



5G-Pentest-UE: A Penetration Testing Framework for Identifying 5G System Vulnerabilities

Richard Riedel¹^[0009-0006-8123-8214] and
Stefan Köpsell²^[0000-0002-0466-562X]

Barkhausen Institut, Dresden, Germany

¹richard.riedel@barkhauseninstitut.org

²stefan.koepsell@barkhauseninstitut.org

<https://www.barkhauseninstitut.org/en/>

This preprint has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this contribution is published in LNCS volume 15999, and is available online at https://doi.org/10.1007/978-3-032-00644-8_20

Abstract. This paper presents and evaluates an open source penetration testing framework for finding vulnerabilities in 5G systems under the assumption of malicious end devices. This is achieved by enabling the creation of arbitrary (5G) messages to be transmitted over the air interface of a 5G system. Our framework is modular and scriptable, allowing the easy creation of test cases. It is based on the OpenAirInterface (OAI) open source 5G stack.

We evaluated our framework by implementing several tests and running it against a well-known open source 5G system. We were able to identify several vulnerabilities.

Keywords: 5G, security, OpenAirInterface, user equipment, penetration testing, vulnerabilities, mobile networks

1 Introduction

In today's society, mobile communications already play an important role in many areas of life. In the future, the importance of mobile communications will continue to grow due to its ever-increasing influence on various aspects of social coexistence. With the 5th generation (5G) of mobile communications and the associated technological innovations, for example, Internet of Things (IoT) applications are increasingly becoming part of mobile networks.

Today, most mobile networks are public networks. They are complex systems run by large operators. However, in the last few years, developments in the field of private networks have gained significant interest. Such private networks usually are smaller-sized 5G systems. They are often used as a more flexible and more performant alternative or complement to WLAN networks. With the

concurrent rise of open architectures like O-RAN and the resulting multi-vendor deployments, the number of different implementations of the 5G specifications is also growing. Besides this, the feature set of 5G is extended continuously to match the large number of new use case ideas, which also leads to many new implementations.

These developments result in a high chance of implementation errors. Also, the specifications for new and old features may not be perfect, which also favours the emergence of vulnerabilities. One obvious attack vector to utilize those errors is the air interface over which the User Equipments (UEs), such as phones or IoT devices, communicate with the 5G network. By the nature of wireless communication, this air interface will always be accessible by attackers, at least to some extent. Therefore, attacks over this interface must be considered wherever a 5G network is deployed.

In summary, in the current situation, many deployments and implementations of different 5G software stacks combined with the ability to attack them over the air demand rigorous tests to identify potential vulnerabilities early on. Therefore, we created a framework to facilitate the pentesting of 5G systems to find vulnerabilities before they can cause problems in production systems. Our contributions can be summarized as follows:

1. We designed a framework that allows the easy creation and execution of vulnerability test cases.
2. We evaluated the framework by testing an existing 5G stack.
3. We provide our framework including an extensive documentation as open source.

The rest of the paper is organized as follows: Section II gives an overview of the goals and requirements for our framework, Section III presents related work, while Section IV describes the design of our framework. We present the related evaluation results in Section V and conclude the paper in Section VI.

2 Development Goals and Requirements

As mentioned in the introduction, our goal is to develop a framework that allows us to test 5G systems with respect to vulnerabilities. Thereby, we concentrate on the case of a malicious user equipment trying to attack the 5G system. We decided on this scenario because it is more easily executable from an attacker's perspective compared to, e.g., manipulating the 5G system itself. This is supported by the fact that freely programmable UEs can be easily realised with the help of existing open source software stacks and cheap software-defined radios implementing the radio part.

To describe what the testing framework shall be capable of, we first need to precisely describe what an UE in a 5G system can do in theory and practice. From this knowledge, the capabilities of our framework can be derived.

