5GPT: 5G Vulnerability Detection by Combining Zero-Shot Capabilities of GPT-4 with Domain Aware Strategies Through Prompt Engineering

Asif Shahriar*, Syed Jarullah Hisham*, K.M. Asifur Rahman*,
Md. Ruhan Islam*, Md. Shohrab Hossain*, Ren-Hung Hwang†, Ying-Dar Lin‡
*Department of CSE, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
†College of AI, National Yang Ming Chiao Tung University, Tainan, Taiwan
‡Department of CS, National Chiao Tung University, Taipei, Taiwan
Email: asif.asr11@gmail.com, syedjarullah@gmail.com, asifurndc8030@gmail.com,
thisisruhan@gmail.com, mshohrabhossain@cse.buet.ac.bd, rhhwang@nycu.edu.tw, ydlin@cs.nctu.edu.tw

***Abstract*—Identifying vulnerabilities in complex 5G network protocols is a challenging task. Manual analysis is time-consuming and often inadequate. Modern ML and NLP methods, though effective, are resource-intensive and struggle to find implicit vulnerabilities. In this research, we utilize GPT-4's advanced language understanding to detect vulnerabilities directly from 5G specifications. To assess GPT-4's fundamental capabilities in this domain, we first adopt a zero-shot approach that relies solely on the specification text without external guidance. For detecting more sophisticated vulnerabilities that require deep contextual understanding, we introduce a novel domain-aware strategy, where we explicitly teach GPT-4 about security properties and hazard indicators from related works using few-shot learning. We further employ chain-of-thought prompting to guide the model through structured reasoning steps to identify violations or exploitations that may lead to vulnerabilities. A two-tier filtering process ensures that only promising test-cases are retained. Our method has identified 47 potential vulnerabilities in 5G mobility management procedures, including 27 previously unreported issues, and generated corresponding test-cases. Simulating 14 of them, we have found 9 vulnerabilities, five of which are new. The zero-shot approach is effective in detecting procedural and validation flaws, while the domain-aware method excels in finding protocol violations and advanced attack scenarios. These findings validate our methodology and demonstrate its strength in discovering both known and novel vulnerabilities in 5G protocols.**

***Index Terms*—5G, vulnerability detection, LLM, network security, AI for security, prompt engineering, few-shot learning.**

I. Introduction

5G standards, developed by the 3rd Generation Partnership Project (3GPP) [1], incorporate complex technologies and protocols that, while innovative, also open up new vulnerabilities. These vulnerabilities arise from the increased network complexity, the integration of legacy and new technologies for backward compatibility, and the implementation of advanced features such as network slicing and virtualization. Finding these vulnerabilities is particularly challenging due to the complexity and vastness of the specifications. Various methods exist for detecting vulnerabilities in LTE and 5G protocols, including formal verification [2]–[6], adversarial testing framework [2], and negative testing framework [7]. However, these methods require extensive manual analysis and expert knowledge, which limit their scalability. Recent advances in the field of Machine Learning (ML) and Natural Language Processing (NLP) have enabled more sophisticated analysis of cellular protocols [8]–[12]. ML-based fuzzing techniques [9], [13], [14] can uncover security flaws that may not be apparent through conventional testing methods. However, custom NLP methods often struggle with the technical language used in these documents, and ML models require substantial resources and high-quality training data. These limitations highlight the need for scalable models that can address the complexity of finding vulnerabilities in 5G specifications.

State-of-the-art LLMs like GPT-4 [15] are increasingly being used in security testing [16]–[24], but their application is still mostly limited to code-based analysis rather than natural language based technical documents. These models, trained on diverse datasets, excel in understanding context and nuances in unstructured text; and identifying subtle cues for vulnerabilities, regardless of sentence structure. Unlike traditional ML models, GPT-4 can adapt to new data without extensive fine-tuning, and handle diverse linguistic styles and technical jargon found in protocol documentation. These capabilities make GPT-4 a strong candidate for vulnerability detection in cellular protocols such as 5G.

In this research, we utilize GPT-4's advanced natural language understanding capabilities to identify procedural or logical flaws, implementation-related issues, complex protocol violations, and cross-procedural vulnerabilities, directly from 5G protocol specifications. We employ two methods for vulnerability detection: a *zero-shot* approach, which exclusively uses GPT-4's pretrained knowledge, and a *domain-aware* strategy for enhancing GPT-4's contextual understanding and adversarial reasoning by integrating structured security knowledge, including security properties, hazard indicators, and reasoning patterns derived from prior research on cellular protocol vulnerabilities [3], [6], [7], [11], [25]–[27]. Using the zero-shot approach, we have identified 24 potential vulnerabilities, with 12 being new findings. The domain-aware

approach has revealed 23 potential vulnerabilities, of which 15 are novel. Altogether, we have found 20 previously reported vulnerabilities and 27 new issues. These vulnerabilities include authentication issues, security flaws, implementation flaws, ambiguities in standards, resource mismanagement, phishing attacks, and replay attacks. The identification of existing vulnerabilities establishes the credibility of our approach, while discovering new vulnerabilities demonstrates the capability to uncover novel security risks. We have simulated 14 of these vulnerabilities using Open5GS and UERANSIM, and confirmed the presence of nine, including five new.

The major contributions of our work are as follows.

- *Novel vulnerability detection method.* By combining GPT-4's advanced contextual understanding with domain-specific insights from related works through effective prompt engineering, we present an innovative black box testing approach for vulnerability detection in 5G.
- *A scalable, lightweight solution.* Our method does not require large-scale dataset or costly fine-tuning, making it scalable across different protocols and applications where white-box fine-tuning is not an option.
- *Innovative task-specific prompt engineering.* We provide detailed illustrations of task-specific prompt engineering, showing how domain context, few-shot examples, and chain-of-thought reasoning can be applied effectively to sophisticated tasks such as detecting 5G vulnerabilities.
- *Software simulation.* We validate our findings through testing in a simulation environment. We also discuss an array of simulation techniques, offering an alternative method when hardware testing is not feasible.

The rest of the paper is organized as follows. Section II provides background knowledge on our research. In section III we discuss existing works on detecting vulnerabilities in cellular protocols. Section IV explains our methodology for vulnerability detection and test-case generation, test-case filtering, and simulation. In section V we present the findings from our experiment in detail and compare our findings against other related works and white-box fine-tuning approaches. In section VI we discuss the effectiveness of our approach and some limitations. Section VII concludes this paper and provides direction on future work.

II. Background

In this section we discuss the basic concepts of 5G network architecture, NAS layer procedures, and prompt engineering techniques relevant to our work.

A. 5G Network Architecture

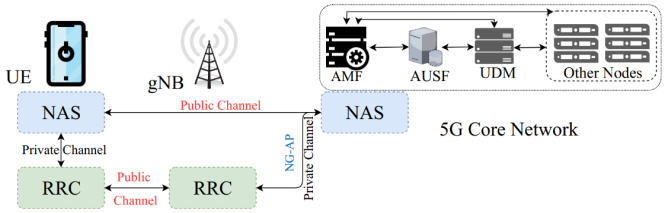


Fig. 1: Simplified 5G Architecture [6]

The 5G network architecture comprises of three main components: User Equipment (UE), 5G Radio Access Network (5G-RAN), and 5G Core Network (5G-CN), as shown in Fig. 1.

UE: The UE includes devices like smartphones, tablets, and IoT devices that connect to the 5G network. Each UE contains a non-access stratum (NAS) for managing connection and mobility states with the 5G-CN.

5G-RAN: The 5G-RAN facilitates wireless communication between UEs and the core network. Its primary component is the gNodeB (gNB), which serves as the base station responsible for managing the radio interface, resources, and connectivity.

5G-CN: The 5G core network adopts a service-based architecture to allow network functions to offer services to each other. Key components include the Access and Mobility Management Function (AMF), which manages user access, mobility, and authentication; the Authentication Server Function (AUSF), responsible for handling the authentication of user devices; the Session Management Function (SMF), which manages session establishment, IP address allocation, and other service policies; and the Unified Data Management (UDM), which manages user subscription data and profiles.

The Non-Access Stratum (NAS) layer manages signaling and communication between UEs and the core network for session management, mobility management, and security. The Radio Resource Control (RRC) layer oversees radio bearer configurations, admission control, mobility measurements, and dynamic resource allocation over secure channels.

B. NAS Layer Procedures

Here we briefly discuss some NAS layer procedures that are relevant to our work.

Authentication and Key Agreement (AKA): AUSF generates an authentication token (AUTN). A long-term secret key (K) that is shared between the UE and the network. Along with the AUTN, a random number (RAND) and an expected response (XRES) are also generated. The network sends AUTN and RAND to the UE. The UE first validates the received AUTN and then computes a response (RES) based on the received RAND and the long-term key (K). The UE sends the RES back to the network. The network then compares this received RES with the expected response (XRES) it generated initially. If RES matches XRES, it confirms the UE's identity and the authentication is considered successful.

Registration: The UE initiates a connection with the 5G network by sending a registration request to the AMF, which includes its Subscription Permanent Identifier (SUPI). The AMF assigns a 5G-Globally Unique Temporary Identifier (GUTI) and updates the UE's location in the UDM for mobility management.

Deregistration: Deregistration can be triggered by either the UE (e.g., when turned off) or the network. The UE sends a Deregistration Request to the AMF, which updates the network's records and releases associated resources. If the UE is inactive, the network may also initiate deregistration to free up resources and maintain efficiency.

Security Mode Control: After the UE is authenticated, the AMF initiates the Security Mode Command, specifying the algorithms for encryption (e.g., 128-NEA2) and integrity protection (e.g., 128-NIA2). The UE responds with a Security Mode Complete message if it successfully configures the security settings. This procedure ensures that all subsequent NAS messages are protected against eavesdropping and tampering.

Configuration Update: To update the UE with new configuration settings such as Tracking Area Identity (TAI) or network slice information, the AMF sends a Configuration Update Command to the UE, which may include updated parameters like TAI lists or changes in session management parameters. The UE applies these updates and responds with a Configuration Update Complete message. This keeps the UE synchronized with the network's parameters, ensuring efficient mobility and service management.

C. Prompt Engineering Techniques

Prompt engineering means creating prompts that are specifically designed to guide an LLM towards generating the desired output. It improves model performance by providing clear instructions, context, and constraints. Recent works [28]–[31] have introduced various techniques to improve LLM responses for specialized tasks.

Zero-Shot Approach: This approach prompts the LLM to perform a task without any examples, relying solely on its pretrained knowledge and reasoning capabilities [28]. It is particularly useful for evaluating the model's baseline understanding of new domains or instructions.

