



Formal Analysis of Access Control Mechanism of 5G Core Network

Mujtahid Akon

The Pennsylvania State University
mba5773@psu.edu

Yilu Dong

The Pennsylvania State University
yiludong@psu.edu

Tianchang Yang

The Pennsylvania State University
tzy5088@psu.edu

Syed Rafiul Hussain

The Pennsylvania State University
hussain1@psu.edu

ABSTRACT

We present 5GCVerif, a model-based testing framework designed to formally analyze the access control framework of the 5G Core. With its modular design, 5GCVerif employs various abstraction techniques to craft an abstract model that captures the intricate details of the 5G Core's access control mechanism. This approach offers customizability and extensibility in constructing the abstract model and addresses the state explosion problem in model checking. 5GCVerif also sidesteps the challenge of exhaustively generating models for all possible core network configurations by restricting the model checker to explore policy violations only within the valid network configurations. Using 5GCVerif, we evaluated 55 security properties, leading to the discovery of five new vulnerabilities in 5G Core's access control mechanism. The uncovered vulnerabilities can result in multiple attacks including unauthorized entry to sensitive information, illegitimate access to services, and denial-of-services.

CCS CONCEPTS

• **Security and privacy** → **Mobile and wireless security**; **Formal security models**; **Access control**.

KEYWORDS

5G Core Network, Access Control, Formal Analysis, Vulnerabilities

ACM Reference Format:

Mujtahid Akon, Tianchang Yang, Yilu Dong, and Syed Rafiul Hussain. 2023. Formal Analysis of Access Control Mechanism of 5G Core Network. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*, November 26–30, 2023, Copenhagen, Denmark. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3576915.3623113>

1 INTRODUCTION

The core network is a crucial component of a cellular network. It orchestrates communications between cellular devices and the network while offering a wide array of services like voice, messaging, and multimedia. To accommodate the increasing demands

for large-scale communication, faster data transfer rates, ultra-low latency, and diverse applications, the Third Generation Partnership Project (3GPP), the standardization body for cellular networks, has replaced the monolithic core network of earlier generations with a disaggregated and cloud-driven service-based architecture (SBA) for the 5G Core (5GC). The introduction of SBA has decomposed the 5GC into multiple Network Functions (NFs), with each accountable for serving a specific set of related services. Moreover, the cloud-based micro-architecture solution for SBA design of 5G enables third-party partners (also known as tenants or non-telco organizations) to deliver a wide variety of third-party services to end-users. Since NFs contain sensitive information about users and 5G system, resource isolation and authorization of NFs are essential for secure interactions among them. To achieve this, 3GPP [1] adopted the industry-standard authorization framework *OAuth 2.0* [22, 46] in its SBA as the basis for the access control mechanisms for NFs in 5GC. However, OAuth 2.0 is a generic authorization framework, and its adoption and integration into 5GC design have not been formally verified. Particularly, flaws in 5GC's access control mechanism can be exploited by malicious or compromised NFs and can lead to critical security and privacy issues, including the unauthorized access to sensitive user information, unwanted modifications of crucial data, and denial-of-services [28, 29, 50]. Given this, we pose the following research question: *Is it possible to formally analyze the design of OAuth-based access control mechanism of a 5G Core?*

While prior efforts [37, 45, 54, 55] have formally analyzed the security of 4G and 5G cellular network protocols, they primarily focus on verifying the authenticity, secrecy, or observational equivalence properties [47]. None of these efforts are directly suitable for analyzing the access control mechanism in 5GC. The primary reason for this is twofold. First, some approaches [54, 55] have modeled only a single NF (i.e., MME/AMF) as the interface of 5GC by combining all NFs' functionalities into one. This prevents reasoning about the interactions between NFs. Second, other methods [37, 45] only model and analyze a subset of the protocol interactions, and quickly run into intractability and scalability issues when faced with more intricate protocol interactions. Additionally, these analyses [37, 45] require manual interventions from human experts to guide the provers, making them less automated and unmanageable for large systems. Moreover, all preceding approaches treat the 5G core network configurations as static, not accounting for changes during network operations. However, 5GC permits dynamic configuration updates. Therefore, prior studies fall short in assessing such key features inherent in the 5GC access control framework.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0050-7/23/11...\$15.00