UEs have at least one thing in common: they communicate with the 5G system through electromagnetic waves. Therefore, on the lowest level, they have

two capabilities: one is sending radio signals, and the other is receiving them. The 5G specifications describe precisely how UEs send and receive radio signals and how they talk to the 5G system. Thereby, protocols and related protocol messages are defined at several layers.

Our goal, therefore, consists of providing easy means to send and receive (manipulated) protocol messages at all layers. In the following, we describe the attacker model we have in mind and derive related requirements regarding our framework.

2.1 Attacker model

Our attacker model is based on the idea of a malicious UE. Therefore, the attacker is able to deviate from the 5G specifications when taking part in 5G communication. Note that following some parts of the specifications is necessary to be able to deviate from other parts. This might sound obvious, but emphasizing this fact is important to understand the framework's value, which makes it possible to deviate on different layers and in different situations easily.

We can subdivide our attacker model into two scenarios: the attacker can be either a legitimate user of a given 5G system or not. In the latter case, the attacker is an outsider.

This translates to the question of whether the attacker has valid authentication information to properly register with the 5G system or not. To a certain extent, our framework will be agnostic to this since it allows manipulation of all messages at all layers. Therefore, both attacker models can be analyzed using our framework.

Nevertheless, one limitation of our framework is that it works only on the digital layer. Therefore, no attacks at the physical layer (i.e., at the layer of the analogous radio signal) like jamming can be executed. Finally, attacks implemented with the help of the framework are always active attacks since the attacker needs to interact with the mobile network.

2.2 Framework requirements

Considering the attacker model, we find the following requirements for our framework.

First We want to be able to adhere to the specifications at any place needed. This means we first need software that can do everything a normal 5G UE can do. For example, basic functionality like finding base stations, connecting to a base station or registering at the core network.

Second We want to deviate wherever we want. This part then poses the requirement of being able to tell the framework where and how it should deviate from normal behaviour. To be more precise, there are two things we need to be able to control:

1. Message order: testing different states of the 5G system
2. Message content: testing the message processing in the different states

This is because different code is used to process the messages received in different connection states. Since we want to be able to test as many of the 5G system code as possible, we need to be able to send anything in any state.

To be able to send at any point, the attacker must adhere to the protocol for every step necessary to get to this point (first requirement) and then be able to send any message to test the implementation (second requirement).

Third An straight forward way of implementing all of this would be to enable the sending of arbitrary messages (streams of bits) at the lowest digital protocol layer. This would certainly allow sending anything in any state and, by that, attacking, e.g., the authentication implementation but also all other interfaces and protocol layers. The drawback is that to get to the wanted state, the attacker would basically need to implement many parts of the 5G protocols until he reaches the point he wants to attack. This leads us to the third and final requirement: the framework's usability. Usability here means it shall be relatively easy to understand the framework and implement vulnerability tests.

In summary, the primary goal of this work is to provide a holistic testing tool for mobile network security from the UE side that is easy to use.

Further requirements are:

- *Open Source*: the tool shall be made available to everyone who needs to enhance the security of a 5G system
- *Documentation*: to make the framework usable and understandable for everyone, it needs to be well documented
- *Update*: Since the 5G specifications are evolving permanently, maintaining the framework shall be easy.
- *Extendability*: extendability here means that it should be rather easy to add new testing functionality to the framework

The three primary and four soft goals constitute the requirements for the developed framework.

3 Related Work

In this section, we look at existing work in the field of security testing for 5G networks. There are quite a few papers that study 5G security and penetration testing. Some of them require direct access to the components that are to be tested or at least their interfaces. For example, some work tries to insert malicious messages between the base stations (gNB) and the core network [11, 12, 15]. Others only use simulation of the UE to test the gNB and core network [8]. This will allow for the testing of the necessary interfaces in most cases. However, there are scenarios where the pentester cannot directly access the interfaces of the system under test. For example, when opting for a private network solution

from a vendor, the customer should be able to check the security of such a system. This is one of the scenarios in which our framework can help.

