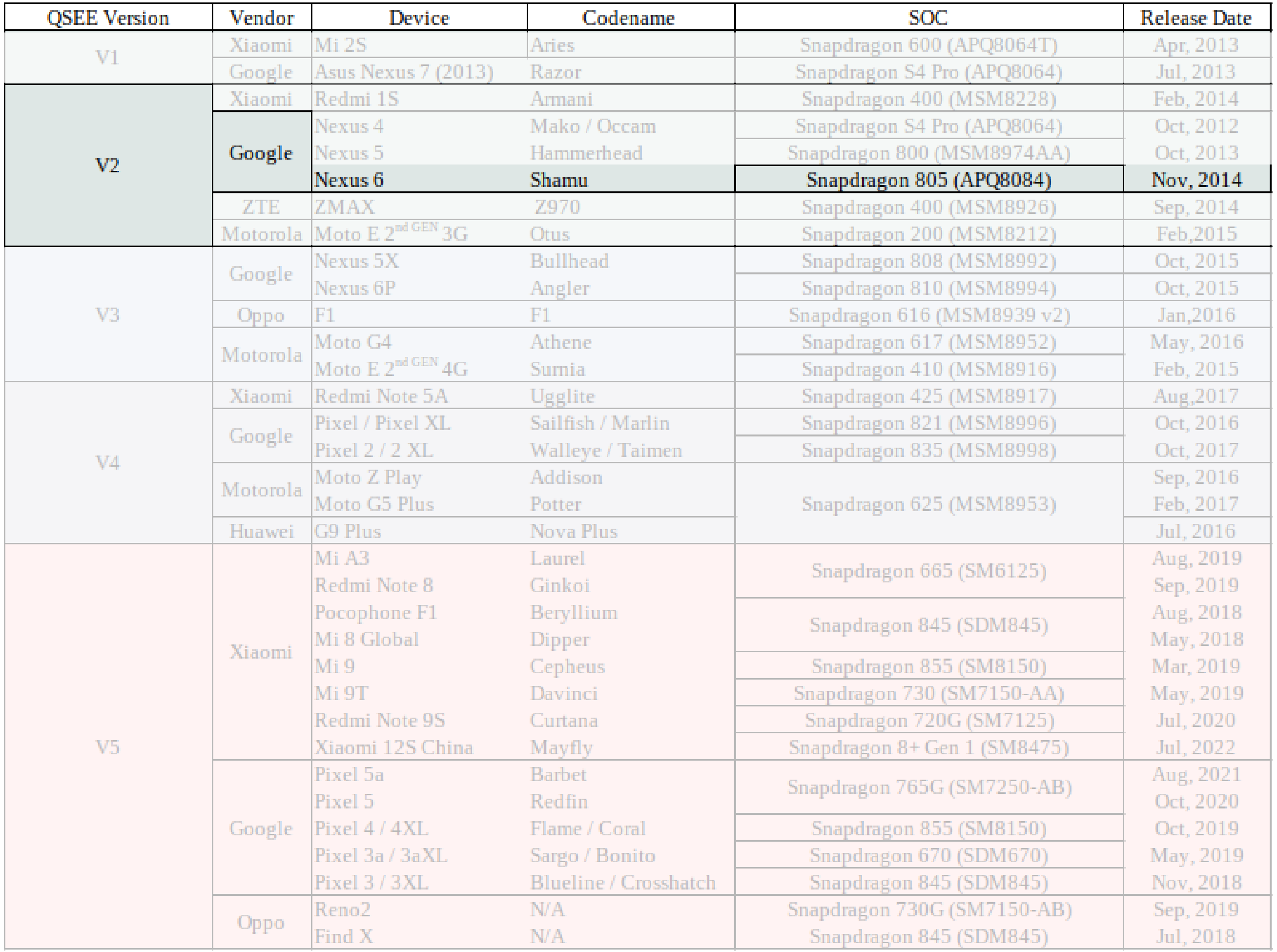
3. Auditing Trusted Applications

3.1 Emulation of Trusted Applications



QSEE Version	Vendor	Device	Codename	SOC	Release Date
V1	Xiaomi	Mi 2S	Aries	Snapdragon 600 (APQ8064T)	Apr, 2013
	Google	Asus Nexus 7 (2013)	Razor	Snapdragon S4 Pro (APQ8064)	Jul, 2013
V2	Xiaomi	Redmi 1S	Armani	Snapdragon 400 (MSM8228)	Feb, 2014
	Google	Nexus 4	Mako / Occam	Snapdragon S4 Pro (APQ8064)	Oct, 2012
		Nexus 5	Hammerhead	Snapdragon 800 (MSM8974AA)	Oct, 2013
		Nexus 6	Shamu	Snapdragon 805 (APQ8084)	Nov, 2014
	ZTE	ZMAX	Z970	Snapdragon 400 (MSM8926)	Sep, 2014
	Motorola	Moto E 2nd GEN 3G	Otus	Snapdragon 200 (MSM8212)	Feb,2015
V3	Google	Nexus 5X	Bullhead	Snapdragon 808 (MSM8992)	Oct, 2015
		Nexus 6P	Angler	Snapdragon 810 (MSM8994)	Oct, 2015
	Oppo	F1	F1	Snapdragon 616 (MSM8939 v2)	Jan,2016
	Motorola	Moto G4	Athene	Snapdragon 617 (MSM8952)	May, 2016
		Moto E 2nd GEN 4G	Surnia	Snapdragon 410 (MSM8916)	Feb, 2015
V4	Xiaomi	Redmi Note 5A	Ugglite	Snapdragon 425 (MSM8917)	Aug,2017
	Google	Pixel / Pixel XL	Sailfish / Marlin	Snapdragon 821 (MSM8996)	Oct, 2016
		Pixel 2 / 2 XL	Walleye / Taimen	Snapdragon 835 (MSM8998)	Oct, 2017
	Motorola	Moto Z Play	Addison	Snapdragon 625 (MSM8953)	Sep, 2016
		Moto G5 Plus	Potter		Feb, 2017
	Huawei	G9 Plus	Nova Plus		Jul, 2016
V5	Xiaomi	Mi A3	Laurel	Snapdragon 665 (SM6125)	Aug, 2019
		Redmi Note 8	Ginkoi		Sep, 2019
		Pocophone F1	Beryllium	Snapdragon 845 (SDM845)	Aug, 2018
		Mi 8 Global	Dipper		May, 2018
		Mi 9	Cepheus	Snapdragon 855 (SM8150)	Mar, 2019
		Mi 9T	Davinci	Snapdragon 730 (SM7150-AA)	May, 2019
		Redmi Note 9S	Curtana	Snapdragon 720G (SM7125)	Jul, 2020
		Xiaomi 12S China	Mayfly	Snapdragon 8+ Gen 1 (SM8475)	Jul, 2022
	Google	Pixel 5a	Barbet	Snapdragon 765G (SM7250-AB)	Aug, 2021
		Pixel 5	Redfin		Oct, 2020
		Pixel 4 / 4XL	Flame / Coral	Snapdragon 855 (SM8150)	Oct, 2019
		Pixel 3a / 3aXL	Sargo / Bonito	Snapdragon 670 (SDM670)	May, 2019
		Pixel 3 / 3XL	Blueline / Crosshatch	Snapdragon 845 (SDM845)	Nov, 2018
	Oppo	Reno2	N/A	Snapdragon 730G (SM7150-AB)	Sep, 2019
		Find X	N/A	Snapdragon 845 (SDM845)	Jul, 2018

3.1 Emulation of Trusted Applications



QSEE Version	Vendor	Device	Codename	SOC	Release Date
V1	Xiaomi	Mi 2S	Aries	Snapdragon 600 (APQ8064T)	Apr, 2013
	Google	Asus Nexus 7 (2013)	Razor	Snapdragon S4 Pro (APQ8064)	Jul, 2013
V2	Xiaomi	Redmi 1S	Armani	Snapdragon 400 (MSM8228)	Feb, 2014
	Google	Nexus 4	Mako / Occam	Snapdragon S4 Pro (APQ8064)	Oct, 2012
		Nexus 5	Hammerhead	Snapdragon 800 (MSM8974AA)	Oct, 2013
		Nexus 6	Shamu	Snapdragon 805 (APQ8084)	Nov, 2014
	ZTE	ZMAX	Z970	Snapdragon 400 (MSM8926)	Sep, 2014
	Motorola	Moto E 2$^{\text{nd GEN}}$ 3G	Otus	Snapdragon 200 (MSM8212)	Feb,2015
V3	Google	Nexus 5X	Bullhead	Snapdragon 808 (MSM8992)	Oct, 2015
		Nexus 6P	Angler	Snapdragon 810 (MSM8994)	Oct, 2015
	Oppo	F1	F1	Snapdragon 616 (MSM8939 v2)	Jan,2016
	Motorola	Moto G4	Athene	Snapdragon 617 (MSM8952)	May, 2016
		Moto E 2$^{\text{nd GEN}}$ 4G	Surnia	Snapdragon 410 (MSM8916)	Feb, 2015
V4	Xiaomi	Redmi Note 5A	Ugglite	Snapdragon 425 (MSM8917)	Aug,2017
	Google	Pixel / Pixel XL	Sailfish / Marlin	Snapdragon 821 (MSM8996)	Oct, 2016
		Pixel 2 / 2 XL	Walleye / Taimen	Snapdragon 835 (MSM8998)	Oct, 2017
	Motorola	Moto Z Play	Addison	Snapdragon 625 (MSM8953)	Sep, 2016
		Moto G5 Plus	Potter		Feb, 2017
	Huawei	G9 Plus	Nova Plus		Jul, 2016
V5	Xiaomi	Mi A3	Laurel	Snapdragon 665 (SM6125)	Aug, 2019
		Redmi Note 8	Ginkoi		Sep, 2019
		Pocophone F1	Beryllium	Snapdragon 845 (SDM845)	Aug, 2018
		Mi 8 Global	Dipper		May, 2018
		Mi 9	Cepheus	Snapdragon 855 (SM8150)	Mar, 2019
		Mi 9T	Davinci	Snapdragon 730 (SM7150-AA)	May, 2019
		Redmi Note 9S	Curtana	Snapdragon 720G (SM7125)	Jul, 2020
		Xiaomi 12S China	Mayfly	Snapdragon 8+ Gen 1 (SM8475)	Jul, 2022
	Google	Pixel 5a	Barbet	Snapdragon 765G (SM7250-AB)	Aug, 2021
		Pixel 5	Redfin		Oct, 2020
		Pixel 4 / 4XL	Flame / Coral	Snapdragon 855 (SM8150)	Oct, 2019
		Pixel 3a / 3aXL	Sargo / Bonito	Snapdragon 670 (SDM670)	May, 2019
		Pixel 3 / 3XL	Blueline / Crosshatch	Snapdragon 845 (SDM845)	Nov, 2018
	Oppo	Reno2	N/A	Snapdragon 730G (SM7150-AB)	Sep, 2019
		Find X	N/A	Snapdragon 845 (SDM845)	Jul, 2018

Test Device

- Google Nexus 6
- QSEE v2

3.1 Emulation of Trusted Applications

Use Case: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

Specifications Table

Codename	Shamu
Developer	Google and Motorola Mobility
Manufacturer	Motorola Mobility
System on chip	Qualcomm Snapdragon 805 (APQ8084)
CPU	Qualcomm 2.7 GHz quad-core Krait 450
GPU	Adreno 420
Memory	3 GB of LPDDR3 RAM

Widevine

- Proprietary Digital Rights Management (DRM) technology from Google
- Allows restricted consumer access to distributed media content according to rules defined by content owners
- **Trusted Application** in QSEE