Few-Shot Learning: Brown et al. [29] demonstrated the effectiveness of providing the LLM with a handful of input-output examples, to help it understand the underlying structure and intent of the task and the pattern of expected response. It is particularly helpful in non-trivial domain specific tasks.

Chain of Thought Prompting: Introduced by Wei et al. [30], this technique encourages the LLM to reveal its intermediate reasoning steps before producing a final answer. This decomposition allows the model to perform multi-step logical inference, improving performance on tasks requiring complex reasoning or rule-based interpretation.

Majority Voting: Wang et al. [31] demonstrated that selecting the most frequently occurring answer among sampled reasoning paths significantly improves accuracy. In essence, recurrence indicates correctness in model-generated outputs.

III. Related Works

Related works in 5G vulnerability detection can be categorized into traditional methods, machine learning and NLP-based models, fuzzing techniques, and LLM-based security testing. While these methods have been effective to varying degrees, they have certain limitations. In this section, we discuss these approaches, their limitations, and how we address them. Table I provides a summary.

Formal verification: These approaches [2]–[4], [6] extract FSMs from protocol specifications for rigorous analysis to detect vulnerabilities. These methods are time consuming, require extensive manual effort, and have limited scalability. They often depend on pre-defined attack vectors and miss dynamic threats. In comparison, 5GPT dynamically analyzes protocol documentation using domain-aware prompt engineering techniques, which is faster, scalable, adaptable, and eliminates the need for extensive human effort.

ML and NLP-based models: ML-based models [8]–[10] are trained on large datasets to identify patterns, anomalies, and new attack vectors in cellular protocols. Chen et al. [11] use ML and NLP to scan LTE documentation for *Hazard Indicators (HIs)*, while Pacheco et al. [27] and Ishtiaq et al. [12] use NLP to automatically generate FSMs from RFC documents and 5G specifications. These models require large labeled datasets, frequent retraining to adapt to evolving threats, and high compute power. Additionally, custom NLP models often struggle with diverse technical jargon found in technical documents like LTE and 5G specifications. In contrast, 5GPT operates purely through domain-aware prompt engineering, allowing it to dynamically adapt to new security contexts without costly training or fine-tuning.

Fuzz testing: Fuzzing techniques [9], [13], [14], [26], [32] generate and test numerous inputs to uncover unexpected flaws, which is powerful but inefficient. It requires massive input mutations to find vulnerabilities, leading to high computational costs but low precision. 5GPT overcomes this limitation by using domain-aware reasoning to prioritize high-risk test cases, which is more precise than blindly mutating inputs and significantly reduces computational overhead.

LLM-based security testing: Recent works have explored the use of LLMs in security testing. Zhang et al. [16] demonstrated ChatGPT's ability to generate security tests for applications primarily through code-level analysis. Wang et al. [17] provide a broader survey of LLM applications in software security testing, covering unit test generation [20], fuzz testing [19], penetration testing [21], bug analysis [22], automated debugging and program repair [23], [24]. However, most of these works focus on code-level security testing, with only two works [18], [19] exploring natural language-based security analysis. LLMs applied to source code benefit from its structured nature, where vulnerabilities can be explicitly traced to specific functions, unhandled conditions, or faulty logic. In contrast, our work, 5GPT, detects vulnerabilities directly from the complex and highly technical 5G specifications, where security flaws emerge from semantic ambiguities, procedural inconsistencies, and implicit security requirements rather than explicit code-level defects.

White-box testing: White-box fine-tuning approaches have shown impressive results in vulnerability detection [17], [22]–[24], especially when there is access to source code, execution traces, or detailed bug reports. However, our black-box approach is designed for situations where that is not an option—such as proprietary systems, closed-source telecoms infrastructure, or protocol-level security assessments. 5GPT is able to uncover meaningful vulnerabilities without access to proprietary code or extensive fine-tuning, making it more scalable than white-box models and a practical choice for security evaluation of cellular protocols like 5G.

TABLE I: Comparison between Existing Vulnerability Detection Approaches and 5GPT

Categories	Related Works	Identified Limitations	How 5GPT Addresses These Limitations
Traditional and Formal Methods	[2]–[4], [6], [7]	Labor-intensive and time-consuming; requires expert knowledge; limited scalability; often predefined attack vectors, missing dynamic threats	Automated and scalable analysis; reduces dependency on manual effort and domain expertise; adapts dynamically to a wider range of vulnerabilities without predefined constraints
ML and NLP-based Models	[8]–[12], [27]	Scarcity of labeled data for training; high compute requirement; difficulty of interpreting results; struggle with diverse technical jargon; need frequent updates	No need for huge dataset or training; better technical understanding of GPT-4; CoT reasoning for better interpretation; dynamically adapts to new data without reprogramming
Fuzzing Techniques	[9], [13], [14], [26]	Resource-intensive but inefficient; difficulty in modeling complex and nuanced 5G protocols precisely	Domain-aware reasoning for identifying precise high-risk scenarios rather than blind input mutation
LLM-based Techniques	[16]–[24]	Requires implementation level details; mostly white box; mainly focused on test case generation from code level analysis rather than natural language analysis	Detects vulnerability directly from protocol specifications, without source code access; black box testing; more scalable and accessible; empirical validation

IV. Proposed Approach

In this work we have used GPT-4 for automated vulnerability detection from 5G protocols, integrating the natural language understanding capabilities of LLMs with structured prompt engineering. The end-to-end workflow can be divided into three key steps. First, we have used GPT-4 for identifying potential vulnerabilities from the 5G specifications and generating comprehensive test-cases. Afterwards, we have applied a filtering process to eliminate the unreliable or inconsistent vulnerability suggestions while retaining those that demonstrate strong logical reasoning. Finally, the identified potential vulnerabilities are tested in a simulation environment to confirm or refute their existence. In this section we describe each process in detail.

A. Vulnerability Detection & Test-case Generation

This is the key step of our approach, where we identify potential vulnerabilities from input specification. We have provided GPT-4 with the 5G-NAS specification document, *3GPP TS 24.501 version 17.9.0 Release 17* [1], focusing on the *Elementary procedures for 5GS mobility management*, including *authentication*, *security mode control*, *identification*, *configuration update*, and *network-slicing*. This section has around 350 pages, which we have divided into smaller subsections. In each iteration, we have extracted the relevant specification subsection and provided it to GPT-4 to identify potential vulnerabilities. For each vulnerability detected by GPT-4, we have also instructed it to generate a concrete test-case in a predefined format. We have used two distinct approaches for detecting vulnerabilities: a zero-shot approach and a domain-aware strategy.

Predefined Test-case Format

Actors: Entities involved in the scenario (e.g., UE, AMF, gNB)
State of Actors: Initial condition of the actors before the event occurs
Event: The main action or sequence of actions being tested (e.g., message exchange, n/w request, state transition, handover)

- **Message:** Request/command sent during the event
- **Direction:** Uplink (UE → AMF) or downlink (AMF → UE)
- **Parameters:** Relevant fields or attributes of the message

Expected behaviour: Normal outcome of the event if no vulnerability
Vulnerability: The potential vulnerability identified, along with an explanation of why it is considered a vulnerability and how it may occur

Zero-Shot Approach: In this approach we have provided GPT-4 with only the document and the test-case format, as shown in Fig. 2. We have not provided any further guidance or details about the document. This kind of prompting technique is called zero-shot prompting [28]. The goal of this approach is to understand the basic capabilities of GPT-4 in detecting vulnerabilities, in the absence of any other domain-specific insight or information. First we have asked GPT-4 to read the document and provide a summary. This is done to make sure that GPT-4's NLP abilities are sufficient for understanding the intricate details of the complex 5G specifications. Afterwards, we have asked GPT-4 to find vulnerabilities in the 5G specifications and to generate test-cases to confirm or reject the presence of the corresponding vulnerability, based on the provided format. A condensed version of the prompt used for this task is shown below.

Zero-shot prompt example

System: You are a security analyst trained to detect vulnerabilities in 5G protocol specifications. Your task is to identify any potential vulnerabilities based on the specification and generate structured test-cases accordingly.
User: Analyze the following 5G specification, identify all possible vulnerabilities, and generate structured test-cases.
Input:

- 5G Specification: {5G_spec_snippet}
- Test-Case Format: {test_case_format}

Task:

- Carefully read and analyze the provided 5G specification section.
- Identify and describe any potential vulnerabilities indicated by the procedure.
- For each identified vulnerability, generate a detailed test-case strictly following the provided test-case format.
- Ensure test-cases are specific, comprehensive, and actionable.

Output: Return a structured list of test-cases adhering to the given format.

The zero-shot approach offers a practical way to utilize the model's extensive pretrained knowledge without requiring additional fine-tuning or curated example sets [33]. An LLM like GPT-4 has already seen and internalized vast amounts of knowledge regarding general security, protocol, and telecommunications during its initial training, including how vulnerabilities are typically classified, what factors contribute to severity, how exploitation scenarios might look, and so forth. As a result, GPT-4 has a generalized understanding of security issues from the large corpus it was trained on [16]–[18], [22], [23]. When GPT-4 reads the 5G protocol specification, it looks for patterns that typically correlate with known classes of security problems. GPT-4 can also reason about attack feasibility, based on attack prerequisites, plausibility of execution steps, and lack of security mechanisms in favour of the attack. While

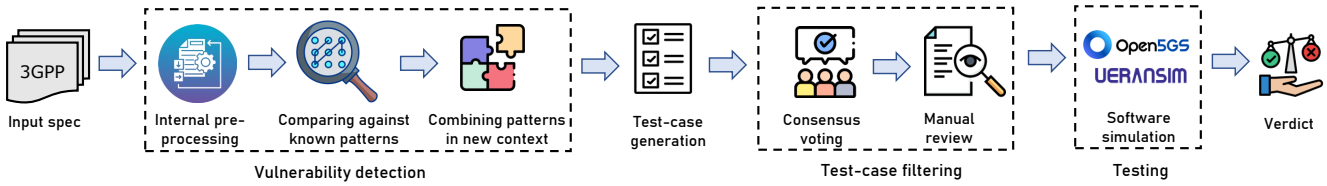


Fig. 2: Zero-shot approach

this is not empirical testing, it helps classify vulnerabilities as low, medium, or high risk by estimating how easy they are to exploit and what their theoretical impact could be. Moreover, through its generative capabilities, the model can also combine these patterns in a new context, allowing it to hypothesize novel vulnerabilities or variations on existing attacks that may not have been explicitly documented before. This can potentially result in some false positives, which is why these outputs should be carefully validated.