In addition, much work on fuzzing 5G systems already exists. Most of it uses the network interfaces directly, as described above. Although there are works that allow for testing a black box network, the focus is on finding the right fuzzing content here. Therefore, the question of how to bring the fuzzing content to the system is only a necessary step, often conquered with some hacky solution that is not very flexible. For example, “Berserker” [14] focuses on generating the fuzzing content with the help of the ASN.1 protocol descriptions given by the 4G/5G specifications. “5G RRC Protocol and Stack Vulnerabilities Detection via Listen-and-Learn” [18] uses machine learning to understand 5G traffic and after that use the gained knowledge to create messages for fuzzing.

Future 5G fuzzing research could massively benefit from a pentesting framework, as it is provided in our paper. Especially because the framework allows for testing many different endpoints and states of the 5G system. Therefore, researchers do not need to work on how to send the fuzzing messages to the relevant interfaces and can instead completely focus on creating good fuzzing messages.

Finally, there are some papers that provide somewhat similar functionality to the framework provided in our work. But those have some shortcomings or different focus points, which makes the creation of an extensive and easy-to-use framework necessary.

The following list provides an overview of all work we are aware of regarding utilizing UEs for security testing of 5G systems.

- “An Automated Vulnerability Detection Method for the 5G RRC Protocol Based on Fuzzing” [17] is an OpenAirInterface [3] based framework. It focuses on the Radio Resource Control (RRC) layer, which might make it hard to test the core network with this tool, due to things like Non Access-Stratum (NAS) integrity checks and encryption. Also, the work is not open source, and we could not find any implementation.
- “An Experimental Testbed for 5G Network Security Assessment” [7] has a similar approach as we have in our paper, but uses srsRAN [5] as a basis for their UE implementation. Also, the source code is only available upon request, and it seems that altering the connection establishment messages is not directly possible (only injection in “already established communications between a UE and a base station” [7]).
- “Towards Automated Fuzzing of 4G/5G Protocol Implementations Over the Air” [10] seems to have a similar approach to ours. Here, the OpenAirInterface [3] UE code is modified to allow intercepting, modifying, and replaying packets. The problem is that the source code is only available upon request, and our requests were not answered. Another issue is that this work, similar to [17], might (based on the explanations given in the paper) have problems with sending correctly encrypted and integrity-protected NAS messages.
- “5G/O-RAN Security Automated Testing” [9] provides an OpenAirInterface [3] framework. The framework is also not open source, or at least not yet,

or not for the general public (“This fuzzing tool being funded by the NTIA grant will be open sourced to the ORAN community” [9]). Again, from what is written in the paper, it seems that NAS integrity checks and encryption could become a problem.

- “CovFUZZ: Coverage-based fuzzer for 4G&5G protocols” [16] provides an srsRAN [5] based framework and does fuzzing with it. The framework is not able to test 5G uplink (UE to network), only 4G downlink and uplink, and 5G downlink. Since investigating 5G uplink is our goal, this work can only be of conceptual help. In addition to that, we could not find the implementation online, although it is stated that the framework is open source.
- “A 5G and Beyond Testbed for Cybersecurity Research and Education” [6] also contains a security testing UE (most likely based on OAI [3]) as part of the work. But in this case, the framework is only a small section of the research done in the paper, and there are no concrete implementation descriptions or code.
- “Soft Tester UE: A Novel Approach for Open RAN Security Testing” [13] provides an srsRAN [5] based framework. Here, additional tools for white box testing are provided, and some practical fuzzing is carried out. We are not sure if the given framework can provide all the functionality we described in subsection 2.2, especially altering NAS messages could again be a problem. This work is the only one where we were able to find the code online as open source [4]. If one is more familiar with srsRAN than with OAI this framework could be an alternative to our OAI-based framework.

The presented collection of related work represents and discusses all relevant papers we found regarding our topic. Therefore, the research gap we try to close consists of a fully open source, well-documented, holistic framework for UE-based 5G pentesting.

