Sherlock on Specs: Building LTE Conformance Tests through Automated Reasoning

Yi Chen[1], Di Tang[1], Yepeng Yao[2,4]*, Mingming Zha[1], XiaoFeng Wang[1]*
Xiaozhong Liu[3], Haixu Tang[1], Baoxu Liu[2,4]
[1]*Indiana University Bloomington*
[2]*{CAS-KLONAT†, BKLONSPT‡}, Institute of Information Engineering, CAS*
[3]*Worcester Polytechnic Institute,* [4]*School of Cyber Security, University of Chinese Academy of Sciences*
{chen481, tangd, mzha, xw7, hatang}@iu.edu, {yaoyepeng, liubaoxu}@iie.ac.cn, xliu14@wpi.edu

Abstract

Conformance tests are critical for finding security weaknesses in carrier network systems. However, building a conformance test procedure from specifications is challenging, as indicated by the slow progress made by the 3GPP, particularly in developing security-related tests, even with a large amount of resources already committed. A unique challenge in building the procedure is that a testing system often cannot directly invoke the condition event in a security requirement or directly observe the occurrence of the operation expected to be triggered by the event. Addressing this issue requires an event chain to be found, which once initiated leads to a chain reaction so the testing system can either indirectly triggers the target event or indirectly observe the occurrence of the expected event. To find a solution to this problem and make progress towards a fully automated conformance test generation, we developed a new approach called *Contester*, which utilizes natural language processing and machine learning to build an event dependency graph from a 3GPP specification, and further perform automated reasoning on the graph to discover the event chains for a given security requirement. Such event chains are further converted by Contester into a conformance test procedure, which is then executed by a testing system to evaluate the compliance of user equipment (UE) with the security requirement. Our evaluation shows that given 22 security requirements from the LTE NAS specification, Contester successfully generated over a hundred test procedures in just 25 minutes. After running these procedures on 22 popular UEs including iPhone 13, Pixel 5a and IoT devices, our approach uncovered 197 security requirement violations, with 190 never reported before, rendering these devices to serious security risks such as MITM, fake base station and reply attacks.

*Corresponding Authors
†Key Laboratory of Network Assessment Technology, CAS.
‡Beijing Key Laboratory of Network Security and Protection Technology

1 Introduction

With the rapid expansion of cellular network systems, their security weaknesses become an increasingly serious concern to the public. Among these weaknesses are the security vulnerabilities exposed by the user equipment (UE), such as mobile phones, which could be exploited to disclose a user's locations [22], deny her service [12, 14, 27], etc. It has been reported that many of these vulnerabilities have actually been caused by failure to follow the 3GPP specifications: for example, prior research shows that many UEs do not verify the presence of correct integrity protection on the DETACH REQUEST message, as required by the specifications, which could result in a disconnection of the UE from the service until it is rebooted [12]. These problems should have been captured by conformance testing meant to determine whether an implementation of a particular standard conforms to the requirements of the standard [38]. Although 3GPP does provide such conformance testing for UEs, the coverage of existing test cases is far from adequate [8, 9]. Particularly, only a very small portion of them are security-related (e.g., just 9 security-related conformance test cases out of 665 for LTE NAS protocols [7]), leaving most security requirements uncovered. The problem comes from the complexity of developing such test cases, which entails a significant amount of effort and time. Therefore, an efficient way to build conformance tests becomes imperative for elevating the security protection cellular network systems can offer.

Challenges in building test cases. Building a conformance test case starts with a given test purpose, for which one needs to design a test procedure and then develop test suites according to the procedure. This has been done completely manually until now, a process that turns out to be time-consuming. Particularly, it has been reported that the 3GPP community has devoted a large amount of resources to developing conformance test procedures, to meet the "strong demands" for the tests as claimed every year by 3GPP working group's annual reports recently [8, 9]: in Year 2020, 37 technical staff coordinated nearly 39 companies to build 326 procedures, and the

manpower has grown to 53 while the number of companies has increased to 52 a year later, with 815 procedures produced (less than 16 per company per year) [10]. Yet still only 37% of the development goals set in 2020 have been reached by the end of 2021 [3], not to mention new requirements being added to the wish list on a yearly basis.

A close look at the test-case development process reveals the unique challenge in building the conformance test procedures: although the test purpose serving as the input to the case development already includes a condition event[1] and a follow-up operation (the event is expected to trigger on a UE), the testing system running the test cases may not be able to directly issue the event and/or observe the occurrence of the target operation, since it only connects to the UE through the air interface for simulating the carrier network's communication with the UE. To address this problem, a chain of events first needs to be identified, through which one or more events the testing system can initiate will lead to the target event, and the sequence of the actions triggered by the expected operation will ultimately induce the effect observable to the system, so as to confirm that the operation indeed takes place.

For example, to evaluate a UE's conformance with the requirement ("*The UE shall initiate tracking area updating on the expiry of timer T3411*") as documented in the LTE NAS specification, we need to induce expiration of the timer T3411. This timer however cannot be directly operated by the testing system through the air interface. To get to the timer, the current 3GPP test procedure [7] requires the testing system to reboot the UE, causing an ATTACH REQUEST message to be issued, which further triggers a sequence of events (see Figure 1) leading to the expiration of the target timer. When developing such a procedure, a human analyst has to identify and weed through a large number of conditions and their combinations from the convoluted descriptions of 3GPP specifications so as to find a path through which the testing system can fire a right event chain. This can be highly complicated: for example, there are at least 25 different conditions for starting T3411, and 13 for its preceding event (start timer T3410), and most of their combinations are not on the event chain that can be initiated by the testing system. Actually, we found that at least 88% of the existing conformance test procedures require either a backward (for finding the event chain leading to the condition event) or a forward (for observing the event indicative of the occurrence of the target operation) inference. Some may also involve logic relations among the events (e.g., the target event can only happen when two preceding events occur together), making any manual attempt to develop the conformance test procedure even more complicated.

Test building through auto-reasoning. To address this unique challenge and facilitate generation of test cases, particularly those related to cellular network security, we designed

[1]Throughout the paper, we consider the terms "event", "action" and "operation" interchangeable since an event essentially describes the occurrence of its corresponding action/operation.

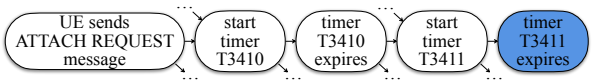

Figure 1: Example of the event dependency graph for the expiration of timer T3411 event.

and implemented an innovative analysis pipeline that automatically infers such chains of events from the LTE NAS specification. These event chains are further utilized to build security conformance test procedures, which are automatically executed in our test environment to evaluate UEs. The pipeline, named *Contester* (CONformance TEST genERator), first performs a semantic analysis on the LTE NAS specification [4] to construct an *event dependency graph* (*EDG*) using natural language processing (NLP) and machine learning (ML). For this purpose, we customized semantic role labeling (SRL) to identify the causal relations from the sentences in the specifications, and further train a new model on the specifications to connect different relation pairs together. Particularly, we addressed the challenge in determining semantically related technical concepts in the absence of adequate instances for training the model.

Given a security requirement whose condition event cannot be directly triggered or expected operation cannot be directly observed by the testing system, Contester conducts automatic backward reasoning on the EDG to find an event chain (or a network in the presence of logical relations among events) that can be set off by the system to trigger the target event, and/or forward reasoning to find the chain (or the network) activated by the target operation that will lead to the events observable to the system. The event chains discovered in this way are then converted into a security conformance test procedure. Our testing system, built on top of a hooked LTE simulator (simulating the cellular network), then executes the procedure on a UE to evaluate its conformance to the 3GPP security requirements.

Evaluation and findings. To understand the efficacy of Contester, we first evaluated its capability to generate conformance test procedures from 3GPP specifications on ground-truth data (20 requirements from the 3GPP conformance testing [7]), which achieved a precision of 80%. We further ran Constester on 22 cellular devices, including 20 mobile phones (e.g. iPhone 13, Pixel 5a, Honor Play20 and others) and 2 IoT devices (both USE Dongles) from 10 device vendors that use baseband processors covering the top 6 major baseband manufacturers, to test their conformance to 22 security requirements in the LTE NAS specification. Contester generated 143 test procedures (136 correct ones) for these security requirements in 25 minutes and spent 34 days to test them in our testing system, and further reported 287 conformance reports and 227 failure reports. Furthermore, from the failure reports, we discovered 197 security requirement violations, with 190 never reported before. More specifically, 19 security requirements in the 22 have been violated by at least 1 UE, and every UE being tested fails to meet at least 3 security

requirements and 8.95 on average. Particularly, even the most recently released mobile phone, iPhone 13, violates 7 security requirements. These violated security requirements are designed to protect the UE against the man-in-the-middle attack, the fake base station attack and the replay attack. Failures to properly implement them can lead to serious security consequences, such as the denial of service, eavesdropping, and location leakage. We reported all violations to the relevant vendors and have received confirmation and rewards from some of them, such as Samsung, Xiaomi and Google. We are continuing to assist vendors in confirming and fixing the problems discovered. The information about the vulnerability disclosure will be updated to our website [1].

Contributions. Following are the contributions of the paper:

• *New technique*. We developed *Contester*, a novel analysis pipeline that makes a step toward fully automated conformance test generation. Our pipeline includes an automated reasoning mechanism that applies existing and new NLP/ML techniques to semantic analysis on the LTE NAS specification, facilitating effective and efficient discovery of event chains critical for building conformance test procedures for security requirements.

• *New findings*. Running Contester on 22 real-world UEs, we discovered 197 violations of 22 LTE NAS security requirements, among which 190 have never been reported before. These violations are shown to have serious security consequences, demonstrating that their discovery can indeed help elevate protection of today's cellular network systems.

2 Background

2.1 3GPP Conformance Testing

Conformance test case. The conformance test case executed by the testing system evaluates whether a SUT (System Under Test) conforms to a set of system requirements, as specified by a given test purpose (an informal description of what is to be tested and what actions the SUT must complete if an event occurs in a certain state). Such a test purpose serves as an input to conformance test generation and is used to design a test procedure (also an informal description about actual test steps) and further develop a test suite (also known as a test script, which is typically written in a test specification language like TTCN-3 [16] in 3GPP, and can be implemented by TTCN compilers and executed on a test platform). So far, test generation has been done manually by 3GPP experts, entailing a significant amount of effort and time. Our research aims to automate the test procedure generation.