Domain-Aware Approach: The goal of this approach is to identify sophisticated vulnerabilities that require contextual knowledge. To achieve this, we have incorporated insights prior works [2], [3], [6], [7], [11], [26], [27] within this domain, as Fig. 3 highlights. Specifically, our domain-aware framework utilizes two well-established techniques. The first involves defining a set of security properties (SPs) – violations of which can be linked to possible vulnerabilities. The second technique leverages hazard indicators (HIs), which are key phrases or conditions that have historically been associated with security risks. To implement this, first we use few-shot learning to *'teach'* GPT-4 about SPs and identify HIs through relevant examples, so that it can extract and identify them from 5G specifications. Afterwards, we use CoT reasoning to analyze whether a given SP is violated in the specification, or if an HI can be exploited in adversarial context – thus linking them to possible vulnerabilities.

Using SPEC5G as specification: The SPEC5G dataset [25] contains expert-annotated security-related sentences from the 5G specification. We have extracted these security-related sentences and grouped them based on contextual similarity to maintain the integrity of the information as presented in the original specification. We have used the *SentenceTransformer* library with the `all-MiniLM-L6-v2` model to compute dense vector embeddings for each sentence and measure cosine similarity between consecutive sentence pairs only. A high similarity score indicates a strong semantic relationship, suggesting that the sentences belong to the same context. We have empirically established a similarity threshold of 0.7 to distinguish between related and unrelated contexts. The grouping process is performed in a sequential, pairwise manner: at each step, we compare sentence pairs (S_{i-1}, S_i). If the similarity between them exceeds the threshold, S_i is appended to the current context group. Otherwise, the current group is closed, and S_i begins a new group. In this way, we propagate context through pairwise chaining of adjacent sentences while avoiding duplication or distortion from group-level embedding aggregation.

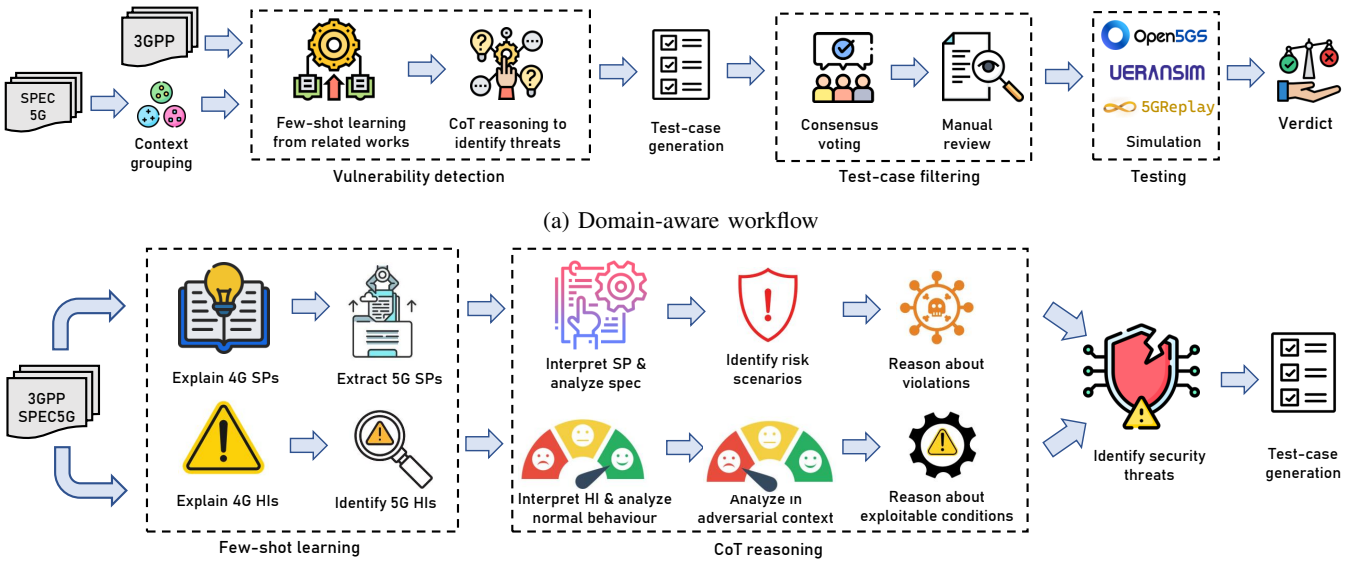
After this preprocessing, SPEC5G serves as a security-oriented subset of the original 5G specification developed by 3GPP, that can be used in its place. Crucially, the size of this dataset is such that it can fit within GPT-4's context window, eliminating the need for segmenting it into smaller parts. During experimentation, we have used both the original specification and the SPEC5G dataset independently for vulnerability detection and compared the results.

TABLE II: 5G Security Properties and Hazard Indicators

Security Property	Hazard Indicator
Protection of Sensitive Subscriber Information	Using stale ngKSI values
UE Identity confidentiality	Resynchronization on same RAND without re-authentication
Seamless security state transitions	Initiation of EAP-AKA' without valid SNN matching
Protection against replay attacks	Processing AUTHENTICATION REJECT without active timers
Ensuring integrity and confidentiality of NAS signalling	Non-delivery of NAS PDU due to handover
Proper handling of invalid or malformed messages	Inclusion of PEI instead of SUCI for emergency services
Network slice isolation	Using previously assigned 5G-GUTIs from foreign PLMNs
Protection of user plane integrity	Aborting registration upon request before completion
Management of temporary identities	Policy-driven rejection without standard cause
Protection against cross-layer attacks	Using null integrity and ciphering algorithms (5G-IA0 / 5G-EA0)
Resilience against service downgrade attacks	Transmission failure handling left to UE implementation

Defining 5G Security Properties: Security properties refer to specific conditions or rules that a system must satisfy to ensure its security. Recent works [2], [6], [26] have shown the effectiveness of defining security properties from protocol specifications that can be used to test LTE and 5G protocols' adherence to security and privacy standards. We improve on these manual approaches by using GPT-4 to autonomously define security properties from 5G specifications, reducing reliance on human expertise. An issue here is that while GPT-4 has broad security knowledge, it does not inherently recognize which properties are critical for assessing protocol security. This is why we use few-shot learning, where we provide GPT-4 with examples of security properties identified in LTE [2], [26]. By learning from these examples, GPT-4 is able to generate relevant security properties for 5G that align with

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2025.3586480



(a) Domain-aware workflow

(b) Domain-aware vulnerability detection process. SP: security property, HI: hazard indicator.

Fig. 3: Domain-aware approach

established expert-driven methodologies. Some of the security properties are mentioned in Table II.

Detecting property violations: After generating the security properties, our model must determine whether there are scenarios in the specification where these properties may be violated. For this assessment, GPT-4 needs to reason about how specific behaviors, procedures, or omissions in the specification could lead to concrete violations of these properties. This is achieved through chain-of-thought (CoT) prompting, in which GPT-4 is guided to break down its reasoning process into a sequence of logical steps. Without CoT, GPT-4 is forced to make a single-shot judgment, often relying on shallow pattern matching using keywords rather than structured analysis. In contrast, CoT prompting enhances GPT-4's reasoning capabilities by enabling it to examine the specified behavior in relation to the intended security property, consider relevant security mechanisms, identify possible attack vectors, and evaluate the conditions under which a violation could occur—much like a human security analyst tracing through system interactions and edge cases. This structured reasoning is especially helpful in detecting violations that are not explicitly stated in the specification but instead arise from subtle procedural or multi-entity interactions (e.g., *mishandling security context during state transitions or inter-system handovers*). Some of the HIs are mentioned in Table II. To illustrate this process, we have provide an example CoT prompt. We have also provided a sample GPT-4 response to CoT prompting, given the specification snippet *"During inter-system handover, the UE continues using its existing NAS security context. Re-establishment of the security context is not mandatory unless explicitly triggered"* and security property *"Seamless Security State Transitions — The integrity and confidentiality of signaling and user data must be preserved during transitions."*

CoT prompt example to detect property violation

System: You are a 5G security analyst. Your job is to identify whether any of the listed security properties are violated by a behavior described in 5G spec. Think step-by-step like a security expert: understand each property, analyze the behavior described in the specification, and determine whether any violations occur.
User: Analyze the following 5G spec snippet against the list of security properties. For each property, determine whether a violation occurs, explain why or why not, and conclude whether it links to a possible vulnerability.
Input:

- **5G Specification:** {5G_spec_snippet}
- **Security Properties:** {security_properties_list}

Reasoning: For each security property:

1) Explain the intended behavior according to the property.
2) Analyze how the specification behavior compares to this intended behavior.
3) Identify any circumstances under which the behavior could violate the property (e.g., replay, downgrade, desynchronization).
4) Conclude whether the property is upheld or violated.

Conclusion: For each property, clearly state whether it can be violated with justification. If yes, indicate whether any vulnerabilities may arise in real-world scenario.

GPT-4 response to CoT prompting (condensed)

1) Expected Behavior: Security context should be validated during inter-system handover to ensure confidentiality and integrity.
2) Specification Behavior: Allows continued use of existing NAS context without mandatory re-validation unless explicitly triggered.
3) Risk: Potential vulnerability if a stale or compromised context is reused—may enable replay or downgrade attacks.
4) Analysis: Lack of enforced validation may allow context desynchronization or reuse across RATs.

Conclusion: The specification likely violates the Seamless Security State Transitions property by not requiring security context validation during handovers, introducing a vulnerability window.

Finding Hazard Indicators: Hazard indicators (HIs) are phrases or conditions that suggest a potential exploitation point, such as unexpected behaviors triggered by specific events or operations that should be aborted under certain conditions. *Bookworm Game* [11] uses NLP and machine learning to scan LTE documentation for explicit hazard indicators, but

it often struggles to capture implicit or context-dependent cues. In contrast, GPT-4's advanced language understanding enables it to interpret nuanced and implicit information across multiple sentences, making it capable of detecting vulnerabilities that tools like *Atomic* might miss. To incorporate this capability into our workflow, first we have provided GPT-4 with examples of how specific HIs led to security flaws in LTE. Through few-shot prompting, GPT-4 learns how certain operations can introduce security issues in adversarial context and is able to identify relevant HIs from 5G specification. Following is an example few-shot prompt used for this task.