4 Design of the framework

This section briefly describes the design and implementation of our framework. Extensive documentation and detailed descriptions will be made available on GitHub [1].

4.1 Concept

Our framework is based on the UE software stack provided by OpenAirInterface (OAI). The basic idea of the framework is to add different interfaces to the OAI code, which allow influencing the behaviour of the UE. To match the requirements stated in subsection 2.2, we choose a three-layer architecture, as shown in Figure 1.

First layer The first layer of our framework is the OAI code with the added interfaces. Layer one, therefore, provides the 5G UE capabilities, which allow

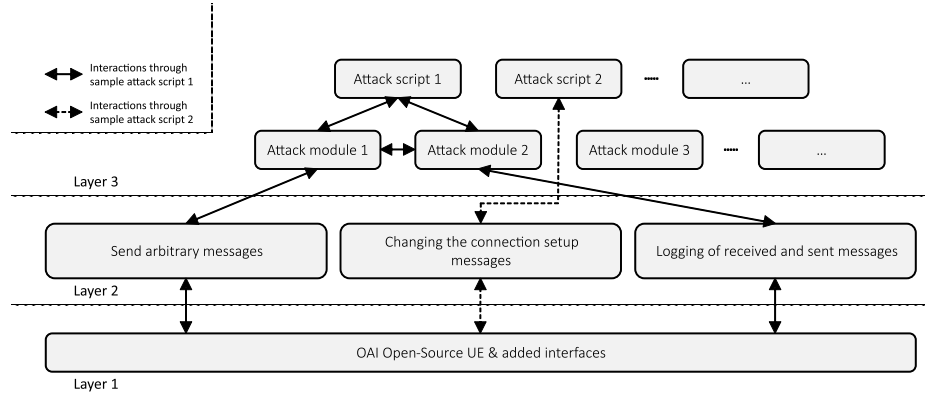


Fig. 1. Basic design of the 5G-Pentest-UE

our framework to interact with 5G networks and the control interfaces, which are to be used by the higher layers.

To enhance maintainability, it is important on this layer to only add code and not change or remove anything. This will support easier upgrades to newer upstream versions of the OAI UE code. For the code added, there also has to be the possibility of disabling its execution to let our UE behave as a vanilla UE. Therefore, the execution of the added code is controlled by a configuration file. This means that if all framework functionalities are deactivated by this configuration file, the OAI UE will behave the same way as a normal, plain OAI UE would.

Second layer On the second layer, we now need to utilize the interfaces created on the first layer. To understand the three capabilities implemented on this layer, we need to recap what we want to accomplish.

Every 5G communication starts with establishing a connection. Here, the UE and the network negotiate different parameters and create the basis for practical use of the connection by exchanging control messages. This predefined message flow is time-critical, and being able to specify the messages sent on the fly is not helpful for us at the moment. Therefore, we implement a capability that allows us to predefine the connection establishment messages before starting the connection. This is done by creating the necessary fields in the configuration file. The control flow of the framework then works in a way that when the UE is executed, it checks if an alternative message is specified for a given connection establishment message and, if so, replaces the message that would usually be sent. By this, we are able to alter the connection establishment messages and test the involved message processing logic on the base station and the core network side.

In addition to being able to alter the connection establishment, we also want to be able to efficiently test all of the interfaces that are available on an existing connection. That means we want to be able to use the UE to send any data we

want to the network after the connection is established. This is done by providing a REST-API inside the UE as shown in Figure 2. Using the REST-API, one can