Generic procedures of LTE. Each 3GPP conformance test purpose describes states of UE at the time of testing. Such states can be reached following one of the generic procedures specified by 3GPP in its technical specification TS 36.508 [6]. For example, UE registration procedure illustrated in Table 4.5.2.3-1 of TS 36.508 describes a general procedure for a UE to connect to the network from power on, which can bring

Figure 2: Example of the semantic role labeling.

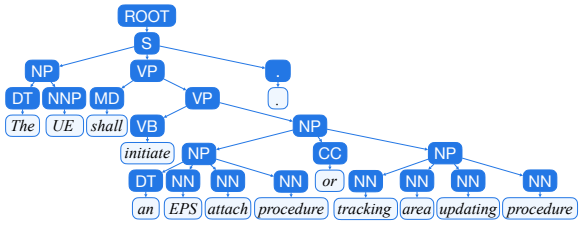

Figure 3: Example of the constituency parsing tree.

the UE to different states, including EMM-REGISTERED-INITIATED state, EMM-REGISTERED state, state of successful completion of EPS authentication and etc. In existing 3GPP test cases for NAS protocol, 44% start from the states led by the above UE registration procedure [7]. This generic procedure is also utilized in our work to generate a conformance test procedure for NAS security-related requirements.

2.2 Natural Language Processing

Semantic role labeling. Semantic role labeling (SRL) is an NLP task to automatically find the semantic roles played by each predicate's argument within a sentence, determining "who did what to whom", "when", "where", and etc [15]. Figure 2 shows a simple example. For the verb "initiate", SRL labels the words "the UE" as *ARG0*, indicating "who initiate", "an EPS attach procedure" as *ARG1*, implying "initiate what" and the statement "Upon expiry of timer T3247" as *ARGM-TMP*, representing the temporal modifier to the verb and describing when the action "initiate" takes place. SRL is useful for several downstream tasks such as question answering [34] and open information extraction [18]. In our research, we utilized SRL to extract the causal relation in a sentence, using the Allennlp SRL model (version 2.1.0), one of the state-of-the-art SRL tools, which achieved 86.49% F1-score on the Ontonotes 5.0 dataset [35] (see Section 3.2).

Constituency parsing. Constituency parsing is an NLP task that aims to generate a structured syntactic parse tree for a sentence by breaking it down into constituents in given grammar categories, such as noun phrase (*NP*) and verb phrase (*VP*) [37]. Figure 3 presents a simple example. Not only can we see in the figure that the phrase "The UE" is labeled as *NP* and the phrase "shall initiate ... procedure" is labeled as *VP*, but we can also see that the phrase "EPS attach procedure" (which is labeled as *NP* as a whole) and "tracking area updating procedure" (which is labeled as *NP* as a whole as well) are in coordinating conjunction (or) to the verb "initiate". In our work, we used the constituency parsing to identify the coordinating conjunction among phrases in a sentence with the Stanza model (version 1.3.0), one of the state-of-the-art constituency parsing models that achieves a test score of 91.5 on the PTB dataset. [30] (see Section 3.2).

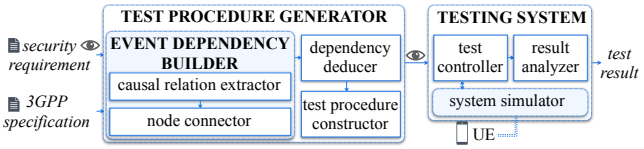

Figure 4: Architecture of our approach.

Data augmentation. Data augmentation is a strategy to generate additional synthetic data from existing data, in order to increase the diversity of training examples and hence reduce overfitting while training ML models [19]. Prominent examples of data augmentation include EDA [40] and BACKTRANSLATION [32], which are often employed to improve the performance of text classification tasks. EDA performs token-level random perturbation on a sentence, such as synonym replacement, random insertion, deletion and swap. BACKTRANSLATION translates a sequence into another language (e.g., from English to French) and then translates it back into the original language, which helps in the generation of textual data with different words but the same semantic meaning. In our study, we utilized both strategies while generating the training set for our ML model (see Section 3.2).

3 Contester: Design and Implementation

3.1 Overview

As aforementioned, to build conformance test procedures, the Contester pipeline we developed leverages a suite of NLP and ML techniques to extract all related event dependency relations from the 3GPP documents to construct an event dependency graph (EDG). Given a security requirement whose condition event cannot be directly triggered or expected operation cannot be directly observed by the testing system, Contester performs automatic reasoning backward or forward on the EDG to identify the target event chains. These chains are further used to generate the test procedure, which is finally executed in our testing system to verify whether a UE conforms to the security requirement.

Architecture. Figure 4 illustrates the architecture of Contester, including a *Test Procedure Generator* (TPG) and a *Testing System* (TS). The TPG runs an *Event Dependency Builder* (EDB) to recover event dependency relations from 3GPP specifications and build an EDG. For a given security requirement with specified condition events and expected operations, the module infers the target chains through reasoning on the graph to construct the test procedure for the requirement. The procedure is then used by the TS to guide a test controller to interact with a UE under testing. The traffic produced by the communication is inspected by a result analyzer to determine whether the UE has passed the test.

Example. Here we use an example to demonstrate how our approach works. Figure 5 presents a security requirement, with its condition event and expected operation being highlighted. For the condition event, the TPG determines that it can be triggered directly by the testing system since the network under the control of the testing system can send the AUTHENTICATION REJECT message to the UE. However, the expected action ("start timer T3247") is an internal operation in the UE and cannot be observed directly by the testing system. So the EDB builds an event dependency graph (EDG) related to the timer based on 3GPP specifications. For this purpose, Contester runs semantic-role labeling on the LTE NAS specification [4] to find all sentences with causal relations, such as *"Upon expiry of timer T3247, the UE shall initiate an EPS attach procedure or ..."* and identify individual semantic roles in the relations, such as "*expiry of timer T3427*" and "*the UE shall initiate an EPS attach procedure*". Then it utilizes a new model we designed to connect semantic roles across different sentences together, which addresses the challenge in determining semantically related technical concepts in the absence of adequate instances for training the model: for example, "*the UE shall initiate an EPS attach procedure*" is found to be related to "*the UE initiates the attach procedure*". The EDG Contester generates is illustrated in Figure 6. On the graph, the TPG infers an event chain (Figure 6), which reveals that once the UE starts T3247, the UE will send an ATTACH REQUEST message to initiate an attach procedure after the timer expires. This indicates that the testing system can determine whether the expected operation happens by examining whether the UE has issued ATTACH REQUEST after the period of time as specified by the value of T3247.

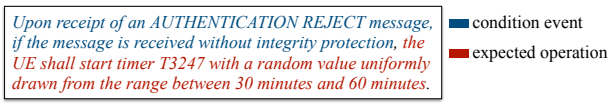

Figure 5: Example of a security requirement from TS 24.301.

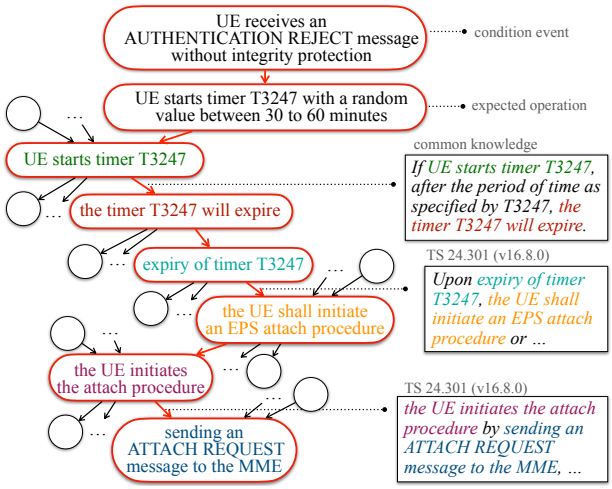

Figure 6: Example of the event chain identified from the dependency graph and related 3GPP documents.

Based on the above reasoning, the TPG then constructs the test procedure, using a generic procedure to initialize the UE state (see Section 2). Table 1 presents the test procedure generated by the TPG for the security requirement: Steps 1

Table 1: Example of the test procedure.

***Initial test state**: before the NAS security activation; **Condition event**: UE receives an AUTHENTICATION REJECT message without integrity protection; **Expected operation**: the UE shall start timer T3247 with a random value between 30 to 60 minutes* (and then try to connect to the network again, instead of stopping making any attempt until reboot).

Step	Procedure	Event Sequence				Verdict
		U-M	Message	Parameter	Sleep	
1	The UE is switched on.	-	-	-	-	-
2	The UE initiates the attach procedure.	→	ATTACH REQUEST	-	-	-
3	The MME transmits an AUTHENTICATION REQUEST message to initiate the authentication and AKA procedure.	←	AUTHENTICATION REQUEST	-	-	-
4	The UE transmits an AUTHENTICATION RESPONSE message and established mutual authentication.	→	AUTHENTICATION RESPONSE	-	-	-
5	The MME transmits an AUTHENTICATION REJECT message without integrity protection.	←	AUTHENTICATION REJECT	security header type = 0	-	-
6	The MME waits for 30-60 minutes.	-	-	-	30-60 min	-
7	The UE transmits an ATTACH REQUEST message	→	ATTACH REQUEST	-	-	Fail

to 4 come from the generic procedure and are used to move the UE into the initial test state; Step 5 triggers the condition event and Steps 6 and 7 check whether the expected operation has occurred. This test procedure is then handed over to our testing system, which runs the test controller and a hooked simulator to issue messages to the UE according to the procedure. After the communication of the first 5 steps, if the TS finds from its traffic log that the UE has sent an ATTACH REQUEST message within 30 to 60 minutes after Step 5, it reports that the UE has passed the conformance testing. Otherwise, the TS reports a failure result, which will be analyzed later to determine whether the UE violates the security requirement. Following we elaborate on the design and implementation of individual components.