Few-shot prompt example for Hazard Indicators

System: You are a security analyst trained to detect hazard indicators (HIs) in 5G protocol specifications. A hazard indicator is an action, trigger, or condition that may, under certain circumstances, introduce a potential security risk. Your task is to identify all potential hazard indicators in the given spec based on the surrounding context.
User: Learn from the following examples to identify HIs and explain their significance using context from the 5G specification.
Example 1:

- Hazard Indicator: Modify Bearer Context
- 5G Specification: {spec_excerpt_1}
- Reasoning: Modification after state change may cause session desynchronization if not validated.

Example 2:

- Hazard Indicator: Abort NR Procedure
- 5G Specification: {spec_excerpt_2}
- Aborting without authentication recheck could allow bypassing protocol state enforcement.

Input: 5G Specification: {5G_spec_section}
Task: Identify all hazard indicators mentioned or implied in the given specification and justify their relevance.

Identifying HI exploitation: After finding a hazard indicator (HI), GPT-4 is guided through CoT prompting to assess whether the HI can be exploited in an adversarial context. First GPT-4 interprets the specification to determine the expected or normal behavior surrounding the HI. It then reasons about how this behavior might be manipulated or triggered under abnormal or malicious conditions. Specifically, it analyzes whether there exist exploitable conditions under which an attacker could use the HI to subvert expected protocol behavior – for example, by inducing premature state transitions, bypassing integrity checks, or misusing abort/retry mechanisms. If GPT-4 concludes that such exploitation is plausible in a real-world scenario, the HI is linked to a concrete vulnerability, and a corresponding test-case is generated.

B. Test-case Filtering

After the test-cases are generated, they undergo a filtering process so that only the promising test-cases (the ones that are most likely to highlight a vulnerability) remain. This process is two-tiered - *consensus-based filtering* and *manual filtering*.

Consensus-based Filtering*:* It is similar to majority voting, where the recurrence of test-cases across various iterations is regarded as an indicator of their potential validity [31]. GPT-4's reasoning is based on statistical association and its own assessment of severity and relevance. So, if a case appears in multiple iterations, it implies that GPT-4 considers it to be a high-severity issue; and if it does not, it suggests that GPT-4 does not internally prioritize it as an important security concern. We consider these outliers to be erroneous suggestion resulting from GPT-4 misunderstanding a part of the specification due to ambiguity or associating a concept incorrectly with a known vulnerability pattern, which is why they are filtered out. Additionally, if a vulnerability recurs but with minor differences in wording, details, or test-case formulation, the filtering process extracts the core overlapping issue and refines the test case accordingly. For example, we got multiple versions of *replay attack* but with different scenarios (SMC, authentication, SCTP), which were merged into a generalized issue covering all scenarios.

Manual Filtering*:* Consensus filtering alone does not guarantee the semantic or contextual correctness of a test case. After a test-case has passed consensus filtering, we manually review it to ensure clarity, logical consistency, and adherence to domain-specific requirements. Test-cases are deemed unclear if they contain ambiguous descriptions (e.g., 'missing proper verification' - unclear what kind of verification) or lack sufficient detail about states , transitions, or message flows. Test-cases are considered redundant if they overlap in purpose or scope with other cases already retained. Finally, test-cases based on invalid or inaccurate assumptions are also excluded. For example, if a test-case assumes unauthorized access to the ngKSI but it was previously found to be impossible, it is considered unpromising and discarded. CoT prompting is particularly effective in this process. When GPT-4 provides step-by-step reasoning, we can follow the logical sequence to see exactly how each step connects the specification to a possible SP violation or HI exploitation, making it easier to determine whether the reasoning is sound. If there are logical inconsistencies (one step of sequence does not entail another), we discard the identified vulnerability.

We note that manual filtering can be a limitation in terms of scalability. However, we deemed it necessary in this work for a reliable evaluation of GPT-4's capability to identify vulnerabilities from complex cellular protocols, which we think has not been explored in detail. Once LLMs like GPT-4 or further advanced models demonstrate reliable consistency in this domain, this process can be automated by fine-tuning language models on labeled data.

C. Test-case Simulation

Simulation using Open5GS and UERANSIM*:* We used Open5GS to simulate the 5G core network and UERANSIM to simulate the user equipment (UE) and gNB. The Open5GS (v2.7.0) setup remained unchanged, running with its default configuration, including MongoDB for database management and Node.js for web-based administration. In UERANSIM, we modified the gNB configuration to use specific IPs: `127.x.x.101` for Radio Link Simulation, `127.x.x.100` for the N2 interface (NGAP), and `127.x.x.200` for the N3 interface (GTP-U). The gNB connected to the AMF at `127.x.x.5` on port `38412`. A single S-NSSAI slice $(sst : 1)$ was defined, and SCTP stream number errors were ignored for better compatibility. For the UE, we kept the default UERANSIM (v3.2.6) settings, except for updating the gNB search list to `127.x.x.101` to match the gNB.

We updated the IMSI of every UE after registering through open5gs UI. It established an IPv4 PDU session with the APN 'internet' and used standard integrity and ciphering algorithms for secure communication. While testing, we had to change some of this base configuration. Here we discuss some of the techniques used to simulate the test-cases.

1) Stopping Certain Responses: We have modified the UERANSIM code to halt specific responses from the UE to simulate the absence of certain network functionalities. This helps in understanding the network's behavior when expected responses, such as `REGISTRATION_COMPLETE`, are not received.

2) Sending Alternate Responses: Here we modify the UERANSIM code to send alternate responses instead of the expected ones, for example, `SECURITY_MODE_REJECT` in place of `REGISTRATION_COMPLETE`. By doing so, we simulate the injection of incorrect data into the network and analyze its effects on network behavior and security.

3) Tampering with Expected Messages: We have tampered some expected messages within the UERANSIM code to evaluate the network's ability to detect and handle message tampering. For example, we modify one or two bits of the message being sent. This experiment aims to uncover vulnerabilities related to message integrity and authenticity.

4) Changing Encrypted Messages: By modifying encrypted messages in the UERANSIM code, we assess the network's susceptibility to message interception and alteration, which can potentially lead to severe security breaches and unauthorized access to the network. This experiment sheds light on encryption vulnerabilities within the 5G network.

5) Disabling Certain Timers: In some cases we disable specific timers in the UERANSIM code, to evaluate the network's ability to handle and recover from DoS attacks without depending on the UE for defense mechanisms.

6) Requesting New Registration Before the Previous is Complete: We introduce a new request before the previous one is complete, to check whether it leads to a race condition or network instability. For example, we stop the registration response from being sent to the AMF and start a new registration request and observe how the AMF reacts to this.

7) Introducing Extra Variables: We introduce additional variables to the UERANSIM code, such as additional MCC and MNC variable, allowing us to toggle between them dynamically at runtime. This experiment aims to identify vulnerabilities related to dynamic variable handling within the 5G network.

8) Custom Function Implementation: In some cases we implement custom functions within the UERANSIM code to bypass encryption and other security checks. By doing so, we assess the network's vulnerability to sophisticated attacks and the effectiveness of security protocols.

***Simulation Using 5GReplay**:* To simulate replay attacks, we have used '5GReplay' [34], an open-source 5G network traffic fuzzer. First we define some rules and configurations to determine which packets will be modified, forwarded, or dropped. For example, to simulate SMC replay attack, we configure 5GReplay to detect the SMC messages sent by UE and replay it twice to AMF. Then we check the AMF to verify whether AMF has actually received the same packet twice. If yes, the attack is successful. To simulate packet modification, we modify the UE RAN identifier in the NGAP authentication response and transmit it with the rest of the traffic. Afterwards, we monitor the AMF log, UE terminal, and Wireshark to see if any unusual activity has been detected by the network.

V. Findings

In this section we present the findings of our experiments. First we provide a summary of the findings from zero-shot and domain-aware approach, including the impact of filtering process on the final output. Then we discuss the results of our simulation, including both true positives and false positives. Finally, we present a quantitative comparison of our findings with similar existing works.

A. Potential Vulnerabilities Identified

***Zero-shot Findings**:* We have identified 25 potential vulnerabilities using the zero-shot approach. To the best of our knowledge, 12 of these are new findings. Another 12 vulnerabilities have been previously identified in other works, while one has been demonstrated to not exist in 5G. These potential vulnerabilities are presented in Table III.

Effectiveness: Zero-shot approach is highly effective for detecting high-level logical inconsistencies, weak validation checks, misconfigurations, and ambiguous protocol rules.

- **Logical and Procedural Flaws:** One of GPT-4's key strengths is analyzing logical flows and detecting contradictions, which is why it is effective in identifying inconsistencies and misalignments like collision between different commands, improper handling of requests (e.g. deregistration) in certain scenarios (e.g. 'switch off indication).
- **Validation and Integrity Issues:** Since GPT-4 is well-trained on common security principles, it is capable of detecting patterns where requests (e.g. identity request) or parameters (e.g. AT_KDF) may not be properly authenticated or verified.
- **Misconfigurations and State Management Issues:** GPT-4 is able to detect flaws in resource allocation, identifier management, and session state handling, like inaccurate updating of 5G-GUTI, mismanagement of truncated 5G-S-TMSI configuration, etc.
- **Ambiguous Guidelines:** As GPT-4 is trained on diverse text sources, it is effective at spotting vague or under-defined rules in technical documentation, for example ambiguous guidelines for SNPN-specific attempt counters, inaccurate processing or storage of Disaster Condition PLMNs, etc.

Limitations: Without domain-specific context, GPT-4 lacks the technical depth to identify advanced, low-level security flaws such as cryptographic weaknesses, exploitable timing and race condition attacks, network layer exploits (including downgrade attacks), and particularly attacks that involve multiple states or entities.