<https://doi.org/10.1145/3576915.3623113>

Challenges. The formal analysis of the OAuth-based 5GC access control mechanism presents several challenges. (A) *Incomplete and non-compliant open-source core*: Available open-source implementations are often incomplete and non-compliant with specifications. As our goal is to formally analyze the access control design, we cannot rely on any existing implementations. Therefore, we base our analysis solely on the technical specifications provided by 3GPP. (B) *Intractability*: A typical 5GC configuration incorporates a vast array of NFs, services, and operations, coupled with dynamic updates and complex authorization logic. This drastically amplifies the complexity and state space of the system. Each component can be defined with hundreds of configuration attributes. When configurations from all NFs are combined, they form a singular 5G Core configuration. Given this structure, the 5GC can potentially have millions of semantically valid network configurations. Modeling 5GC systems for all potential network configurations and then analyzing each of them present an intractable task. (C) *Ambiguity and underspecification*: The technical specifications often lack clarity and are underspecified, potentially leading to multiple interpretations and incorrect implementations.

Approach. We reformulate the problem of analyzing the access control mechanisms of the 5G Core into a model-checking problem. However, the highly configurable and customizable nature of the 5G Core turns the problem to an undecidable parameterized model checking problem. We, therefore, draw inspirations from parameterized model checking paradigm and design 5GCVerif with an emphasis on soundness rather than completeness. In essence, 5GCVerif employs the *cutoff* principle [33] by leveraging *small model theorem* [34]. This principle stipulates that if a certain property is verified for a system up to a certain size, i.e., for a certain number of NFs in the 5G Core, then the property is verified for a system of any size. Empirically, we observe that a 5GC system with five NFs effectively represents most access control communications in larger 5GC systems. Therefore, we choose to model a 5GC system with five NFs to capture the complex details of 5GC's access control mechanism. However, exhaustively generating and analyzing models for all possible network configurations is infeasible. We address this challenge by employing our model to initialize the network configurations randomly while ensuring the validity of configurations by imposing specification-compliant constraints on the generated attributes. This approach allows us to reason about different configurations of the 5GC using a single model, and ensures that the model checker focuses on discovering policy violations only within semantically valid network configurations.

5GCVerif adopts a modular design for modeling messages, resources, and NFs of a 5G Core. This enhances customizability and extensibility, allowing the inclusion of arbitrary numbers of NFs in our model. 5GCVerif addresses scalability issues by designing several domain-specific data, behavior, and predicate abstraction techniques. Such abstractions effectively capture the essential characteristics of the access control mechanism while reducing complexity. In our model, we assume that an adversary gains control over a single compromised NF. This allows the adversary to inherit the NF's original capabilities, and it actively attempts to initiate requests with possibly forged parameters, aiming to ultimately elevate its privileges to access resources or perform operations that the compromised NF was not initially authorized to. We incorporate the

adversary's capabilities into our abstract model and obtain a threat instrumented model. 5GCVerif analyzes if the threat instrumented model satisfies the access control properties. Counterexamples unveiled by 5GCVerif represent violations of 5GC's access control policies. We manually inspect the trace of each counterexample and record the reported attacks. Inspired by the Counterexample-Guided Abstraction Refinement (CEGAR) technique, we adopt an iterative workflow. However, instead of refining the model, we modify each access control property to ignore already observed vulnerabilities and test the updated property against the model until violations are no longer encountered.

Findings. We have tested 55 properties against our 5G Core model and identified 5 new vulnerability types, each resulting in multiple attack scenarios. These vulnerabilities can allow attackers to obtain unauthorized entry to sensitive user information, illegitimate access to restricted services, and denial-of-service against benign NFs.