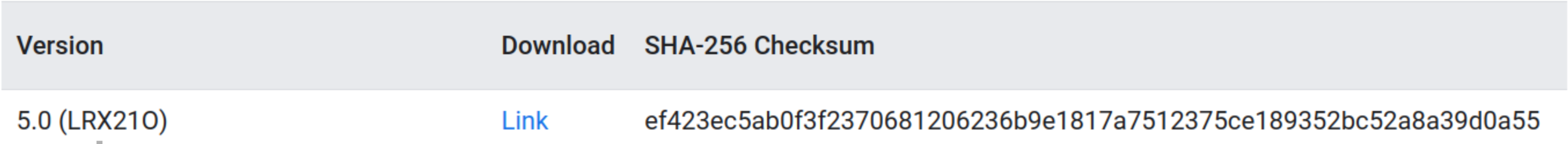
3.1 Emulation of Trusted Applications

Getting Widevine TA from Firmware

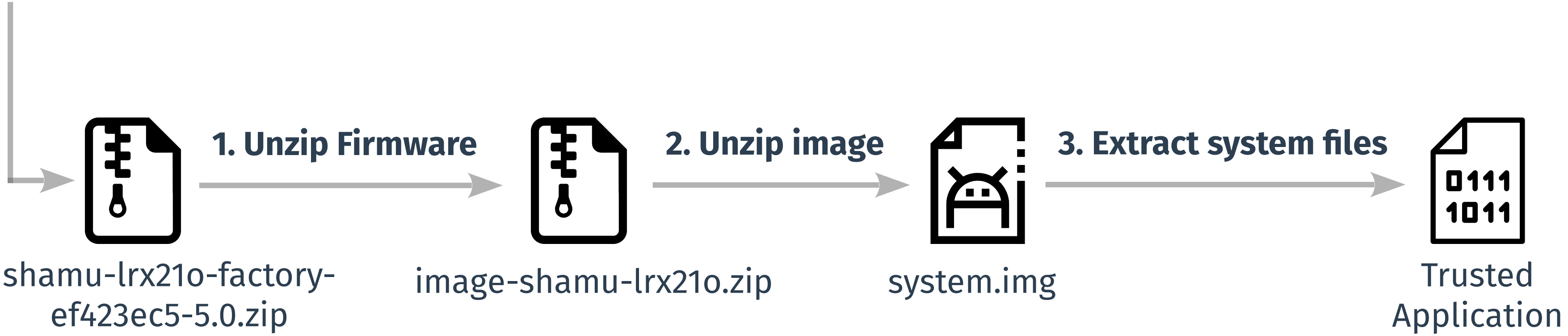
`https://developers.google.com/android/images`

"shamu" for Nexus 6

Unpatched firmware for development of the tools



Version	Download	SHA-256 Checksum
5.0 (LRX21O)	Link	ef423ec5ab0f3f2370681206236b9e1817a7512375ce189352bc52a8a39d0a55

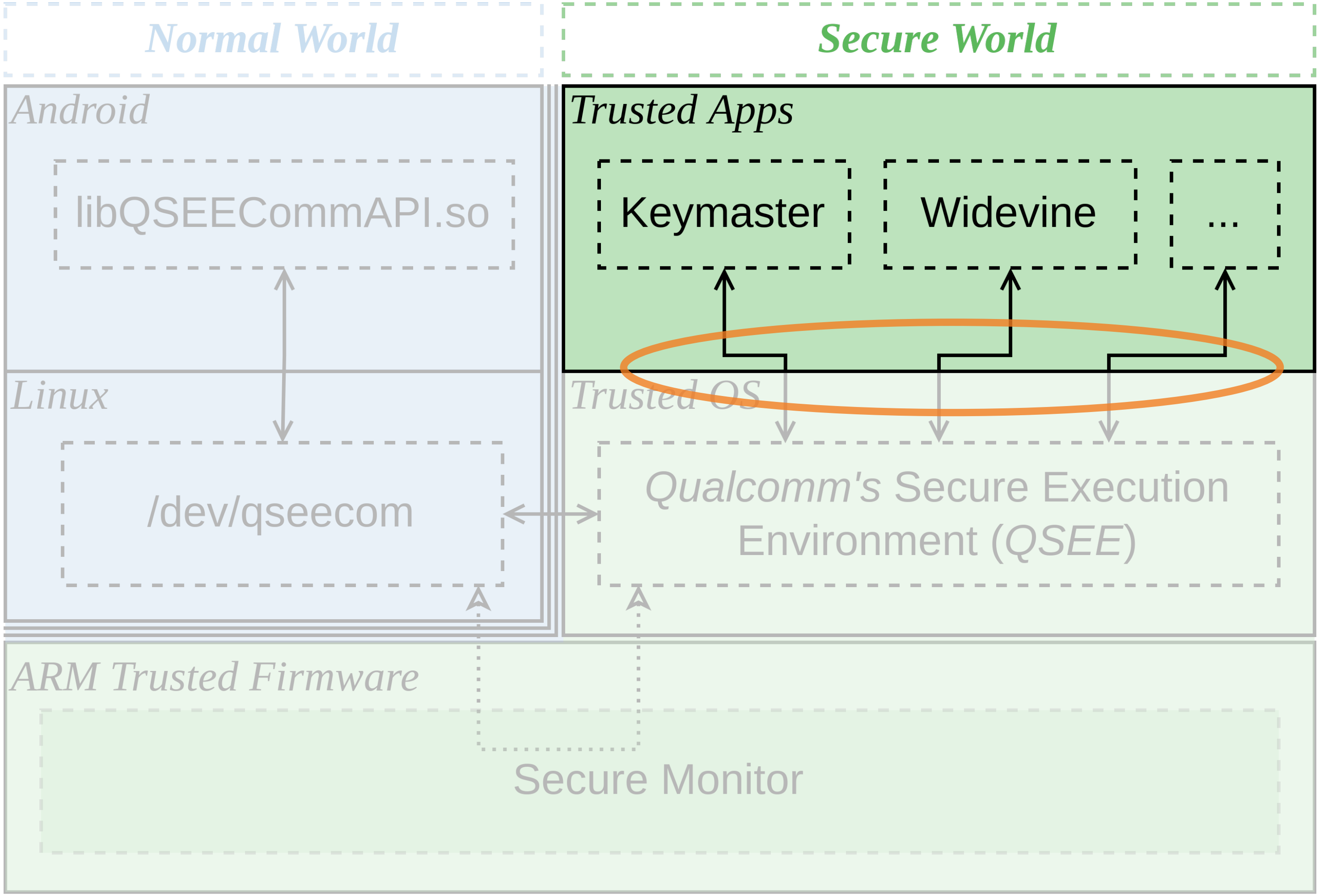


1. Unzip Firmware

2. Unzip image

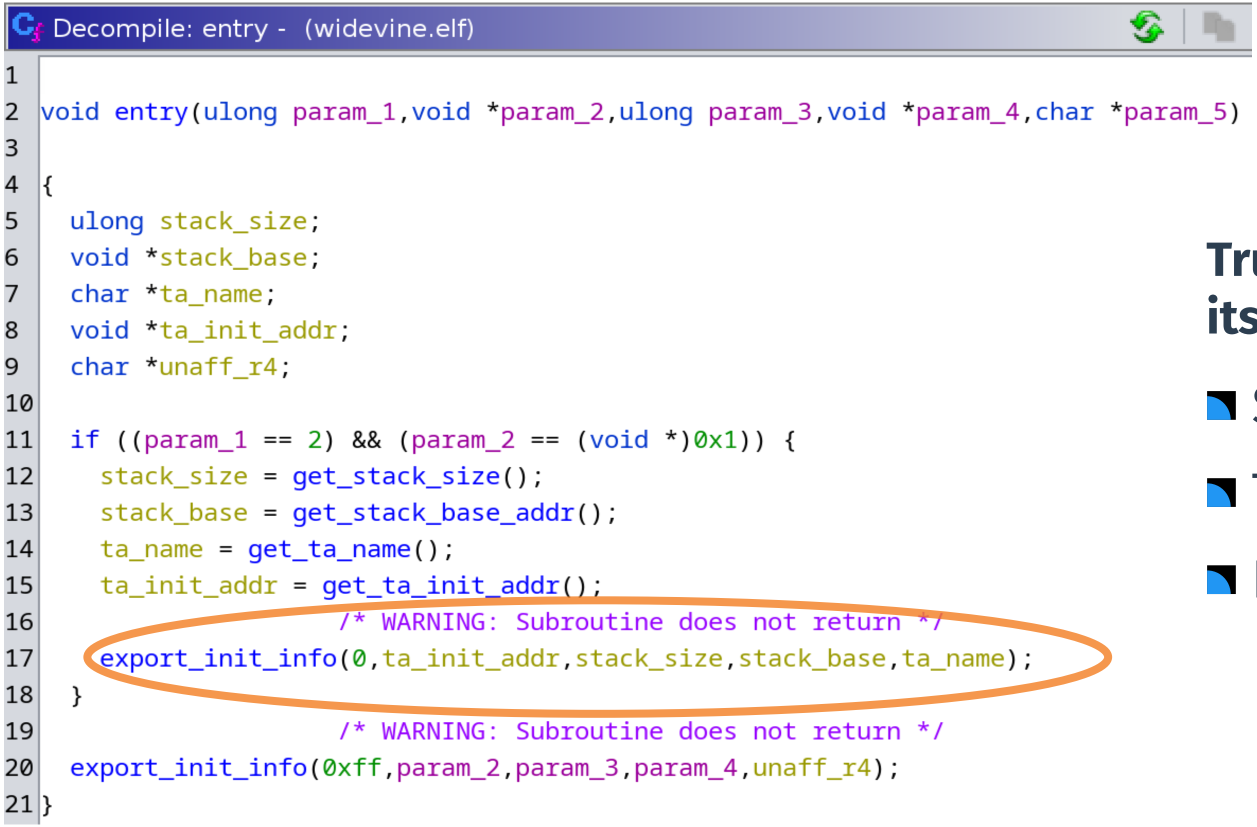
3. Extract system files

2.1 QSEE Overview



3.1 Emulation of Trusted Applications

Brief run-through on Widevine TA loading



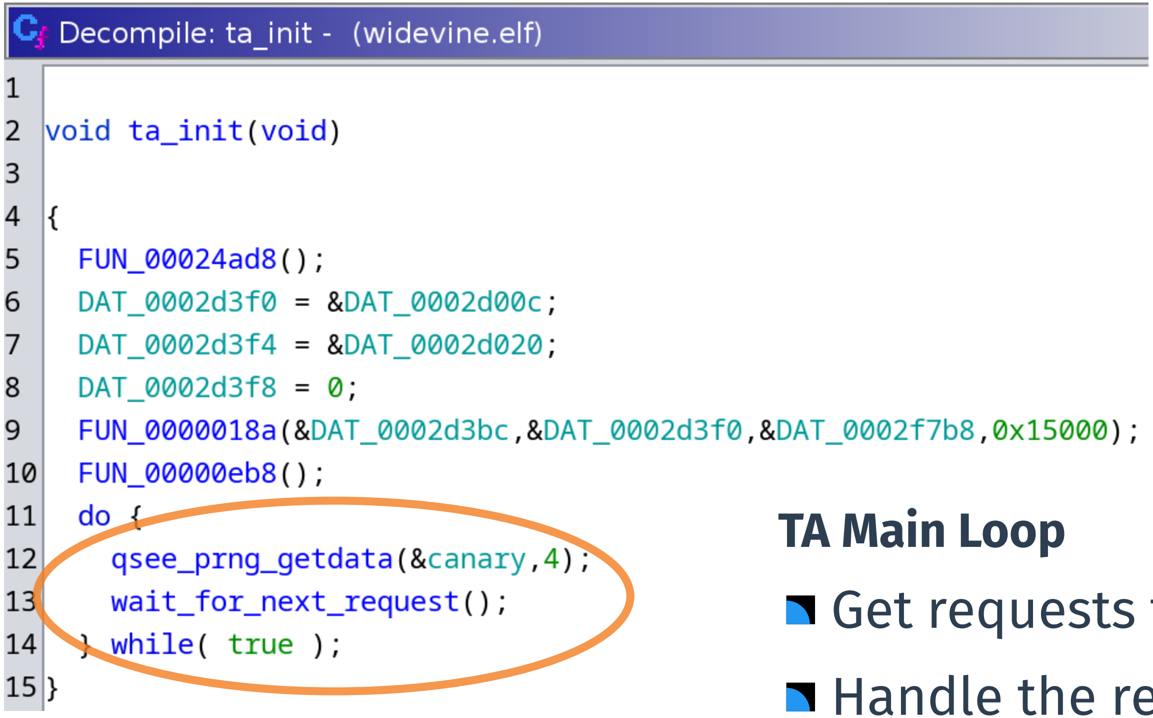
```

void entry(ulong param_1,void *param_2,ulong param_3,void *param_4,char *param_5)

{
  ulong stack_size;
  void *stack_base;
  char *ta_name;
  void *ta_init_addr;
  char *unaff_r4;

  if ((param_1 == 2) && (param_2 == (void *)0x1)) {
    stack_size = get_stack_size();
    stack_base = get_stack_base_addr();
    ta_name = get_ta_name();
    ta_init_addr = get_ta_init_addr();
                    /* WARNING: Subroutine does not return */
    export_init_info(0,ta_init_addr,stack_size,stack_base,ta_name);
  }
                    /* WARNING: Subroutine does not return */
  export_init_info(0xff,param_2,param_3,param_4,unaff_r4);
}
```

Trusted Application registers itself to the trusted OS (QSEE)

