Monitoring for Open5GS in Secure and Low-Footprint 5G Deployments

Janeba Matej, Oleksii Biletskyi, Galinski Marek, Kotuliak Ivan
Slovak University of Technology in Bratislava
Ilkovicova 2, Bratislava, Slovak Republic
matej.janeba@stuba.sk

***Abstract*—This work presents low-latency-focused monitoring for secure 5G control-plane mobile networks. We emphasize encrypted internal communication and minimal performance overhead along with effective OS-specific deployment, including potential ARM support. The solution is built over Open5GS Packet Core implementation project to match the best performance and smooth deployment over different hardware. The proposed system extracts data from the protocol level by modifying the Open5GS source code. The system uses asynchronous reporting using the UDP.The core processing unit for the incoming logs is solved by the ELK stack (ElasticSearch, Logstash, and Kibana) as a robust solution with wide options of performance configuration depending on the system's needs. The alert mechanisms are managed by logstash custom filters and the ElastAlert2 project. A custom light-weight alert dispatcher server is introduced to gather and forward the alerts to the end-user. For demonstration purposes, the Discord Webhook API was chosen. Although not covering the entire spectrum of complex alert scenarios, these tools provide us with enough capabilities to demonstrate the fundamental approach of the project.**

***Keywords*—5G; Open5GS; Monitoring; Private Cellular Network; Packet Core**

I. Introduction

Fifth-generation (5G) mobile networks have already changed the way we perceive wireless communication both in civil commercial and critical private sectors. More scalable control-plane design enables developers to implement complex policies of effective management, effectively lowering the latency and extending throughput for critical data. However, complexity comes at the cost of extended attack surfaces. Although substantial effort was made to mitigate the exposed potential vulnerable APIs, critical infrastructure requires enhanced security features such as alert-driven monitoring.

Regular deployments tend to prioritize pure performance over enhanced security. However, this project targets another niche of 5G networks, namely private deployments for campuses, governments, or possible military applications. The main requirement for secured internal communication is the configuration of TLS encrypted communication and IPSec protected interfaces for certain scenarios where the encryption of the application layer is not covered by the 3GPP standard (N1, N2, N3, N4) [1]. These measures prevent visibility of the data for monitoring. The encrypted traffic can only be accessed at the application layer upon reaching the Packet Core.

The widely used Open5GS project is chosen as the best-fit implementation for the project. The Open5GS includes basic monitoring features that are designed for debugging purposes and produce large volumes of unstructured text directly to the filesystem. The security scenario expects protocol-level information that is exclusively accessible at the initial processing. The system requires the deepest logging level to access the required data at the expected initial point, therefore, produces extensive output significantly affecting computational performance and physical storage.

To address this gap, this project introduces hooks injected into the Open5GS source code to collect the required data at the earliest stage before any further processing. This approach effectively gathers data at the protocol level to be evaluated independently of the following Open5GS flow. The data is further consumed by the ELK stack. This battle-tested solution is a balanced choice between a scalable and a performance-configurable processing pipeline. Debian packages and OS-specific management is considered as the deployment approach to enhance performance and avoid virtualization complexity. Further discussion is presented in Section III to compare and justify the selected approach. Comparison to existing solutions is provided below in Section II.

II. Related Work

Monitoring of mobile networks and 5G systems is a widely adopted topic in the academic field. Multiple research projects are focused on performance benchmarks and service integration across different Packet Core implementations, especially on the most widely adopted Open5GS. On contrary, fewer works address the security aspect of the monitoring and its own performance overhead in low-latency deployments.

MonArch [2] system presents a modular monitoring approach that focuses on KPI collection and telemetry with slice-based data gathering. This system also introduces native cloud deployment techniques for hardware independence, but omits security-specific measures in exchange for modularity and robustness.

The alternative approach of Bhattacharjee et al. [3] emphasizes lightweight and open-source tools such as Prometheus and Netdata to gather telemetry data on the performance of the system over time. Despite near-real-time performance, it does not mitigate potential security constraints and is not designed to gather low-level data for event-driven analysis.

A slightly different approach is presented by the 5G-EVE framework [4] which addresses geographically distributed multi-site monitoring powered by the Kafka pub-sub mechanism. The Apache Kafka system enables the authors to gather near-real-time metrics across physically distant services. This

979-8-3315-9679-8/25/$31.00 ©2025 IEEE

system provides insight into distributed system which is expected by certain critical use cases, but the monitoring is focused on metrics instead of events and assumes plain-text visibility without consideration of TLS-encrypted traffic.