Contributions. This paper makes the following contributions:

- We present the first scalable formal model of the 5G core network's access control mechanism.
- Based on the formal model, we introduce 5GCVerif, an adversary-controlled framework to conduct systematic formal analysis of the access control mechanism of 5G core network. To the best of our knowledge, no prior work has formally specified or verified core components of 5G systems.
- We tested 55 security properties with 5GCVerif and found 5 new classes of exploitable privilege escalation vulnerabilities in the technical specifications. We confirmed that the identified attacks are possible in open-source 5G Core implementations. The model, properties, and findings are all available on GitHub [4].

Responsible disclosure. We shared our findings with GSMA [9]. They acknowledged with CVD-2023-0069 [10] for all our findings.

2 BACKGROUND

In this section, we provide a primer of 5GC network architecture, its communication model, and the enforced access control schemes.

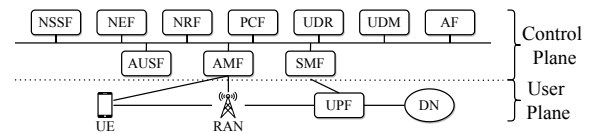


Figure 1: Core network architecture of 5G

2.1 Service Based Architecture

The 5G Core adopts a microservice-like design, where it divides its operations into functional blocks called Network Functions (NFs). NFs are designed to be hosted in virtual machines or containers on the cloud. Figure 1 demonstrates the core network architecture of 5G systems. NFs interact with each other to provide functionalities like authentication, security, and session management of cellular devices (also known as User Equipment or UE) traffic. A Network Repository Function (NRF) allows other NFs to register and discover each other, and is the main focus of this study. The Access and Mobility Management Function (AMF) provides UE registration, connection, and reachability management. The Unified Data Repository (UDR) provides storage for subscriber and policy-related data. The Unified Data Management (UDM) accesses UDR and manages user identity and generates authentication credentials.

NF communication model. NFs communicate with each other through Application Programming Interface (API) over HTTP. The interactions are request/response and subscribe/notify messages between NFs. In a typical API interaction between two NFs, the requester NF is referred to as the NF service consumer, and the target NF is referred to as the NF service producer. For ease of exposition, we refer NF service consumer as consumer NF or only consumer (NF_C) and NF service producer as producer NF or only producer (NF_P) unless otherwise stated. In general, an NF can act as a consumer, a producer, or both.

Services and operations. To perform its functionalities, each NF provides a set of services. For example, UDM provides different services for subscriber data management, UE context management, and UE authentication. Each NFService performs multiple operations designed to work on related data or achieve similar objectives. These operations include retrieval, modification, or removal of specific data and creation, modification, or deletion for subscriptions of data changes. Specifically, UDM's subscriber data management service provides operations for retrieval of UE context data and subscription for notifications of change in subscriber data.

2.2 Network Slicing

5G architecture also enables the ability to differentiate the levels of service offered to different applications and customers through the logical slicing of networks. At a high level, network slicing is a resource isolation mechanism where the physical network resources are broken down into isolated virtual blocks. To achieve such isolation, slicing in the core network is essential to allow the managing of data flow. Based on the network requirements, each slice can selectively implement different combinations of NFs, and customize or scale multiple instances of the same NFType. 3GPP uses the term sNssai, or single network slice selection assistance information, to uniquely identify a network slice. It can be considered as an identifier for a network slice for the scope of this paper.

2.3 Access Control in 5G Core

The SBA design of the 5G Core requires a careful specification of how communications between NFs are limited and how operations provided by each NF are protected. 3GPP specifies to use OAuth 2.0 framework [46] to authorize API interactions between NFs. The use of OAuth framework in the 5GC is optional but recommended as the access control mechanism between NFs. OAuth 2.0 is a well-established framework to govern authorization in a virtualized system. It is based on a central authorization server that issues accessTokens to clients to grant access for invoking API calls. In a 5G Core, the NRF plays the role of the authorization server.