- Stack Base & Size
- TA Name
- Init function pointer

3.1 Emulation of Trusted Applications

Brief run-through on Widevine TA loading



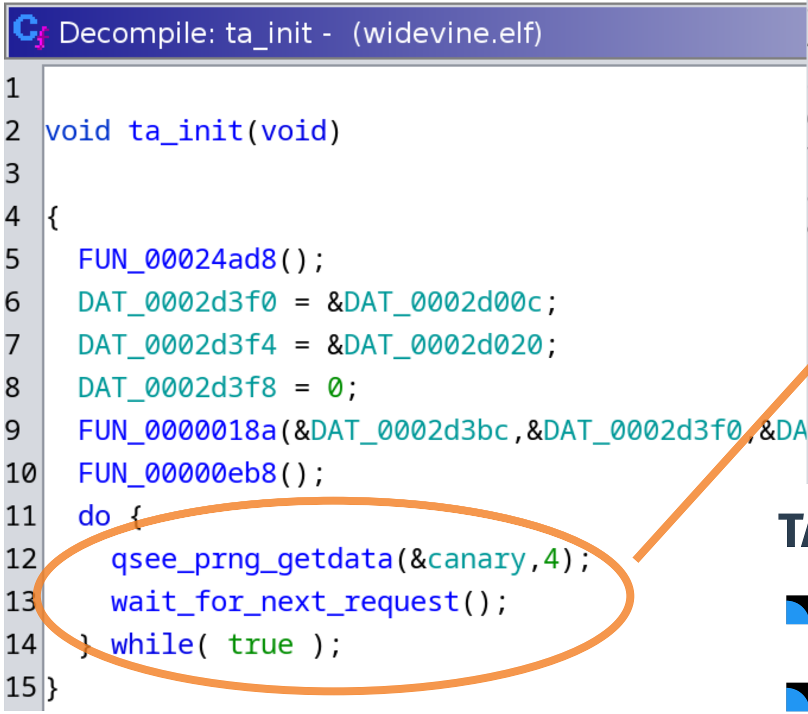
```
void ta_init(void)

{
  FUN_00024ad8();
  DAT_0002d3f0 = &DAT_0002d00c;
  DAT_0002d3f4 = &DAT_0002d020;
  DAT_0002d3f8 = 0;
  FUN_0000018a(&DAT_0002d3bc,&DAT_0002d3f0,&DAT_0002f7b8,0x15000);
  FUN_00000eb8();
  do {
    qsee_prng_getdata(&canary,4);
    wait_for_next_request();
  } while( true );
}
```

TA Main Loop

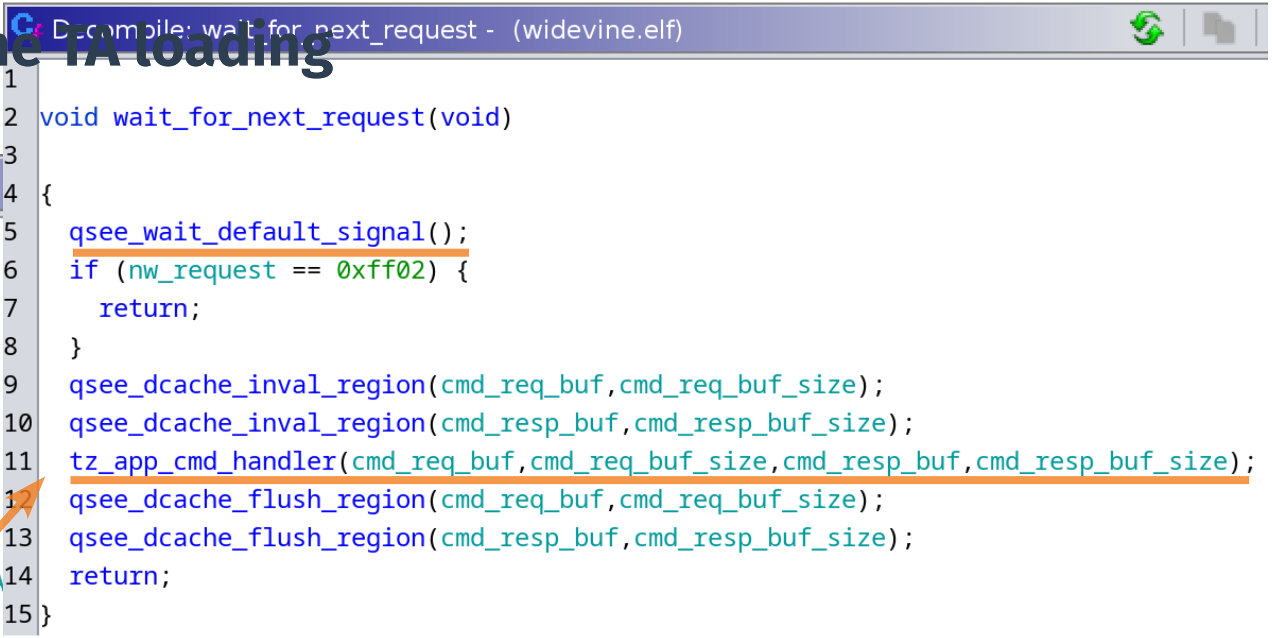
- Get requests from Normal World
- Handle the received input

3.1 Emulation of Trusted Applications

Brief run-through on Widevine TA loading



```
void ta_init(void)

{
  FUN_00024ad8();
  DAT_0002d3f0 = &DAT_0002d00c;
  DAT_0002d3f4 = &DAT_0002d020;
  DAT_0002d3f8 = 0;
  FUN_0000018a(&DAT_0002d3bc,&DAT_0002d3f0,&DA
  FUN_00000eb8();
  do {
    qsee_prng_getdata(&canary,4);
    wait_for_next_request();
  } while( true );
}
```



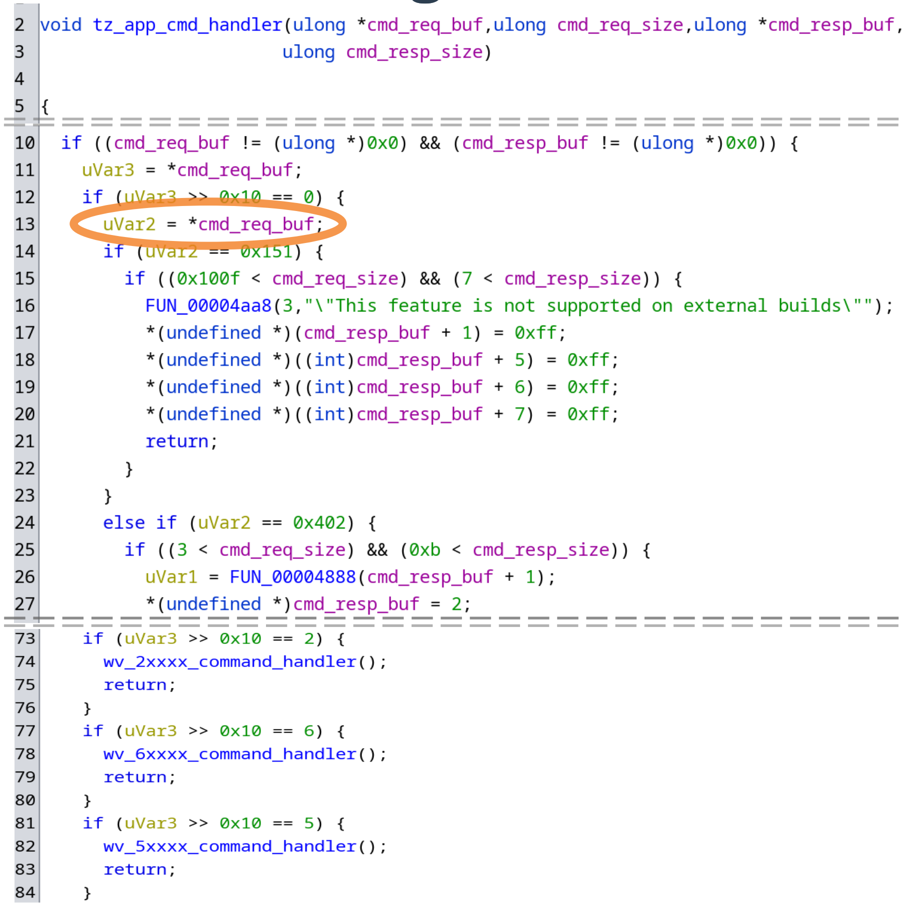
```
void wait_for_next_request(void)

{
  qsee_wait_default_signal();
  if (nw_request == 0xff02) {
    return;
  }
  qsee_dcache_inval_region(cmd_req_buf,cmd_req_buf_size);
  qsee_dcache_inval_region(cmd_resp_buf,cmd_resp_buf_size);
  tz_app_cmd_handler(cmd_req_buf,cmd_req_buf_size,cmd_resp_buf,cmd_resp_buf_size);
  qsee_dcache_flush_region(cmd_req_buf,cmd_req_buf_size);
  qsee_dcache_flush_region(cmd_resp_buf,cmd_resp_buf_size);
  return;
}
```

TA Main Loop

- Get requests from Normal World
- Handle the received input

3.1 Emulation of Trusted Applications

Brief run-through on Widevine TA loading



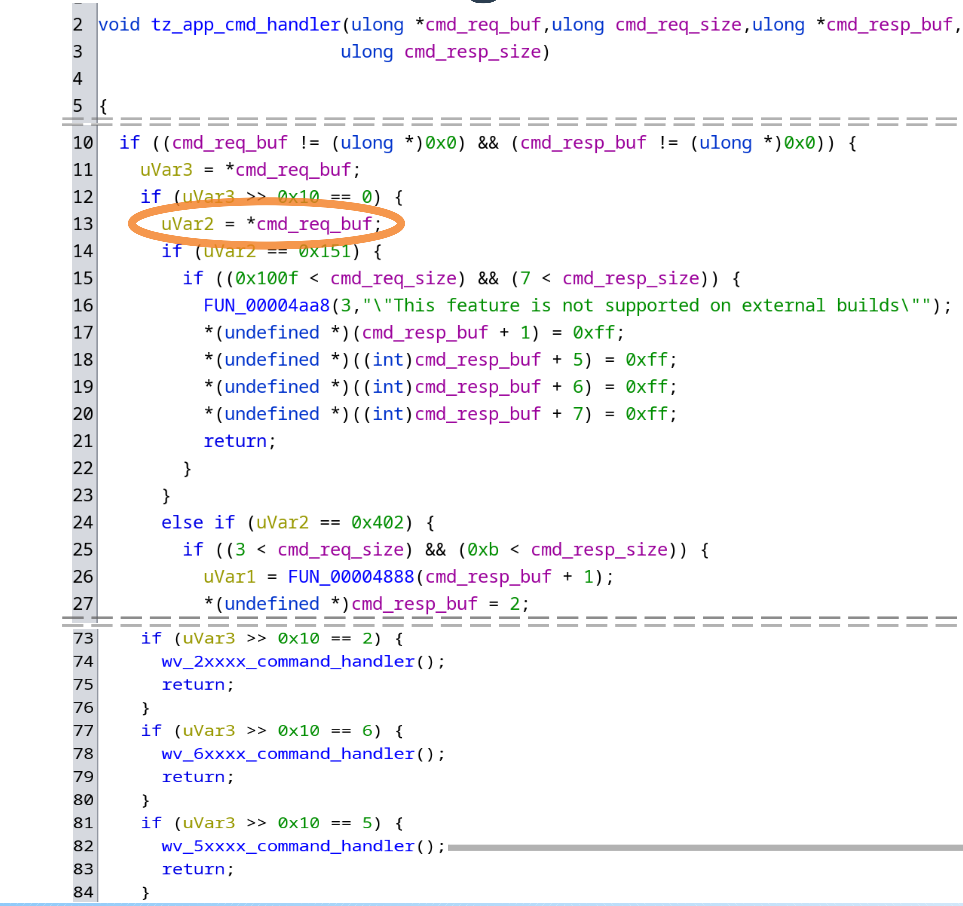
```
void tz_app_cmd_handler(ulong *cmd_req_buf,ulong cmd_req_size,ulong *cmd_resp_buf,
                        ulong cmd_resp_size)

{
  if ((cmd_req_buf != (ulong *)0x0) && (cmd_resp_buf != (ulong *)0x0)) {
    uVar3 = *cmd_req_buf;
    if (uVar3 >> 0x10 == 0) {
      uVar2 = *cmd_req_buf;
      if (uVar2 == 0x151) {
        if ((0x100f < cmd_req_size) && (7 < cmd_resp_size)) {
          FUN_00004aa8(3,"\"This feature is not supported on external builds\"");
          *(undefined *)(cmd_resp_buf + 1) = 0xff;
          *(undefined *)((int)cmd_resp_buf + 5) = 0xff;
          *(undefined *)((int)cmd_resp_buf + 6) = 0xff;
          *(undefined *)((int)cmd_resp_buf + 7) = 0xff;
          return;
        }
      }
      else if (uVar2 == 0x402) {
        if ((3 < cmd_req_size) && (0xb < cmd_resp_size)) {
          uVar1 = FUN_00004888(cmd_resp_buf + 1);
          *(undefined *)cmd_resp_buf = 2;
    if (uVar3 >> 0x10 == 2) {
      wv_2xxxx_command_handler();
      return;
    }
    if (uVar3 >> 0x10 == 6) {
      wv_6xxxx_command_handler();
      return;
    }
    if (uVar3 >> 0x10 == 5) {
      wv_5xxxx_command_handler();
      return;
    }
```