Several studies were selected to explore the attack surfaces of 5G networks. The Chinese study, conducted by You et al. [5], provides an in-depth overview of common attack vectors, including API vulnerabilities in control plane and user plane. The work motivates us to use TLS encryption with a combination of IPSec on defined interfaces [1], however, it does not propose solutions to overcome visibility challenges for monitoring. Another research focuses on Open5GS and other popular 5G Core implementations revealing the actual attack surfaces for the selected Packet Core [6]. Another military-targeted article provides surface-level insights into the scope of typical challenges faced in military-grade communication systems [7], providing valuable information on concerns related to critical infrastructure. For government usage, South Korea's example is described in the paper [8] published by Seoul's government that describes a successful example of mobile network deployment to replace VPNs.

In contrast to these approaches, the system introduced in the current study focuses on zero-visibility data and minimized performance overhead of security-first event-driven monitoring. The project emphasizes cost-effective OS-specific deployment over dockerized virtualization and suggests event-specific data pipeline tools, namely the ELK stack.

III. System Design

This section presents a description of the internal architecture and technical justifications for the selected approaches. We target a monitoring system tailored for a secure, low-latency 5G Core network based on Open5GS. The main unique constraints of this solution are internal encryption and full OS-level deployment to improve performance efficiency. The system contains three main layers:

- **Open5GS**: modified Packet Core logic with a custom UDP-driven report sender
- **ELK stack**: ElasticSearch + Logstash + Kibana with reduced resources and custom logstash filter
- **Alert system**: Includes ElastAlert2 open-source project as an aternative to Elastic Watchers and custom lightweight dispatcher server to gather alerts and forward to Discord webhook.

Open5GS: This part consists of the main Open5GS project, the release of v2.6.2, and the custom dynamically linked monitoring library, coupled with the main project by the Meson build system. This library receives internal reports from certain predefined events, transforms it to a human readable version, serializes to JSON and sends it to Logstash via UDP. Specific hooks were inserted on the lowest protocol level at the moment the data is only decoded from ASN.1 message, before any logical processing. This approach allows for seamless updates considering up-to-date software as a crucial part of security-focused deployment. In addition, it allows one to catch metadata that is dropped off in the future processing.

ELK stack: In order to set up a scalable and stable data processing pipeline, the ELK stack was chosen as an industry standard. Custom processing rules and filters were created for Logstash to perform basic checks and store data. Several examples of single-event alerts are defined in the Logstash processing configuration. ElasticSearch and Logstash use a significant portion of system resources (6GB+ RAM) by default and were reduced to testbed values of 512MB each. Kibana provides a browser-based user interface for a human to manually analyze logs with rich graphical tools.

Alert system: The alert mechanism includes ElastAlert2 to manage complex scenarios that require analysis and Logstash for single event primitive alerts on suspicious activity. These two systems send the alerts to a custom lightweight dispatcher which retransmits them to a predefined Discord Webhook API i.e. to the end-user.

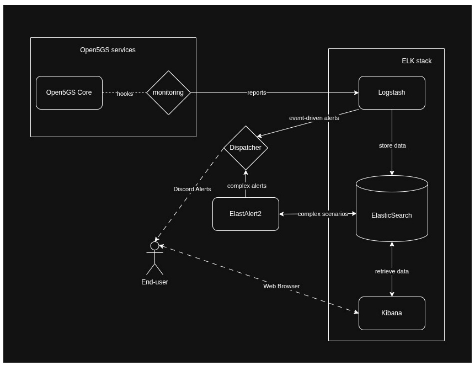


Figure 1. System architecture for Transport Layer Security resilient monitoring in Open5GS.

IV. Implementation

In this section, the exact testing scenarios will be described. Given the secure internal communication, the user-plane exposed interface N2 was chosen. The gNB initial message and the UE initial message are monitored to demonstrate the capabilities of the system. Dynamically linked monitoring library is located at `open5gs/lib/monitoring/`

gNB initial message: To capture the data about every gMN connection attempt, a custom hook was injected into the flow of the function `ngap_handle_ng_setup_request (...)` located at `open5gs/src/amf/ngap_handler.c`. This hook consumes fields from the initial gNB message defined in ASN.1 format [9] (Clause 8.7.1), namely: gNB name, PLMN ID, and gNB ID. This data is sent to the monitoring library via void function call. gNB ID and PLMN ID are encoded for internal usage, therefore the helper functions transform these data to human readable format. The data is then serialized to JSON and sent via UDP to Logstash.