AccessToken scope. A *scope* field in the accessToken defines the range of operations an NF_C with the token is allowed to access. Scopes are predefined by 3GPP and are included as part of OpenAPI specifications [3, 21]. Two types of scopes are defined, *service-level* and *operation-level* scopes. *Service-level* scopes are coarse-grained and shared across all operations of a service. For instance, scope *namf-comm* is shared across all the communication services provided by AMF. *Operation-level* scopes, on the other hand, are more fine-grained, and specific to a set of related operations, e.g., *nudm-uecm:amf-registration:write* is only used for operations in UDM that update amf-registration information. Operation-level scopes can

still be shared between a few related operations accessing the same group of resources. Accessing an operation controlled by operation-level scopes requires the accessToken to contain both the operation- and service-level scopes. 3GPP defines operation-level scopes only for a few services, while for others, it specifies only service-level scopes. In fact, 3GPP considers the use of operation-level scopes or even service-level scopes optional when network operators opt out of enforcing access control mechanisms. For our analysis in this paper, we consider the most conservative/strictest scenarios, i.e., operation-level scopes are enforced whenever they are defined, and if not, service-level scopes are imposed.

Table 1: Attributes in NFProfile of an NF instance.

Attribute Name	Description
nfInstanceId	Unique identifier of the NF Instance
nfType	Type of NF (AMF, SMF, ...)
sNssais	Identifiers of network slices this NF serves
allowedNFTypes	Types of NFs allowed to access this NF
allowedNssais	Identifiers of slices that are allowed to access the NF
nfServices	Services produced by this NF instance
...	...

NFProfile. Each NF manages a list of properties/attributes called Network Function Profile (NFProfile). Table 1 outlines essential fields present in an NF profile. The nfServices attribute within an NFProfile contains a list of service-specific attributes, including lists of allowed NFTypes (allowedNFTypes) and allowed slices (allowedNssais) that are permitted to access the corresponding service. If there is a conflict between the attributes specified in NFService and those in NFProfile, the attributes in NFService take precedence.

NFProfile is managed by network vendors who have the freedom to modify most fields. For example, a vendor can modify *allowedNssais* of a producer to restrict the slices this NF serves. Values of certain fields are, however, guided by 3GPP. Unfortunately, these instructions are sometimes vague and scattered across multiple Technical Specifications (TS).

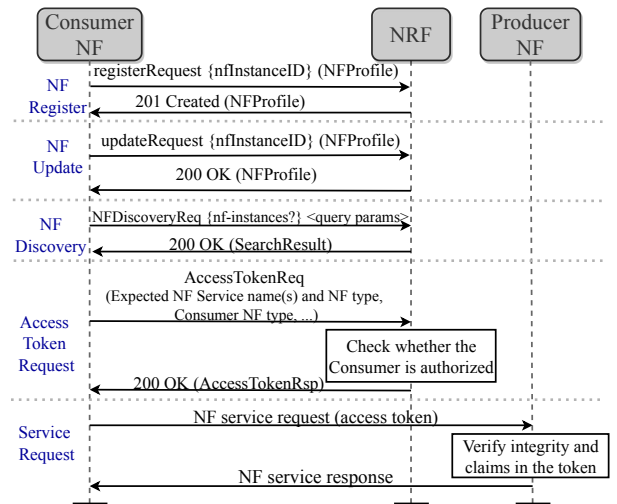


Figure 2: Sample interactions between NFs

NF interactions. Figure 2 shows key interactions between a pair of generic consumer and producer NFs, as well as the NRF, which

authorizes the communications. Before requesting or providing a service to others, each NF goes through a registration process in which it issues its NFProfile to the NRF. Any NF can also update its profile at any time with an *NFUpdateRequest* to the NRF. An NF_C can query for information on available producers, including a list of services and operations each producer provides through an *NFDiscoveryRequest*. An NF_C can also specify the NFType and sNssai information (i.e., slice ID) of the producers it hopes to discover in the query parameters, and NRF responds with the NFProfiles and endpoints of the producers the NF_C has access to.