***Domain-aware Findings**:* Using the domain-aware approach, we have identified 24 potential vulnerabilities, including 15 new discoveries according to our knowledge. Eight vulnerabilities align with findings from previous works, while

TABLE III: Summary of Zero-shot Findings

Potential Vulnerability	Message/Command	New/Existing
Authentication-related Issues		
Failure to detect malicious authentication request with forged ngKSI	EAP-authentication request	Disproven [35]
Vulnerabilities in EAP message processing in network slice-specific authentication	Slice-Specific Authentication	Existing [36]
Exploiting lack of timer validation via repeated AUTH REJECT messages	Authentication reject	New
Insufficient or missing Serving Network Name (SNN) verification	EAP-authentication challenge	New
Lack of EAP-Request Validation (incorrect validation of AT_KDF)	EAP-authentication request	New
Security Flaws		
Vulnerabilities in EAP message processing in Network Slice-Specific authentication	Slice-Specific Authentication	Existing [37]
Incorrect NAS MAC calculation risking	UL NAS Transport	Existing [26]
Incomplete or malformed security mode commands	Security mode command	Existing in 4G [2]
Inadequate validation of identity request with malformed parameters	Identity request	Existing in 4G [2]
Inadequate handling of security mode reject command	Security mode reject	Existing in 4G [7]
Implementation Flaws		
Improper handling of deregistration request with 'switch off' indication	Deregistration request	New
Lower layer failures leading to incorrect invalidation of old 5G-GUTIs and TAIs	Config Update, deregistration request	New
Collision between configuration update and deregistration request	Config update, deregistration request	New
Inability to detect transmission failures of configuration update complete message during tracking area change	Config Update	New
Improper handling of rejected NSSAI in mixed access scenarios	NSSAI Config Update	New
Inadequate handling of emergency services registration	Config Update	New
Routing failures in NAS 5GSM message transport	DL NAS TRANSPORT	New
Inaccurate processing or storage of Disaster Condition PLMNs	Config Update	New
Mismanagement of truncated 5G-S-TMSI configuration	Config Update	Existing [6]
SMF selection failure during PDU session establishment	UL NAS Transport	Existing [37]
Inadequate handling of security mode reject command	Security mode reject	Existing in 4G [7]
Inconsistent NSSAI parameters handling across different access types	NSSAI config update	Existing [37]
Inaccurate Updating of 5G-GUTI	Config Update	Existing [37]
Standards and Resource Management Issues		
Ambiguous guidelines for SNPN-specific attempt counters in the specification	Authentication reject	New
Ineffective congestion control and resource allocation	UL NAS Transport	Existing [37]

one has been proven to not exist in 5G. Table IV summarizes these findings. We have also mentioned which domain aware technique was used to find a particular vulnerability and whether 3GPP specification or SPEC5G was used as input.

Effectiveness: Since this approach benefits from techniques such as property-directed analysis and hazard indicators identification from established literature, it is able to capture sophisticated vulnerabilities that require a combination of deep procedural understanding and adversarial thinking.

- **Multi-State and Cross-Procedure Attacks:** Domain-aware GPT-4 is capable of evaluating how security properties should persist across states and transitions, and which interactions are linked with potential security breaches. As a result, it is able to detect vulnerabilities when multiple protocol steps interact, like security context mishandling during state transitions or inter-state handovers, sequence number mismanagement during disconnect requests, session resumption without re-authentication, and so on.
- **Cryptographic and Integrity Violations:** Using security property enforcement, GPT-4 is guided to check whether messages that require cryptographic protection are actually being secured, allowing it to uncover issues like replay attacks, tampering with integrity-protected messages, or lack of security mechanisms altogether.
- **Message Spoofing and Injection:** GPT-4 now has enough contextual understanding to identify spoofing requests before or even after security activation, unauthorized message injection, and rogue network service delivery risks.
- **Privacy and Identity Exposure:** By explicitly guiding GPT-4 to focus on data confidentiality and identity protection, the approach ensures that user identity and sensitive information leakage issues like SIP credentials interception and SUPI exposure in plaintext.
- **Network and Resource Management Exploits:** Although zero-shot approach also has this capability, domain-aware strategies help GPT-4 identify advanced attack scenarios like message flooding to overwhelm gNB and AMF, repeated PDU session re-establishments etc.

Impact of SPEC5G: Among the 24 potential vulnerabilities, 19 were identified using the 3GPP specification of 5G while 11 were found using SPEC5G. SIX of them were found using both. There are two key observations here. First, as we discussed earlier, segmenting the full 3GPP specification into smaller portions to fit them within GPT-4's context window risks losing contextual coherence. If a relevant passage or prerequisite condition appears in a different segment than the place where a particular vulnerability arises, GPT-4 will not be able to detect it. In contrast, the SPEC5G dataset is a carefully curated subset of the entire specification and contains security related texts only, which fits within GPT-4's context window. This is why using SPEC5G we have found FIVE vulnerabilities that were not found using the segmented specifications. Second, while SPEC5G helps consolidate key security details, it inevitably omits certain procedural, architectural, or corner-case nuances that do not seem directly 'security-related' — but still introduce vulnerabilities when taken in full context. In other words, detecting some vulnerabilities requires the depth of information found only in the original, larger specification.

TABLE IV: Summary of Domain-aware Findings

Potential Vulnerability	Message/Command	Spec	SP/HI	New/Existing
Authentication-related Issues				
Incorrect validation of authentication token	Authentication request	both	SP	Disproven [35]
Session resumption without re-authentication	Service request	3GPP	HI	New
Unauthorized EAP Server Change during EAP-TTLS authentication	EAP-request/response	3GPP	HI	New
Accepting service from rogue networks without proper authentication	Unauthenticated service delivery	3GPP	SP	New
Privacy/Phishing Attacks				
SIP credentials exposure via REGISTER requests	Integrity protected SIP register	SPEC5G	SP	New
SUPI exposure in clear text during transmission	Authentication or reg request	both	SP	Existing [6]
Inadequate SUCI handling during initial registration	Registration request	3GPP	SP	Existing [38]
IP-SM-GW short message spoofing	SIP message request	SPEC5G	HI	New
Security Flaws				
Spoofing RRC connection request before security activation	RRC connection request	both	SP	New
Spoofing NAS attach request after security activation	NAS attach request (plain)	both	SP	New
Tampering Integrity-Protected Message with Valid Sequence Number	Integrity protected NAS message	SPEC5G	SP	New
Misuse of access barring (e.g. by manipulating network signals) to trigger false local deregistration	Connection release command with access barring conditions	3GPP	HI	New
Reusing previously assigned GUTIs without context validation	Deregistration request	3GPP	HI	New
Null security for messages that should be security protected	NAS message	SPEC5G	SP	Existing [6]
Unauthorized access to paging procedures	Paging request	3GPP	SP	Existing [6]
NAS Signalling Spoofing via duplicate TEID values in GTP-U header	NAS messages	SPEC5G	HI	Existing [37]
Injection of malicious NAS messages into protected sessions	NAS messages	3GPP	SP	Existing [35]
Implementation Flaws				
Mishandling security contexts during state transitions	Handover command	3GPP	HI	New
Mishandling security contexts during inter-system handovers	Handover command	3GPP	HI	New
Sequence number mismanagement in PDN disconnect request	PDN disconnect request	both	SP	New
CAG information list mismanagement	Deregistration request	3GPP	HI	Existing [12]
Resource Management and Replay Attacks				
NAS message flooding to overwhelm gNB and AMF RM	NAS messages (plain/protected)	3GPP	SP	Existing [39]
DoS via Timer T3560 expiry handling in burst message attacks	Authentication request	3GPP	HI	New
Replaying old security protected messages	Replayed protected messages	both	SP	New

TABLE V: An ablation study of CoT-reasoning

HI	Output without CoT	Output with CoT
Initiation of EAP-AKA' without valid SNN matching	Vulnerable. If SNN matching fails or is omitted, the UE may derive keys based on incorrect domains, enabling man-in-the-middle attacks on authentication.	Vulnerable. The SNN anchors trust in EAP-AKA'. If the UE skips this check, a rogue base station could impersonate a legitimate network and hijack authentication, leading to MITM or credential theft.
Processing auth reject without active timers	Vulnerable. If auth reject is received without T3516/T3520 running and is not integrity-protected, the UE may discard it, allowing desynchronization or access continuation in invalid state.	Depends on implementation. The spec mandates discarding such messages if no timers are active. If a UE erroneously processes unauthenticated messages without checking timers, an attacker could trigger deregistration or blacklisting without cause.
Using stale `ngKSI` values	If `ngKSI` is reused (5GMM Cause #71), a stale key context may be activated, leading to key reuse and opening doors for key replay and session hijacking.	If the UE accepts an `ngKSI` that was already used in an active context, a replayed auth request could cause NAS context confusion. Vulnerable only if UE does not implement 5GMM cause #71 rejection.
Emergency PDU session use without authentication	Vulnerable. When timer T3520 expires during emergency sessions, the UE continues using current context even if authentication fails—potentially exploitable for maintaining sessions without auth.	Not vulnerable. Emergency PDU sessions are allowed by specification to bypass authentication under strict conditions (e.g., user unconscious). This is a trade-off, not a design flaw. Abuse is mitigated by policy, timers, and network-side constraints (not UE).
Deletion of valid keys upon receiving EAP-success	Vulnerable. On EAP-success, the ME deletes previously valid KAUF/KSEAF even if partial context was not updated. If success is spoofed or injected, it causes premature context removal and DoS.	Not vulnerable. Keys are only deleted after the UE has successfully validated the EAP exchange and received an authenticated EAP-success message. Forging such a message requires full control of the network, which is infeasible for an external attacker.

As a result, we have found 13 vulnerabilities only using the full 3GPP specification.

Impact of CoT-reasoning: In Table V, we compare GPT-4's reasoning on the exploitation of several authentication-related Hazard Indicators (HIs), both with and without Chain-of-Thought (CoT) prompting. Without CoT, GPT-4 mostly makes single-shot decisions based on surface-level cues and keyword-specific pattern matching, overlooking key protocol safeguards or feasibility constraints. With CoT prompting, GPT-4 demonstrates more structured and context-aware reasoning — analyzing the context surrounding an HI to determine whether it can be exploited in real-world scenario."

Impact of Filtering: As shown in Table VI, zero-shot approach produced 46 test-cases, of which 33 passed the consensus filtering. Out of these 33, 25 passed the manual review. In domain-aware approach, 30 out of the 34 test-cases passed consensus filtering, from which 24 passed manual filtering. Consensus filtering eliminated 13 test-cases in zero-shot and 4 in domain-aware, indicating that domain-aware test-cases were more robust and consistent. 8 zero-shot test cases were removed through manual review, with most rejections due to missing details. 6 domain-aware test-cases were also filtered out. In general, the zero-shot approach retained 54% of its original test cases, while domain-aware retained 71%,

TABLE VI: Impact of Consensus-based and Manual Filtering

Filtering Type	Zero-shot Approach				Domain-aware Approach			
	Input	Accepted	Discarding reason	Count	Input	Accepted	Discarding reason	Count
Consensus	46	33	Infrequent	13	34	30	Infrequent	4
Manual	33	25	Ambiguous	1	30	24	Ambiguous	1
			Lacks detail	4			Lacks detail	2
			Redundant	2			Redundant	2
			Inaccurate assumption	1			Inaccurate assumption	1

suggesting that domain-aware strategies produce higher quality and more contextually grounded test-cases.

