SUMMARY REPORT – UDG 20.2.0

HUAWEI TECHNOLOGIES CO., LTD. AND ITS SUBSIDIARIES

Version:	1.0
Date:	09.05.2020

ERNW
providing security.

TABLE OF CONTENT

1 Handling

The present document is classified as SECRET. Any distribution or disclosure of this document REQUIRES the permission of the document owner as referred in Section “Document Status and Owner”.

1.1 Document Status and Owner

As the owner of this report, the document owner has exclusive authority to decide on the dissemination of this document and responsibility for the distribution of the applicable version in each case to the places defined in the respective section.

The possible entries for the status of the document are “Initial Draft”, “Draft”, “Effective” (currently applicable) and “Obsolete”.

Title:	**SUMMARY REPORT – UDG 20.2.0 – HUAWEI TECHNOLOGIES CO., LTD. and its subsidiaries**
Document Owner:	HUAWEI TECHNOLOGIES CO., LTD., AND ITS SUBSIDIARIES
Version:	1.0
Status:	Effective

2 Initial Situation and Scope

Huawei Technologies Co., Ltd. (Huawei) provides the Unified Distributed Gateway (UDG). UDG bears functions of the so called User Plane Function (UPF) on the 5G core network and those of the Serving Gateway for User Plane (SGW-U) and Packet Data Network Gateway for User Plane (PGW-U) on the traditional network. This enables the UDG to process both 4G and 5G subscriber services. UDG contains product and platform components. The product functionalities include functionality like path management, backup and fault recovery. The platform components provide services like routing and operational management.

ERNW performed a technical software engineering capability review in February 2020 in the Huawei Cyber Security Transparency Centre in Brussels, Belgium. Access to source code and the build processes was granted with high security requirements in an isolated environment providing a strict separation of concerns based on the components in scope. The scope of the technical review was the code base of UDG 20.2.0 5G, the code bases of UDG 20.2.0 platform components and their corresponding build processes prioritized by Huawei. The technical review focused on the product components and the key platform components. The source code in scope contained approximately 30 million lines of code.

The objective of the technical review was to provide Huawei an assessment for the overall source code quality and maturity of the code base of UDG. The overall quality of the source code was measured by key indicators like complexity, usage of unsafe functions and test coverage. In addition to automated tooling, manual checks were performed by ERNW senior auditors to identify general usage of anti-patterns, indicators of insecure coding, and general violations of best practice.

The technical review of the build processes focused on the application of hardening measures and the reproducibility of the build itself.

Lifecycle management for open source components was also in scope of the technical review with a focus on common best practices.

During the whole technical review the auditors were supported with documentation by Huawei's specialists. The auditors were also provided output for different types of tests performed by Huawei.

The technical review shall support Huawei in improving the software development process, the overall quality of the source code and the resulting artifacts.

This summary report includes high level information on the results of the technical review. Detailed information was provided by ERNW to Huawei in separate technical reports.

3 Results

3.1 Source Code Quality

The following section covers the results of the technical review of the overall source code quality for the components in scope. Source code was provided by Huawei in archive files that also contained documentation about build processes and all dependencies needed to create a corresponding artifact.

The quality of source code can be determined by quantitative and qualitative aspects. To cover both aspects in an efficient way, the auditors focused on three major approaches during the project: complexity analysis, variant analysis and static code analysis. In addition, a limited dynamic analysis was performed.

Source Code Complexity Analysis

Complexity of source code is a quantitative indicator that can be measured by different metrics. The auditors used cyclomatic complexity as a metric to determine the susceptibility to defects of the source code in scope. To determine the cyclomatic complexity, tools supplied by the auditors were used on the code provided by Huawei. The threshold was set by the auditors to a value of 100 per file considering industry best practices to represent the inherent complexity of the components. Further, a threshold per file was chosen to also account for large files that are hard to maintain.

For the UDG 5G components the average complexity per file was calculated with 90.284. The platform average was calculated with 77.22.

The results show that the source code complexity on average is below the threshold with a low dispersion of the individual complexity values for each component. This is a good indicator that the complexity of the source code is appropriate and accidental complexity must be reduced only in the minority of the components. Additionally, only less than 5% of all product component functions are to be considered complex. Taking literature on clean coding practices into account this is to be considered a good result.

To address the shortcomings of cyclomatic complexity following McCabe's definition and to provide a more holistic view on aspects like readability and maintainability, the auditors also used an approach to determine the complexity of the source code from a qualitative perspective. The methods used included automated analysis tools and a representative manual examination of the source code.

