# A Formal Approach for Complex Attacks Generation based on Mutation of 5G Network Traffic

Zujany Salazar, Fatiha Zaidi, Wissam Mallouli, Ana Rosa Cavalli, Huu Nghia Nguyen, Edgardo Montes de Oca

# A Formal Approach for Complex Attacks Generation based on Mutation of 5G Network Traffic

Zujany Salazar [1][a], Fatiha Zaidi[2][b], Wissam Mallouli[1][c], Ana Rosa Cavalli[1][d], Huu Nghia Nguyen[1][e] and Edgardo Montes de Oca[1][f]

[1]*Montimage EURL, Paris, France*

[2]*Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France.*

*{zujany.salazar, wissam.malloui,ana.cavalli,huunghia.nguyen, edgardo.montesdeoca}@montimage.com, fatiha.zaidi@lri.fr*

Keywords: Formal Approach, Mutation Technique, Network Traffic, 5G networks, Fuzzing Techniques

Abstract: We present a formal approach based on mutation techniques for the modelling of cybersecurity attacks and its application to 5G networks. We introduce formal definitions of the main concepts of network protocols, mutation operators, flow of network packets and network traffic. We design a formal approach based on different mutation operators that allows to design models that can be assimilated with known and unknown attacks. This approach has been implemented in our open source 5G network traffic fuzzer, 5Greplay, and has been applied to two use cases that are representative of attacks against 5G networks.

## 1 INTRODUCTION

Different techniques have been developed in order to detect and execute attacks in 5G networks. One of the techniques used is fuzz testing, which is a software testing technique that relies on the injection of random, invalid or unexpected data to cause the malfunctioning or a crash of the system.

The work described in this paper extends the research and open-source solution 5Greplay, presented by the authors in (Salazar et al., 2021). Notably, in this paper, we present a formal approach based on mutation techniques for the modelling of cybersecurity attacks and its application to 5G networks, that we demonstrate by means of our 5G network traffic fuzzer, 5Greplay.

The idea is to develop a formal approach based on different mutation operators that will allow to design models that can be assimilated with known attacks and with 0-day attacks. Attacks can be simple attacks consisting of simple actions, or more complex attacks based on the composition of different actions. This approach will facilitate the description and execution of attacks in order to check the robustness of 5G networks and their components. It will also facilitate the automated detection and design of mitigation techniques to resist attacks.

The main contribution of this paper is the design of a formal approach for mutation techniques by introducing a formal definition of the concepts of network protocol, mutation operators, network packet, flow of network packets, network traffic, as well as, the notion of independence of network protocols fields, packets, and flows.

The paper is organized as follows: Section 2 introduces related work; Section 3 presents the 5G network fuzzer; Section 4 presents the formalization of the network mutation functions, and Section 5 the experimentation to illustrate our approach. Finally, Section 6 provides the conclusion and perspectives.

## 2 RELATED WORK AND BACKGROUND

In this Section we present the research works that deal different topics of this paper, and the gaps we could identify in the literature.

[a] https://orcid.org/0000-0002-8655-0585
[b] https://orcid.org/0000-0003-3414-8815
[c] https://orcid.org/0000-0003-2548-6628
[d] https://orcid.org/0000-0003-2586-9071
[e] https://orcid.org/0000-0002-5778-0752
[f] https://orcid.org/0000-0001-6771-0689

## 2.1 5G core network

5G networks divide the LTE core and implement network functions separately, so that they can run independently from each other. Moreover, 5G network functions are virtualized, therefore they do not need dedicated hardware. 5G core Service-Based Architecture proposed by the 3GPP[1] is composed of the following major components (Brown, 2017):

1. The Access and Mobility Management Function (AMF) is as a single-entry point for the UE connection. Based on the service demanded by the UE, the AMF selects a Session Management Function (SMF) for handling the user session.
2. The User Plane Function (UPF) transports the IP data traffic (user plane) between the User Equipment (UE) and the external networks.
3. Further functions like the Session Management Function (SMF), the Policy Control Function (PCF), the Application Function (AF) and the Unified Data Management (UDM) function manage the policy control framework, applying policy decisions and accessing subscription information, to control the network behavior.