3.2 Test Procedure Generation

A conformance test procedure involves three steps: initializing the UE to the test state, triggering the condition events, and confirming the occurrence of the expected operations. However, the types of the events that can be directly triggered or observed by the testing system are limited. Our observation is that all such events fall into two categories: 1) network environment change, which is directly set by the testing system to simulate the complicated testing scenarios for the UE in mobile communication, such as downgrading from 4G to 3G and switching between different core networks or service providers; 2) message transmission, through which the testing system can command the simulator (simulating the network) to affect the UE's state if the message is from the MME to the UE, and monitor the full interactions between the network and the UE through the air interface. So, when a condition event or the expected operation cannot be directly triggered or observed, we need to find an event chain starting with an actionable event and ending with an observable event. Due to the limitation of today's simulator, which does not fully support network environment changes, we only consider message transmission events in our research.

EDG modeling. An event chain (or a subgraph) is discovered from an event dependency graph (EDG), which is formally defined as follows: EDG $:= \{V, E, C_V, W_E\}$, where V is the set of nodes, E is the set of directed edges, C_V is the set of weights on nodes and W_E is the set of weights on edges. For an edge $e =< u, v >$, it connects the node u to v with a weight $W(e)$, which is either 1, indicating that the message issued by the event u once it happens will be passed to node v, or 0.5, denoting that the message may or may not be delivered to v (e.g., Figure 8(d)). For a node v, its event will only be triggered if it has received sufficient number message from its preceding events (nodes), no less than its weight $C(v)$, which essentially models an AND relation among these preceding nodes. For instance, the "AND" node in Figure 8(a) has weight 2, indicating that this "AND" event will only be triggered if all of the two preceding events happened.

To construct the EDG, the TPG runs the EDB to first identify the causal relations from the sentences in 3GPP specifications, and then infer the connections between different relations to link them together, so as to build up the whole graph. On the graph, the deducer performs automatic reasoning to find event chains (or subgraph) for a given security requirement, which are later converted to the security conformance test procedures by the procedure constructor (Figure 4).

Causal relation extraction. 3GPP specifications contain detailed textual descriptions of the UE's working procedures, including all the information about how a specific operation will trigger another operation. Such causal relations, in the LTE specifications, are usually encapsulated in a single sentence like "*If the SECURITY MODE COMMAND message can be accepted, the UE shall send a SECURITY MODE COMPLETE message ...*", indicating a causal relation between the condition event "acceptance of the SECURITY MODE COMMAND message" and the consequent event "send a SECURITY MODE COMPLETE message". So, our EDB inspects every single sentence of the LTE NAS specification to extract all the causal relations. Note that there is actually a unique presentation structure extensively used by 3GPP specifications, which involves multiple sentences but can be converted into a single sentence for causal relation discovery. Such a structure tends to describe a set of actions taken by a subject or triggered by an event, or a set of conditions for a given event, as illustrated in Figure 7. The description involving the structure can be easily identified from its format and these related sentences can be directly combined

Upon expiry of timer T3247, the UE shall
- *remove all tracking areas ...*
⋮
- *initiate an EPS attach procedure* ***or*** *tracking area updating procedure ...*

Figure 7: Example of statements across sentences.

into a single sentence by concatenating the verb phrases or conditional clauses together. We found that in the LTE specifications many causal relations are described in such a structure. Hence, the EDB pre-processes these statements, converting them into single sentences before extracting causal relations from the sentences.

The causal relation discovery is performed by the EDB using an NLP technique, called *Semantic Role Labeling* (SRL), which can extract sentence semantics by labeling its key elements, e.g., words and phrases, with their syntactic roles associated with predicates (see the example in Figure 2). Leveraging such semantic role labels and the leading word of each labeled semantic element, we use a set of rules to identify the causal relation in a sentence. Take the sentence in Figure 2 as an example. Here the rule applied is as follows: *ARGM-TMP* together with a set of leading words describing actions such as "upon" indicates the presence of a causal relation, in which the *ARGM-TMP* element will trigger the action labeled by *V* and other labels. So, for the sentence in Figure 2, the EDB determines the presence of the causal relation between "Upon expiry of timer T3247" and "the UE shall initiate an EPS attach procedure". Then, the EDB creates a node on the EDG for each element and connects them using a directed edge with a weight 1 to indicate that the occurrence of one event causes the other to take place. These rules were created manually in our research, based upon our summary of typical causal relations as discovered from the sample sentences drawn from 3GPP conformance requirements [7]. The complete set of rules is presented in Table 2 in Appendix.

More complicated are the presence of logic relations (*AND* and *OR*) in an element, indicating that the element actually describes multiple related events. Consider the example in Figure 7: the element "*initiate an EPS attach procedure or tracking area updating procedure ...*" contains two events, which are lumped together by SRL but can be separated by another NLP technique called constituency parsing (see Section 2). So, after a causal relation has been identified, the EDB further runs a constituency parsing model (Stanza [30], version 1.3.0) on every semantic element involved in the relation, in an attempt to refine the element by detecting the presence of the logic relation in the element. Once detected, the element is broken into two or more elements, each assigned to a node created by the EDB. If these nodes describe the operations to be triggered, they are connected to their preceding node (the condition event) in a way illustrated in Figure 8 (c) and (d), which also describe the assignments of both edge and node weights. Otherwise, these newly created nodes are connected to their succeeding nodes as specified in Figure 8 (a) and (b). Note that for the AND relation, a new node is created for an

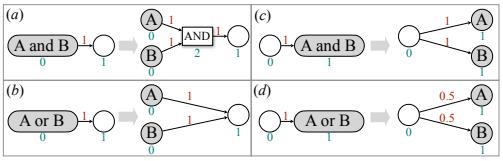

Figure 8: Node breakdown and graph transformation for describing logic relations. (a) and (b) describe the graph transformation for the condition event involving logic relations (AND and OR); (c) and (d) are for converting the consequent operation with logic relations.

AND event. For example, for the highlighted element in Figure 7, the EDB creates two nodes with node weight 1, one for "the UE shall initiate an EPS attach procedure" and the other for "the UE shall initial tracking area updating procedure", which are connected to the preceding node (condition event) with a weight 0.5 (Figure 8 (d)).

Moreover, in our research, we found that some causal relations can be identified using common knowledge though 3GPP specifications do not document them (e.g., after the time as specified by a timer's value has passed, the timer becomes expired). To address this problem, we added these causal relations by automatically generating artificial sentences as the EDB's inputs to put back the links missed by the 3GPP specifications. The sentences used in our research are all related to timers. Specifically, for each timer, our approach generates a sentence that describes a causal relation between two events ("start a timer" and "expiry of a timer"): e.g., "If UE starts timer T3247, after the period of time as specified by T3247, the timer T3247 will expire", as shown in Figure 6. In total, we generated 50 artificial sentences for timers.

In our research, we used the EDB to analyze the 3GPP LTE NAS specification [4], which has 575 pages with 8,522 sentences and 284,047 words, together with the 50 generated artificial sentences. The EDB extracted 4,721 causal relations with 7,450 nodes including 251 AND-logic nodes.

Node connection. From the extracted causal relations, the EDB connects two nodes from different sentences if they are semantically related. For example, Figure 6 shows that two nodes ("*the UE shall initiate an EPS attach procedure*" and "*the UE initiate the attach procedure*") are linked because they share the same semantics. Furthermore, the EDB connects two nodes if one of them is an instance of another. For example, the nodes "*UE takes a partial native EPS security context into use*" and "*UE takes an EPS security context into use*" are related because they refer to the same action (taking something into use), and the latter's object of the action "*partial native EPS security context*" is an instance of the former's object "*EPS security context*", so the invocation of the second event will cause the invocation of the first event, and the nodes should be linked together.

Connecting these dots (nodes) turns out to be complicated. Specifically, traditional NLP techniques for determining the semantic similarity between two sets of textual data, Sen-

tence Similarity in particular, cannot help here. For example, "*the UE sends an ATTACH REQUEST message*" and "*the UE sends a SERVICE REQUEST message*" describe completely different events while the current sentence similarity model [39] considers them to be similar with high confidence (86.4%), due to its focus on the message-sending action, which is less important here. Also, when it comes to establishing the relationship between a more generic concept ("EPS security context") and its more specific instance ("partial native EPS security context"), no existing NLP technique can provide such support, up to our knowledge. To address these challenges, we built a new classification model designed specifically to identify such relations from 3GPP specifications. Our model is based upon an existing fine-tuning BERT for 3GPP [12], which has learned the 3GPP language model and produces high-quality embeddings, followed by a fully connected neural network as the classifier. The model's input is two sentences, and its output is a label for relation between the two sentences, i.e., 0: the inputs have no relation, 1: they have the same semantic meaning; 2: the first is an instance of the second; 3: the second is an instance of the first.

Most important for building the model is the collection of training data, which is nontrivial. Specifically, we cannot randomly collect sentences from the 3GPP specifications, since these receiving the class label 1, 2 and 3 are very rare. Our solution is an automatic generation of the training instances together with data augmentation [32, 40]. Specifically, for the class 1, we utilize BACKTRANSLATION [32], translating a candidate sentence into another language using the Google translate API [21] and then translate it back into the original language to generate textual data with different wording but the same semantic meaning. In our research, for each sentence in the LTE NAS specification, we generated a new sentence and took the pair of the original sentence and the new one as a training instance for the label 1. To generate training instances for label 2 and 3, we need to produce sentence pairs with one being an instance of the other: that is, both sentences describe the same action (e.g., "take ... into use"), while the object of the action in one sentence ("partial native EPS security context") is an instance of the object in another sentence ("EPS security context"). The objects we care about are those defined as *proper nouns* by 3GPP at the beginning of every specifications. From the definitions of these terms, we can determine that one such noun describes the instance of another (e.g., "*An EPS security context has type "mapped", "full native" or "partial native".*") to establish such relations across proper nouns. Leveraging such relations, we automatize the generation of the training pairs for the label 2 and 3 by simply identifying from the LTE NAS specification all sentences carrying these "instance proper nouns" as the objects of their action terms, and replace these nouns with the proper nouns for their general concepts, to form pairs for training.

Finally to produce the training set for the class 0, we run a sentence similarity model [39], on randomly selected sentence pairs from 3GPP specifications. The pairs predicted to be "not similar" with high confidence (90%) by the model are then used as training data for the label 0. Also to help our model differentiate between two sentences describing different events but including the same action term ("*the UE sends an ATTACH REQUEST message*" vs. "*the UE sends a SERVICE REQUEST message*"), we transform every sentence in the LTE NAS specification by replacing its object with a randomly-selected different proper noun to form a training pair. In this way, we generated 90,959 distinct inputs (pairs) for the class 1, 40,609 for the class 2, 40,609 inputs for the class 3, and 175,441 inputs for the class 0. To balance the training dataset, we further applied data augmentation algorithms [32,40] (see Section 2) to increase the number of inputs in each class to 200,000.