UE initial message: UE initial message scenario contains two hooks in AMF and UDM network functions:

- **AMF**: UE initial message first comes to the `gmm_handle_registration_request(...)` function located at `open5gs/src/amf/gmm_handler.c`

[9]. The extracted data is: SUCI, UE PLMN ID, radio PLMN ID, TAC, Cell ID. The transformed fields are PLMN IDs, Cell ID split to gNB ID + Sector ID. The remaining flow is similar to the initial message from gNB.
- **UDM**: SUCI is a unique identifier per initial message and must be decrypted in order to identify the user by IMSI. The SUCI decryption process takes place in UDM [10] (Clause 6.12.4). The hook is placed in the `udm_ue_add(...)` function located in the `open5gs/src/udm/context.c` file. The reason for this placement is because different encryptions can be used, and the report should not concern itself with it at this stage. The state machine part still meets the project requirements.

Alert scenarios: There are several monitoring scenarios made to demonstrate the basic capabilities of the system:

- **Suspicious PLMN**: Logstash filter sends an alert on specific forbidden PLMNs (North Korean 467XXX in our example). Might be useful to identify strange devices in the area
- **Suspicious gNB name**: Testing solutions like PacketRusher [11]] or UERANSIM [12] use specific names to identify their devices. However, this field is ignored by the Open5GS project above the protocol decoding. Might be useful to identify nonproprietary devices trying to connect.
- **PLMN mismatch (No Roaming)**: The system expects UE's PLMN to match gNB PLMN, alerts otherwise. Filters out all visiting UEs.
- **SUCI not decoded**: With the help of ElastAlert2, a check is performed every 30 seconds to retrieve decrypted SUCI reports or alert of absence. Alerts a mismatch of encryption keys.

V. Demonstration and Evaluation

The result of the unencrypted IMSI scenario would be demonstrated in the following with the scenario of successful registration without alert.

Selected report scenarios

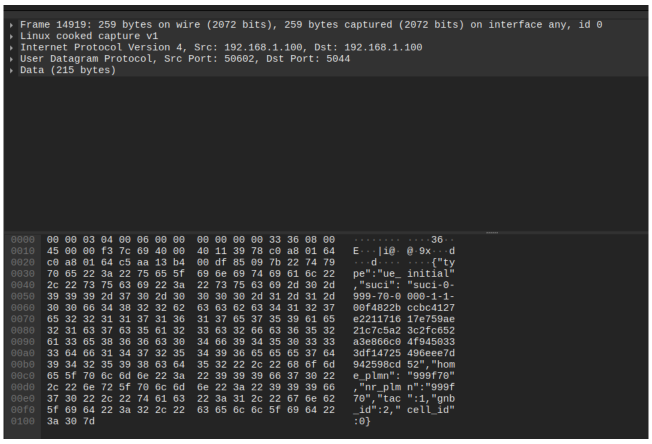
To test the system, the PacketRusher [11] project was used. This is an actively developing tool that does not fully support heavy-load testing; however, it works effectively for general testability. The installation might be tricky due to a custom kernel module usage; the testing system is Ubuntu 24.04.

To demonstrate the results of this work, two scenarios were selected: with failed SUCI decryption and expected report flow. This is not the full coverage of example scenarios and not the limits of these system capabilities.

- **SUCI not decrypted**: If run with the wrong `homeNetworkPublicKey` field, IMSI would never be decrypted, which means that Logstash never saved any decrypt report. An example of this report is shown in Fig. 2 without the corresponding SUCI decryption report. Causes ElastAlert2 to produce an alert that is sent to the Discord Webhook API depicted in Figure 3.
- **Successful registration**: Upon successful registration, no alert is sent in Figures 4 and 5.