The results of the qualitative complexity analysis support the outcome of the cyclomatic complexity analysis for the product components. Again, the vast majority of the product components is below the threshold and can be considered to have an appropriate qualitative complexity.

Static Analysis of the Source Code

To cover additional qualitative aspects of the source code, static code analysis was performed for the product and platform components. The analysis focused on the identification of anti-patterns, bad practices and classic software bugs. Static analysis was also used to assist generating the results for Code Duplication and unsafe functions described below.

Automated tooling for the analysis reached a full coverage of the source code for the product components and a representative coverage for the platform components. Additional manual checks and verifications have been performed. For the identification of certain bug classes caused by complex calculations or formulas the auditors applied theorem prover checks. The majority of the issues was detected by automated tooling.

Code Duplication

For the product components code duplication was found to be below 3% for all the components and on 2.2% on average. The auditors recommend inspecting the duplications in the components with higher values to further improve the quality of the code. However, this is to be considered a good result.

The results for the platform components are incomplete due to a technical failure of the tools used to analyze the source code. The code duplication was found to be below 3% for the components that could be evaluated in the given time frame. This is to be considered a good result.

Unsafe Functions

The use of unsafe functions was also considered during the technical review. Huawei provides own safe function alternatives and their developers are required to make use of those alternatives.

The significant amount of problematic functions identified are type conversions from strings to numbers with undefined behavior. Huawei is aware of the problems coming from that class of functions and is using additional safeguards to mitigate potential issues.

In addition, some occurrences of calls to *system* were identified. Where a call to *system* is unavoidable, additional safeguards were implemented.

Huawei should strive for a minimization of the usage of the functions mentioned above. The auditors recommend a regular review of the density of those functions. Ideally, this value can be decreased steadily.

As Huawei provides safe function alternatives in the software development process they seem to be aware of the general risk of unsafe functions and try to solve it with a holistic approach. Another indicator that those risks are handled appropriately is the internal automated vulnerability scanning that is performed by Huawei on a regular basis. From an auditor's perspective, Huawei seems to be giving a reasonable amount of attention to this risk.

Unit Tests

Huawei provided the auditors with unit tests for the corresponding components and associated documentation. Representative test cases were reviewed manually and were found to be technically suitable. Based on the provided information and test cases the auditors recommended that the test coverage should be increased for product and platform components with high value functionality, where a unit test coverage[1] of 75% was not achieved.

Variant Analysis of the Source Code

The purpose of a variant analysis is to identify a bad pattern used in the source code and then using it in a generalized approach to identify identical or similar patterns in other parts of the source code.

During the technical review, code that was assumed to decrease the quality of the components was used as an input for the variant analysis. The patterns used for the analysis were derived from a representative manual source code review and automated scanning results focusing on reliability and from the static analysis described above.

The variant analysis identified additional bad patterns. The recommendations documented in the technical report have been based on a differentiation between generalized issues identified with the variant analysis and individual issues from the static code analysis.

Dynamic Analysis

Code parts of the product components were subjected to dynamic analysis. The auditors chose isolated function blocks from representative core product components. Those isolated function blocks were built and supplied to a Fuzzer. The used Fuzzer was chosen to maximize coverage. Input for the fuzzing process was generated mutation-based and aware of the input structure expected by the isolated code. Positive test cases with either crashes or undefined states were manually verified by analyzing the corresponding source code and documented.

The auditors identified positive test cases and recommended mitigating measures. The results are common for projects with similar characteristics in terms of complexity and size of the code base.

Maintainability and Reliability

Taking the results of previous sections and the rating of supporting tools into account, the maintainability of the product and platform components seems to meet industry best-practices. As described in the sections above, certain aspects that could impact reliability have been identified. Applicable solutions for those issues should be integrated into the software engineering process in a sustainable way.

[1] *The value of 75% coverage was chosen by the auditors considering the overall complexity and safety requirements for UDG. Industry standards like ISO 26262 and IEC 61508 where considered to set this threshold to a reasonable value.*

Conclusion

The following section is a summary of the technical results of the Source Code Quality review.

Source Code Complexity Analysis: The source code complexity is below the threshold and appropriate.
Code Duplication: Duplicate code is present only in a very low and appropriate amount.
Unsafe Functions: Unsafe functions have been avoided extensively.
Unit Tests: Test coverage can be improved in some critical modules.