Graph building. On the generated data, we trained the model and used it to construct the EDG on LTE NAS specification. Specifically, Contester runs the model to analyze each pair of all causal relations discovered, in an attempt to determine whether any relation (1: same semantics, 2 and 3: instance and concept) exists between any node (condition event or expected action) of one causal relation and a node of the other relation. Once any one of the 3 relations is found, the EDB connects them. Specifically, for class 1 relation, a directed edge connects one node to the other and another edge connects the latter back to the former. The weights of both edges are set to 1, indicating that the occurrence of one event is equivalent to the occurrence of the other. For the relation in the class 2 or 3, we use an example in Figure 9 (N_{ins}: "take a partial native EPS security context into use", N_{cpt}: "take an EPS security context into use") to show how the EDB connects related nodes. First, the EDB builds a direct edge from the N_{ins} to N_{cpt} and assigns it a weight of 1, because a concept event always happens once its instance occurs. On the other hand, when both the concept event and an event related to creating an instance for the object of the operation described by the concept event happen together, the instance event should also be triggered. In the example, the EDB creates a node N_p for the event "create a new partial native EPS security context", which together with the concept event "take an EPS security context into use", causes the instance event "take a partial native EPS security context" to take place; so these two nodes N_p and N_{cpt} are all connected to an AND node N_{AND} before linked to N_{ins}. The weights for these nodes and edges are set as described in Figure 9. The events such as the one described by N_p can be generated automatically but need to be reviewed by the expert to remove those inconsistent with the description of specifications. In this way, we built up the whole EDG for the LTE NAS specification, including 7,450 nodes and 22,110 directed edges that represent 1,884 sentences.

Inference on the graph. As mentioned earlier, in our research, we only consider the events that transmit messages through the air interface to be directly invocable by and observable to our testing system (TS). The former includes the

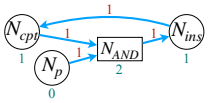

Figure 9: The node connection between an instance and its concept.

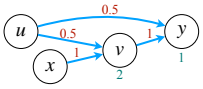

Figure 10: Toy example of EDG to demonstrate reasoning algorithm.

MME events that sent messages to the UE and the latter are the UE events that transmit messages to the MME. These "directly invocable" and "directly observable" nodes are automatically discovered from the EDG using the ML classifier we trained, which compares the event described by each node with automatically generated sentences such as "the UE sends an ATTACH REQUEST message" to find the node with the same meaning (in the class 1). For this purpose, we constructed two templates for the sentences, one for UE to MME and the other for MME to UE. Then from the section named "message functional definitions and contents" provided by the NAS specification, our approach automatically discovers the related messages and their directions to fill up the templates.

The nodes identified by our ML model form the set T of "directly invocable" nodes, which should be the start point of a conformance test procedure, and the set O of "directly observable" nodes, which should be the end point of the procedure. However, the condition event or the expected operation of a security requirement may not be in these sets. So Contester performs automatic reasoning to find event chains (or subgraphs) to link these requirement events to those in T and O. Formally, for an edge $e =< u, v >$, we use $in(e)$ and $out(e)$ to represent its start node u and end node v respectively. For a node v, we use $In(v)$ to denote the set of its inbound edges, i.e., $In(v) = \{e : out(e) = v\}$, and $Out(v)$ the set of its outbound edges, i.e., $Out(v) = \{e : in(e) = v\}$. The number of messages the node v accepts, $m(v, \delta)$, is calculated as follows:

$$m(v,\delta) \quad = \sum_{e \in In(v)} \mathbb{1}[f(e) \geq \delta], \tag{1}$$

where $f(e) = f(u) \times W(e)$ and $f(v) = \mathbb{1}[m(v,\delta) \geq C(v)]$. Here $\delta = 1$ indicates that the node v refuses to accept the messages delivered through the edges with the weight of 0.5, and $\delta = 0.5$ indicates that v accepts message from all edges. $f(v)$ describes how many messages from v are accepted by their recipients and $f(e)$ denotes the number of accepted messages delivered through e. Following we describe the reasoning process using a simple Dynamic Programming (DP) algorithm:

• For an expected operation node $x \notin O$, we first set the f value of the node x and those in T to 1: i.e., $f(u) = 1, \forall u \in T \cup \{x\}$, then propagate messages from these nodes using Eq. 1 with $\delta = 0.5$, until a valid directed path is found from x to a node $y \in O$ and $f(y) = 1$, or none of the f values (for edges or nodes) can be further updated.

• For a condition event node $y \notin T$, we first set the f value of those nodes in T to 1: $f(u) = 1, \forall u \in T$, and then propagate messages from these nodes using Eq. 1 with $\delta = 1$, until $f(y) = 1$ or none of the f values (for edges or nodes) can be further updated.

• To find a event chain (or a subgraph) that connects x to "directly observable" nodes or connects "directly invocable" nodes to y, our approach records $R(v) \subseteq In(v)$, a set of inbound edges of v that makes $\sum_{e \in R(v)} f(e) = C(v)$, for those nodes v satisfying $f(v) = 1$, when running the DP algorithm. After an observable end point or an invocable start point has been found, our approach derive the desired event chain (or subgraph) through a backward induction using these records.

Here, we use a toy example of EDG (Figure 10) to demonstrate how our reasoning algorithm works. To infer the event chain that makes x observable, we first set $T = \{u\}$ and $O = \{y\}$. Using our reasoning algorithm, we initialize $f(x) = 1$ and $f(u) = 1$. Then by iteratively applying Eq. 1 with $\delta = 0.5$ to the nodes receiving messages and the edges delivering messages, we derive that $f(v) = 1$ and $f(y) = 1$. Since $y \in O$ is a successor of x and $f(y) = 1$, we determine x is observable and the event chain is $x \to v \to y$. If we start with $T = \emptyset$ and $O = \{y\}$, we will derive that $f(v) = 0$ and $f(y) = 0$, thus x is unobservable in this case.

Now let us find out how to make the event y invocable, with $T = \{x, u\}$. We initialize $f(x) = 1$ and $f(u) = 1$ to apply Eq. 1 with $\delta = 1$ on the EDG. In this way, we derive that $f(v) = 0$ and $f(y) = 0$, which indicates that there does not exist an event chain to invoke y. However, if we change $C(v) = 1$ and set $T = \{x, u\}$, our algorithm reports that $f(v) = 1$ and $f(y) = 1$, thus y can be invoked through $x \to v \to y$. Also, if we set $W(< u, y >) = 1$ and $T = \{x, u\}$, our approach will derive that $f(v) = 0$ and $f(y) = 1$, thus y can also be invoked.

Test procedure generation. Given an event chain (or a subgraph) from an invocable action to an observable event, the TPG component of Contester converts it into a test procedure. A test procedure specifies the actions the UE must take once a condition event occurs at a certain UE state. More specifically, the procedure starts with the steps that lead the UE to the initial test state, then issue messages to invoke the condition events and finally takes action to determine whether or not the expected operations have been performed successfully. Table 1 shows a test procedure for a security requirement: Steps 1 to 4 guide the UE into the specified state (before the NAS security activation); Step 5 triggers the condition event (UE receives an AUTHENTICATION REJECT message without integrity protection); Step 6 to 7 determine whether the expected action (start timer T3247) has been taken.

To bring the UE to the initial test state, our TPG leverages a generic procedure (the UE registration procedure [6]), from which, the TPG extracts the steps that move the UE from power-on to the target state through message transmission. Such a target state is typically described in a security requirement in the form of before or after the occurrence of a certain event: e.g., "before the network has established secure exchange of NAS message". Based upon such description,

the TPG searches the EDG to find the message transmission event(s) in the set T that leads to the target event. In the example, the message event is "MME sends a SECURITY MODE COMMAND message to the UE", which will trigger an event chain leading to "the network has established secure exchange of NAS message". Then, our approach locates the message mentioned in the transmission event (SECURITY MODE COMMAND) to extract all steps from the general procedure before the transmission event or after the event, depending on the description of the target state. In the example, the list of steps before the "MME sends a SECURITY MODE COMMAND message to the UE", is used to initiate the test procedure. Note that in the presence of multiple event chains toward the initial state, our approach produces one test procedure for each chain. Moreover, we found that some security requirements do not contain the description of the initial state, and instead are supposed to be followed throughout the UE's operations. So for these requirements, the TPG provides two initiate states: before and after the NAS security activation, two crucial NAS security states that have also been studied by related prior research [14, 29]. For each such state, our approach creates a test procedure.

Right after the steps to set the UE's initial state, the TPG adds the event chain (or subgraph) for invoking the condition event, and the operations to observe the occurrence of the expected operation directly or through the observable event(s) the operation initiates. Specifically, along the event chain, the TPG adds each event's statement to the procedure under construction, together with the message automatically identified during the inference step. For example, for the step 5 in Table 1, the TPG takes the condition event's statement as its description, enters the message name "AUTHENTICATION REJECT" (identified when determing the invocable and observable sets). In addition to the message transmission event, on the procedure are also events related to timers: whenever a timer is set, we retrieve its expiration time from the specification and enter the information into the Sleep column in the test procedure sequence (Step 6 in Table 1). In the end, the conformance test procedure includes the field verdict attribute [7], which is set to "pass", indicating that the expected operation is observed directly or indirectly. Notably, the message's specific parameters are set manually. However, this manual effort is moderate since the parameter information is usually provided by the event's statement. Taking step 5 in Table 1 as an example again, the parameter "security header type" is set to 0 according to the condition event description "without integrity protection" and the message functional definitions and content in the NAS specification [4].

3.3 Testing and Test Result Analysis

For each security requirement, our testing system (TS) uses the test procedures generated by the TPG to evaluate a UE's conformance to the requirement. Following, we elaborate the implementation of our test environment and the way the TS performs the test and determine the test result.