Command Handler

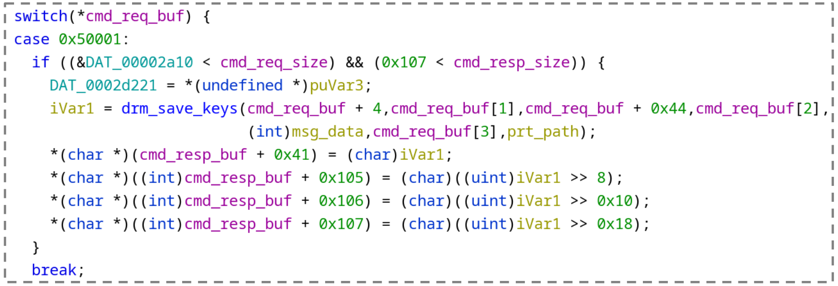
- Get Command ID
- Switch-Case
- Handle Command

3.1 Emulation of Trusted Applications

Brief run-through on Widevine TA loading



```
void tz_app_cmd_handler(ulong *cmd_req_buf,ulong cmd_req_size,ulong *cmd_resp_buf,
                       ulong cmd_resp_size)


{
  if ((cmd_req_buf != (ulong *)0x0) && (cmd_resp_buf != (ulong *)0x0)) {
    uVar3 = *cmd_req_buf;
    if (uVar3 >> 0x10 == 0) {
      uVar2 = *cmd_req_buf;
      if (uVar2 == 0x151) {
        if ((0x100f < cmd_req_size) && (7 < cmd_resp_size)) {
          FUN_00004aa8(3,"\"This feature is not supported on external builds\"");
          *(undefined *)(cmd_resp_buf + 1) = 0xff;
          *(undefined *)((int)cmd_resp_buf + 5) = 0xff;
          *(undefined *)((int)cmd_resp_buf + 6) = 0xff;
          *(undefined *)((int)cmd_resp_buf + 7) = 0xff;
          return;
        }
      }
      else if (uVar2 == 0x402) {
        if ((3 < cmd_req_size) && (0xb < cmd_resp_size)) {
          uVar1 = FUN_00004888(cmd_resp_buf + 1);
          *(undefined *)cmd_resp_buf = 2;
    if (uVar3 >> 0x10 == 2) {
      wv_2xxxx_command_handler();
      return;
    }
    if (uVar3 >> 0x10 == 6) {
      wv_6xxxx_command_handler();
      return;
    }
    if (uVar3 >> 0x10 == 5) {
      wv_5xxxx_command_handler();
      return;
    }
```

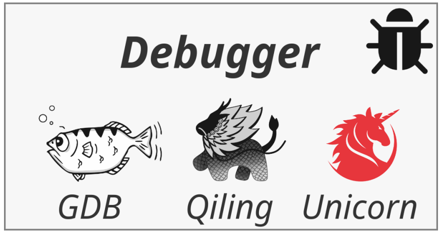


```
switch(*cmd_req_buf) {
case 0x50001:
  if ((&DAT_00002a10 < cmd_req_size) && (0x107 < cmd_resp_size)) {
    DAT_0002d221 = *(undefined *)puVar3;
    iVar1 = drm_save_keys(cmd_req_buf + 4,cmd_req_buf[1],cmd_req_buf + 0x44,cmd_req_buf[2],
                          (int)msg_data,cmd_req_buf[3],prt_path);
    *(char *)(cmd_resp_buf + 0x41) = (char)iVar1;
    *(char *)((int)cmd_resp_buf + 0x105) = (char)((uint)iVar1 >> 8);
    *(char *)((int)cmd_resp_buf + 0x106) = (char)((uint)iVar1 >> 0x10);
    *(char *)((int)cmd_resp_buf + 0x107) = (char)((uint)iVar1 >> 0x18);
  }
  break;
```

Command Handler

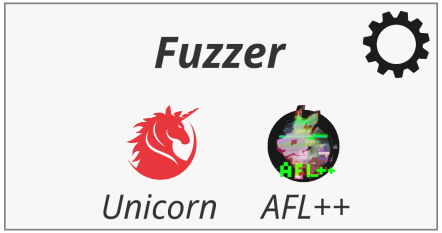
- Get Command ID
- Switch-Case
- Handle Command

3.2 Show Tool: Debugger of Trusted Applications

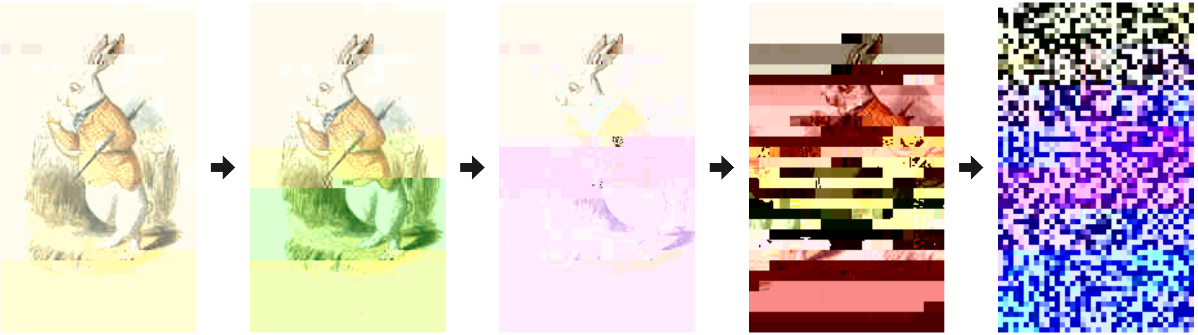


- Based on Qiling framework
- Support for dynamic debugging using GDB
- Integration with Ghidra GUI
  - Allows usage of plugins and Ghidra scripts
- Generate coverage files in drcov format
  - Easy visualization
- Save & Load state dumps
  - Feed fuzzers
- Execute and debug other types of software (e.g., bootloaders or kernels)

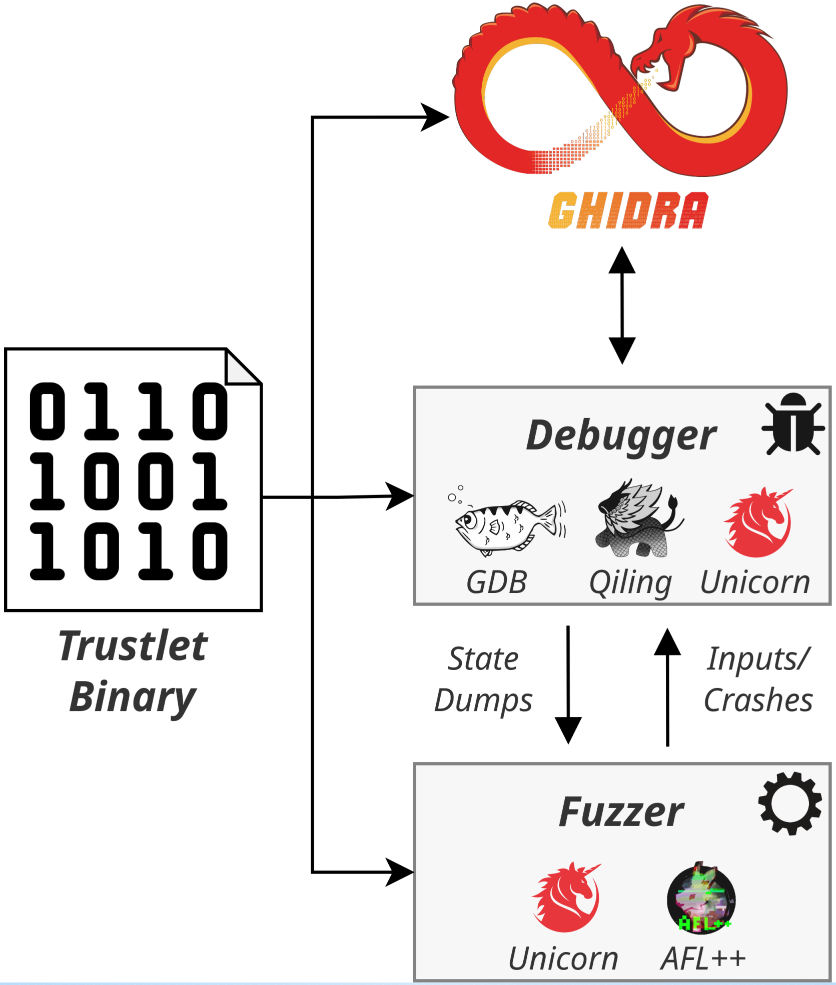
3.3 Show Tool: Fuzzer of Trusted Applications



- Coverage-based Greybox Fuzzing (CGF)
- Input mutation
- Based on AFL++
- Flexible & Easy configuration
- Load state dumps
- Support for fuzz “filters”
- Fuzz-start, fuzz-stop
- Emulation entry point
- Reproduce crashes (dry-run)



3.4 Tool Interaction Overview



3.5 Debugger and Fuzzer Demo

Objective: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

nexus6

Specifications Table

Codename	Shamu
Developer	Google and Motorola Mobility
Manufacturer	Motorola Mobility
System on chip	Qualcomm Snapdragon 805 (APQ8084)
CPU	Qualcomm 2.7 GHz quad-core Krait 450
GPU	Adreno 420
Memory	3 GB of LPDDR3 RAM

Widevine

- Proprietary Digital Rights Management (DRM) technology from Google
- Allows restricted consumer access to distributed media content according to rules defined by content owners
- **Trusted Application** in QSEE

3.5 Debugger and Fuzzer Demo

Objective: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

- Debug the Widevine Trusted Application
  - Emulate the Google Nexus
  - Load memory contents from Ghidra Bridge

3.5 Debugger and Fuzzer Demo

Objective: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

- Debug the Widevine Trusted Application
  - Emulate the Google Nexus
  - Load memory contents from Ghidra Bridge
- Attach with `gdb-multiarch` (target remote :9999)

3.5 Debugger and Fuzzer Demo

Objective: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

- Debug the Widevine Trusted Application
  - Emulate the Google Nexus
  - Load memory contents from Ghidra Bridge
- Attach with `gdb-multiarch` (target remote :9999)
- Synchronize the debugger’s execution with Ghidra GUI (ret-sync)

3.5 Debugger and Fuzzer Demo

Objective: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

- Debug the Widevine Trusted Application
  - Emulate the Google Nexus
  - Load memory contents from Ghidra Bridge
- Attach with `gdb-multiarch` (target remote :9999)
- Synchronize the debugger’s execution with Ghidra GUI (ret-sync)
- Breakpoint @ Command Handler

3.5 Debugger and Fuzzer Demo

Objective: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

- Debug the Widevine Trusted Application
  - Emulate the Google Nexus
  - Load memory contents from Ghidra Bridge
- Attach with `gdb-multiarch` (target remote :9999)
- Synchronize the debugger’s execution with Ghidra GUI (ret-sync)
- Breakpoint @ Command Handler
- Save a machine state dump

3.5 Debugger and Fuzzer Demo

Objective: Fuzz-test the Widevine TA for the Nexus 6 device (shamu)

- Debug the Widevine Trusted Application
  - Emulate the Google Nexus
  - Load memory contents from Ghidra Bridge
- Attach with `gdb-multiarch` (target remote :9999)
- Synchronize the debugger’s execution with Ghidra GUI (ret-sync)
- Breakpoint @ Command Handler
- Save a machine state dump
- Load the saved state dump into the fuzzer