B. Zero-shot Simulation Results

Among the 25 potential vulnerabilities we found using the zero-shot approach, we have tested eight, and found the presence of vulnerabilities in four cases (true positives). We have been unable to detect the vulnerabilities in other 4 cases (false positives). The findings from our testing and their impacts are presented below.

True Positives: A summary of the test-cases that yielded vulnerabilities are presented in the Table VII. Now we briefly discuss about these test-cases and their outcomes.

TABLE VII: Summary of Zero-shot True Positives

Sl	Potential Vulnerability	Our Findings
A1	Inadequate Handling of sec mode reject (SMR)	If we send `SMR` before AMF has completed registration, the UE still remains operational (Should have been de-registered)
A2	Improper Handling of dereg request	Sending a dereg request with switch-off indication results in irregularities
A3	SMF Selection Failure in PDU Session Establishment	Mismatched APN and SST-id halts PDU Session to be created
A4	Inability to Detect Transmission Failure during TAI Change	If transmission failure occurs when the device has moved to a new area, the network does not re-initiate the registration procedure

A1 Inadequate Handling of `SECURITY MODE REJECT`

During registration, when the AMF receives a `SECURITY MODE REJECT (SMR)` message from the UE, it is expected to immediately halt the ongoing procedure.

Vulnerability: If the AMF does not stop the ongoing registration process immediately and allows the process to continue, it will lead to an incomplete security setup, leaving the connection vulnerable to attacks.

Our Findings: After making the UE send an `SMR` during the registration process, we have observed the following:

- As AMF was expecting `REGISTRATION COMPLETE` message, it treats the `SMR` as `'UNKNOWN MESSAGE'` and does not stop the procedure.
- If we send `SMR` before `REGISTRATION COMPLETE`, UE remains operational. Here, UE should have been deregistered by AMF.
- If we send `SMR` only, UE becomes un-operational but keeps switching between `IDLE` and `CONNECTED` states.

Impact: This vulnerability can be exploited in several ways:

- **Bypassing Security Controls:** An attacker can exploit this vulnerability to maintain a connection with the network, bypassing security measures that would be re-established during the registration process. This can allow for continued access to the network without proper security credentials.
- **Impersonation and Session Hijacking:** An attacker can also exploit it to impersonate the UE. Since the AMF still considers the UE registered, the attacker can send messages pretending to be the UE, and gain access to ongoing sessions and sensitive information. This can result in data theft, unauthorized access to user accounts, and interception of confidential communications.

A2 Improper Handling of `DEREGISTRATION REQUEST`

During the identification procedure, the 5G protocol specifies that if a `DEREGISTRATION REQUEST` with a `'switch off'` indication is received, the process should be immediately aborted. This measure ensures that the network can promptly respond to potential security issues or user-initiated requests, maintaining integrity and security of the network.

Vulnerability: Failure to correctly handle this request can leave the network susceptible to malicious activities.

Our Findings: To simulate this, we have sent a normal `DEREGISTRATION REQUEST`, and another one that includes a `'switch off'` indication. On inspecting the AMF logs in both cases, we have found dissimilarities. The invoked deregistration (switch off indication) AMF log does not have clear UE identification (no IMSI or SUCI mentioned initially), contains errors regarding the identity type and the absence of SUCI, and shows an unexpected sequence of events.

Impact: This vulnerability can have significant consequences.

- **Identity Spoofing and Anonymity:** The inability to properly identify UE can allow attackers to spoof UE identity or to maintain anonymity while conducting malicious activities, thus evading detection.
- **Security Breach:** The identified irregularities and the network's failure to handle them correctly can be exploited by attackers to bypass security measures, leading to unauthorized access and potential compromise of network security.

A3 SMF Selection Failure in PDU Session Establishment

During Packet Data Unit (PDU) session establishment, the AMF is responsible for selecting a Session Management Function (SMF) based on various criteria, including routing contexts such as the Access Point Name (APN) or the Session and Service Continuity (SSC) modes.

Vulnerability: If the AMF is unable to select an appropriate SMF for PDU session establishment due to missing or mismatched routing contexts, such as the APN or Service

Selection Subscription Identifier (SST-id), it can prevent the completion of the PDU session establishment process, rendering the UE unable to access intended services.

Our Findings: Our testing has revealed that when an APN or SST-id that does not correspond with the Digital Network Name (DNN) or SST-id configured in the OPEN5GS is used, the AMF fails to establish the PDU session. Although the UE successfully registers with the network, it remains unoperational due to the inability to establish a PDU session. This indicates a lack of robustness in handling routing context mismatches, leading to service disruption for the end user.

Impact: The inability of the AMF to select an SMF and establish a PDU session under certain conditions can lead to disruption of services such as internet access and voice communication, degradation in the quality of service, and increased signaling traffic and network load.

A4 Inability to Detect Transmission Failure during TAI Change

When a UE moves from one area to another, resulting in a change of the Tracking Area Identity (TAI), it requires an update to reflect the new location. This is done through a `CONFIGURATION UPDATE` command. If this command does not reach the AMF for any reason (such as a transmission failure), the network should re-initiate the registration procedure to ensure that the UE does not suffer from outdated configuration settings or get disconnected.

Vulnerability: If the AMF does not re-initiate the registration procedure, the UE will either get disconnected from the network or be left with outdated settings.

Our Findings: When the TAI changes dynamically (on the fly), the AMF does not re-initiate the registration procedure and the UE is disconnected.

Impact: The failure to handle dynamic TAI changes and transmission failures properly can have several impacts:

- **Inaccurate UE Location Tracking:** The network's inability to keep up with the UE's current location can lead to inaccuracies in location-based services or in the network's ability to efficiently manage resources.
- **Service Disruption:** Users may experience service disruptions or disconnections when moving across different tracking areas, affecting their ability to use network services seamlessly.

False Positives: We did not find vulnerabilities in four cases, which are described below.

B1 Handling of Repeated Identity Requests

Supposed vulnerability: If the AMF allows repeated `IDENTITY REQUEST` messages even after timer expiry, a malicious actor can exploit this to launch a Denial of Service (DoS) attack by sending a fake registration request but not responding to AMF's identity request.

Our Findings: To simulate this, we have stopped sending response from the AMF so that the UE keeps sending `IDENTITY REQUEST`s. AMF allows re-transmissions for only as long as its timer allows, which is 5 times per request, then the UE context is released; so a direct DoS attack is not possible.

Implication: A single UE cannot directly cause a DoS due to the AMF's timer-based retransmission limits. However, it may be possible to launch a DDoS (Distributed denial of service) attack by sending numerous fake registration requests from different sources. Each request triggers the AMF to allocate resources and retry identity requests up to 5 times, so continuous fake requests can overwhelm the AMF, potentially leading to service disruption.

B2 Inadequate Handling of Emergency Registrations

Supposed vulnerability: If the AMF improperly sets the registration result, it can lead to incorrect registration statuses, potentially denying the UE access to non-emergency services.

Our Findings: To simulate this, we have established a connection using `EMERGENCY` status and another one as a normal registration. On inspecting various status-related parameters, we have found difference only in S-NSSAI parameters (which is expected). There are no considerable irregularities.

Implication: The AMF correctly handles emergency service registrations without any irregularities, ensuring that emergency and non-emergency services are appropriately differentiated based on S-NSSAI parameters.

B3 Collision with `DEREGISTRATION REQUEST`

Supposed vulnerability: If there is a `CONFIGURATION UPDATE COMMAND` during deregistration, it can lead to conflicts in network registration state, causing service disruption or incorrect network access.

Our Findings: To simulate this, we have configured the UE to send a `CONFIGURATION UPDATE COMMAND` while the deregistration process was ongoing. We have observed that UE ignores the `CONFIGURATION UPDATE COMMAND` if there is already a `DEREGISTRATION REQUEST` for the same access type and so there is no collision.

Implication: The UE correctly prioritizes the ongoing deregistration process by ignoring configuration updates for the same access type, ensuring there are no conflicts in the network registration state.

B4 Unauthorized Access to ngKSI

Supposed vulnerability: If the AMF does not properly validate an `AUTHENTICATION REQUEST` with a forged ngKSI, it can lead to unauthorized access to network resources.

Our Findings: To test this, we have configured the UE to send an `AUTHENTICATION REQUEST` with a forged ngKSI value. We have found that if the computed ngKSI value does not match the received value, AMF immediately issues a `SECURITY MODE REJECT` and deregisters the UE.

Implication: AMF is effective in preventing unauthorized access through tampered ngKSI values.

C. Domain-aware Simulation Results

Among the 24 potential vulnerabilities found using this approach, we have simulated six, and confirmed five vulnerabilities (true positives). We have not detected any vulnerability in the remaining case (false positive).

True Positives: A summary of the domain-aware test-cases that resulted in vulnerabilities are presented in the Table VIII. Here we briefly discuss these vulnerabilities, our findings in the simulation and the impacts.

TABLE VIII: Summary of Domain-aware True Positives

Sl	Potential Vulnerability	Our Finding
C1	NAS Message Flooding (Before and After Security Activation)	It is possible to exhaust the resources of the gNB by sending connection requests from a large number of UEs
C2	Spoofing RRC connection request before security activation	Spoofing a legitimate IMSI and connecting to AMF results in service loss for both UEs
C3	Null Security for Messages that should be Security Protected	Open5GS lacks a proper security profile, offering only null-security
C4	Tampering Messages with Valid Sequence Number	UE connects to AMF even with tampered Registration payload (NAS KSI)
C5	Replay Attack on Security Protected Messages	Attack succeeds using lost or retransmitted messages but alteration fails

C1 NAS Message Flooding (Before and After Security Activation)

In a 5G network, the gNB is the primary point of radio access, responsible for managing all radio communications between the network and the UE. The AMF manages the Non-Access Stratum (NAS) messages which include important signaling between the UE and the core network, such as session management and mobility management.

Vulnerability: By sending a large volume of NAS messages from the UE to both the gNB and AMF, a coordinated DoS attack can potentially deplete their computational and memory resources. These messages can be both 'plain' (unencrypted and unprotected) or 'protected' (encrypted and integrity-protected).

Our Findings: To simulate this attack, we have used a bash script to initiate simultaneous connection attempts from numerous UEs with the same configuration file and IMSI number. The gNB could not handle the volume of requests, resulting in a segmentation fault in our simulated environment. This suggests that the gNB does not track the number of requests per UE; otherwise, it would likely restrict unusually high numbers of connection attempts from a single UE.