Test environment building. During the testing, the TS controls the network to communicate with the UE following the test procedure sequence (the U-M, Message, Parameter and Sleep columns in Table 1). To this end, our TS is built on a hooked LTE simulator (simulating the network) with a component called *Test Controller*, which guides the network to perform the specified actions on the procedure. By analyzing the traffic logs between the UE and the network, the *Result Analyzer* component of the TS determines the test result (see Figure 4) and outputs the verdict for the test.

In our implementation, we utilized an SDR board (LimeSDR USB v1.4) connecting to a computer that runs a simulator software (srsRAN v21.04 [36]), which simulates the network to communicate with the UE through the air interface. To let the TS monitor the NAS message the network received from the UE and instruct the network to reply with a specified message or perform other actions, we implanted *hooks* in the simulator. Specifically, we placed instructions to trigger the function HOOK (illustrated in Figure 11 at Appendix) right before the code for processing a specific inbound message and for preparing a specific outbound message. The HOOK before inbound message processing first notifies the Test Controller of the message received from UE and then fetches instructions from the Controller for the follow-up action network shall take. If the follow-up action is to let network wait for a timer to expire, HOOK will acquire the instruction for the next action again after sleep. If the follow-up action is to send a specific message to UE, the HOOK stops and lets simulator continue processing the inbound message until another HOOK in front of outbound message delivery is encountered. Also before issuing a specific output message, the HOOK queries the Test Controller for the name and parameters of the message to be sent and then calls the simulator API (e.g., liblte_mme_pack_attach_request_msg) to prepare message and send it out, in accordance with test procedure.

The Test Controller follows the test procedure step by step to issue messages, run a listener to receive notifications and requests from the simulator and respond to them with the requested information according to the test procedure. Take the procedure in Table 1 as an example. After the Controller commands the UE to reboot at the beginning of the test, it moves onto the step 2 – waiting for an ATTACH REQUEST message from the UE. During the test, if a notification from the simulator, such as the reception of a message or completion of a sleep, indicates the occurrence of the event expected, the Test Controller will conclude that this step has been completed and move to the next step. When the Controller receives a request from a HOOK about the simulator's next action, it replies with the type of the actions for the next step, either message transmission or sleeping. If a request is about the information of the message to send to the UE, the Controller replies with the message name and parameters according to the current step in the test procedure. When the procedure runs to an end,

the Test Controller will invoke the Result Analyzer. Notably, if the notification from the simulator is different from the expected action and no value is given in the verdict field, the TS stops and issues an error message.

Result. At the end of the test, the Test Controller issues a message to the Result Analyzer, to indicate either a success of the test when the expected last message has been received by the Controller, or a failure, when the last message does not arrive after a predetermined waiting time. The Analyzer keeps track of all conformance tests related to the security requirement and if any of them passes, it reports conformance to the requirement; otherwise, it reports nonconformance.

4 Evaluation and Discovery

This section reports our evaluation of Contester, which provides evidence that our approach is capable of automatically generating conformance test procedures and determining whether a UE's implementation complies with the LTE NAS specification. In our research, we further ran Contester on LTE security requirements to create test procedures and evaluate UEs on them to understand whether the implementations of these UEs conform to the NAS security requirements. Our evaluation has led to the discovery of many conformance failures, which we analyzed and present findings in this section.

4.1 Evaluation

Settings. In our experiments, we installed the TPG on a server operating a Ubuntu 20.04 LTS, with a 1.5GHz CPU, 256GB memory and 3.5TB hard drive, as well as CUDA 11.7 on four GPUs (NVIDIA A40). Our Testing System includes an SDR board (LimeSDR USB v1.4), a back-end host running a simulator (srsRAN v21.04), Test Controller and Result Analyzer on Ubuntu 18.04 with a 2.10GHz CPU, 8GB memory and a 512GB hard drive. The UE devices we evaluated include 20 mobile phones(e.g., iPhone 13, Pixel 5a, Honor Play20) and 2 IoT devices (both USB Dongles), which were connected to the SDR board through an air interface. All experiments were conducted in a radio-isolated shield box to prevent any interference with the communication outside. The detailed information about the UEs is presented in Table 4 at Appendix.

Effectiveness. We evaluated the effectiveness of Contester on its two key components: the TPG and the TS. The effectiveness of the TPG mostly depends on the quality of the EDG, a graph constructed by the EDB. The EDB includes two important components: causal relation extractor and node connector. In our research, we first evaluated the quality of the causal relation extractor and the node connector respectively, and then the EDG's quality.

To evaluate the causal relation extractor, we built a dataset with 100 randomly selected sentences from the LTE NAS specification, 50 with causal relations (and logic relations) and the rest not, and further manually drew the ground truth graph for each sentence (empty graph for those carrying no causal relations). On this dataset, we ran our causal relation extractor to generate the graph for each sentence and calculated the Graph Edit Distance (GED) [20] between the generated graph and the ground truth graph. We found that our causal relation extractor correctly identified the relation for 99 sentences, and missed only on 1 sentence where the GED is 1. Specifically, on this sentence, the extractor generated a wrong node with a wrong statement. We analyzed this error and found that it was caused by the failure of the SRL that wrongly labeled a semantic element: missing the phrase "to 5" in the element "UE may set the attach attempt counter to 5". In general, the average GED between the graphs generated by our causal relation extractor and the ground truth graphs is 0.01, showing the high efficacy of our extractor.

To evaluate our node connector, which classifies the relation between a pair of semantic elements into 4 classes, we constructed another dataset with 400 pairs of semantic elements (100 pairs for each class). These pairs were gathered from 500 randomly selected sentences from the LTE NAS specification. On this dataset, our node connector correctly classified 98 pairs for the class 0, 91 pairs for the class 1, and all the pairs in the class 2 and class 3. In general, the classification accuracy of our node connector is 97.25%.

To evaluate the correctness of the whole EDG, which is built on the outputs of our causal relation extractor and node connector, is hard, since the event dependency graph is huge, making it impossible to manually build up a ground truth graph for the whole specifications. Thus, we randomly selected a set of nodes and edges to evaluate their correction. Specifically, we randomly chose 200 nodes from the EDG built on the NAS specification and checked the correctness of the semantic elements represented by these nodes through analyzing their origin sentences. Among these 200 nodes, 14 nodes carry wrong semantic elements and the rest contain correct elements. Thus, we estimated that 93% nodes in the EDG is correct. To evaluate the correctness of edges we first randomly selected 100 pairs of nodes such that there is no edge connecting two nodes in the same pair and built an edge between them. Together with 100 real edges randomly selected from the EDG, we constructed a set of 200 edges: 100 real edges and 100 false edges that do not exist in the EDG. For those 100 false edges, we treated them as the negative samples predicted by the EDB and manually checked whether they should exist. For the 100 real edges, we considered them as the positive samples and manually checked each edge to determine whether both its direction and existence are correct. In this way, we got 88 true positives, 97 true negative, 12 false positives and 3 false negatives. So, we estimate the recall of the edges to be 96.70% and the precision to be 88.00%.

To evaluate the end-to-end effectiveness of the TPG, we constructed a dataset by randomly selecting 20 requirements (with given initiate states, condition events and expected operations) from existing 3GPP NAS conformance test cases [7] as the TPG's input. On this dataset, the TPG successfully generated 36 test procedures for 19 requirements, with 29

being correct. We first checked the requirement for which our TPG did not generate the test procedures. This requirement contains a condition event "attach attempt counter is less than 5", which could not be handled since the specifications do not explain how the event can be invoked. Then, we looked into 7 incorrectly generated procedures, and ascribed the failures to the wrong classification results predicted by our node connector. Specifically, our node connector predicted that the event "the UE shall set the attach attempt counter to 5" and the event "the UE shall reset the attach attempt counter" have the same semantics (labeled as class 1). So, our TPG wrongly believed that the events causing "reset the attach attempt counter" can also trigger the event "set attach attempt counter to 5". The result indicates that precision of 80% was achieved by our TPG in generating test procedures.

To understand the effectiveness of the TS, we manually checked its execution of the correctly generated test procedure for the 19 requirements on 22 UEs, which were all correct. Also, all results for the UEs were confirmed manually.

Performance. Our implementation of Contester took 25 minutes to build EDG for the whole LTE NAS specification, including 5 minutes to extract the causal relations from the specifications, which contains 575 pages and 8,522 sentences with 284,047 words, 20 minutes to link nodes from different sentences by the ML model, and 10 seconds to determine whether each a node is invocable, observable, or not. Based on EDG, for 20 security requirements, our TPG took only 2 seconds to infer the event chains (or subgraphs) on EDG for 5 condition events that cannot be directly triggered by the TS and 3 expected operations that cannot be directly observed, and generated 36 test procedures for all the requirements, and the TS used around 66 hours to test them on all 22 UEs. Specifically, running each test procedure on a UE took around 6 minutes on average (ranging from 30 seconds to 13 minutes) and testing each security requirement on a UE took around 9 minutes averagely (ranging from 30 seconds to 2 hours). This result offers strong evidence that Contester is capable of generating security conformance test procedures for the 3GPP specifications and also executing the procedures on UEs to determine their compliance with the requirements.

4.2 Processing Real Security Requirements

In our research, we used the Contester, with the same setting as introduced in the evaluation (Section 4.1), to generate test procedures for the security requirements in the LTE NAS specification and tested 22 UEs to understand whether implementations of real devices meet NAS security requirements.

Security requirements. We extracted 50 security requirements, which are from three sources: 1) 34 from the NAS security chapter, 2) 6 from the rest of the LTE NAS specification [4] that refer to the security-focusing 33 series technical specifications [2], and 3) 10 from the rest are the sentences that require additional protection from the UE before the NAS security context has been established to protect its communication with the MME. Note that all of these requirements carry the verb "shall", which is used in 3GPP specifications to indicate that the requirements must be strictly followed [17]. Among them, 23 are the requirements under the handover situation (network exchange), which today's LTE simulators do not fully handle. So we had to focus on the remaining 27 security requirements. Before running Contester on them, we manually specified the initiate state, condition event and expected operation in the security requirement sentence (such as the example illustrated in Figure 5) as the TPG's input. The security requirement is presented in Table 3 at Appendix.