Evaluation

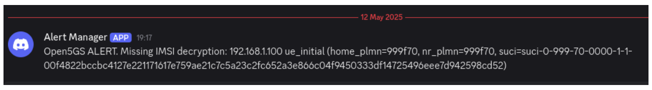
The system provides expandable capabilities for resource-efficient monitoring under security constraints. Demonstrated examples provide surface understanding of the project capabilities and principles in practical examples. The project does not claim to be ready for deployment as is, but rather proposes a unique approach of gathering and analyzing data in 5G Core Networks.



```
▸ Frame 14919: 259 bytes on wire (2072 bits), 259 bytes captured (2072 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.100
▸ User Datagram Protocol, Src Port: 50602, Dst Port: 5044
▸ Data (215 bytes)

0000  00 00 03 04 00 06 00 00  00 00 00 00 33 36 08 00   ········ ····36··
0010  45 00 00 f3 7c 69 40 00  40 11 39 78 c0 a8 01 64   E···|i@· @·9x···d
0020  c0 a8 01 64 c5 aa 13 b4  00 df 85 09 7b 22 74 79   ···d···· ····{"ty
0030  70 65 22 3a 22 75 65 5f  69 6e 69 74 69 61 6c 22   pe":"ue_ initial"
0040  2c 22 73 75 63 69 22 3a  22 73 75 63 69 2d 30 2d   ,"suci": "suci-0-
0050  39 39 39 2d 37 30 2d 30  30 30 30 2d 31 2d 31 2d   999-70-0 000-1-1-
0060  30 30 66 34 38 32 32 62  63 63 62 63 34 31 32 37   00f4822b ccbc4127
0070  65 32 32 31 31 37 31 36  31 37 65 37 35 39 61 65   e2211716 17e759ae
0080  32 31 63 37 63 35 61 32  33 63 32 66 63 36 35 32   21c7c5a2 3c2fc652
0090  61 33 65 38 36 36 63 30  34 66 39 34 35 30 33 33   a3e866c0 4f945033
00a0  33 64 66 31 34 37 32 35  34 39 36 65 65 65 37 64   3df14725 496eee7d
00b0  39 34 32 35 39 38 63 64  35 32 22 2c 22 68 6f 6d   942598cd 52","hom
00c0  65 5f 70 6c 6d 6e 22 3a  22 39 39 39 66 37 30 22   e_plmn": "999f70"
00d0  2c 22 6e 72 5f 70 6c 6d  6e 22 3a 22 39 39 39 66   ,"nr_plm n":"999f
00e0  37 30 22 2c 22 74 61 63  22 3a 31 2c 22 67 6e 62   70","tac ":1,"gnb
00f0  5f 69 64 22 3a 32 2c 22  63 65 6c 6c 5f 69 64 22   _id":2," cell_id"
0100  3a 30 7d                                           :0}
```

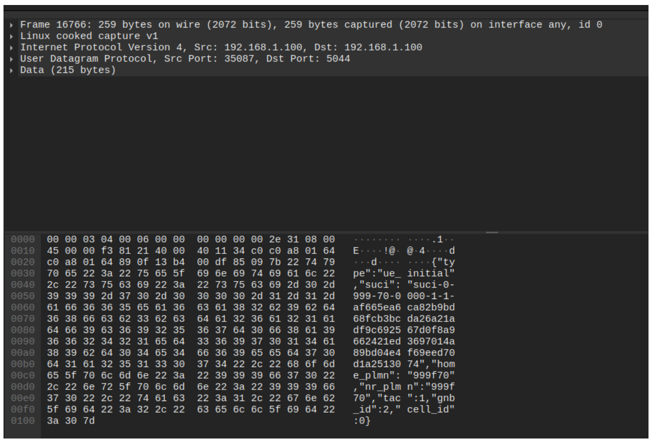
Figure 2. User equipment initial request with no subscriber concealed identifier decryption



12 May 2025

Alert Manager APP 19:17
Open5GS ALERT. Missing IMSI decryption: 192.168.1.100 ue_initial (home_plmn=999f70, nr_plmn=999f70, suci=suci-0-999-70-0000-1-1-00f4822bccbc4127e221171617e759ae21c7c5a23c2fc652a3e866c04f9450333df14725496eee7d942598cd52)