Impact: The implication of these vulnerabilities are:

- **gNB Resource Exhaustion:** The gNB faces an immediate resource drain due to the excessive number of connection attempts, which can prevent it from handling legitimate user connections effectively, leading to network-wide instability.
- **AMF Resource Drainage:** Similar to the gNB, the AMF is tasked with processing a large volume of NAS messages, further straining the network's core resources and leading to potential service degradation or failure. The mixture of plain and protected messages can further complicate the processing logic of the AMF, as it needs to handle both types differently – decrypting and verifying integrity where necessary.

C2 Spoofing RRC Connection Request Before Security Activation

Before the activation of security measures, the gNB is expected to verify the plausibility of incoming RRC connection requests to prevent unauthorized access.

Vulnerability: If gNB does not verify the plausibility of incoming RRC connection requests even before security is activated, then a malicious actor can impersonate a legitimate UE without needing to go through proper authentication checks.

Our Findings: First we have established a connection using a legitimate UE, then we have attempted to establish another connection using the same (spoofed) IMSI of the legitimate UE. Open5GS allowed the spoofed connection, but then rejected service for both UEs, citing the cause `'UE IDENTITY CANNOT BE DERIVED FROM NETWORK'`. This allows an attacker to deprive a legitimate UE of access by spoofing IMSI.

Impact: This vulnerability has several potential impacts:

- **Denial of Service to Legitimate Users:** By spoofing the IMSI of a UE, attackers can cause the network to reject services for legitimate users. For example, an attacker can capture the IMSI of a UE and use it to send an RRC connection request to the gNB. Since the gNB does not verify the plausibility of the request before security activation, it allows the connection but later rejects service for both the legitimate and spoofed UEs, thus depriving the legitimate user of network access.
- **Security Bypass:** The ability for attackers to initiate connections without proper authentication checks can be exploited to bypass the network's security mechanisms, allowing unauthorized access to network resources and services.

C3 Null Security for Messages that should be Protected

In a secure 5G network, the AMF is responsible for ensuring the integrity of data exchanged between the UE and the network through robust encryption methods.

Vulnerability: If the AMF only implements null security profile (no security), it will be unable to detect any tampering of messages.

Our Findings: UERANSIM has three security profiles: *Null-security*, *Profile A* (protection scheme 1), *Profile B* (protection scheme 2). When we tried to simulate this test-case, we could not establish a connection with AMF (Open5GS) using profile-A. We could only connect using null-security profile; suggesting that Open5GS may not have implemented proper encapsulations. We have raised an issue regarding this in the Open5GS GitHub and found that other users are facing it too.

Impact: The inability to establish a secure connection using Profile A raises concerns about the overall reliability and security of Open5GS-based systems. It can undermine trust in Open5GS as a viable solution for 5G deployments, especially for applications requiring stringent security standards.

C4 Tampering Messages with Valid Sequence Number

Messages which can affect the network state or user authentication (e.g., Registration Request, Service Request) should be both integrity protected and verified for sequence number validity. This dual check ensures that the messages have not been tampered with and are sent in the correct order, preventing replay attacks and ensuring the message's authenticity and integrity.

Vulnerability: If the AMF only verifies the sequence number without ensuring the integrity of the message content, it may accept tampered messages as legitimate. It can lead to unauthorized access and manipulation of network functions.

Our Findings: To simulate this test-case, we have sent a Registration Request to the AMF that had valid sequence

number but tampered payload. Specifically, we have tampered the NAS key set identifier (ngKSI). We have found that despite this tampering, AMF still allows the UE to be connected.
Impact: This issue can have significant practical impacts.

- **Impersonation:** Attackers can impersonate legitimate users by tampering with the ngKSI or other critical parameters. For example, an attacker can capture a legitimate Registration Request, modify the ngKSI to match their own device's identifier, and send it to the AMF. If the AMF only verifies the sequence number, it may allow the attacker's device to connect, mistaking it for the legitimate user.
- **Session Hijacking:** During an active session between a legitimate user and the network, an attacker can send a tampered Service Request with a valid sequence number but modified payload. If the AMF accepts this tampered message, the attacker can take over the session, potentially accessing or altering sensitive communications.

C5 Replay Attack on Security Protected Messages

In 5G, encrypted messages can still be susceptible to *replay attacks*, where a malicious actor captures valid network communication and re-transmits it later to attempt to gain unauthorized access or disrupt operations.
Vulnerability: Lack of replay protection will allow old messages to be used in attacks.
Our Findings: We used 5GReplay to test replay attacks by capturing network packets and either replacing the original packet with an older one (drop-replay) or sending a modified version of the original packet (forward-replay). The results of these tests are summarized in Table IX. The drop-replay attack succeeded for security mode control (SMC) messages and sequence number modification. However, both of these attacks failed when we tried to forward the modified messages, evident by errors in AMF log and connection loss. Other attacks, such as modification of SCTP parameters, malformed NGAP procedure codes, and NAS fuzzing, also failed.
Implication: The success of drop-replay attacks suggests weaknesses in 5G protocols in handling lost or retransmitted messages. The failed attacks indicate strong integrity protection, sequence number validation, and cryptographic checks in 5G to reject altered packets. However, another factor to consider here is that integrity protection and sequence validation rely on timestamps, expected delays, and state synchronization. A real-world adversary can exploit latencies, network congestion or handover delays to time the replay attack more effectively, which cannot be replicated in simulation.
Impact: Adversaries can exploit the vulnerability to retransmission scenarios to bypass authentication or replay old security commands under certain conditions.

False Positives: We have not found any vulnerability in one case, which is discussed below.

Inaccurate Validation of Authentication Token

During authentication, the UE computes a response (RES) based on a random number (RAND) received from the network and sends it back to the AMF.
Supposed vulnerability: If a malicious UE generates a RES with a fake or invalid RAND and sends it to the AMF, and the AMF does not properly validate it, this can lead to serious issues like authentication bypassing and session hijacking.
Our Findings: To simulate this, after the UE receives AUTN and RAND from the AMF, we have generated a RES using a fake RAND. Upon receiving this forged RES, AMF sends an `AUTHENTICATION REJECT` and terminates the process.
Implication: AMF effectively prevents any attempt of authentication bypass using fake RES values. Nonetheless, another way of carrying out this attack is to use an old but valid RAND value, which the AMF might not properly keep track of. This case has not been covered in our simulation.

TABLE IX: Summary of Replay Attack Findings

Attack Description	Packet Type	Warning or Error	UE-gNB Connection Status	Verdict
NAS-5G SMC Replay Attack	DROP	NO	Fruitful Connection	**SUCCESS**
NAS-5G SMC Replay Attack	FORWARD	YES	Connection lost	FAILED
Malformed NGAP procedure code	BOTH	YES	Connection lost	FAILED
Modification of SCTP parameters	BOTH	YES	Connection lost	FAILED
UE Random ID Generation	BOTH	YES	Connection lost	FAILED
NAS Sequence Number Modification	DROP	NO	Fruitful Connection	**SUCCESS**
NAS Sequence Number Modification	FORWARD	YES	Connection lost	FAILED
NAS Custom Fuzzer	BOTH	YES	Connection lost	FAILED

D. Comparison with Existing Works

TABLE X: Comparing 5GPT with existing works

Approach	#Issues identified/ test-cases	#Tested	#Confirmed
5GPT	**47**	**14**	**9**
Bookworm Game [11]	23	23	10
5GReasoner [6]	11	-	-
Hermes [12] (5G NAS)	11	-	-

A large-scale comparison with the existing works in vulnerability detection is difficult due to their different scope. Some works [6], [11], [12], like ours, focus on protocol-level issues, while others attempt to uncover device- or vendor-specific implementation issues. In Table X we have quantitatively compared our findings with other works of similar scope. It should be mentioned here that 5GReasoner and Hermes use formal verification methods instead of hardware or simulation testing. Moreover, Bookworm Game [11] employs two threat models, only one of which are related to our work. The other threat model focuses on UE-specific issues, which has uncovered further 32 vulnerabilities.

We have also compared our black-box approach against a white-box model, `MobileLLaMA` [40], which is an instruction fine-tuned variant of the LLaMA-2 13B model and is

TABLE XI: Comparison between MobileLLaMA (fine-tuned on LLaMA-2 13B) and 5GPT

Criteria	MobileLLaMA	5GPT
Common Findings	Both models point out missing validation of key EAP parameters (e.g., AT_KDF, SNN checks, message integrity), unauthorized access issues, ngKSI mishandling, replay/duplicate attacks, and timer-related DoS risks.	
Unique Findings	MobileLLaMA identifies some unique protocol field missing vulnerabilities such as "AUSF not including AT_RESULT_IND" and "UE not verifying network name inside AT_KDF_INPUT."	5GPT identifies deeper process-level and adversary-driven vulnerabilities, such as "Unauthorized EAP Server Change during EAP-TTLS authentication", "Accepting service from rogue networks without proper authentication", "DoS via Timer T3560 expiry handling in burst message attacks", and "EAP method filtering failure (e.g., other than AKA' and TLS)".
Specificity & Technical Depth	MobileLLaMA's findings are often phrased from high level. Some are superficial without enough details (e.g., "No protection against replay attacks", "Possible unauthorized access"). The more technical ones, like "AMF does not set the authenticator retransmission timer, leading to potential delays or failures" also do not specify which message types or timers are involved, why it matters, and whether it can be exploited in real-world scenario.	5GPT findings are highly specific with strong technical reasoning. For example, "Lack of Timer Validation: If an AUTH REJECT message is received without integrity protection, the UE starts timer T3247 with a random value. This could be abused by an attacker sending numerous unprotected AUTH REJECT messages, causing the UE to initiate multiple timers and leading to resource exhaustion" specifies the timer, identifies the vulnerable message, explains the attacker's strategy (flooding UE with fake messages), and shows the security impact (resource exhaustion).
Precision & Confidence	Almost similar vulnerabilities found across iterations, suggesting high confidence. However, due to lack of details and ambiguity, more test-cases would have to be filtered out before testing.	There are occasional outliers that would not appear across iterations. However, due to sufficient details, strong reasoning, and clear exploitation paths present, less filtering is required.
Testability	MobileLLaMA generated test-cases are often surface-level, without details about how to trigger a vulnerability or which protocol fields are involved ("replay attack", "lack of timers"). Attacker actions are often unclear ("AUSF sends incorrect parameters" – hard to understand what message or which parameters) and expected outcomes are vague ("potential delays or failures"). Some test-cases did not follow the provided format.	5GPT generated test-cases specify exact message types and parameters to manipulate (e.g., "EAP-Request with AT_KDF $\neq$ 1", "AUTHENTICATION REQUEST with malicious ngKSI"), describe clear attacker actions (e.g., "modifying AT_KDF", "forging ngKSI"), define concrete expected outcomes (e.g., UE rejects, AMF blocking unsupported EAP methods), and require less guesswork and minimal interpretation overall.

specifically optimized for telecom and 5G-related tasks. For a focused comparison, we tested both models on EAP-based primary authentication and key agreement procedure. Due to LLaMA 2's limited context window of 4096 tokens, we were unable to input the entire section at once and had to split it page by page. We compared both models' outputs in terms of coverage, specificity, technical detail, and testability. Table XI provides a detailed comparison. In summary, 5GPT consistently identified system-level procedural issues and attacker-driven vulnerabilities, while MobileLLaMA focused more on protocol field omissions and validation issues. However, MobileLLaMA often failed to recognize logical flaws and real-world exploits involving message flow manipulation or adversarial behavior. These results demonstrate that GPT-4, when guided by domain-specific prompt engineering, has better context understanding and reasoning capabilities than white-box models in the task of identifying vulnerabilities from natural language specifications.