Before an operation provided by some service is granted by the NF_P, the NF_C needs to first obtain a valid accessToken with appropriate scopes from the NRF by invoking the *accessTokenRequest* operation. A valid *accessTokenRequest* includes the consumer's NFInstanceID, the target service and the corresponding scopes it wishes to access, the NFInstanceID or NFType of the NF_P, etc. Upon receiving such a request, the NRF checks whether the NF_C is authorized to access the requested services by comparing the request parameters against the authorization attributes (e.g., *allowedNFTypes*, *allowedNssais*, etc.) of the NF_P's NFProfile, and issues an accessToken if the check succeeds [22]. The accessToken is a digitally signed JSON Web Token (JWT) [57, 58] which carries various essential information, including the NFInstanceID of the issuer (NRF), the subject (NF_C), the NFInstanceID or NFType of the audience (NF_P), scopes, and the expiration time. However, 3GPP does not detail exactly how the NRF performs the whole authorization checks [22, 23]. In this work, we consider a conservative approach by assuming that the NRF performs all possible checks within its capabilities properly. More discussion can be found in Section 4.

Finally, the NF_C initiates *NFServiceRequest*, which is the API call to NF_P and includes accessToken. The NF_P verifies the accessToken attributes, e.g., *scope*, expiration time, etc., and grants access to the resource or service to the NF_C only if the verification is successful.

3 DESIGN OVERVIEW

In this section, we present our threat model and provide a motivating example that highlights the need for formal analysis of 5G access control. We then discuss the high-level workflow and insights underpinning the design of 5GCVerif.

3.1 Threat Model

As opposed to previous generations, the significant shift in the openness of the 5G core network, coupled with the potential inclusion of third-party NFs, e.g., by Mobile Virtual Network Operators or MVNOs [5, 20], greatly amplifies the risk of malicious entities gaining control over an NF within the 5G systems. As shown by prior research [2, 16], a variety of vulnerabilities can lead to an NF being compromised and controlled by an attacker, including flaws in cloud-based microservices, configuration mishaps [11, 17], malicious or vulnerable dependencies [12, 13], or software vulnerabilities [7]. Additionally, the 5G Core allows the inclusion of third-party NFs, some of which could have malevolent intentions. This paper considers that the attacker's ultimate goal would be to elevate its privilege to access resources or perform operations the compromised NF was originally not allowed.

For our analysis, we consider an adversary with the following capabilities: **(A-1)** The adversary gains full control of an already

registered consumer NF while other NFs remain benign. As a result, the adversary can create and send any network packets on behalf of the compromised NF. **(A-2)** All communications between the compromised NF and other NFs are properly authenticated and encrypted and thus honest NFs cannot directly identify the malicious NF. Contrary to the Dolev-Yao [48] adversary model generally utilized for analyzing the *authenticity* of communication protocols [6, 36], the adversary in our threat model does not need to intercept, drop or alter messages in transit. **(A-3)** Provided that the Operations, Administration, and Management (OAM) system permits, an NF including the attacker's controlled one can alter the NFProfile by modifying certain attributes of the compromised NF. Our model governs the permissions of the adversary, providing the flexibility to switch between conditions under which no NFProfile update is allowed, only specific fields can be updated, or all fields are allowed to be updated. Further discussion on OAM can be found in Section 4.3. **(A-4)** We assume that NRF cannot be compromised by the adversary because (i) it has higher trust requirements than others; and (ii) since the NRF is tasked with performing all access control checks in the OAuth 2.0 framework [46], a compromised NRF negates the entire access control mechanism. Our threat model only considers a single malicious NF and does not account for the possibility of multiple malicious NFs colluding. This threat model aligns with 3GPP's Technical Report (TR) on SBA security [25].

3.2 A Motivating Example

We present a simplified core network scenario (Figure 3) to demonstrate the necessity of our model checking approach. It includes a consumer NF instance (C_1), two candidate producer NF instances (P_1 and P_2), and the NRF. The relevant NFProfiles for C_1 , P_1 , and P_2 are detailed in Table 2.