## 2.2 Network-enabled and 5G protocol fuzzers

Dedicated fuzzers for protocols commonly used in telecommunications have been proposed. T-Fuzz (Johansson et al., 2014) follows a generation-based fuzzing approach, relying on message models with full protocol specifications to test the 3GPP Non-Access Stratum (NAS) protocol, used in LTE and 5G networks. For the Resource Control layer (RCC) protocol that operates between the UE and gNB, a fuzzer based on the ASN.1 description language has been proposed (Potnuru and Nakarmi, 2021). The fuzzer extracts information about ongoing RRC messages by means of protocol description files of RRC from 3GPP, and uses it to mutate RRC messages. The adaptive fuzzer recognizes individual fields, submessages, and custom data types according to specifications when fuzzing the content of existing messages.

The Next Generation Application Protocol (NGAP) that operates between the UE and the AMF, has been also tested with fuzzing techniques. Mutation algorithms based on partition weight have shown to be able to reduce fuzzing time by generating samples that are more likely to produce anomalies in the 5G core (Hu et al., 2022). The main limitation of this approach is that the number of samples that are required to test and calculate the weights of each protocol field during the tuning phase of the algorithm, must be adequate to get an accurate calculation of field weights, and this process could take a considerable amount of time.

[1]https://www.enisa.europa.eu/publications/sdn-threat-landscape

## 2.3 Complex Attack Definition and Generation

In the context of this paper, we define the following complex attacks:

- Multi-layers attacks: these attacks that are composed of two events happening on at least two layers (for instance, transport layer and application layer). At the network level, this can impact at least 2 different protocols.
- Collaborative attacks: these attacks have different sources of packets and needs coordination between them to succeed the attack. A typical attack of this kind is a Distributed Denial of Service (DDoS) attacks.
- Multi-target attacks: these attacks targets at least two different nodes (or virtual functions) in the network following thus two different paths or interfaces.
- Distributed attacks: these attacks combines the two last types of attacks where we can have multi-source and/or multi-target flows.

## 2.4 Mutation-based Fuzzing Testing Formalization

Various formal bases to improve this critical fuzzing approaches has been proposed in the literature. On the software testing field, research (Salls et al., 2020) has characterized the input evaluation and selection components of fuzzing, based on static analysis concepts. In their model, they define the notions of input, concrete and abstract states, and how the inputs are mapped from a concrete state trace to an abstract state by an abstraction function. Moreover, authors have defined some software fuzzing technique using their model concepts in order to demonstrate its generality. Their results proved that the choice and combination of abstraction functions are significant and can modify the effectiveness of fuzzing.

Fuzzing testing has been also represented using mutation trees, in particular (Dong et al., 2018) applied it to an industrial control systems, in order to improve the efficiency of the mutation process and the

insufficient amount of tests in the old fuzzing technology for industrial control systems (ICS).

In conclusion, nowadays fuzzing testing is common practice for searching vulnerabilities. Nevertheless, to the best of our knowledge there is an absence of a unified formalism for describing network traffic mutation. Some works have approached this problem, but mainly in the software engineering domain. We believe that such a formalism would improve network security by allowing automation of test case creation, and facilitating the creation of complex test cases scenarios.

The formalism we described in Section 4 differs from the state of the art because it was specially designed to mutate network protocols and also allows the design of complex attacks. Network-enabled mutation has the peculiarity that there may be dependency between network packets or between packet flows. Therefore, when mutating a packet, our formalism also considers the packets that depend on it, in addition to other characteristics of its context that we will discuss later in this article. Finally, the operators that we defined in our formalism can be combined to create complex protocol-based attacks.

# 3 5G NETWORK FUZZER: 5GREPLAY

5Greplay[2] is an open-source solution entirely developed by the authors that generate *mutants* of the network traffic by using *mutant operators*, in order to perform specified security and functional tests on a system, as well as, fuzzing testing. In (Salazar et al., 2021) we presented 5Greplay mutant operators, and rule syntax. 5Greplay needs the following elements as an input to operate:

A **filtering set of rules**, that explicitly indicating which packets must be considered and ignored, to perform 5Greplay actions

A **configuration file**, that specify the default actions to be applied on the packets that are not managed by the rules, i.e., if they should be forwarded or not

**Network traffic**, online or offline, i.e. a pcap file

When defining a 5Greplay rule, users must indicate the following three elements in the rule: (i) which packet will be processed, (ii) which action will be applied, and in the case where the actions is a modification (iii) how to modify the packet.