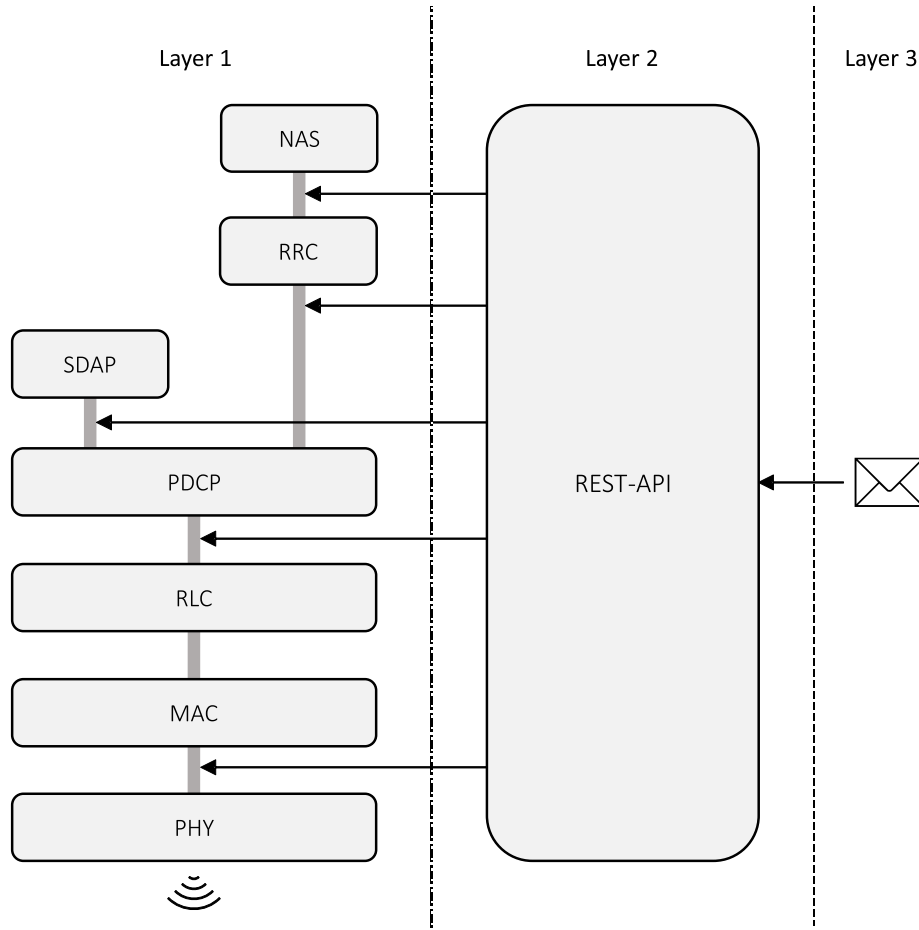


Fig. 2. Visualization of the integration of the REST-Socket

let the UE insert a specific message on a given layer.

Besides adding means to influence what is sent by the UE, the framework also provides the capability to log everything sent or received by the UE. This enables the framework user to understand better what is happening and to react to specific messages sent by the network. The details of the log output (e.g., which messages at which layers should be logged) can be specified in the configuration file.

Third layer Although the two layers discussed so far already provide enough functionality to execute meaningful vulnerability testing, we added another logical layer to our design to make the framework easier and more efficient. Layer three contains generic and reusable modules that use and wrap the functionalities given on layer two. These modules combine common steps that happen in many attack paths. Separating these steps into modules makes the creation of new vulnerability tests easier.

For example, the framework user can automate the testing or create different attack scripts and analysis tools on this layer. An example for “Attack module 1” in Figure 1 could be a module that takes inputs from the user, validates them, and sends them to the REST-API. Such a module could then be used by a proprietary fuzzing generator to eventually send the generated messages to the 5G network.

In summary, the attack modules on layer three implement common functionality shared among different attack scripts. The attack scripts themselves represent different vulnerability tests or attack ideas.

4.2 Implementation and Feature set

This section describes some basic implementation details and the currently implemented features. A much more complete description will be provided in the GitHub repository [1].

Command line parameter The first step of implementation was to add a command-line parameter to the OAI UE. This parameter allows us to specify a configuration file and, through that, activate the framework. If the parameter is not given when running the UE, it will behave like a vanilla OAI UE.