Table 2: Simplified NFProfiles for C_1 , P_1 and P_2

NFProfile of C_1	NFProfile of P_1	NFProfile of P_2
nfType: AMF	nfType: UDM	nfType: UDM
sNssai: 1	sNssai: 1	sNssai: 2
NFStatus: Registered	NFStatus: Registered	NFStatus: Registered
NFService:	NFService: Nudm_UECM	NFService: Nudm_UECM
Namf_comm	scope: Nudm_UECM	scope: Nudm_UECM
	sNssai: 1	sNssai: 2
	allowedNFTypes: AMF	allowedNFTypes: AMF
	allowedNssais: 1	allowedNssais: 2
	operation: Nudm_UECM_Get	operation: Nudm_UECM_Get

In this setup, both P_1 and P_2 of NFType UDM offer the same NFService Nudm Context Management (Nudm_UECM), and allow consumers of NFType AMF to access the service through *allowedNFType* attribute. However, P_1 serves only sNssai 1 (i.e., slice #1), so only AMF instances from sNssai 1 can access P_1 's resources (indicated by *allowedNssais*). Similarly, P_2 is associated with sNssai 2 and only allows access of AMF instances from sNssai 2.

In this simplified setup, the consumer AMF instance C_1 , being a participant of only sNssai 1, is authorized to *discover* P_1 , and to receive an accessToken to access Nudm_UECM services provided by P_1 . On the other hand, C_1 should neither be authorized to discover nor be able to acquire an accessToken for P_2 as P_2 serves only the consumers of sNssai 2. The accessToken that C_1 received from NRF to access Nudm_UECM service provided by P_1 contains the following information: **<issuer:** NRF, **<subject:** C_1 , **<audience:** P_1 , **<scope:** Nudm_UECM). Upon discovering P_1 's NFProfile and acquiring the

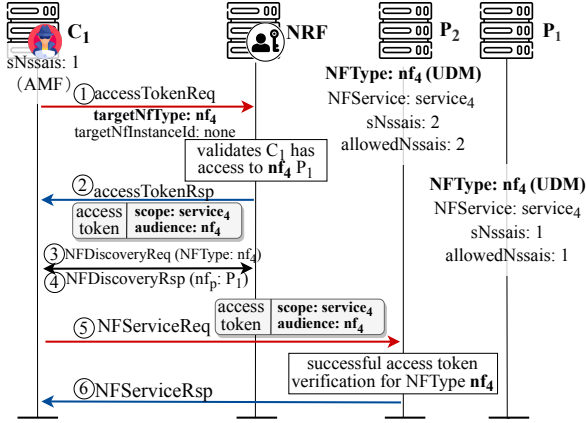


Figure 3: Demonstration of the motivating example.

accessToken, C_1 can invoke *Nudm_UECM_Get* operation and thus obtain the access to desired information, e.g., UE Context from P_1 .

At first glance, the above setup looks typical for an access control interaction in the 5GC and appears to be not vulnerable. However, if C_1 discovers P_2 's endpoint information (e.g., IP address or domain name of P_2) using techniques like network scanning [32, 49], it may exploit a flaw in the current authorization logic enforced in the 5GC to gain unauthorized service from P_2 . In the *accessTokenRequest* packet, instead of setting the *targetNFInstance* attribute to P_2 , which would result in NRF denying the request, C_1 can choose to pass in only the *targetNFType* attribute, and set it to be UDM. In this case, while verifying the *accessTokenRequest*, NRF will examine all registered NFs of the specified NFType in the *targetNFType* attribute (in this scenario, P_1 and P_2). If at least one of them (here P_1) allows access to the consumer, NRF will approve the *accessTokenRequest*, and respond with an accessToken with the producer's NFType set as the *audience* (i.e., $\langle \text{audience: UDM} \rangle$), instead of the specific authorized producer's NFInstanceID (i.e., $\langle \text{audience: } P_1 \rangle$). As a result, the generated accessToken will authorize C_1 's access to all UDM producers in the network, irrespective of consumer's sNssais. Later in an *NFServiceRequest*, if C_1 passes this accessToken to P_2 , P_2 will verify the *audience* attribute of the accessToken and learn that C_1 is allowed to access any UDM instances. Hence, P_2 will approve the *NFServiceRequest*, providing unauthorized access to C_1 .