[2]http://5greplay.org

Today, 5Greplay can operate over NAS-5G and NGAP protocols. However, the tool incorporates a plugin architecture for the addition of new protocols. In order to perform different mutations on the incoming 5G traffic, 5GReplay defined a set of mutation operator that can be applied either on the packet or on the flow levels. The list of these operators are provided in the Table 1.

# 4 NETWORK MUTATION FUNCTIONS FORMALIZATION

We propose a formalism to handle combinations of mutation operators applied to network traffic. The proposed approach relies on syntactic and behavioral changes.

Indeed, we consider that network protocols can be studied from several perspectives. Every protocol has

a **syntax** that defines sequencing of data elements or bits that are considered to be valid, and determines how to read the data in the form of fields

a **semantic** that refers to the interpretation or meaning that computers give to each field, and

a **behavior** that considers the data in its context, this is when the data should be sent, and how fast for non-functionnal behavioral aspects.

To illustrate our formalism, let us consider as a running example, a small part of Non-Access-Stratum (NAS) protocol for 5G System (5GS)[3]. NAS-5G protocol operates between user equipment (UE) and AMF, through 5G N1 interface. Its main function is to support of mobility of the UE, together with procedures such as authentication, identification, and generic UE configuration update and security control mode procedures.

The NAS-5G syntax is the group of rules that defines, for example, that during the UE registration procedure for mobility and periodic registration update a NAS REGISTRATION REQUEST message is composed by pieces of information of a determined length, that must be interpreted as different fields. The semantics of the protocol gives to the first field in the header the meaning of the *Extended protocol discriminator*, followed *Security header type*, etc. Finally, the functional behavior determines that for instance a NAS REGISTRATION ACCEPT message must arrive after a NAS REGISTRATION REQUEST message.

[3]https://itectec.com/archive/3gpp-specification-ts-24-501/

Table 1: 5Greplay mutantion operators

| Level | Atomic operator | Description |
|---|---|---|
| Packet, Flow | DROP | Delete a packet/flow of packets |
| | DUPLICATE | Duplicate packet/flow of packets |
| | PERMUTE | Exchange the order of two consecutive packets/flow of packets |
| | MODIFY | Change a specific attribute on the header of a network packet/flow of packets |

In a mutation-based fuzzing testing strategy, the idea is to generate new test cases, called also mutants, by making syntactic operations changes in already existing test cases, to therefore inject them into the system under test. Ideally, the mutants must be syntactically correct in order to discard test cases that the systems under test cannot interpret. Syntactic operators generate new network packets by making operations in a recorded packet. The syntactically incorrect and repeated packets must be discarded.

Moreover, mutants can be also generated by modifying the inputs from a behavioral' perspective. This is the packet in its context. Furthermore, these operators consider the time between two packets, and the order of the packets. Behavioral operators operate in a group of network packets, that later we will define as network packet flows to generate a new flow of packets with different time delay between them, repeated packets, or different order. Analogous to syntactic operators, syntactically incorrect and repeated sequences of packets must be discarded.

Considering the NAS-5G protocol, supposing that the system under test is an AMF, and a test case is a group of NAS-5G packets, that semantically constitute the UE authentication procedure. A syntactical mutant would be a NAS-5G packet with the field *Extended protocol discriminator* modified. A behavioral mutant of the test case can be the same message exchange but with a NAS REGISTRATION ACCEPT message before the UE sends the NAS REGISTRATION REQUEST message. The AMF authentication response would be syntactically and semantically correct, but it would violate the behavior of the protocol.

In our methodology, mutant operators can only make changes of the syntax, and behavior of an object. Semantic changes are not possible as the ultimate objective of this formalization is to represent a testing process, and modifying the interpretation or meaning of a message would imply to modify the system under test.