Configuration file Inside the configuration file, which is in JSON format, we have three sections that allow us to control the three different capabilities at layer two. In the first section, we configure the REST-API. There, we have the option to activate or deactivate it and set the port number we want to use for the REST-API. In the second section, the connection establishment messages can be configured. Here, for every message of the connection establishment procedure, alternative message content can be specified by providing the message in hexadecimal format. The last part of the configuration file configures the logging capability. Here, we can independently activate/deactivate the output of sent and received messages at the MAC, RRC, NAS, and Data layer of the 5G protocol stack.

Code structure The configuration file and all other implementations except the interfaces added to the OAI code are located in a separate folder called “security_testing” inside the base directory of the OAI UE. The ability to exclude all of those additions is gained by including them inside IF-Statements, which only allows the execution if the command line parameter is specified and the corresponding configuration is active. For more information on why, where, and

how the individual additions are included in the OAI code, please look at the documentation in the GitHub repository [1].

Feature set Regarding the current feature set, the framework provides the following capabilities:

- REST-API — the REST-API provides the ability to send any message content on the following layers:
 - NAS (Non Access-Stratum)
 - * security protected as well as unprotected messages
 - RRC (Radio Resource Control)
 - * SRB0 (Signaling Radio Bearer 0)
 - * SRB1 (Signaling Radio Bearer 1)
 - * SRB2 (Signaling Radio Bearer 2)
 - SDAP (Service Data Adaptation Protocol)
 - PDCP (Packet Data Convergence Protocol)
 - * SRB1 (Signaling Radio Bearer 1)
 - * SRB2 (Signaling Radio Bearer 2)
 - MAC (Medium Access Control)
- Connection establishment — for the connection establishment, the following messages can be adjusted
 - RRC Setup Request
 - RRC Setup Complete
 - NAS Registration Request
 - NAS Authentication Response
 - NAS Security Mode Complete
 - RRC Security Mode Complete
 - RRC Reconfiguration Complete
 - NAS Registration Complete
 - NAS Session Establishment Request
 - NAS Deregistration Request
- Message logging — the framework allows logging the received (RX) and transmitted (TX) messages on the following layers:
 - MAC
 - RRC
 - NAS
 - User data

5 Evaluation

In this section, we want to investigate if the proposed framework matches the requirements from subsection 2.2. We also want to discuss some practical tests we did with the framework and the results we gathered.

Requirements In subsection 2.2, we set 3 main goals for the framework.

The **first goal** was to be able to adhere to the 5G specifications wherever needed. This goal is reached by allowing our framework to dynamically activate and deactivate all parts of added code and functionality.

The **second goal** was to be able to deviate wherever we want. Therefore, different interfaces were added to the OAI UE code. The possibilities given by those additions were sufficient for the investigations we did. However, some special attacks may require additional functionalities. The clear structure of the framework and the extensive documentation given in the GitHub repository will enable easy implementation of such possibly missing features.

The **third goal** was to enable the framework user to test different interfaces easily. We tried to reach this goal using a logical approach to the problem. With the connection establishment being time-critical, we allowed preconfiguring the involved messages inside the configuration file. For the fully connected state, after the connection establishment, we offer different interfaces to be able to create different attacks on different layers easily. Also, a logging feature was added to the framework to be able to output and, after that, process the received messages. This will enable the user to create adaptive attacks and, therefore, find complicated vulnerabilities in the network.

We want to evaluate the soft goals set in subsection 2.2 as follows. The first goal was to open source the whole framework and its documentation to be used as a basis for research and security testing of 5G networks. This is done via a GitHub repository [1]. With this repository, the second soft goal is also tackled. The repository contains the code we wrote and extensive documentation on the framework. In addition, there are examples of the usage of the framework, a tutorial on setting up a local testbed, and some documentation of the OAI UE. Also, a basic introduction to 5G and the protocol stack is available in the repository. The third and fourth soft goals were to make updates to the framework possible and allow easy extendability. We accomplish this by only making minimal additions to the OAI code and not changing anything. This has already proven to make it relatively easy to transfer the framework to a newer version of the OAI UE code, which is regularly updated, to add new features and stay up to date with the 5G specifications. Also, this, combined with the given documentation, should be very helpful for extending the framework's basic functionality. Besides that, extending also means implementing new test cases. Most of those test cases can already be realized with the available layer one and two functionality. Therefore, extending the framework may only mean adding some utility scripts to the third layer to be used by your test case.