Debugger and Fuzzer Demo

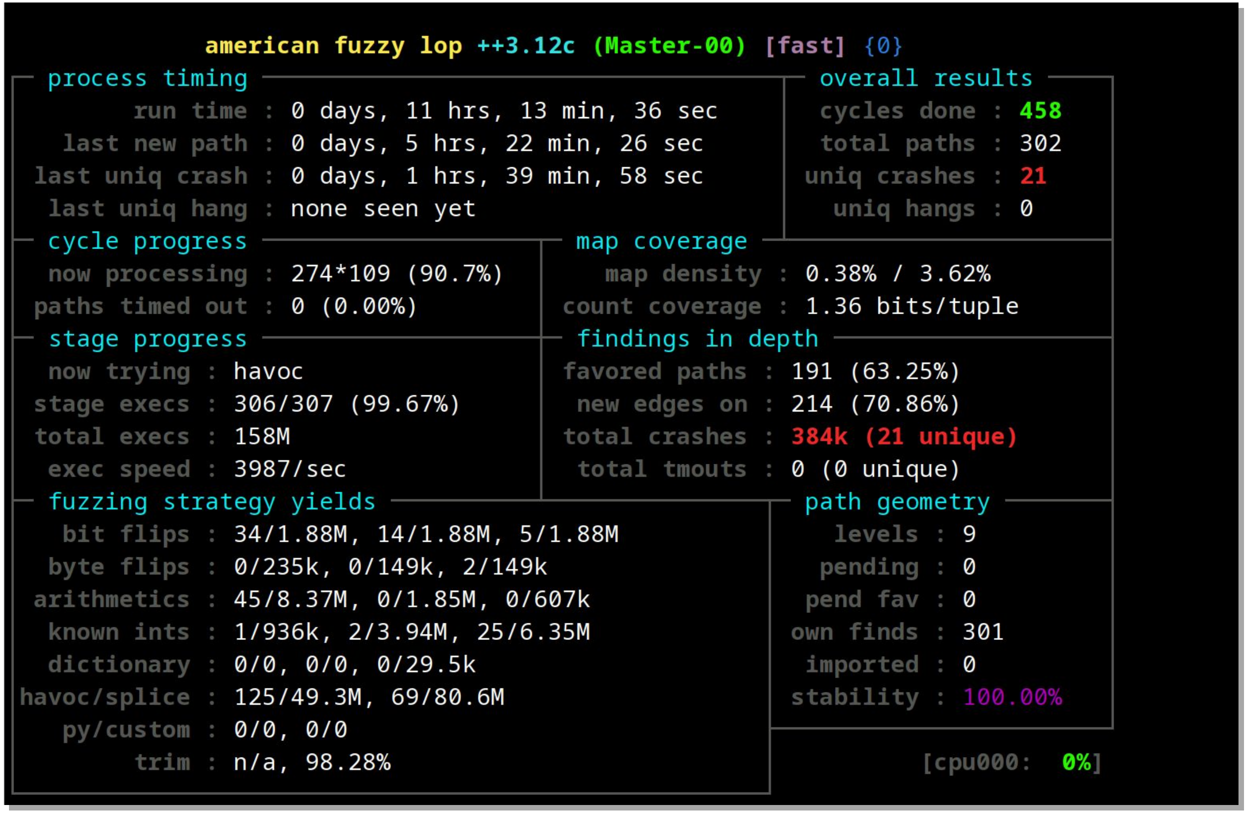
4. Results

4.1 Bugs Found

Widevine Trusted Application
Google Nexus 6
Firmware lrx21o

Server Specifications

CPU	AMD EPYC 7713 64-Core Processor
Clockspeed	2.0 – 3.7 GHz
Memory	256 GB DDR4 3.2 GHz
Cache Size	L1: 8128 KB, L2: 63.5 MB, L3: 512 MB



```
              american fuzzy lop ++3.12c (Master-00) [fast] {0}
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 11 hrs, 13 min, 36 sec     │  cycles done : 458   │
│   last new path : 0 days, 5 hrs, 22 min, 26 sec      │  total paths : 302   │
│ last uniq crash : 0 days, 1 hrs, 39 min, 58 sec      │ uniq crashes : 21    │
│  last uniq hang : none seen yet                      │   uniq hangs : 0     │
├─ cycle progress ──────────────────┬─ map coverage ─┴──────────────────────┤
│  now processing : 274*109 (90.7%)  │    map density : 0.38% / 3.62%        │
│ paths timed out : 0 (0.00%)        │ count coverage : 1.36 bits/tuple      │
├─ stage progress ──────────────────┼─ findings in depth ───────────────────┤
│  now trying : havoc                │ favored paths : 191 (63.25%)          │
│ stage execs : 306/307 (99.67%)     │  new edges on : 214 (70.86%)          │
│ total execs : 158M                 │ total crashes : 384k (21 unique)      │
│  exec speed : 3987/sec             │  total tmouts : 0 (0 unique)          │
├─ fuzzing strategy yields ─────────┴─────────────┬─ path geometry ────────┤
│   bit flips : 34/1.88M, 14/1.88M, 5/1.88M        │    levels : 9          │
│  byte flips : 0/235k, 0/149k, 2/149k             │   pending : 0          │
│ arithmetics : 45/8.37M, 0/1.85M, 0/607k          │  pend fav : 0          │
│  known ints : 1/936k, 2/3.94M, 25/6.35M          │ own finds : 301        │
│  dictionary : 0/0, 0/0, 0/29.5k                  │  imported : 0          │
│havoc/splice : 125/49.3M, 69/80.6M                │ stability : 100.00%    │
│   py/custom : 0/0, 0/0                           ├────────────────────────┘
│        trim : n/a, 98.28%                        │          [cpu000:  0%]
└──────────────────────────────────────────────────┘
```

4.1 Bugs Found

Widevine Trusted Application
Google Nexus 6
Firmware lrx21o

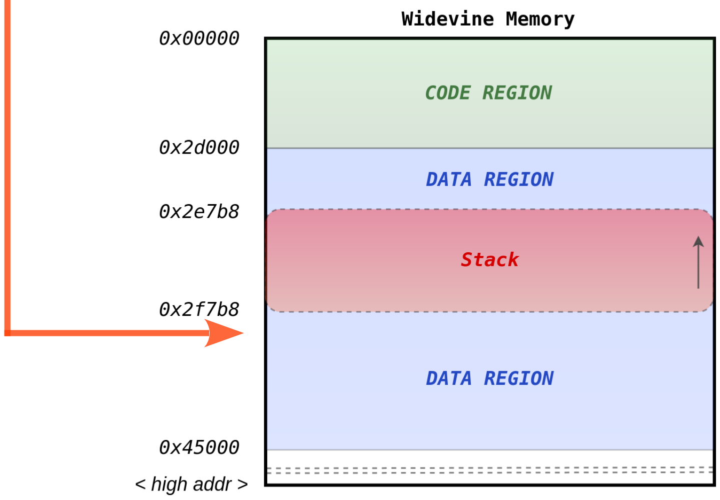
- **50001 (drm_save_keys)**
  - BUG-01: Buffer Overflow (0x69b0)
  - BUG-02: Buffer Overflow (0x6a18)
- **50002 (drm_verify_keys)**
  - BUG-03: Buffer Overflow (0x730c)
  - BUG-04: Buffer Overflow (0x7370)
- **50003 (PRDiagMaintenance)**
  - BUG-05: Buffer Overflow (PRDiagClearProvisioning @ 0x583c)
  - BUG-06: Buffer Overflow (PRDiagVerifyProvisioning @ 0x5f90)
- **50004 (PRDiagProvisionData)**
  - BUG-07: Buffer Overflow (PRDiagParseAndStoreData @ 0x5c9c)
  - BUG-08: Buffer Overflow (PRDiagParseAndStoreData @ 0x5cc8)

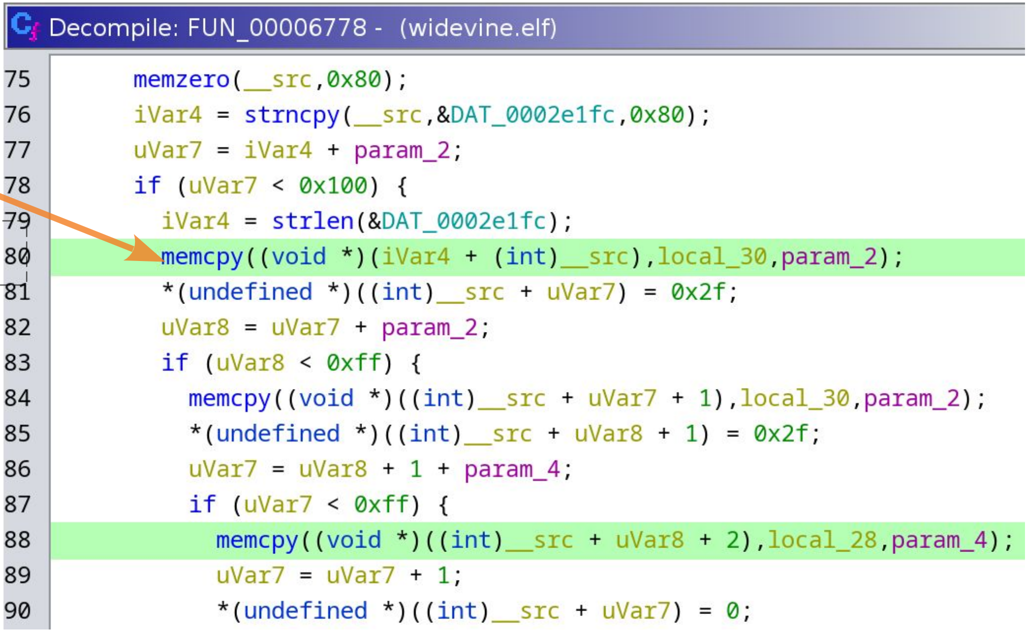
4.1 Bugs Found

50001 (drm_save_keys)

- BUG-01: Buffer Overflow (0x69b0)

```
memcpy(0x2f7d6, &feature_name, feature_name_len)
```





```
    memzero(__src,0x80);
    iVar4 = strncpy(__src,&DAT_0002e1fc,0x80);
    uVar7 = iVar4 + param_2;
    if (uVar7 < 0x100) {
      iVar4 = strlen(&DAT_0002e1fc);
      memcpy((void *)(iVar4 + (int)__src),local_30,param_2);
      *(undefined *)((int)__src + uVar7) = 0x2f;
      uVar8 = uVar7 + param_2;
      if (uVar8 < 0xff) {
        memcpy((void *)((int)__src + uVar7 + 1),local_30,param_2);
        *(undefined *)((int)__src + uVar8 + 1) = 0x2f;
        uVar7 = uVar8 + 1 + param_4;
        if (uVar7 < 0xff) {
          memcpy((void *)((int)__src + uVar8 + 2),local_28,param_4);
          uVar7 = uVar7 + 1;
          *(undefined *)((int)__src + uVar7) = 0;
```

4.1 Bugs Found

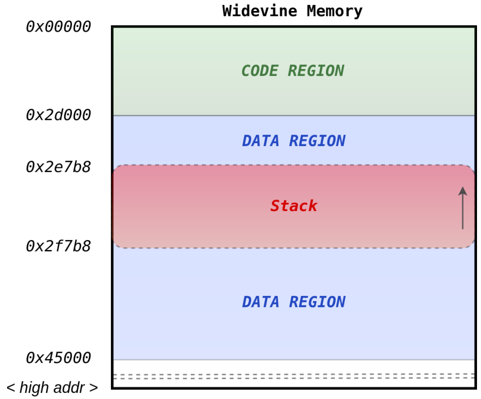
50001 (drm_save_keys)

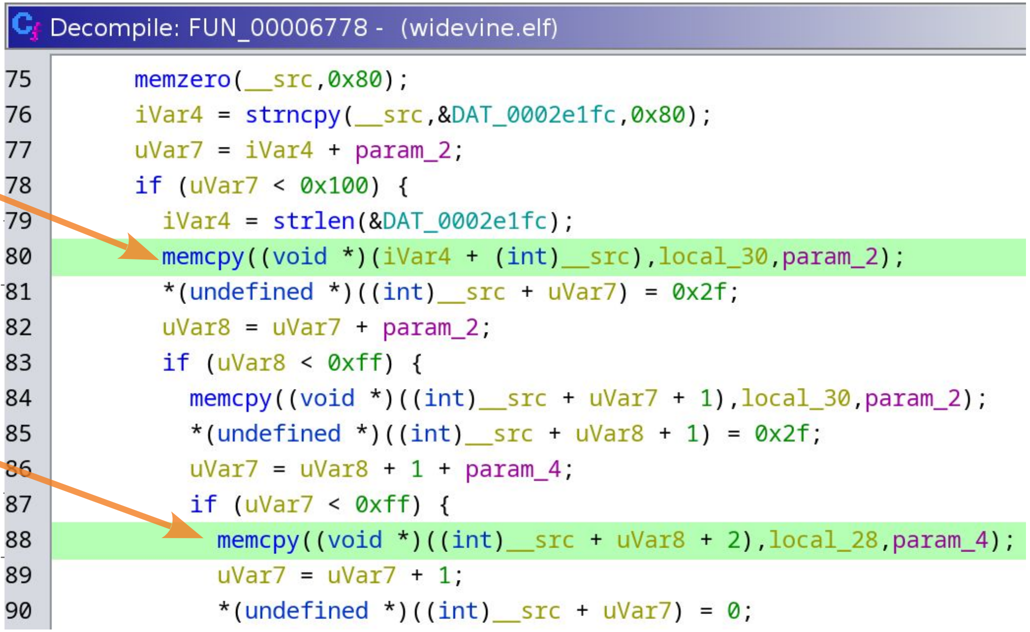
- BUG-01: Buffer Overflow (0x69b0)

```
memcpy(0x2f7d6, &feature_name, feature_name_len)
```

- BUG-02: Buffer Overflow (0x6a18)

```
memcpy(0x2f88a, &file_name, file_name_len)
```