## 4.1 Basic Definitions

In the following subsection we present some definitions that will be used throughout our mutation formalism:

Let $Pr$ denote a **network protocol** formed by $i$ number of **fields**, which are ordered pairs of **field names** $FD = \{fd1, fd2, ..., fdi\}$ with input domain $Dfd = \{Dfd1, Dfd2, ..., Dfdi\}$, and **field values** $V = \{value1, value2, ..., valuei\}$, such that $Pr = \{(fd1, value1), (fd2, value2), ..., (fdi, valuei)\}$, where $valuei \in Dfdi$

Let $P$ denote a **network packet** formed by $n$ number of network protocols with input domain $Dp$, such that $P = \{pr1, pr2, ..., prn\}$, where $prn \in Dp$

Let $F$ denote a **flow of network packets** formed by $x$ number of network packets with input domain $Df$, such that $F = \{P1, P2, ..., Px\}$, where $Px \in Df$

Let $NT$ denote a **network traffic** formed by $y$ number of flow of network packets with input domain $Dnt$, such that $NT = \{F1, F2, ..., Fy\}$, where $Fy \in Dnt$

## 4.2 Notions of Independence

In the following subsection we define the independence notions applied to the 4 basic concepts we defined in the Section above.

Two **fields** are independents, if a change on the field value of one does not affect the field value of the other. This notion applies for fields in the same and in different protocols in the same packet, or for fields in different packets. We represent this as: $fdi \perp\!\!\!\perp fdj$

Two **protocols** are independent if all their fields are independent. $pri \perp\!\!\!\perp prj$

Two **packets** are independent if all their protocols are independent. $Pi \perp\!\!\!\perp Pj$

Two **flow of packets** are independent if all their packets are independent. $Fi \perp\!\!\!\perp Fj$

## 4.3 Mutants

Let $P', F', NT'$ be a syntactically correct network packet, flow of network packets, and network traffic

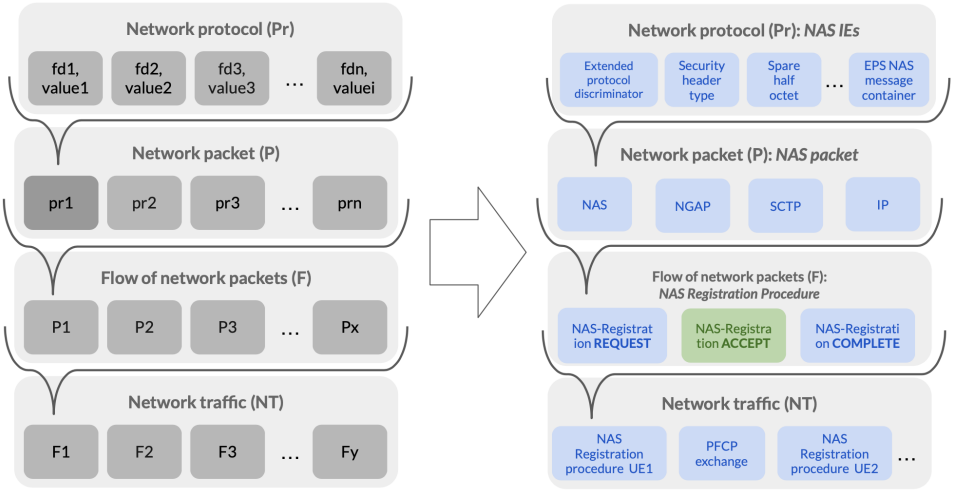


Figure 1: Summary of the basic definitions of our formalism (left), and instantiation using 5G protocols (right). Messages sent by the UE are in blue, and messages sent by the AMF are in green

respectively, obtained by making a syntactic, or behavioral changes of $P, F, NT$. $P', F', NT'$ are known as **mutants** of $P, F, NT$.

## 4.4 Mutant Operators

Let $R$ be a rule according to which $P, F$, or $NT$ are changed. $R$ is known as a **mutant operator**, and it is composed by a context $\gamma$ and an action $\sigma$ such that $R = \{\gamma, \sigma\}$.