Testing result. For the 27 security requirements, Contester generated 143 test procedures for the 22 requirements among them, with spending 34 days, including only 4 seconds in test procedure generation and the rest in testing on 22 UEs. We found that the majority of time (96.26%) is consumed sleeping for timers. For example, when testing the security requirement S_{14} in Table 3, for each test on a UE with a test procedure, the TS must sleep for 30 to 60 minutes to allow the timer T3247 to expire and then check whether the MME receives a specific message (e.g., ATTACH REQUEST message) from the UE at that time. Among the 143 generated test procedures, we manually confirmed that the 136 procedures are correct, which can be used to tested for the 22 security requirements, while the remaining 7 are wrong. First, we looked into the 7 wrongly generated test procedures, which are all generated for the same security requirement S_{13}. Its related event "attach attempt counter is equal to 5" has been incorrectly linked to the event "reset the attach attempt counter" by the ML model too, as we analyzed in the evaluation. So, the TPG wrongly thought that those events to trigger the event ("reset the attach attempt counter") can also be used to trigger the event ("attach attempt counter is equal to 5"). Then, we checked the 5 security requirements, for which our TPG does not generated procedures. Among the rest 5 security requirements, three share a special condition event called *NAS COUNT WRAP AROUND*, which could not be handled since the specifications do not explain how the event can be invoked at all: the event is considered to be common knowledge that the NAS counter needs to be reset once overflown. Without the knowledge, the TPG does not know how to issue a message to cause the event to happen. The other two have the identical condition event "establish a new NAS signalling connection". The problem is that throughout the specifications, no description provides the complete information about how to invoke the event. Specifically, the only clue we can find indicates that "initial message" can trigger the event, yet the "initial message" is only loosely specified as follows: "*For this purpose the initial NAS message (i.e., ATTACH REQUEST, TRACKING AREA UPDATING REQUEST, ...*" (Page 42, LTE NAS [4]). In the end, still there is no clear information whether the message should go from the MME to the UE or the other way around.

After testing the 136 successfully generated test procedures for the 22 security requirements on the 22 UEs, Contester suc-

cessfully reported 287 conformance reports, which verifies that the implementation of UEs follow the security requirement in the LTE NAS specification.

4.3 Findings and Analysis

During the security conformance testing on the 22 UEs, Contester reported 227 failures, in which the UEs did not pass the test procedures. Such a failure indicate that the corresponding security requirement could be violated by the UE's implementation. In our research, we manually analyzed each failure, to determine whether it was caused by a problematic implementation of the security requirement or by a disruption of the event chain so the condition event of the requirement was not invoked or the occurrence of the expected operation did not lead to an observable event. For this purpose, we leveraged the 3GPP current test cases, which all commercial UEs are required to pass before being released. Specifically, if all the causal relations in the chain (except the one describes the security requirement under test) have conformance tests already released by the 3GPP [7], we have reason to believe that the conformance of these relations to the specifications has already been evaluated on the commercial UEs used in our study, and therefore a failure in our security conformance test should be attributed to a violation of the security requirement by the UE. In this way, we discovered 197 security requirement violations, including 190 never reported before, up to our knowledge, on the 22 commercial UEs. Every UE under test violates at least 3 security requirements, 14 at most, 8.95 (=197/22) on average. Among the 22 security requirements, only 3 of them (S_{17}, S_{19} and S_{21}) have been correctly followed by all UEs. The remaining 19 are violated by at least one UE. Especially, there are 2 security requirements (S_{11} ans S_{20}) that no UE implements correctly. This result reveals severe violations of LTE NAS security requirements by today's commercial phones and IoT devices, which are provided by the mainstream device vendors and baseband manufacturers. Detailed information can be found in Table 4 at Appendix.

Analysis of security violations. In our research, we found that the purpose of these security requirements is to protect the UE against three types of classic threats to the cellular network: the man-in-the-middle attack, the fake base station attack, and the replay attack. Failure to implement these security requirements may expose the UE to these attacks, further resulting in serious consequences, such as denial of service, eavesdropping, and location leakage. Below, we elaborate on the violated security requirements and discuss their potential consequences if they are not followed by the UE.

The man-in-the-middle (MITM) attack could have serious consequences on a cellular network, e.g., enabling the adversary to eavesdrop on sensitive communication and modify the commands issued by the MME to operate on the UE [33]. To avoid such risks, the LTE NAS protocol utilizes integrity protection and encryption for the communication between the UE and the MME. To this end, the LTE specification provides security requirements to specify when and how to perform integrity protection and encryption. For example, the security requirement S_{16} states that with the exception of two emergency situations, the UE must always activate such protection; the security requirements (S_7, S_8 and S_9) state that once the security context has been established between the UE and the MME, all the messages issued by the UE must be protected and the UE shall only process the encrypted messages with integrity signatures. Failure to implement these security requirements will result in end-to-end attacks in the real-world network, as prior research demonstrated [14, 33]. However, among the UEs we tested, still there are 8 devices (e.g., Galaxy A71 5G, Honor 6X) that violate these requirements, including 20 cases that have never been discovered before. In addition, in order to achieve the integrity protection and encryption, the security parameters (e.g., keys for calculating integrity signatures) are exchanged by the security mode control procedure first, whose security protection must also be assured. Thus, to prevent the MITM attackers from undermining these parameters, the security requirement S_{12} asks the UE to compare a hash value from the MME with a locally calculated hash value on the parameters the UE originally sent to the MME so as to detect any unauthorized change to the parameters. If a UE fails to implement this security requirement, it will face a bidding down attack [12]. In our testing, we found that 13 UEs do not meet the requirement, even including the mobile Honor Play20, a new phone published last year in June. In addition to the protection on exchange of security parameters, the LTE specification also mandates the UE to carefully handle the parameters on the UE side. For example, when the UE and the MME refresh the security parameters (by creating or using a new EPS security context), the security requirements (e.g., S_1, S_2, S_3 and S_4)) demand that the UE must delete the previously used or currently used EPS security context. However, we found that 14 devices do not delete the specified EPS security context as required. They continue to accept the messages encrypted and integrity-protected with the supposed to-be-deleted EPS security context, which can be utilized to perform a replay attack to locate the UE, as the LTEInspector [22] does. To the best of our knowledge, our work is the first to uncover this type of security requirement violation, in which the UE wrongly handles the critical security parameters (EPS security context), resulting in a failure to meet the security requirement of the NAS specification.

The fake base station attack is one of the most prevalent attacks in the cellular network, using a fake base station to simulate a network to communicate with the victim UE in order to steal sensitive information or deny the UE's service for a period of time. Against these attacks, the LTE offers an authentication procedure that allows the UE to identify fake base stations. To successfully proceed with this procedure, the LTE specification provides security requirements (e.g., S_{10} and S_{11}). However, we found that 14 UEs fail to meet S_{10} and all the UEs fail to meet S_{11}, which will cause

the authentication procedure fail. Sending reject messages to the victim UE to command it to disconnect from the network is an attack performed through the fake base station, which has been widely reported in recent years [12, 14, 41]. To safeguard the UE against this type of attacks, the 3GPP releases a brunch of security requirements mandating the UE to carefully handle these reject messages (S_6, S_{13}, S_{14}, S_{15}, S_{17}, S_{18}, S_{19}, S_{20}, S_{21}, and S_{22}). For example, when a UE receives an AUTHENTICATION REJECT message without integrity protection, which could be sent by a fake base station, the security requirement S_{15} requires the UE to reconnect the network after a default period of time (30-60 minutes), rather than remaining out of service until reboot. In our tested UEs, we found that all the 22 UEs fail to implement some of these security requirements. Even the most recently released iPhone 13 (09/24/21) violates security requirements S_6, S_{15}, S_{18}, S_{20} and S_{22}.

Replaying specific messages to the UE would result in a traceability attack [11, 22]. The 3GPP specification claims that they will provide replay protection in the communication between the UE and the MME [5]. The security requirement S_5 in the LTE NAS specification states that the UE can only accept a message with a given sequence number once. However, we found 12 UEs do not follow this security requirement, including the Google Pixel 5a and Honor Play 20, which are released on 08/26/21 and 06/16/21 respectively.

Our discovery reveals a pervasive failure of today's UE implementations in meeting 3GPP NAS security requirements, exposing them to the security risks for which the 3GPP has already provided mitigation. Although the 3GPP has issued 9 conformance test cases for 2 security requirements, our findings show that this is far from enough for protecting the UE. So the LTE NAS security requirement conformance testing is urgently needed and our technique will facilitate the development of these tests.

5 Discussion

Limitation. Although running Contester can automatically generate test procedures for given security requirements by automated reasoning on the EDG, there is still some manual labor involved in building the graph: 1) during extracting causal relations, we manually created rules to extract it from the SML's output and added those causal relations that the 3GPP specification does not mention but are the common knowledge being used by the specification (e.g., "start a timer" and "expiry of timer"); 2) to generate the training examples of class 2 and 3 for the ML model, we manually found the proper noun that contains a specific instance or belongs to a generic concept from the specification. These manual labors render Contester less effective in building the EDG. Moreover, limited by the capability of today's simulators, which do not fully support the exchange of network environment, we did not generate test procedures for 27 security requirements that is related to network exchange and did not test UEs for them. These will be tested once the simulator fully supports it.

Future work. 3GPP specifications contain a wealth of information, which could be extracted by NLP-enabled semantic analysis to enhance the security assurance of the cellular network. A meaningful attempt is this research that uses NLP techniques to extract semantic meanings from 3GPP specifications and, based on them, builds the EDG to facilitate automatic LTE conformance test generation. Down this road, a more comprehensive and precise EDG could be built by bringing in more advanced techniques in the future to improve the test procedure generation. An imminent future work could be automatizing the manual step included to check whether the failure of a test procedure is caused by the incorrect implementation on the security requirement or not through using NLP-based techniques to understand current 3GPP conformance testing specifications.

6 Related Work

Testing in LTE network. Prior research revealed a significant number of implementation issues in the LTE network [14, 23, 25–27, 29, 31, 33]. Rupprecht *et al.* [31] develope the first LTE testing framework and uncovered several UEs that accepted insecure security algorithm. Kim *et al.* [27] propose a semi-automatic method for fuzzing the LTE network based on some basic security properties. Park *et al.* [29] introduce a negative testing framework that can comprehensively test UE devices. Different from these negative testing approaches, Contester is designed to perform conformance (positive) testing on the end-user devices through automatically generating test procedures. Moreover, Kim *et al.* [26] study how to verify whether the communication message structure conforms with the LTE specification design. Hussain *et al.* [23] compare two UEs' behaviors represented by FSMs to identify deviant activities. Unlike these conformance testing studies on LTE networks and UEs, Contester is used to determine whether the implementation of actions on UEs adheres to security requirements as laid out by specifications.