Decompile: FUN_00006778 - (widevine.elf)

```
    memzero(__src,0x80);
    iVar4 = strncpy(__src,&DAT_0002e1fc,0x80);
    uVar7 = iVar4 + param_2;
    if (uVar7 < 0x100) {
      iVar4 = strlen(&DAT_0002e1fc);
      memcpy((void *)(iVar4 + (int)__src),local_30,param_2);
      *(undefined *)((int)__src + uVar7) = 0x2f;
      uVar8 = uVar7 + param_2;
      if (uVar8 < 0xff) {
        memcpy((void *)((int)__src + uVar7 + 1),local_30,param_2);
        *(undefined *)((int)__src + uVar8 + 1) = 0x2f;
        uVar7 = uVar8 + 1 + param_4;
        if (uVar7 < 0xff) {
          memcpy((void *)((int)__src + uVar8 + 2),local_28,param_4);
          uVar7 = uVar7 + 1;
          *(undefined *)((int)__src + uVar7) = 0;
```

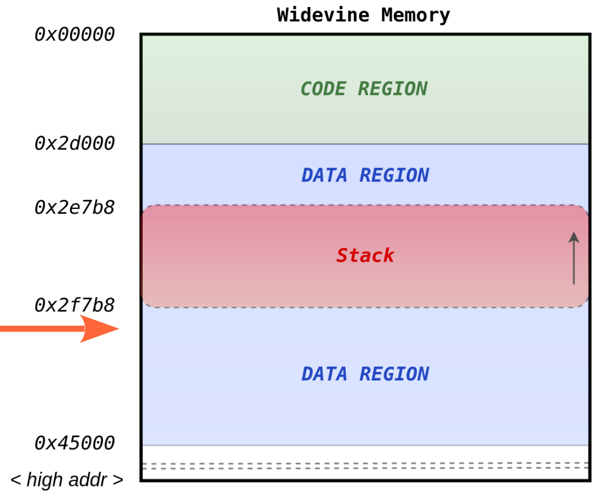
CYBER INTELLIGENCE

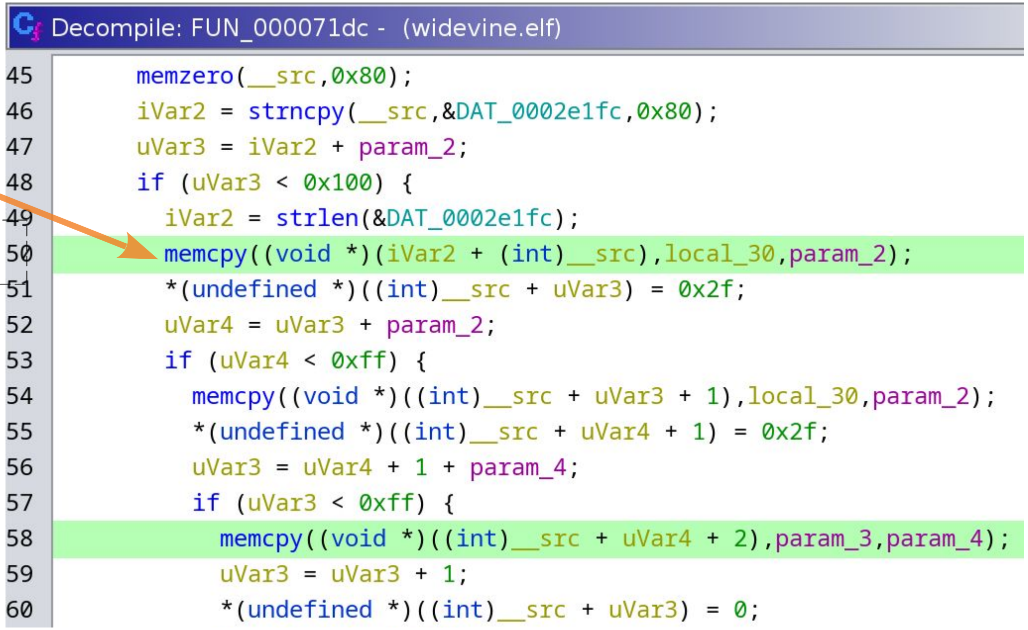
4.1 Bugs Found

50002 (drm_verify_keys)

- BUG-03: Buffer Overflow (0x730c)

```
memcpy(0x2f7d6, &feature_name, feature_name_len)
```





```
memzero(__src,0x80);
iVar2 = strncpy(__src,&DAT_0002e1fc,0x80);
uVar3 = iVar2 + param_2;
if (uVar3 < 0x100) {
  iVar2 = strlen(&DAT_0002e1fc);
  memcpy((void *)(iVar2 + (int)__src),local_30,param_2);
  *(undefined *)((int)__src + uVar3) = 0x2f;
  uVar4 = uVar3 + param_2;
  if (uVar4 < 0xff) {
    memcpy((void *)((int)__src + uVar3 + 1),local_30,param_2);
    *(undefined *)((int)__src + uVar4 + 1) = 0x2f;
    uVar3 = uVar4 + 1 + param_4;
    if (uVar3 < 0xff) {
      memcpy((void *)((int)__src + uVar4 + 2),param_3,param_4);
      uVar3 = uVar3 + 1;
      *(undefined *)((int)__src + uVar3) = 0;
```

4.1 Bugs Found

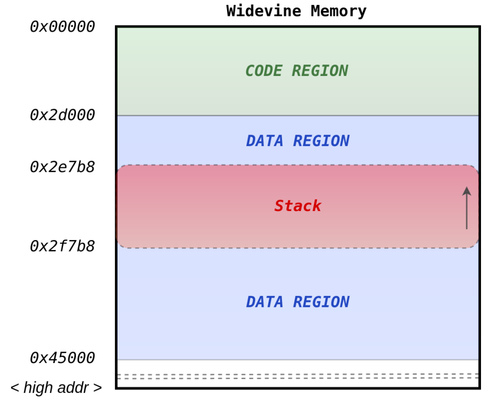
50002 (drm_verify_keys)

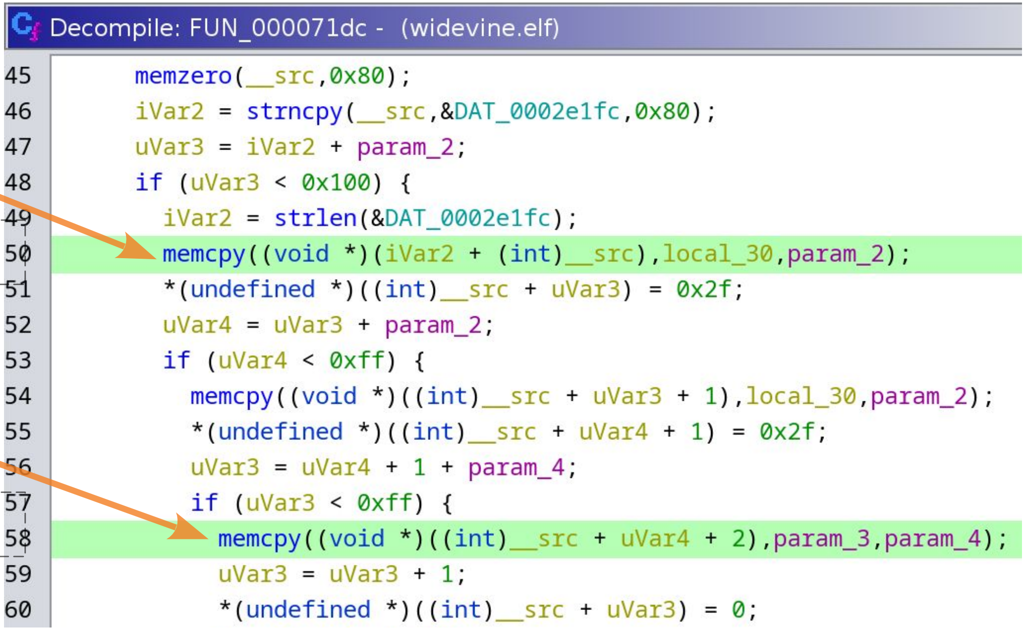
- BUG-03: Buffer Overflow (0x730c)

```
memcpy(0x2f7d6, &feature_name, feature_name_len)
```

- BUG-04: Buffer Overflow (0x7370)

```
memcpy(0x2f7e2, &file_name, file_name_len)
```





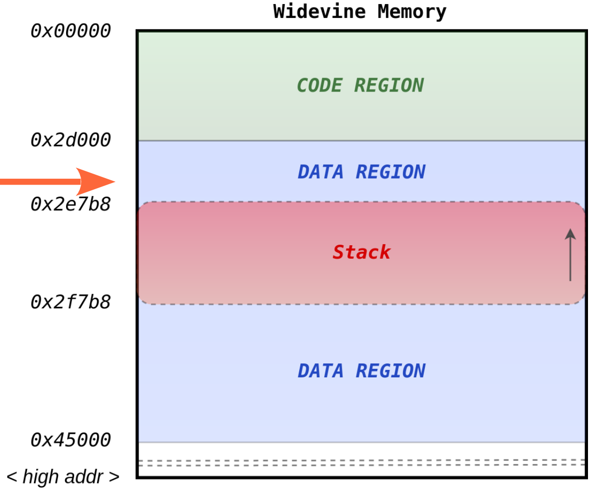
```
memzero(__src,0x80);
iVar2 = strncpy(__src,&DAT_0002e1fc,0x80);
uVar3 = iVar2 + param_2;
if (uVar3 < 0x100) {
  iVar2 = strlen(&DAT_0002e1fc);
  memcpy((void *)(iVar2 + (int)__src),local_30,param_2);
  *(undefined *)((int)__src + uVar3) = 0x2f;
  uVar4 = uVar3 + param_2;
  if (uVar4 < 0xff) {
    memcpy((void *)((int)__src + uVar3 + 1),local_30,param_2);
    *(undefined *)((int)__src + uVar4 + 1) = 0x2f;
    uVar3 = uVar4 + 1 + param_4;
    if (uVar3 < 0xff) {
      memcpy((void *)((int)__src + uVar4 + 2),param_3,param_4);
      uVar3 = uVar3 + 1;
      *(undefined *)((int)__src + uVar3) = 0;
```

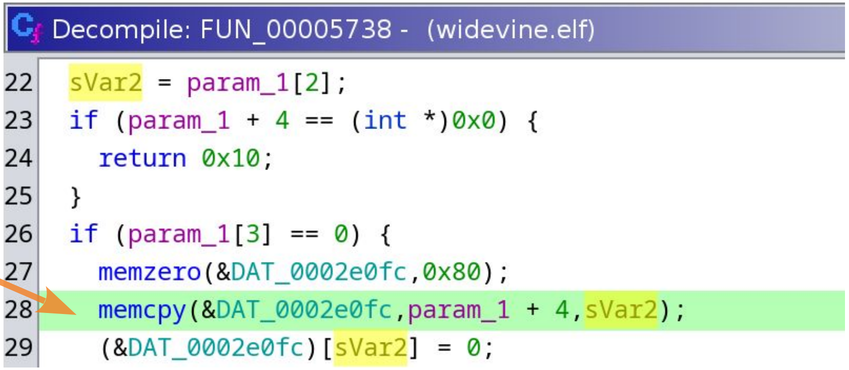
4.1 Bugs Found

50003 (PRDiagMaintenance)

- BUG-05: Buffer Overflow (PRDiagClearProvisioning @ 0x583c)

```
memcpy(0x2e0fc, &msg_buf, msg_buf_len)
```





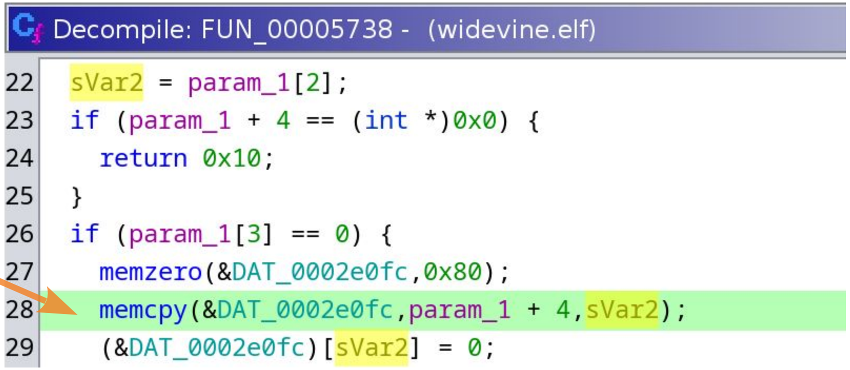
```
sVar2 = param_1[2];
if (param_1 + 4 == (int *)0x0) {
  return 0x10;
}
if (param_1[3] == 0) {
  memzero(&DAT_0002e0fc,0x80);
  memcpy(&DAT_0002e0fc,param_1 + 4,sVar2);
  (&DAT_0002e0fc)[sVar2] = 0;
```

4.1 Bugs Found

50003 (PRDiagMaintenance)

- BUG-05: Buffer Overflow (PRDiagClearProvisioning @ 0x583c)

```
memcpy(0x2e0fc, &msg_buf, msg_buf_len)
```