3.2 Build Processes

The following section covers the results of the technical review of the build processes. The auditors had access to build environments running on infrastructure provided by Huawei and were able to execute and create builds for all the components in scope.

Huawei provided documentation on how to execute the build processes. The tools used during the build processes are state-of-the-art and compose effective build systems.

Secure Compilation Options

The auditors reviewed the hardening measures that are applied to the binaries during the compilation process. The build processes are set up to apply those hardening measures to resulting binaries. The following compiler settings were checked in a representative set of binaries out of the release files provided by Huawei: RELRO, Stack Canary, NX, PIE, RPATH and RUNPATH.

The results show that the vast majority of binary files had the hardening measures applied. An exception are stack canaries. These seem to be missing for about half of the binaries. The auditors assume that the usage of the compiler flag *-fstack-protector-strong* might be the cause for this, as it omits stack canaries for functions without local array definitions or without references to local frame addresses. For 99% of the binaries the hardening measures RELRO and PIE were enabled.

The auditors recommend to investigate the missing hardening measures and to continuously monitor the binaries with every release. Under the assumption made for the missing stack canaries the amount of hardening seems reasonably and can be considered good practice.

Binary Equivalence

To establish a chain of trust from source code to the resulting build artifact, reproducible builds are a common method used in modern software engineering. Huawei's build infrastructure is tailored to enable reproducible and deterministic builds by default. The source code built by the auditors resulted in binary equivalent outputs for different builds for all binaries. Minor deviations concerning reproducibility were observed for the resulting build artifacts (e.g. in Manifest files). All of the observed deviations were clarified by Huawei's specialists during the analysis. From the auditor's perspective these deviations can be considered as only minor issues

for the product components and are acceptable. The build artifacts created by the auditors were binary equivalent to the ones provided by Huawei via the Huawei Support Website with only the mentioned minor deviations. Artifacts resulting from the build process contain information (configuration files) about the modifications to the build environment. This increases transparency on how binary equivalence was achieved. The modifications applied by Huawei during the build process are reasonable from the auditor's perspective.

Conclusion

The following section is a summary of the technical results of the Build Processes review.

Secure Compilation Options: The binaries are compiled comprehensively with secure compilation options.

Binary Equivalence: All binaries are built with binary equivalence. Overall an acceptable amount of binary equivalence is achieved.

3.3 Open-source Lifecycle Management

Open-source libraries are commonly used in proprietary products. As these libraries usually cannot be maintained in the same way as the proprietary components, special requirements must be met to ensure an efficient and secure integration into the product.

To improve traceability, the usage of open-source dependencies, version updates and vulnerability fixing should be managed in a comprehensive approach. The used open-source dependencies should be updated in a timely manner to ensure security and reliability.

Replacing and updating third party open-source dependencies is only possible in an efficient way if the design principle of separation of concerns is applied. Functionality of the proprietary components should therefore not be mixed into open-source dependencies. If changes to the open-source libraries must be made they should be traceable and easy to apply on future versions of the libraries.

The auditors examined the usage of open-source dependencies in the product and platform components under the considerations mentioned above. This was done by a manual approach with tool support to improve coverage.

Open-source dependencies were provided as original source code and were kept in a separate folder next to the source code.

Modifications (e.g. for backporting issues) implemented by Huawei are provided as patch files to improve traceability and transparency. In addition to the libraries and the corresponding patch files documentation exists. The auditors were able to determine the used versions and the changes made to the libraries. The used dependencies are mostly up-to-date or just a few minor versions behind.

Due to the high number of open-source dependencies used in the product and platform code, representative checks were performed to identify missing security patches in the dependencies. The auditors could not

identify any outdated dependencies with missing security patches. The known security issues have either been patched into the code by Huawei themselves or were already in place because an up-to-date version was used.

Conclusion

Only minor improvements for the open-source lifecycle management were recommended. From the auditor's perspective the separation of the code, the handling, documentation and patch management are reasonable and meet all requirements of a state-of-the-art open-source lifecycle management.

3.4 Summary

Considering the results of the technical review and the project constraints mentioned above the overall source code quality is a good indicator that Huawei has established a mature and appropriate software engineering process for UDG. Given the complexity and size of the product and platform code, it is not surprising that aspects like test coverage and general code quality can be improved in some components. The auditors were able to identify shortcomings in a minority of the components, but were also able to identify compliance with best practices in the majority of the components in scope.