NLP/ML in LTE security analysis. Leveraging NLP/ML technologies, two prior studies show that semantic information from 3GPP specifications can be automatically extracted for LTE security analysis [12, 14]. Specifically, Chen *et al.* [14] apply textual entailment and dependency parsing to identify hazard indicators from the LTE specifications, which are used to generate test cases for discovering vulnerabilities in the UE and the network. Chen *et al.* [12] built an NLP/ML pipeline including PU learning and self training to detect security-relevant Change Requests from 3GPP specifications, so as to understand security risks in the 3GPP ecosystem. Our research differ from these studies in both the research purpose and the techniques developed to serve the purpose. We aim at enabling automated reasoning about an event chain from the specifications for a conformance test, and leverage NLP/ML techniques not used in the prior studies, including semantic

role labeling, constituency parsing, data augmentation, etc.

NLP/ML techniques in network protocol analysis. The NLP/ML techniques have been leveraged to not only facilitate LTE security analysis, but also improve network protocol security [13, 24, 28]. Chen *et al.* [13] employ a suite of NLP techniques, including dependency parsing and word embedding, to recover the FSMs of different payment service for logic-vulnerability discovery. Jero *et al.* [24] propose a system that uses NLP techniques to extract network protocol rules for grammar-based fuzzing. Pacheco *et al.* [28] developed a method for automatically extracting FSMs from RFC documentation, which could then facilitate a network protocol security analysis, such as attack synthesis. Our research, unlike these prior studies, aims to establish causal relations across events to reason about an event chain for triggering the action in a conformance test and for observing its outcome.

7 Conclusion

Conformance testing is critical to security assurance. However, conformance test case generation faces a unique challenge that a testing system often cannot directly invoke condition event for a security requirement and/or directly observe occurrence of expected operation to be triggered by the condition event. In our research, we proposed an approach (*Contester*) to address it, which uses NLP/ML techniques to build EDG from 3GPP specifications and performs automated reasoning on graph to discover event chains, which once invoked leads to a chain reaction so that the testing system can either indirectly trigger the target event or indirectly observe the occurrence of the expected event. Such chains are then converted by Contester into conformance test procedures and our testing system execute them on the UE to evaluate its conformance to the 3GPP security requirements. After running Contester for 22 security requirements and testing 22 real-world UEs, our research discovered 197 security requirement violations, 190 of which had never been reported before.

8 Acknowledgment

We would like to thank the anonymous reviewers for their insightful comments, particularly our shepherd for the guidance for preparing the final version. The research is supported in part by NSF CNS-2154199.

References

[1] https://sites.google.com/view/contester.

[2] 3GPP. Specifications by Series. https://www.3gpp.org/specifications-technologies/specifications-by-series.

[3] 3GPP. Work Plan. https://portal.3gpp.org/#/55935-work-plan.

[4] 3GPP, TS 24.301, v16.8.0. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3, 2022.

[5] 3GPP, TS 33.401, v16.3.0. 3GPP System Architecture Evolution (SAE); Security architecture, 2022.

[6] 3GPP, TS 36.508, v16.8.0. Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Packet Core (EPC); Common test environments for User Equipment (UE) conformance testing, 2022.

[7] 3GPP, TS 36.523-1, v16.8.0. Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Packet Core (EPC); User Equipment (UE) conformance specification; Part 1: Protocol conformance specification, 2022.

[8] 3GPP, TSG-RAN Meeting #90-e. MCC Task Force 160 (TF160) Description and Terms of Reference for 2021, 2021.

[9] 3GPP, TSG-RAN Meeting #94-e. MCC Task Force 160 (TF160) Description and Terms of Reference for 2022, 2021.

[10] 3GPP Working Group 5. Meeting records. https://www.3gpp.org/ftp/tsg_ran/WG5_Test_ex-T1/.

[11] Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 205–216, 2012.

[12] Yi Chen, Di Tang, Yepeng Yao, Mingming Zha, XiaoFeng Wang, Xiaozhong Liu, Haixu Tang, and Dongfang Zhao. Seeing the forest for the trees: Understanding security hazards in the {3GPP} ecosystem through intelligent analysis on change requests. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 17–34, 2022.

[13] Yi Chen, Luyi Xing, Yue Qin, Xiaojing Liao, XiaoFeng Wang, Kai Chen, and Wei Zou. Devils in the guidance: Predicting logic vulnerabilities in payment syndication services through automated documentation analysis. In *USENIX security symposium*, pages 747–764, 2019.

[14] Yi Chen, Yepeng Yao, XiaoFeng Wang, Dandan Xu, Chang Yue, Xiaozhong Liu, Kai Chen, Haixu Tang, and Baoxu Liu. Bookworm game: Automatic discovery of lte vulnerabilities through documentation analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1197–1214. IEEE, 2021.

[15] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA, 2009.

[16] ETSI. TTCN-3. http://www.ttcn-3.org/.

[17] ETSI. ETSI Drafting Rules (EDR), 2022.

[18] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1535–1545, 2011.

[19] Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*, 2021.

[20] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.

[21] Google. Translation AI. https://cloud.google.com/translate.

[22] Syed Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. Lteinspector: A systematic approach for adversarial testing of 4g lte. In *Network and Distributed Systems Security (NDSS) Symposium 2018*, 2018.

[23] Syed Rafiul Hussain, Imtiaz Karim, Abdullah Al Ishtiaq, Omar Chowdhury, and Elisa Bertino. Noncompliance as deviant behavior: An automated black-box noncompliance checker for 4g lte cellular devices. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1082–1099, 2021.

[24] Samuel Jero, Maria Leonor Pacheco, Dan Goldwasser, and Cristina Nita-Rotaru. Leveraging textual specifications for grammar-based fuzzing of network protocols. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9478–9483, 2019.

[25] Imtiaz Karim, Syed Rafiul Hussain, and Elisa Bertino. Prochecker: An automated security and privacy analysis framework for 4g lte protocol implementations. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 773–785. IEEE, 2021.

[26] Eunsoo Kim, Dongkwan Kim, CheolJun Park, Insu Yun, and Yongdae Kim. Basespec: Comparative analysis of baseband software and cellular specifications for l3 protocols. In *NDSS*, 2021.

[27] Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. Touching the untouchables: Dynamic security analysis of the lte control plane. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1153–1168. IEEE, 2019.

[28] Maria Leonor Pacheco, Max von Hippel, Ben Weintraub, Dan Goldwasser, and Cristina Nita-Rotaru. Automated attack synthesis by extracting finite state machines from protocol specification documents. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 51–68. IEEE, 2022.

[29] C Park, Sangwook Bae, B Oh, Jiho Lee, Eunkyu Lee, Insu Yun, and Yongdae Kim. Doltest: In-depth downlink negative testing framework for lte devices. In *USENIX Security Symposium*, 2022.

[30] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*, 2020.

[31] David Rupprecht, Kai Jansen, and Christina Pöpper. Putting {LTE} security functions to the test: A framework to evaluate implementation correctness. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[32] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[33] Altaf Shaik, Ravishankar Borgaonkar, N Asokan, Valtteri Niemi, and Jean-Pierre Seifert. Practical attacks against privacy and availability in 4g/lte mobile communication systems. *arXiv preprint arXiv:1510.07563*, 2015.

[34] Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 12–21, 2007.

[35] Peng Shi and Jimmy Lin. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*, 2019.

[36] srsLTE, v21.04. https://www.srslte.com/.

[37] Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. Improving constituency parsing with span attention. *arXiv preprint arXiv:2010.07543*, 2020.

[38] Jan Tretmans. An overview of osi conformance testing. *Formal Methods Tools group University of Twente*, 2001.

[39] Ubiquitous Knowledge Processing Lab. Semantic Textual Similarity. https://www.sbert.net/docs/usage/semantic_textual_similarity.html#semantic-textual-similarity.

[40] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[41] Chuan Yu and Shuhui Chen. On effects of mobility management signalling based dos attacks against lte terminals. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2019.

APPENDIX

Table 2: Causal relation extraction rules.

Label	Leading term	Example
ARGM-TMP (Modifier-Temporal)	when, while, once, upon, after	Once the encryption of NAS messages has been started between the MME and the UE, the receiver shall discard the unciphered messages ...
ARGM-MNR (Modifier-Manner)	by	The UE initiates the attach procedure by sending an ATTACH REQUEST message to the MME.
ARGM-ADV (Modifier-Adverbial)	if	If the attach attempt counter is equal to 5, the UE shall delete any GUTI, TAI list ...

```
1:  procedure HOOK(hook_place)
2:      if hook_place is before processing inbound message then
3:          NOTIFY(msg)
4:          next_action ⟵ TEST CONTROLLER
5:          if next_action is to issue message then
6:              return
7:          if next_action is to wait then
8:              SLEEP(time)
9:              NOTIFY(time)
10:             goto 4
11:     end if
12:     if hook_place is before processing outbound message then
13:         msg ⟵ TEST CONTROLLER
14:         SEND_MSG(msg)
15:     end if
16:     exit
17: end procedure
```

Figure 11: The pseudocode of the HOOK function.

Table 3: Summary of security requirements.

S: The sources where the security requirement extracted (∗: source 1, •: source 2, ○: source 3 (see Section 4.2)) **TP**: # of test procedures; **P**: The purpose of a security requirement (*M: against the MITM attack, F: against the fake base station attack, R: against replay attack*). Note: The all security requirements are present in our website [1], here we list the ones that were tested in our TS.