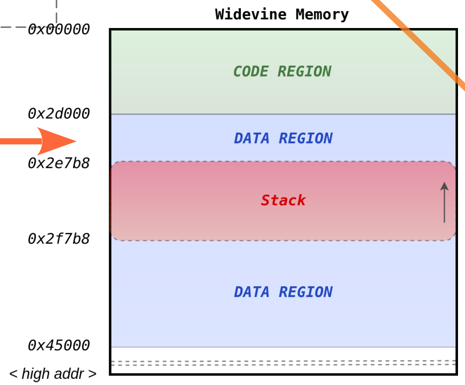


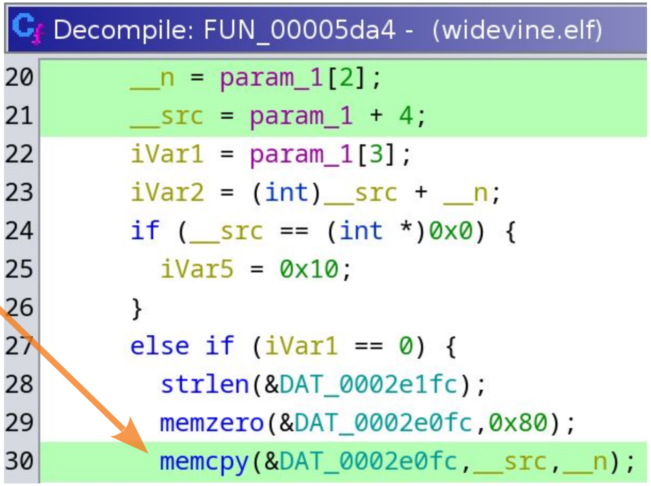
Decompile: FUN_00005738 - (widevine.elf)

```
sVar2 = param_1[2];
if (param_1 + 4 == (int *)0x0) {
  return 0x10;
}
if (param_1[3] == 0) {
  memzero(&DAT_0002e0fc,0x80);
  memcpy(&DAT_0002e0fc,param_1 + 4,sVar2);
  (&DAT_0002e0fc)[sVar2] = 0;
```

- BUG-06: Buffer Overflow (PRDiagVerifyProvisioning @ 0x5f90)

```
memcpy(0x2e0fc, &msg_buf, msg_buf_len)
```





Decompile: FUN_00005da4 - (widevine.elf)

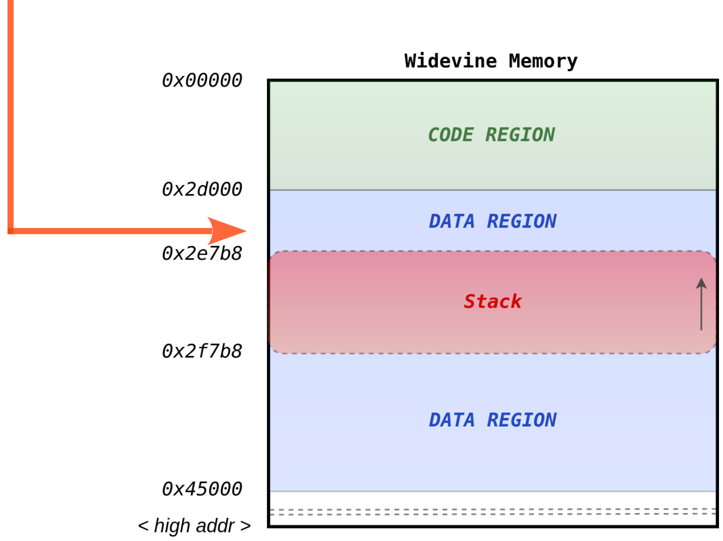
```
    __n = param_1[2];
    __src = param_1 + 4;
    iVar1 = param_1[3];
    iVar2 = (int)__src + __n;
    if (__src == (int *)0x0) {
      iVar5 = 0x10;
    }
    else if (iVar1 == 0) {
      strlen(&DAT_0002e1fc);
      memzero(&DAT_0002e0fc,0x80);
      memcpy(&DAT_0002e0fc,__src,__n);
```

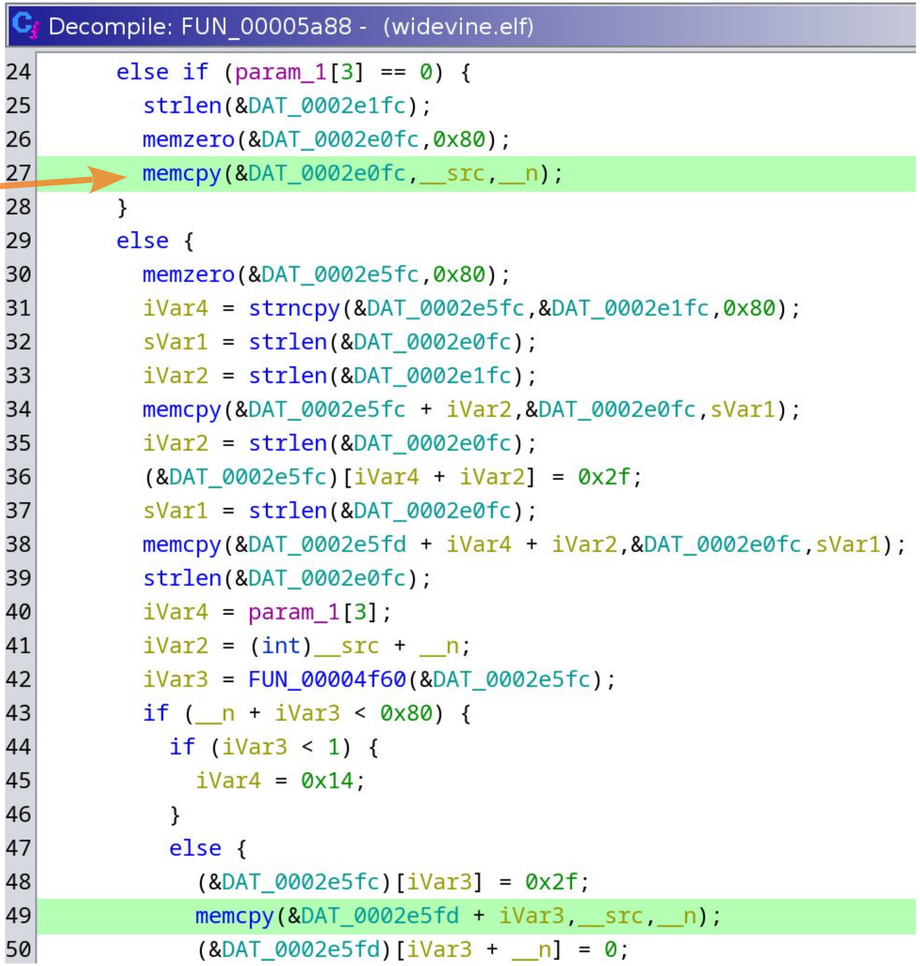
4.1 Bugs Found

50004 (PRDiagProvisionData)

- BUG-07: Buffer Overflow (PRDiagParseAndStoreData @ 0x5c9c)

```
memcpy(0x2e0fc, &msg_buf, msg_buf_len)
```





```
    else if (param_1[3] == 0) {
      strlen(&DAT_0002e1fc);
      memzero(&DAT_0002e0fc,0x80);
      memcpy(&DAT_0002e0fc,__src,__n);
    }
    else {
      memzero(&DAT_0002e5fc,0x80);
      iVar4 = strncpy(&DAT_0002e5fc,&DAT_0002e1fc,0x80);
      sVar1 = strlen(&DAT_0002e0fc);
      iVar2 = strlen(&DAT_0002e1fc);
      memcpy(&DAT_0002e5fc + iVar2,&DAT_0002e0fc,sVar1);
      iVar2 = strlen(&DAT_0002e0fc);
      (&DAT_0002e5fc)[iVar4 + iVar2] = 0x2f;
      sVar1 = strlen(&DAT_0002e0fc);
      memcpy(&DAT_0002e5fd + iVar4 + iVar2,&DAT_0002e0fc,sVar1);
      strlen(&DAT_0002e0fc);
      iVar4 = param_1[3];
      iVar2 = (int)__src + __n;
      iVar3 = FUN_00004f60(&DAT_0002e5fc);
      if (__n + iVar3 < 0x80) {
        if (iVar3 < 1) {
          iVar4 = 0x14;
        }
        else {
          (&DAT_0002e5fc)[iVar3] = 0x2f;
          memcpy(&DAT_0002e5fd + iVar3,__src,__n);
          (&DAT_0002e5fd)[iVar3 + __n] = 0;
```

4.1 Bugs Found

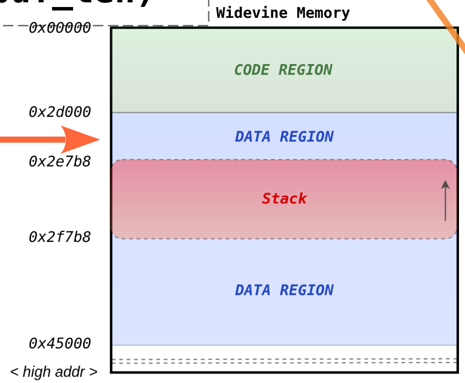
50004 (PRDiagProvisionData)

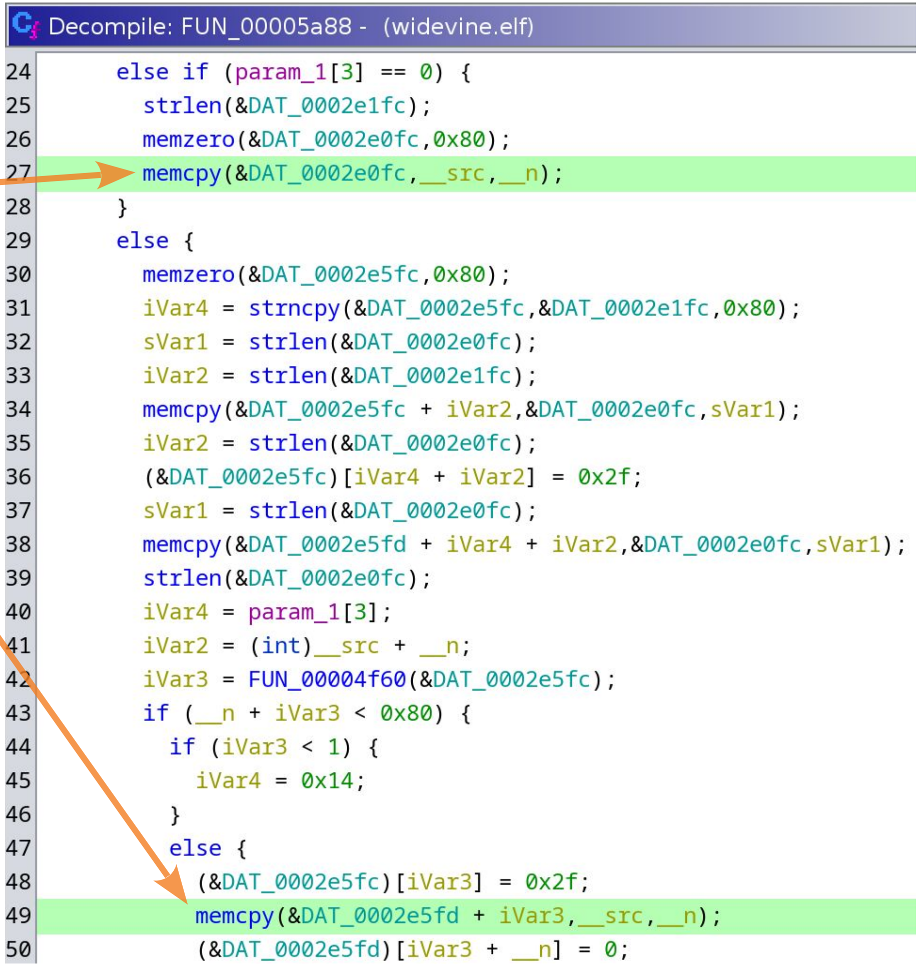
- BUG-07: Buffer Overflow (PRDiagParseAndStoreData @ 0x5c9c)

```
memcpy(0x2e0fc, &msg_buf, msg_buf_len)
```

- BUG-08: Buffer Overflow (PRDiagParseAndStoreData @ 0x5cc8)

```
memcpy(0x2e60a, &msg_buf, msg_buf_len)
```