VI. Discussion

A. Validation and Novelty of Our Approach

Among the 47 potential vulnerabilities we have identified, 20 have already been documented in existing research. This alignment with known vulnerabilities confirms the credibility of our methodology. In addition, the ability to uncover new vulnerabilities demonstrates the innovative potential of our model. By detecting 27 issues that have not yet been addressed in the literature, our approach not only contributes to the advancement of knowledge in 5G security, but also highlights its capability to push the boundaries of current research.

B. Limitations

Simulator limitations: Simulators like Open5GS, UERANSIM, and 5GReplay are designed primarily for conformance testing, which limits the scope of creating adversarial scenarios. For example, despite security measures, an attacker can potentially carry out a replay attack by exploiting network latencies, congestion, and handover delays, which cannot be simulated in 5GReplay. Moreover, certain vulnerabilities like timing attacks, downgrading security protocols during handovers, or injecting maliciously crafted messages to exploit improper validation, are difficult to test in a simulation environment, but can be verified using a proper hardware setup.

Risk of losing context due to segmentation: When using the 3GPP 5G specification, we segmented it into smaller sections for a more focused analysis. This can make it difficult to capture vulnerabilities that arise during interaction of multiple procedures described in different sections.

Untested vulnerabilities: Due to time constraints as well as simulator limitations, we were unable to test all potential vulnerabilities identified by our approach.

Manual filtering: The filtering process requires manual review for a reliable evaluation of GPT-4's performance in this domain, which may limit scalability. This process can be automated by using pre-trained language models fine-tuned on labeled datasets containing examples of valid and invalid test cases. This was beyond the scope of our current study.

VII. Conclusion & Future Work

In this work, we introduced an innovative approach for detecting vulnerabilities in 5G networks by combining GPT-4's zero-shot capabilities with domain-specific insights through prompt engineering. We identified a total of 47 potential vulnerabilities, with 27 new discoveries and 20 previously reported issues, thereby establishing both credibility and novelty of our method. Our analysis shows that zero-shot prompting is effective in identifying high-level procedural and logical flaws, whereas context-aware strategies are more suitable for detecting complex protocol violations, cross-procedural attacks, and

adversarial exploits. Our qualitative evaluation highlights that black-box LLMs, guided by prompt engineering, are more effective than white-box fine-tuning in identifying vulnerabilities from natural language protocol documents. In future work we plan on expanding real-world validation through hardware testing, automating the filtering process, and exploring mitigation strategies, thus further advancing the practical applicability of our findings in securing next-generation networks.

REFERENCES

[1] 3GPP, "5G system overview," 2020, accessed: 2024-06-03. [Online]. Available: https://www.3gpp.org/technologies/5g-system-overview

[2] S. R. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino, "LTEInspector: A systematic approach for adversarial testing of 4G LTE," in *25th Network and Distributed System Security Symposium (NDSS)*, 2018.

[3] I. Karim, S. Hussain, and E. Bertino, "ProChecker: An automated security and privacy analysis framework for 4G LTE protocol implementations," in *ICDCS*, 2021.

[4] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5G authentication," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[5] J. Zhang, L. Yang, W. Cao, and Q. Wang, "Formal analysis of 5G EAP-TLS authentication protocol using Proverif," *IEEE Access*, vol. 8, 2020.

[6] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, and E. Bertino, "5GReasoner: A property-directed security and privacy analysis framework for 5G cellular network protocol," in *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[7] C. Park, S. Bae, B. Oh, J. Lee, E. Lee, I. Yun, and Y. Kim, "DoLTEst: In-depth downlink negative testing framework for LTE devices," in *31st USENIX Security Symposium*, 2022.

[8] T. Saha, N. Aaraj, and N. K. Jha, "Machine learning assisted security analysis of 5G-network-connected systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 2006–2024, 2022.

[9] H. A. Kholidy, "Multi-layer attack graph analysis in the 5G edge network using a dynamic hexagonal fuzzy method," *Sensors*, vol. 22, 2021.

[10] B. Hussain, Q. Du, B. Sun, and Z. Han, "Deep learning-based DDoS-attack detection for cyber–physical system over 5G network," *IEEE Transactions on Industrial Informatics*, vol. 17, 2020.

[11] Y. Chen, Y. Yao, X. Wang, D. Xu, C. Yue, X. Liu, K. Chen, H. Tang, and B. Liu, "Bookworm Game: Automatic discovery of LTE vulnerabilities through documentation analysis," in *IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1197–1214.

[12] A. A. Ishtiaq, S. S. S. Das, S. M. M. Rashid, A. Ranjbar, K. Tu, T. Wu, Z. Song, W. Wang, M. A.-I. Akon, R. Zhang, and S. R. Hussain, "Hermes: Unlocking security analysis of cellular network protocols by synthesizing finite state machines from natural language specifications," in *Proceedings of USENIX Security Symposium*, 2024.

[13] Z. Wang and Y. Wang, "NLP-based cross-layer 5G vulnerabilities detection via fuzzing generated run-time profiling," in *IEEE 12th International Conference on Cloud Networking (CloudNet)*. IEEE, 2023.

[14] F. He, W. Yang, B. Cui, and J. Cui, "Intelligent fuzzing algorithm for 5G NAS protocol based on predefined rules," in *International Conference on Computer Communications and Networks (ICCCN)*, 2022.

[15] OpenAI, "GPT-4 technical report," 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[16] Y. Zhang, W. Song, Z. Ji, N. Meng *et al.*, "How well does llm generate security tests?" *arXiv preprint arXiv:2310.00710*, 2023.

[17] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language models: Survey, landscape, and vision," *IEEE Transactions on Software Engineering*, 2024.

[18] A. Mathur, S. Pradhan, P. Soni, D. Patel, and R. Regunathan, "Automated test case generation using T5 and GPT-3," in *9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2023.

[19] J. Ackerman and G. Cybenko, "Large language models for fuzzing parsers (registered report)," in *Proceedings of the 2nd International Fuzzing Workshop*, 2023, pp. 31–38.

[20] M. Tufano, D. Drain, A. Svyatkovskiy, S. K. Deng, and N. Sundaresan, "Unit test case generation with transformers and focal context," 2021. [Online]. Available: https://arxiv.org/abs/2009.05617

[21] G. Deng, Y. Liu, V. Mayoral-Vilches, and et al., "PentestGPT: Evaluating and harnessing large language models for automated penetration testing," in *33rd USENIX Security Symposium*, 2024.

[22] T. Zhang, I. C. Irsan, F. Thung, and D. Lo, "Cupid: Leveraging chatgpt for more accurate duplicate bug report detection," 2024. [Online]. Available: https://arxiv.org/abs/2308.10022

[23] A. Z. H. Yang, C. Le Goues, R. Martins, and V. Hellendoorn, "Large language models for test-free fault localization," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3623342

[24] S. Gao, X.-C. Wen, C. Gao, W. Wang, H. Zhang, and M. R. Lyu, " What Makes Good In-Context Demonstrations for Code Intelligence Tasks with LLMs? ," in *38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2023.

[25] I. Karim, K. S. Mubasshir, M. M. Rahman, and E. Bertino, "SPEC5G: A dataset for 5G cellular network protocol analysis," in *IJCNLP-AACL 2023 (Findings)*, 2023.

[26] H. Kim, J. Lee, E. Lee, and Y. Kim, "Touching the Untouchables: Dynamic security analysis of the LTE control plane," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1153–1168.

[27] M. L. Pacheco, M. Hippel, B. Weintraub, D. Goldwasser, and C. Nita-Rotaru, "Automated attack synthesis by extracting finite state machines from protocol specification documents," in *IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 51–68.

[28] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *International Conference on Learning Representations*, 2022. [Online]. Available: https://ar5iv.org/abs/2109.01652

[29] T. Brown et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[30] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[31] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[32] J. Yang, S. Arya, and Y. Wang, "Formal-guided fuzz testing: Targeting security assurance from specification to implementation for 5G and beyond," *IEEE Access*, 2024.

[33] W. X. Zhao et al., "A survey of large language models," 2024. [Online]. Available: https://arxiv.org/abs/2303.18223

[34] Z. Salazar, H. N. Nguyen, W. Mallouli, A. R. Cavalli, and E. Montes de Oca, "5GReplay: A 5G network traffic fuzzer-application to attack injection," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–8.

[35] E. Bitsikas, S. Khandker, A. Salous, A. Ranganathan, R. Piqueras Jover, and C. Pöpper, "UE security reloaded: Developing a 5G standalone user-side security testing framework," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2023. [Online]. Available: https://doi.org/10.1145/3558482.3590194

[36] V. P. Singh, M. P. Singh, S. Hegde, and M. Gupta, "Security in 5g network slices: Concerns and opportunities," *IEEE Access*, vol. 12, 2024.

[37] European Union Agency for Cybersecurity (ENISA), "Enisa threat landscape for 5g networks," 2020, accessed: 2024-08-22.

[38] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz, "5G SUCI-catchers: Still catching them all?" in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, p. 359–364. [Online]. Available: https://doi.org/10.1145/3448300.3467826

[39] H. Wen, P. Porras, V. Yegneswaran, A. Gehani, and Z. Lin, "5G-Spector: An O-RAN compliant layer-3 cellular attack detection service," in *Proceedings of NDSS'24*, 2024.

[40] K. B. Kan, H. Mun, G. Cao, and Y. Lee, "Mobile-llama: Instruction fine-tuning open-source llm for network analysis in 5g networks," *IEEE Network*, vol. 38, no. 5, pp. 76–83, 2024.