ID	SECURITY REQUIREMENT	S	TP	P
S_1	In the present document, when the UE is required to delete an eKSI, the UE shall set the eKSI to the value "no key is available" and consider also the associated keys KASME or K'ASME, EPS NAS ciphering key and EPS NAS integrity key invalid (i.e. the EPS security context associated with the eKSI as no longer valid).	∗	19	*M*
S_2	When a partial native EPS security context is taken into use through a security mode control procedure, the MME and the UE shall delete the previously current EPS security context.	∗	1	*M*
S_3	When the MME and the UE create an EPS security context using null integrity and null ciphering algorithm during an attach procedure for emergency bearer services, or a tracking area updating procedure for a UE that has a PDN connection for emergency bearer services (see subclause 5.4.3.2), the MME and the UE shall delete the previous current EPS security context.	∗	1	*M*
S_4	The UE shall mark the EPS security context on the USIM or in the non-volatile memory as invalid when the UE initiates an attach procedure as described in subclause 5.5.1 or when the UE leaves state EMM-DEREGISTERED for any other state except EMM-NULL.	∗	13	*M*
S_5	Specifically, for a given EPS security context, a given NAS COUNT value shall be accepted at most one time and only if message integrity verifies correctly.	∗	10	*R*
S_6	Except the messages listed below, no NAS signalling messages shall be processed by the receiving EMM entity in the UE forwarded to the ESM entity, unless the network has established secure exchange of NAS messages for the NAS signalling connection: ...	∗	8	*F*
S_7	Once the secure exchange of NAS messages has been established, the receiving EMM or ESM entity in the UE shall not process any NAS signalling messages unless they have been successfully integrity checked by the NAS.	∗	10	*M*
S_8	If any NAS signalling message is received as not integrity protected even though the secure exchange of NAS messages has been established by the network, then the NAS shall discard this message.	∗	10	*M*
S_9	Once the encryption of NAS messages has been started between the MME and the UE, the receiver shall discard the unciphered NAS messages which shall have been ciphered according to the rules described in this specification.	∗	10	*M*
S_{10}	If the UE finds that the "separation bit" in the AMF field of AUTN supplied by the core network is 0, the UE shall send an AUTHENTICATION FAILURE message to the network, with the EMM cause #26 "non-EPS authentication unacceptable" (see subclause 6.1.1 in 3GPP TS 33.401 [19]).	○	1	*F*
S_{11}	If the UE finds the SQN (supplied by the core network in the AUTN parameter) to be out of range, the UE shall send an AUTHENTICATION FAILURE message to the network, with the EMM cause #21 "synch failure" and a re-synchronization token AUTS provided by the USIM (see 3GPP TS 33.102 [18]).	○	1	*F*
S_{12}	If HASH_{MME} and the locally calculated hash value are different, the UE shall include the complete ATTACH REQUEST or TRACKING AREA UPDATE REQUEST message which the UE had previously sent in the Replayed NAS message container IE of the SECURITY MODE COMPLETE message.	○	1	*M*
S_{13}	If an ATTACH REJECT message including timer T3402 value different from "deactivated", was received integrity protected, the UE shall apply this value until a new value is received with integrity protection or a new PLMN is selected. Otherwise, the default value of this timer is used.	•	2	*F*
S_{14}	If the UE receives an ATTACH REJECT, TRACKING AREA UPDATE REJECT or SERVICE REJECT message without integrity protection with EMM cause value #3, #6, #7, #8, #11, #12, #13, #14, #15, #31 or #35 before the network has established secure exchange of NAS messages for the NAS signalling connection, the UE shall start timer T3247 with a random value uniformly drawn from the range between 30 minutes and 60 minutes.	•	33	*F*
S_{15}	Upon receipt of an AUTHENTICATION REJECT message, if the message is received without integrity protection, the UE shall start timer T3247 with a random value uniformly drawn from the range between 30 minutes and 60 minutes, if the timer if not running.	•	2	*F*
S_{16}	The UE shall accept a SECURITY MODE COMMAND message indicating the "null integrity protection algorithm" EIA0 as the selected NAS integrity algorithm only if the message is received for a UE that has a PDN connection for emergency bearer services established, or a UE that is attached for access to RLOS, or a UE that is establishing a PDN connection for emergency bearer services or a UE that is requesting attach for access to RLOS.	•	1	*M*
S_{17}	If the ATTACH REJECT message with EMM cause #25 was received without integrity protection, then the UE shall discard the message.	•	1	*F*
S_{18}	The UE shall take the following actions depending on the EMM cause value received in the ATTACH REJECT message. #22 (Congestion); If the ATTACH REJECT message is not integrity protected, the UE shall start timer T3346 with a random value from the default range specified in 3GPP TS 24.008 [13].	•	2	*F*
S_{19}	If the TRACKING AREA UPDATE REJECT message with EMM cause #25 was received without integrity protection, then the UE shall discard the message.	•	2	*F*
S_{20}	The UE shall take the following actions depending on the EMM cause value received in the TRACKING AREA UPDATING REJECT message. #22 (Congestion); If the TRACKING AREA UPDATING REJECT message is not integrity protected, the UE shall start timer T3346 with a random value from the default range specified in 3GPP TS 24.008 [13].	•	2	*F*
S_{21}	If the SERVICE REJECT message with EMM cause #25 or #31 was received without integrity protection, then the UE shall discard the message.	•	4	*F*
S_{22}	The UE shall take the following actions depending on the EMM cause value received in the SERVICE REJECT message. #22 (Congestion); If the SERVICE REJECT message is not integrity protected, the UE shall start timer T3346 with a random value from the default range specified in 3GPP TS 24.008 [13].	•	2	*F*

Table 4: Summary of tested devices and security requirement violations.

Last update: MM/DD/YY, -: no public information. Note: The security requirement violations never reported before are highlighted in red.

#	Name	Device vendor	Baseband vendor	Chipset model	Fireware version	Last update	Violated security requirement
1	iPhone 13 Pro Max	Apple	Qualcomm	X60M	1.59.03	03/15/2022	S_6, S_{10}, S_{11}, S_{15}, S_{18}, S_{20}
2	iPhone 13	Apple	Qualcomm	X60M	15.5(19F77)	07/01/2022	S_6, S_{10}, S_{11}, S_{15}, S_{18}, S_{20}, S_{22}
3	iPhone 12 Pro	Apple	Qualcomm	X55M	15.5(19F77)	07/01/2022	S_6, S_{10}, S_{11}, S_{15}, S_{18}, S_{20}, S_{22}
4	iPhone 11	Apple	Intel	XMM 7660	3.04.01	07/21/2022	S_{11}, S_{18}, S_{20}
5	iPhone 6	Apple	Qualcomm	MDM9625	7.80.04	07/15/2021	S_1, S_4, S_5, S_{10}, S_{11}, S_{12}, S_{15}, S_{18}, S_{20}
6	Pixel 5a	Google	Qualcomm	Snapdragon 765G	b9-0.4-7617867	10/05/2021	S_1, S_4, S_5, S_6, S_{10}, S_{11}, S_{15}, S_{18}, S_{20}, S_{22}
7	Pixel 3	Google	Qualcomm	Snapdragon 845	g845-00194-210812-B-7635520	03/01/2022	S_1, S_4, S_5, S_{10}, S_{11}, S_{12}, S_{15}, S_{18}, S_{20}, S_{22}
8	Pixel 2	Google	Qualcomm	Snapdragon 835	g8998-00034-2006052136	10/05/2020	S_1, S_4, S_5, S_6, S_{10}, S_{11}, S_{12}, S_{18}, S_{20}, S_{22}
9	Honor 30S	Honor	Hisilicon	Kirin 820	21C93B377S000C000,21C93B377S000C000	03/01/2022	S_6, S_{11}, S_{15}, S_{18}, S_{20}, S_{22}
10	Honor 6X	Honor	Hisilicon	Kirin 655	21C60B269S007C000,21C60B269S007C000	01/01/2021	S_6, S_7, S_8, S_9, S_{11}, S_{12}, S_{15}, S_{18}, S_{20}, S_{22}
11	Honor Play20	Honor	Unisoc	T610	FM_BASE_19C_W22.04.3	04/01/2022	S_1, S_4, S_5, S_{11}, S_{12}, S_{14}, S_{15}, S_{18}, S_{20}, S_{22}
12	Redmi K30i	Xiaomi	Qualcomm	Snapdragon 756G	MPSS.HI.2.0.c7-00269-1213_0042_eec7a04f5	12/01/2021	S_1, S_4, S_5, S_{10}, S_{11}, S_{15}, S_{18}, S_{20}
13	Redmi Note2	Xiaomi	Mediatek	Helio X10	MOLY.LR9.W1423.MD.LWTG.MP.V24.P58	12/01/2016	S_8, S_9, S_{11}, S_{12}, S_{15}, S_{18}, S_{20}
14	Xiaomi MIX 2S	Xiaomi	Qualcomm	Snapdragon 845	4.0.c2.6-00335-0220_1946_40a1464	12/01/2020	S_1, S_4, S_5, S_{10}, S_{11}, S_{12}, S_{15}, S_{18}, S_{20}, S_{22}
15	Galaxy A71 5G	Samsung	Samsung	Exynos 980	A7160ZCU5CUL9	01/01/2022	S_2, S_4, S_5, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{20}, S_{22}
16	Galaxy S10	Samsung	Qualcomm	Snapdragon 855	G9730ZCU6GVA7	01/01/2022	S_4, S_6, S_{10}, S_{11}, S_{18}, S_{20}, S_{22}
17	Huawei E3276 USB Dongle	Huawei	-	-	-	-	S_3, S_5, S_{11}, S_{12}, S_{16}, S_{18}, S_{20}
18	Nexus 6P	Huawei	Qualcomm	Snapdragon 810	angler-03.61	02/01/2016	S_1, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{18}, S_{20}, S_{22}
19	Nexus 6	Motorola	Qualcomm	Snapdragon 805	MDM9625_104670.31.05.51R	02/01/2016	S_1, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{15}, S_{18}, S_{20}, S_{22}
20	Nexus 4	LG	Qualcomm	Snapdragon S4 Pro	M9615A-CEFWMAZM-2.0.1700.33	07/08/2015	S_6, S_7, S_8, S_9, S_{11}, S_{12}, S_{15}, S_{16}, S_{18}, S_{20}
21	Realme GT	OPPO	Qualcomm	Snapdragon 888	Q_V1_P14,Q_V1_P14	04/01/2022	S_4, S_6, S_{10}, S_{11}, S_{15}, S_{18}, S_{20}, S_{22}
22	SINELINK MF782 USB Dongle	SineLink	-	-	-	-	S_1, S_2, S_4, S_5, S_6, S_7, S_8, S_9, S_{11}, S_{12}, S_{18}, S_{20}