The **context** determine if the operation will be performed in a single packet, a flow of packets, or the whole network traffic. It is a **filter** function $\gamma$ of $NT$ with image egal to

1. A single network packet: $\gamma : NT \rightarrow P$ or
2. A flow of network packets: $\gamma : NT \rightarrow F$ or
3. The whole network traffic: $\gamma : NT \rightarrow NT$

For describing the context, we use the following notation:

$$\gamma(NT) = [Level(P \text{ or } F)].[protocol].[field].[value]$$

Where **level**, indicates if the results of the filtering will be individual packets (P), or a flow of network packets (F). **Protocol**, **field**, and **value** refer to the name of network protocol, its respective field, and the field value according to which the filtering will be done. For example:

1. $\gamma(NT) = P.NAS\_5G.message\_type$ is a function that filters all the network packets of NAS_5G protocol, with and specific *message_type* value
2. $\gamma(NT) = F.IP.ip\_source$ is a function that filters all the network flows with a specific IP source address.

And, the **action** is a function $\sigma$ of $P$, $F$, *or* $NT$. We define the different actions applicable to $P$, $F$, *or* $NT$ in Sections 4.5, 4.6, and 4.7.

## 4.5 Mutation of a Single Packet

Let $R$ be a mutant operator, with context $\gamma$ applied to a network traffic $NT$, such that, the image of $\gamma(NT)$ is equal to a single network packet $Pi = \{pr1, pr2, ..., pri, ..., prh, ..., prn\}$, where $Fx = \{P1, P2, ..., Pi, Pj, ..., Px\} \subseteq NT$, there are the following list of mutation operators applicable to $Pi$:

1. **Drop:** Drop a packet.
   $\sigma = P\_DROP(Pi)$
   $\Rightarrow Fx = \{P1, P2, ..., Pj, ..., Px\}$
2. **Duplicate:** Duplicate a packet or a group of packets.
   $\sigma = P\_DUPLICATE(Pi)$
   $\Rightarrow Fx = \{P1, P2, ..., Pi, Pi, Pj, ..., Px\}$
3. **Permute:** Swap the position of Qx with Pi.
   $\sigma = P\_PERMUTE(Qx, Pi)$
   (a) **Permute two single packets:** Qx is a single packet $Px$.
   $\Rightarrow Fx = \{P1, P2, ..., Px, Pj, ..., Pi\}$
   (b) **Permute a group of packets:** Qx is a group of packets $\{Pi, Pj\}$.
   $\Rightarrow Fx = \{P1, P2, ..., Px, ..., Pi, Pj\}$
4. **Modify field value:** Change the value of field $fdi$, in the protocol $pri$, of the packet $Pi$, for the value $valuei'$. As a result we get a mutation of $Pi$, that we call $Pi'$.
   $\sigma = P\_MODIFY(Pi, pri, fdi, valuei')$

(a) **Independent fields:** If the field $fdi$ of the protocol $pri$ is independent from all the other fields on the same protocol, which is independent from all the other protocols in the packet $Pi$, then
$\Rightarrow Fx = \{P1, P2, ..., Pi', Pj, ..., Px\}$, where $Pi' = \{pr1, pr2, ..., pri', ..., pn\}$, with $pri' = \{(fd1, value1), (fd2, value2), ..., (fdi, valuei')\}$

(b) **Dependent fields in the single protocol:** If the field $fdi$ of the protocol $pri$ is dependent on a another field $fdh$ in the same protocol, which is independent from all the other protocols in the packet $Pi$, then
$\Rightarrow Fx = \{P1, P2, ..., Pi', Pj, ..., Px\}$, where $Pi' = \{pr1, pr2, ..., pri', ..., prn\}$, with $pri' = (fd1, value1), (fd2, value2), ..., (fdh, valueh'), (fdi, valuei')\}$

(c) **Dependent fields in several protocols:** If the field $fdi$ of a network protocol $pri$ is dependent on one field $fdh$ in another protocol $prh$ of the same packet $Pi$, which is independent from all the other packets in its flow $Fx$, then
$\Rightarrow Fx = \{P1, P2, ..., Pi', Pj, ..., Px\}$, where $Pi' = \{pr1, pr2, ..., prh', pri', ..., prn\}$