Practical evaluation This paper is mainly about describing and publishing the framework we build. Nevertheless, we want to discuss some tests we did and what we learned. With the help of the framework, we discovered many different vulnerabilities in an existing open source implementation of the base station and the core network. Most of those vulnerabilities were crashes of the 5G stack (base station and core part) as well as other problems (e.g., dead locks) leading to availability issues. For example, we found an infinite loop in the core network

session management function (SMF). Here, a switch statement inside a loop was used to decrypt the message identifiers. Providing an identifier unknown to the switch statement caused the loop to run forever. This led to a situation where no UEs could register to the core network until the SMF was restarted.

Besides that, we also discovered an attack that allowed us to skip the authentication of the UE towards the core network. This vulnerability even allows to impersonate other UEs. This attack could be executed by rearranging and adjusting the connection establishment messages.

The issues found were reported to the developers of the 5G stack and are meanwhile mostly fixed.

In summary, we were able to make good use of our framework and improve some 5G implementations. In the future, we plan to extensively study different mobile network implementations to get an overall view of their security when assuming the malicious UE attacker model.

6 Conclusion

In this work, we present a framework for security testing of 5G networks. In the preceding sections, we described our requirements and goals, investigated related work and proposed and evaluated the framework. Our work closes the gap in open source, well-documented, and easy-to-use security testing software. This will allow future research to find and fix vulnerabilities in 5G protocol implementations faster.

Outlook In the future, we want to use the framework extensively to test existing 5G implementations. Therefore, we want to combine the framework with existing fuzzing solutions, e.g., libfuzz [2]. In addition to that, we plan on creating specialized attacks and testing our assumptions in different networks. By that, we hope to gain a comprehensive overview of mobile network security investigated from the UE perspective.

Ethical considerations With regard to misuse of the framework for criminal activities, the large amount of existing work (partly also open source) on attacking mobile networks shows that attacks are possible in many different ways. For example, not much knowledge is necessary to implement an easy Fuzzer with the help of OAI or srsRAN UE. The difficulty is in implementing concrete attacks or finding the correct values to fuzz test. Our work does not provide help for those parts of the attack. Therefore, we believe that providing a framework that makes security testing easier will improve security in mobile networks rather than endanger it.

Acknowledgments. This work has been supported by the German Federal Office for Information Security (BSI) project 6G-ReS (grant no. 01MO23013D), the Federal Ministry of Research, Technology and Space of Germany through the 6G-CampuSens project (grant no. 16KISK205) and by the Federal Ministry of Transport, Germany through the InnoDCon project (grant no. 19OI23013C). Additionally, the authors are also financed based on the budget passed by the Saxonian State Parliament in Germany.