Decompile: FUN_00005a88 - (widevine.elf)

```
else if (param_1[3] == 0) {
  strlen(&DAT_0002e1fc);
  memzero(&DAT_0002e0fc,0x80);
  memcpy(&DAT_0002e0fc,__src,__n);
}
else {
  memzero(&DAT_0002e5fc,0x80);
  iVar4 = strncpy(&DAT_0002e5fc,&DAT_0002e1fc,0x80);
  sVar1 = strlen(&DAT_0002e0fc);
  iVar2 = strlen(&DAT_0002e1fc);
  memcpy(&DAT_0002e5fc + iVar2,&DAT_0002e0fc,sVar1);
  iVar2 = strlen(&DAT_0002e0fc);
  (&DAT_0002e5fc)[iVar4 + iVar2] = 0x2f;
  sVar1 = strlen(&DAT_0002e0fc);
  memcpy(&DAT_0002e5fd + iVar4 + iVar2,&DAT_0002e0fc,sVar1);
  strlen(&DAT_0002e0fc);
  iVar4 = param_1[3];
  iVar2 = (int)__src + __n;
  iVar3 = FUN_00004f60(&DAT_0002e5fc);
  if (__n + iVar3 < 0x80) {
    if (iVar3 < 1) {
      iVar4 = 0x14;
    }
    else {
      (&DAT_0002e5fc)[iVar3] = 0x2f;
      memcpy(&DAT_0002e5fd + iVar3,__src,__n);
      (&DAT_0002e5fd)[iVar3 + __n] = 0;
```

4.2 Post-Analysis

Widevine Trusted Application
Google Nexus 6
Firmware lrx21o

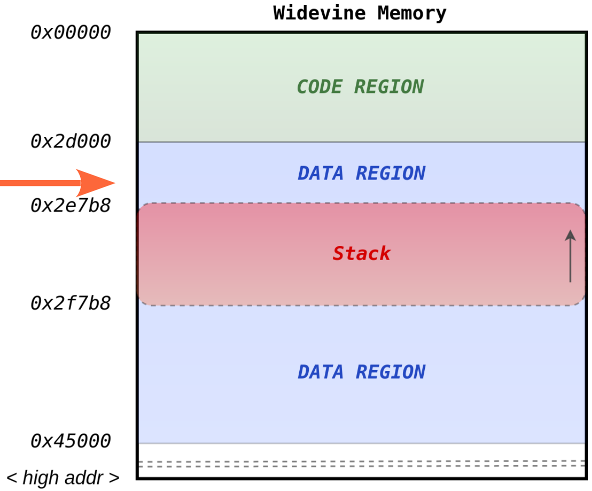
- **50001 (drm_save_keys)**
  - BUG-01: Buffer Overflow (0x69b0)
  - BUG-02: Buffer Overflow (0x6a18)
- **50002 (drm_verify_keys)**
  - BUG-03: Buffer Overflow (0x730c)
  - BUG-04: Buffer Overflow (0x7370)
- **50003 (PRDiagMaintenance)**
  - BUG-05: Buffer Overflow (PRDiagClearProvisioning @ 0x583c)
  - BUG-06: Buffer Overflow (PRDiagVerifyProvisioning @ 0x5f90)
- **50004 (PRDiagProvisionData)**
  - BUG-07: Buffer Overflow (PRDiagParseAndStoreData @ 0x5c9c)
  - BUG-08: Buffer Overflow (PRDiagParseAndStoreData @ 0x5cc8)

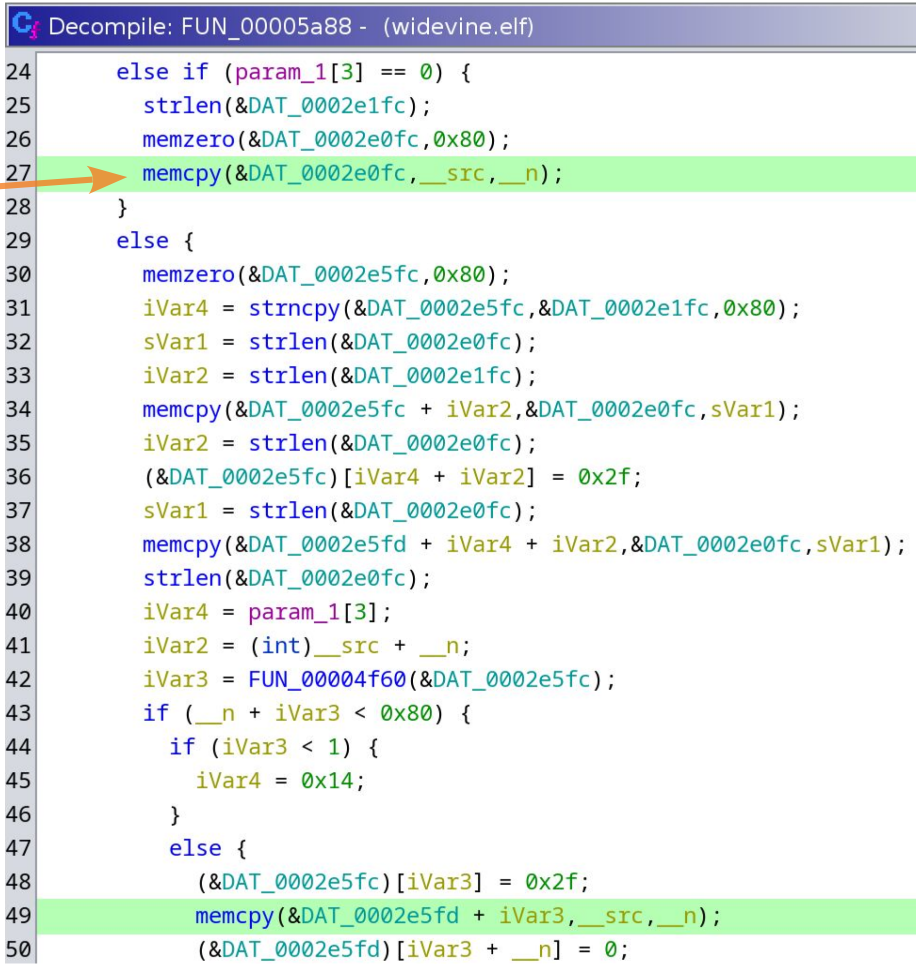
4.2 Post-Analysis

50004 (PRDiagProvisionData)

- BUG-07: Buffer Overflow (PRDiagParseAndStoreData @ 0x5c9c)

```
memcpy(0x2e0fc, &msg_buf, msg_buf_len)
```



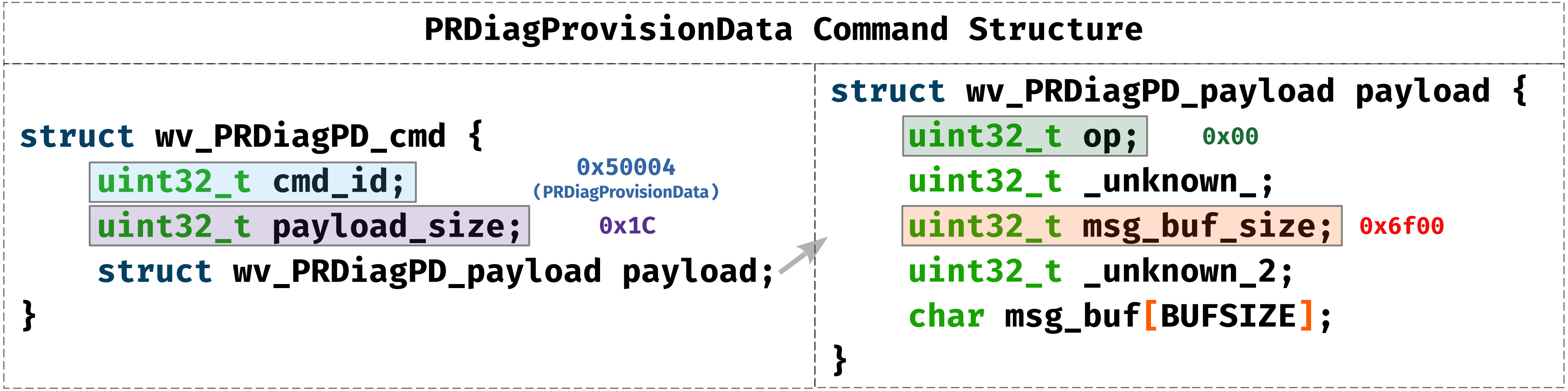

Decompile: FUN_00005a88 - (widevine.elf)

```
  else if (param_1[3] == 0) {
    strlen(&DAT_0002e1fc);
    memzero(&DAT_0002e0fc,0x80);
    memcpy(&DAT_0002e0fc,__src,__n);
  }
  else {
    memzero(&DAT_0002e5fc,0x80);
    iVar4 = strncpy(&DAT_0002e5fc,&DAT_0002e1fc,0x80);
    sVar1 = strlen(&DAT_0002e0fc);
    iVar2 = strlen(&DAT_0002e1fc);
    memcpy(&DAT_0002e5fc + iVar2,&DAT_0002e0fc,sVar1);
    iVar2 = strlen(&DAT_0002e0fc);
    (&DAT_0002e5fc)[iVar4 + iVar2] = 0x2f;
    sVar1 = strlen(&DAT_0002e0fc);
    memcpy(&DAT_0002e5fd + iVar4 + iVar2,&DAT_0002e0fc,sVar1);
    strlen(&DAT_0002e0fc);
    iVar4 = param_1[3];
    iVar2 = (int)__src + __n;
    iVar3 = FUN_00004f60(&DAT_0002e5fc);
    if (__n + iVar3 < 0x80) {
      if (iVar3 < 1) {
        iVar4 = 0x14;
      }
      else {
        (&DAT_0002e5fc)[iVar3] = 0x2f;
        memcpy(&DAT_0002e5fd + iVar3,__src,__n);
        (&DAT_0002e5fd)[iVar3 + __n] = 0;
```

4.2 Post-Analysis

`id:000004,sig:06,src:000087,time:129926,op:flip2,pos:8` → **`bug-07.bin`**

```
dev@cyberintel:fuzzer-results$ hexdump -C bug-07.bin
00000000  04 00 05 00 1c 00 00 00  00 00 00 00 00 00 21 00  |..............!.|
00000010  00 6f 00                                          |.o.|
```



PRDiagProvisionData Command Structure

```
struct wv_PRDiagPD_cmd {
    uint32_t cmd_id;                        0x50004
                                            (PRDiagProvisionData)
    uint32_t payload_size;                  0x1C
    struct wv_PRDiagPD_payload payload;
}
```

```
struct wv_PRDiagPD_payload payload {
    uint32_t op;                0x00
    uint32_t _unknown_;
    uint32_t msg_buf_size;      0x6f00
    uint32_t _unknown_2;
    char msg_buf[BUFSIZE];
}
```

4.2 Post-Analysis

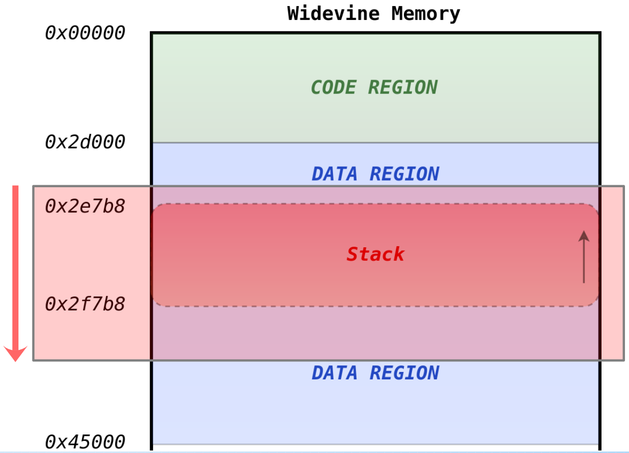
PRDiagProvisionData Command Structure

```
struct wv_PRDiagPD_cmd {
    uint32_t cmd_id;             0x50004
    uint32_t payload_size;       0x1C
    struct wv_PRDiagPD_payload payload;
}
```

```
struct wv_PRDiagPD_payload payload {
    uint32_t op;         0x00
    uint32_t _unknown_;
    uint32_t msg_buf_size;  0x6f00
    uint32_t _unknown_2;
    char msg_buf[BUFSIZE];
}
```

```
memcpy(0x2e0fc, &msg_buf, msg_buf_len)
```

```
0x2e0fc + 0x6f00 = 0x34ffc
```



Proof of Concept

Conclusions

Conclusions

- Mobile devices are a highly attractive target for attackers
- Security of sensitive data in smartphones rely on TEEs
  - The ability to audit its security is critical
  - Trusted OS and Applications can also have vulnerabilities
- Mobile TEEs are typically less tested than Normal World
- We presented our observations about the Qualcomm's TEE
  - Contributing knowledge about their trusted OS and TAs
- We have developed tools especially valuable for auditing their security
  - Working on real devices
- We found 8 vulnerabilities that were already fixed in `7.1.1`

EXFILES Grant Agreement No. 883156

“The **EXFILES** project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement
No **883156**.”

If you need further information, please contact the coordinator:

Technikon Forschungs- und Planungsgesellschaft mbH
Burgplatz 3a, 9500 Villach, AUSTRIA
Phone: +43 4242 233 55   Fax: +43 4242 233 55 77
Mail: **coordination@exfiles.eu**

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author`s view – the European Commission is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.