(d) **Dependent packets:** If the field $fdi$ of the network protocol $pri$ on the packet $Pi$ is dependent on one field $fdh$ in another protocol $prh$ of another packet $Px$ in the same flow $Fx$, which is independent from all the other flows in its network traffic $NT$, then
$\Rightarrow Fx = \{P1, P2, ..., Pi', Pj', ..., Px'\}$, where $Px' = \{pr1, pr2, ..., pri', ..., prn\}$ and $Pj' = \{pr1, pr2, ..., prh', ..., prn\}$

5. **Function composition:** Two operators f and g produces a new operator h such that $h(Px) = g \circ f(Px) = g(f(Px))$. In this operation, the operators g is applied to the result of applying the function f to $Px$. Operator composition is not necessarily commutative. Successive transformations applying and composing to the right agrees with the left-to-right reading sequence.

## 4.6 Mutation of a Flow of Packets

Let $R$ be a mutant operator, with context $\gamma$ applied to a network traffic $NT$, such that the image of $\gamma(NT)$ is equal to $Fi = \{P1, P2, ..., Pi, ..., Px\}$, where $Fi \subseteq NT = \{F1, F2, ..., Fi, Fj, ..., Fy\}$, there are the following list of mutation operators applicable to $Fi$:

1. **Drop:** Drop all the packets contained in a flow of packets.
$\sigma = F\_DROP(Fi)$
$\Rightarrow NT = \{F1, F2, ..., Fj, ..., Fy\}$

2. **Duplicate:** Duplicate all the packets in a flow of packets.
$\sigma = F\_DUPLICATE(Fi)$
$\Rightarrow NT = \{F1, F2, ..., Fi, Fi, Fj, ..., Fy\}$

3. **Permute:** Swap two flow of packets.
$\sigma = F\_PERMUTE(Fx, Fi)$
$\Rightarrow NT = \{F1, F2, ..., Fy, Fj, ..., Fi\}$

4. **Modify field value:** Change the value of field $fdi$, in the protocol $pri$, in all the the packets $Pi$ of the flow $Fx$, for the value $valuei'$. As a result we get a mutation of $Pi$, that we call $Pi'$.
$\sigma = F\_MODIFY(Fi, Pi, pri, fdi, valuei')$

   (a) **Independent flows:** $Fi$ is an independent flow of packets from all the other flows in network traffic $NT$.
   $\Rightarrow NT = \{F1, F2, ..., Fi', Fj, ..., Fy\}$, where $Fi' = \{P1, P2, ..., Pi', ..., Px\}$

   (b) **Dependent flows:** $Fi$ is dependent on the flow $Fj$ in the network traffic $NT$.
   $\Rightarrow NT = \{F1, F2, ..., Fi', Fj', ..., Fy\}$, where $Fi' = \{P1, P2, ..., Pi', ..., Px\}$ and $Fj' = \{P1, P2, ..., Ph', ..., Px\}$

Furthermore, all mutation operators for flow packets can be seen as a composition of operators of single packets. For example:
$F\_DROP(Fx) = P\_DROP(P1) \circ P\_DROP(P2) \circ ... \circ P\_DROP(Px) = \{\}$

## 4.7 Mutation of a Network Traffic

The formalization described in Section 4.6 can be generalized to be applied to the entire network traffic input.

## 4.8 Theorems

From our formalism definitions the following theorems can be deduced. These theorems allow optimization of the mutation process by reducing the possible number of mutants, and they are easily provable through mathematical manipulations.

1. PERMUTATION is **commutative** operation: The arguments of the PERMUTATION operation can be exchanged without altering the result
2. DUPLICATE and MODIFY are **commutative**: Both mutant operators can be exchanged without altering the result
3. Composition with DROP operator is always equal to an **empty flow**: Any operator composition including the DROP operator will be equal to an empty flow

### 4.9 Generating complex attacks

The proposed formalism envisage the possibility of combining mutation operators, in order to enable fuzzing testing and the generation of complex attacks, according to the definition proposed in Section 2.3. As a first approach, we proposed the combination of these operators to be performed randomly. Although inefficient in terms of the number of mutations that we could generate, this first approach has allowed us to find vulnerabilities in the 5G core (see Section 5). Furthermore, thanks to the theorems proposed in Section 4.8, the number of possible mutants can be reduced.