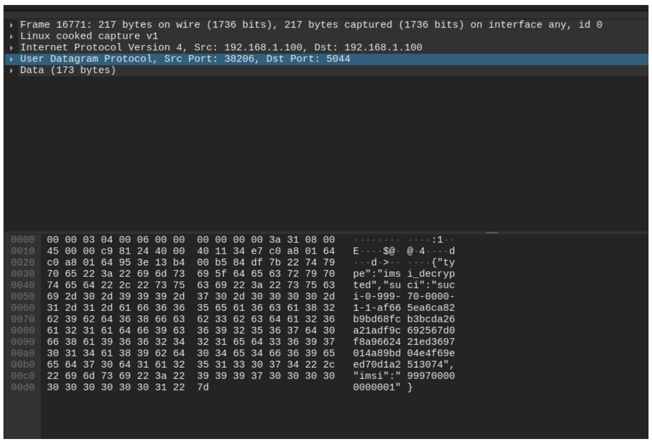
Figure 3. Alert sent to the Discord Webhook API about missing decryption report



```
▸ Frame 16766: 259 bytes on wire (2072 bits), 259 bytes captured (2072 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.100
▸ User Datagram Protocol, Src Port: 35087, Dst Port: 5044
▸ Data (215 bytes)

0000  00 00 03 04 00 06 00 00  00 00 00 00 2e 31 08 00   ········ ····.1··
0010  45 00 00 f3 81 21 40 00  40 11 34 c0 c0 a8 01 64   E····!@· @·4····d
0020  c0 a8 01 64 89 0f 13 b4  00 df 85 09 7b 22 74 79   ···d···· ····{"ty
0030  70 65 22 3a 22 75 65 5f  69 6e 69 74 69 61 6c 22   pe":"ue_ initial"
0040  2c 22 73 75 63 69 22 3a  22 73 75 63 69 2d 30 2d   ,"suci": "suci-0-
0050  39 39 39 2d 37 30 2d 30  30 30 30 2d 31 2d 31 2d   999-70-0 000-1-1-
0060  61 66 36 36 35 65 61 36  63 61 38 32 62 39 62 64   af665ea6 ca82b9bd
0070  36 38 66 63 62 33 62 63  64 61 32 36 61 32 31 61   68fcb3bc da26a21a
0080  64 66 39 63 36 39 32 35  36 37 64 30 66 38 61 39   df9c6925 67d0f8a9
0090  36 36 32 34 32 31 65 64  33 36 39 37 30 31 34 61   662421ed 3697014a
00a0  38 39 62 64 30 34 65 34  66 36 39 65 65 64 37 30   89bd04e4 f69eed70
00b0  64 31 61 32 35 31 33 30  37 34 22 2c 22 68 6f 6d   d1a25130 74","hom
00c0  65 5f 70 6c 6d 6e 22 3a  22 39 39 39 66 37 30 22   e_plmn": "999f70"
00d0  2c 22 6e 72 5f 70 6c 6d  6e 22 3a 22 39 39 39 66   ,"nr_plm n":"999f
00e0  37 30 22 2c 22 74 61 63  22 3a 31 2c 22 67 6e 62   70","tac ":1,"gnb
00f0  5f 69 64 22 3a 32 2c 22  63 65 6c 6c 5f 69 64 22   _id":2," cell_id"
0100  3a 30 7d                                           :0}
```

Figure 4. User euqipment initial request with successful subscriber concealed identifier decryption



```
▸ Frame 16771: 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.100
▸ User Datagram Protocol, Src Port: 38206, Dst Port: 5044
▸ Data (173 bytes)

0000  00 00 03 04 00 06 00 00  00 00 00 00 3a 31 08 00   ········ ····:1··
0010  45 00 00 c9 81 24 40 00  40 11 34 e7 c0 a8 01 64   E····$@· @·4····d
0020  c0 a8 01 64 95 3e 13 b4  00 b5 84 df 7b 22 74 79   ···d·>·· ····{"ty
0030  70 65 22 3a 22 69 6d 73  69 5f 64 65 63 72 79 70   pe":"ims i_decryp
0040  74 65 64 22 2c 22 73 75  63 69 22 3a 22 73 75 63   ted","su ci":"suc
0050  69 2d 30 2d 39 39 39 2d  37 30 2d 30 30 30 30 2d   i-0-999- 70-0000-
0060  31 2d 31 2d 61 66 36 36  35 65 61 36 63 61 38 32   1-1-af66 5ea6ca82
0070  62 39 62 64 36 38 66 63  62 33 62 63 64 61 32 36   b9bd68fc b3bcda26
0080  61 32 31 61 64 66 39 63  36 39 32 35 36 37 64 30   a21adf9c 692567d0
0090  66 38 61 39 36 36 32 34  32 31 65 64 33 36 39 37   f8a96624 21ed3697
00a0  30 31 34 61 38 39 62 64  30 34 65 34 66 36 39 65   014a89bd 04e4f69e
00b0  65 64 37 30 64 31 61 32  35 31 33 30 37 34 22 2c   ed70d1a2 513074",
00c0  22 69 6d 73 69 22 3a 22  39 39 39 37 30 30 30 30   "imsi":" 99970000
00d0  30 30 30 30 30 30 31 22  7d                        0000001" }
```

Figure 5. Subscriber concealed identifier decryption report

VI. Discussion

The author acknowledges several limitations of this work in advance, namely:

- **ARM and Debian**: The project is partially tested on ARM architecture, namely Open5GS SA was tested on Raspberry Pi 5 hardware, the ELK stack is claimed to be fully compatible with ARM architecture, as well as Python and Go standard library used for the scripts in this project. Debian building tool is selected over Docker for better performance, with the partial loss of OS-independent deployment.
- **Performance testing**: The project aims to achieve the best computational efficiency by utilizing Debian-based builds, ARM compatibility, and balanced resource management for the ELK stack; however, the exact deployment depends on the real-world usage which is not strictly defined within this work. Performance testing with a single simulator on a local machine would not show any valuable results. This testing would require more specific use cases and extended monitoring/alert scenarios with additional hardware equipment to measure the performance of this solution. This project is outside the scope of this project due to time and resource limitations.
- **ElastAlert2**: The ideal alerting mechanism is Elastic Watchers due to complex scenario support and efficient integration with ElasticSearch. However, this solution is only available on a paid subscription of the enterprise level. ElastAlert2 is used in this work as the best available open-source alternative; however, it lacks complex joins and grouping pushing the user to create inefficient queries manually per each scenario.

VII. Conclusion and Future Work

This project proposes a unique approach to event-driven security monitoring in encrypted 5G packet core based on the Open5GS project. It provides several examples and complete processing pipeline for the alert mechanisms based on protocol-level controlled data. The design emphasizes minimal performance overhead and TLS resilience with the Debian-based OS-optimized deployment with the consideration of ARM architecture compatibility.

Future work includes extending the monitoring scenarios based on precise use-cases, deployment of Elastic Watchers in case of real-world deployment if financially possible and performance testing of this solution using specific hardware and real-world scenarios.

Acknowledgment

This project has been supported by Connected and Cooperative Overtake Maneuver in smart mobility (CCOM) project within the internal university grant scheme for excelent teams of young STU researchers and EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I05-03-V02-00014 and by APVV-23-0519 project “Legal and Technical Challenges of Smart Mobility to Increase Road Traffic Safety”.

References

[1] ETSI, “TS 133 501 - 3GPP System Architecture Evolution (SAE); Security architecture and procedures for 5G System (Release 15),” 2018. Accessed: 2025-05-09.

[2] N. Saha, N. Shahriar, R. Boutaba, and A. Saleh, “Monarch: Network slice monitoring architecture for cloud native 5g deployments,” in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–7, 2023.

[3] D. Giannopoulos, P. Papaioannou, L. Ntzogani, C. Tranoris, and S. Denazis, “A holistic approach for 5g network slice monitoring,” in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pp. 240–245, 2021.

[4] R. Perez, J. Garcia-Reinoso, A. Zabala, P. Serrano, and A. Banchs, “A monitoring framework for multi-site 5g platforms,” in *2020 European Conference on Networks and Communications (EuCNC)*, pp. 52–56, 2020.

[5] W. You, M. Xu, and D. Zhou, “Research on security protection technology for 5g cloud network,” in *2021 International Conference on Advanced Computing and Endogenous Security*, pp. 01–11, 2022.

[6] F. Giambartolomei, M. Barceló, A. Brighente, A. Urbieta, and M. Conti, “Penetration testing of 5g core network web technologies,” in *ICC 2024 - IEEE International Conference on Communications*, pp. 702–707, 2024.

[7] J. M. Batalla, K. Wrona, D. Brown, F. Wiącek, U. Ruuto, and T. Wichary, “Threat assessment of 5g networks for military applications,” in *2024 International Conference on Military Communication and Information Systems (ICMCIS)*, pp. 01–10, 2024.

[8] N. Kim, T.-Y. Shin, H.-S. Lee, J.-Y. Jung, D. Kang, C.-H. Hong, D. Kim, and E. Kim, “A study on implementation issues of 5g-based government network services,” in *2023 IEEE Future Networks World Forum (FNWF)*, pp. 1–6, 2023.

[9] ETSI, “ 5G; NG-RAN; NG Application Protocol (NGAP) (3GPP TS 38.413 version 17.4.0 Release 17) ,” 2022. Accessed: 2025-05-09.

[10] ETSI, “ 5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 17.5.0 Release 17) ,” 2022. Accessed: 2025-05-09.

[11] HewlettPackard, “PacketRusher,” 2025. Accessed: 2025-05-09.

[12] aligungr, “UERANSIM,” 2025. Accessed: 2025-05-09.