References

1. 5G-Pentest-UE GitHub Repository. <https://github.com/Barkhausen-Institut/UE-based-5G-Pentesting-Framework>, accessed: 12-06-2025
2. libFuzzer. <https://l1vm.org/docs/LibFuzzer.html>, accessed: 30-04-2025
3. Open Air Interface GitLab. <https://gitlab.eurecom.fr/oai/openairinterface5g>, accessed: 17-03-2025
4. Soft Tester UE GitHub Repository. <https://github.com/oran-testing/oran-tester-ue>, accessed: 10-04-2025
5. srsRAN Project. <https://www.srslte.com/>, accessed: 09-04-2025
6. Almazyad, I., Elmadani, S., Hariri, S.: A 5G and Beyond Testbed for Cybersecurity Research and Education. In: 2024 IEEE/ACS 21st International Conference on Computer Systems and Applications (AICCSA). pp. 1–6 (2024). <https://doi.org/10.1109/AICCSA63423.2024.10912627>
7. Baccar, K., Lahmadi, A.: An Experimental Testbed for 5G Network Security Assessment. In: NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium. pp. 1–6 (2023). <https://doi.org/10.1109/NOMS56928.2023.10154283>
8. Chianese, L., Granata, D., Palmiero, P., Rak, M.: Leveraging Threat Modelling for Effective Penetration Testing in 5G Systems. In: 2024 IEEE International Conference on Cyber Security and Resilience (CSR). pp. 180–185 (2024). <https://doi.org/10.1109/CSR61664.2024.10679437>
9. Dessources, D., Appiah-Mensah, S., Amato, C., Parikh, D., Bull, J.D., Burger, E.W.: 5G/O-RAN Security Automated Testing. In: MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM). pp. 129–134 (2024). <https://doi.org/10.1109/MILCOM61039.2024.10774015>
10. Garbelini, M.E., Shang, Z., Chattopadhyay, S., Sun, S., Kurniawan, E.: Towards Automated Fuzzing of 4G/5G Protocol Implementations Over the Air. In: GLOBECOM 2022 - 2022 IEEE Global Communications Conference. pp. 86–92 (2022). <https://doi.org/10.1109/GLOBECOM48099.2022.10001673>
11. He, F., Yang, W., Cui, B., Cui, J.: Intelligent Fuzzing Algorithm for 5G NAS Protocol Based on Predefined Rules. In: 2022 International Conference on Computer Communications and Networks (ICCCN). pp. 1–7 (2022). <https://doi.org/10.1109/ICCCN54977.2022.9868872>
12. Mancini, F., Da Canal, S., Bianchi, G.: AMFuzz: Black-Box Fuzzing of 5G Core Networks. In: 2024 19th Wireless On-Demand Network Systems and Services Conference (WONS). pp. 17–24 (2024). <https://doi.org/10.23919/WONS60642.2024.10449510>
13. Moore, J., Abdalla, A.S., Ueltschey, C., Marojevic, V.: Soft Tester UE: A Novel Approach for Open RAN Security Testing. In: 2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall). pp. 1–5 (2024). <https://doi.org/10.1109/VTC2024-Fall163153.2024.10757739>
14. Potnuru, S., Nakarmi, P.K.: Berserker: ASN.1-based Fuzzing of Radio Resource Control Protocol for 4G and 5G. In: 2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). pp. 295–300 (2021). <https://doi.org/10.1109/WiMob52687.2021.9606317>
15. Salazar, Z., Nguyen, H.N., Mallouli, W., Cavalli, A.R., Montes de Oca, E.: 5GReplay: a 5G Network Traffic Fuzzer - Application to Attack Injection. In: Proceedings of the 16th International Conference on Availability, Reliability and Security. ARES '21, Association for Computing Machinery, New York, NY,

- USA (2021). <https://doi.org/10.1145/3465481.3470079>, <https://doi.org/10.1145/3465481.3470079>
16. Siroš, I., Singelée, D., Preneel, B.: CovFUZZ: Coverage-based fuzzer for 4G & 5G protocols (2024), <https://arxiv.org/abs/2410.20958>
 17. Wang, H., Cui, B., Yang, W., Cui, J., Su, L., Sun, L.: An Automated Vulnerability Detection Method for the 5G RRC Protocol Based on Fuzzing. In: 2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC). pp. 1–7 (2022). <https://doi.org/10.1109/CTISC54888.2022.9849690>
 18. Yang, J., Wang, Y., Tran, T.X., Pan, Y.: 5G RRC Protocol and Stack Vulnerabilities Detection via Listen-and-Learn. In: 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC). pp. 236–241 (2023). <https://doi.org/10.1109/CCNC51644.2023.10059624>