## 5 Experimental Evaluation

To illustrate our formalism, we have formalized the experiments we already performed in our previous work, where we presented our 5Greplay fuzzer (Salazar et al., 2021).

We performed these scenarios against two 5G core open-source solutions, free5GC and open5GS. In both cases we used the RAN simulator UERANSIM.

### 5.1 Threat 1: NAS Replay attack

Attackers with access to the NAS traffic (described in Section 4) in the 5G interface N1, could intercept a NAS SMC *Security Mode command* clear message sent from the AMF to the UE, copy its NAS sequence number (NAS SQN), and use it to build a NAS SMC *Security Mode complete* message that is replayed to the AMF, or directly intercept a NAS SMC *Security Mode complete* message and replay it to the AMF. If the AMF does not implement a proper integrity protection against this type of attack, the network will not drop the replayed packet.

#### 5.1.1 Formalization of Threat 1

To perform the NAS-5G SMC Replay attack, a malicious actor must perform the two following actions:

- Duplicate a NAS SMC packet with a specific *Security Mode Complete* field. (NAS_5G.message_type == 93 means that it is a SMC packet)
- Change the value of this *Security Mode Complete* field to a lower level and recompute the checksum of the packet. (for instance, NAS_5G.security_type == 4 means that there will be no encryption)

The **formalization** of this threat is a follow:

Let $NT = \{F1, F2, ..., Fi, ..., Fy\}$, where $Fi = \{P1, P2, ..., Pi, ..., Px\}$, where $Pi = \{pr1, pr2, ..., pri, ..., pn\}$, where $pri$ corresponds to the $NAS\_5G$ protocol and $pri = \{(fd1, value1), (fd2, value2), ..., \textbf{(message\_type,93)}, ..., (fdi, valuei)\}$. And $R = \{\gamma, \sigma\}$ a mutant operator, according to which a subset of $NT$ will be filtered, and mutated.

if $\gamma(NT) == P.NAS\_5G.message\_type.93 \rightarrow Pi$
$\Rightarrow \sigma = P\_MODIFY(Pi, NAS\_5G, security\_type, 4) \circ P\_DUPLICATE(Pi)$
$\Rightarrow NT = \{F1, F2, ..., Fi, ..., Fy\}$,
$Fi = \{P1, P2, ..., Pi', Pi', ..., Px\}$,
$Pi = \{pr1, pr2, ..., pri', ..., pn\}$, and
$pri' = NAS\_5G = \{(fd1, value1), (fd2, value2), ...,$ **(message_type,4)**$, ..., (fdi, valuei)\}$.

#### 5.1.2 Experimentation

We implement a mutant operator in 5Greplay with context: NAS SMC *Security Mode complete* messages sent by the UE after its authentication; and action: replay it twice to the AMF. Then, we checked the AMF logs and we monitored the network to verify that the AMF actually received the same packet twice.

After the NAS SMC *Security Mode Command* message, the AMF received a legitimate NAS SMC *Security Mode complete message*, and two NGAP packets with the same UE NGAP ID as the legitimate user. The AMF identified this as not belonging to the same NGAP security context. These two packets corresponded to the replayed packets by 5Greplay and allow us to conclude that the free5Gc AMF is protected again this type of replay attack.

### 5.2 Threat 2: Denial of Service by Sending Malformed NGAP Packets

This threat intends to check the robustness of the AMF function by sending inconsistent values of NGAP protocol sent over SCTP protocol. For instance, we can change the SCTP protocol identifier from 60 to 0 , and put the UE identifier of NGAP to an arbitrary value.

#### 5.2.1 Formalization of Threat 2

To perform this attack, a malicious actor must perform the two following actions:

- Change the value of the SCTP protocol identifier to 0. (This identifier should be SCTP.protoid == 60)

- Change the value of the UE identifier field to a random value. (for instance, NAS_5G.amf_UE_id == 1234 which is an random value)

This actions can be done for an authentication response identified by a message type $NAS\_5G.message\_type == 93$. The **formalization** of this threat is a follow:

Let $NT = \{F1, F2, ..., Fi, ..., Fy\}$, where $Fi = \{P1, P2, ..., Pi, ..., Px\}$, where $Pi = \{pr1, pr2, ..., pri, prj, ..., pn\}$, $pri$ corresponds to the $SCTP$ protocol, and $pri = \{(fd1, value1), (fd2, value2), ..., \textbf{(proto\_id,60)}, ..., (fdi, valuei)\}$; $prj$ corresponds to the $NAS\_5G$ protocol, and $prj = \{(fd1, value1), (fd2, value2), ..., \textbf{(amf\_UE\_id, 93)}, ..., (fdi, valuei)\}$. And $R = \{\gamma, \sigma\}$ a mutant operator, according to which a subset of $NT$ will be filtered, and mutated.

if $\gamma(NT) == P.NAS\_5G.message\_type.93 \rightarrow Pi$
$\Rightarrow \sigma = P\_MODIFY(Pi, NAS\_5G, amf\_UE\_id, 1234) \circ P\_MODIFY(Pi, SCTP, proto\_id, 0)$
$\Rightarrow NT = \{F1, F2, ..., Fi, ..., Fy\}$,
$Fi = \{P1, P2, ..., Pi', ..., Px\}$,
$Pi = \{pr1, pr2, ..., pri', prj', ..., pn\}$,
$pri = SCTP = \{(fd1, value1), (fd2, value2), ..., \textbf{(proto\_id,0)}, ..., (fdi, valuei)\}$, and
$prj = NAS\_5G = \{(fd1, value1), (fd2, value2), ..., \textbf{(amf\_UE\_id,1234)}, ..., (fdi, valuei)\}$.

### 5.2.2 Experimentation

We implement a mutant operator in 5Greplay with context: NGAP protocol messages sent by the UE during the authentication exchange; and action: replay them to the AMFs with two modification of the SCTP and NAS_5G fields. Then, we checked the AMF logs and we monitored the network to verify that the AMF actually received the same packet twice.

When replaying against free5GC, we got an AMF warning, but the simulator keep running and allowed new UE connections. On the other hand, open5GS was not able to handle this packet and the simulator crashed, preventing new connections to the AMF.

## 6 CONCLUSION AND FUTURE WORK

In this paper we have defined a formal approach for network mutation that provides a scientific basis for research work and application of these techniques. Based on this formalism, we have designed models of simple and complex attacks that we have applied to 5G networks. The proposed approach has been applied to two use cases that represent different attacks against a 5G network. In future work, we plan to introduce ML/AI techniques in order to improve the perform a smart fuzzing.

## ACKNOWLEDGEMENTS

This research is supported by the H2020 projects SANCUS N° 952672, INSPIRE-5Gplus N° 871808, and SPATIAL N° 101021808.

## REFERENCES

Brown, G. (2017). Service-based architecture for 5g core networks. *Huawei White Paper*, 1.

Dong, G., Sun, P., Shi, W., and Choi, C. (2018). A novel valuation pruning optimization fuzzing test model based on mutation tree for industrial control systems. *Applied Soft Computing*, 70:896–902.

Hu, Y., Yang, W., Cui, B., Zhou, X., Mao, Z., and Wang, Y. (2022). Fuzzing method based on selection mutation of partition weight table for 5g core network ngap protocol. In Barolli, L., Yim, K., and Chen, H.-C., editors, *Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 144–155, Cham. Springer International Publishing.

Johansson, W., Svensson, M., Larson, U. E., Almgren, M., and Gulisano, V. (2014). T-fuzz: Model-based fuzzing for robustness testing of telecommunication protocols. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*, pages 323–332.

Potnuru, S. and Nakarmi, P. K. (2021). 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)*, pages 295–300.

Salazar, Z., Nguyen, H. N., Mallouli, W., Cavalli, A. R., and Montes de Oca, E. (2021). 5greplay: A 5g network traffic fuzzer - application to attack injection. In *The 16th International Conference on Availability, Reliability and Security*, ARES 2021, New York, NY, USA. Association for Computing Machinery.

Salls, C., Machiry, A., Doupe, A., Shoshitaishvili, Y., Kruegel, C., and Vigna, G. (2020). Exploring abstraction functions in fuzzing. In *2020 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9.