It is evident from the example that in order to allow a resource access from a consumer, both the NRF and the producer need to collaboratively perform intricate authorization checks. Note, for the illustration purpose, we only present and discuss the key information of an attack on the above. Real-world 5G system deployments involve numerous core network settings and NF configurations with diverse attributes. Furthermore, the NFProfiles and core network setup within a 5G Core are dynamic, subject to updates by the network operator or the NF itself. This leads to numerous attribute combinations and network behaviors, making it challenging to identify subtle vulnerabilities, such as the one illustrated. Manual inspection of network setups, NF configurations, and potential complex NF interactions is extremely time-consuming and error-prone. Hence, a formal and systematic analysis of the 5G Core's access control is vital for its correctness and security.

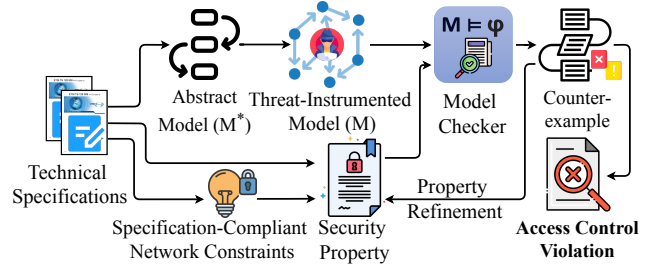


Figure 4: 5GCVerif architecture

3.3 Overview of 5GCVerif

Model reduction. A 5G Core may comprise an arbitrary number of producer and consumer NFs, each containing an arbitrary number of services and operations. This makes the analysis of 5GC access control policies an instance of the parameterized system verification problem: an undecidable problem [60] that is parameterized by the numbers of producers, consumers, services per NF, and operations per service. We tackle this problem with the notion of *cutoff* [33], that is, if a particular property is verified (or violated) for a system up to a certain size - specifically, up to a certain number of producer and consumer NFs, then it can be considered as verified (or violated) for a system of any size. The precise cutoff is determined by system resources and the required level of accuracy in the abstract model. According to *small model theorem* [34], a sufficiently large system with N producers and M consumers can be considered symmetric if we can project its set of reachable states $S(N, M)$ onto a smaller system with K (where $K < M$) producers and L (where $L < N$) consumers. This implies that if we can capture all possible interactions between M producers and N consumers in the large system within K producers and L consumers in the smaller system, then any property that holds true for the smaller system will equally hold true for the larger system.

As per 3GPP's TS 33.501 [22], all NFs use the same sets of authorization parameters and authorization checks. For instance, although API requests *Nausf_UEAuthentication_authenticate* (from AMF to AUSF) and *Nudm_UEAuthentication_Get* (from AUSF to UDM) differ in functionalities, similar authorization checks are required before granting access. Hence, by abstracting specific API functionalities and applying the above insight, we can map each API request to an abstract API request (i.e., *NFServiceRequest*), and the participating NF pair to a consumer and a producer NF instances.

However, using just one pair of consumer/producer, we cannot distinguish resources owned by distinct consumers/producers in multi-consumer/producer scenarios. Thus, we need to account for an additional consumer/producer NF pair in our model. Leveraging this observation and the small model theorem, we analyze the *design* of 5G access control mechanisms with five NFs: the NRF, two consumer NFs and two producer NFs. This strategy also aligns with the general notion of security analysis, where the focus is on soundness rather than completeness. In other words, if our methodology reports a violation, it is indeed a violation. However, we do not claim to detect all possible violations.

A detailed discussion on 5GCVerif components is provided in the following sections, while an overview can be found in Figure 4.

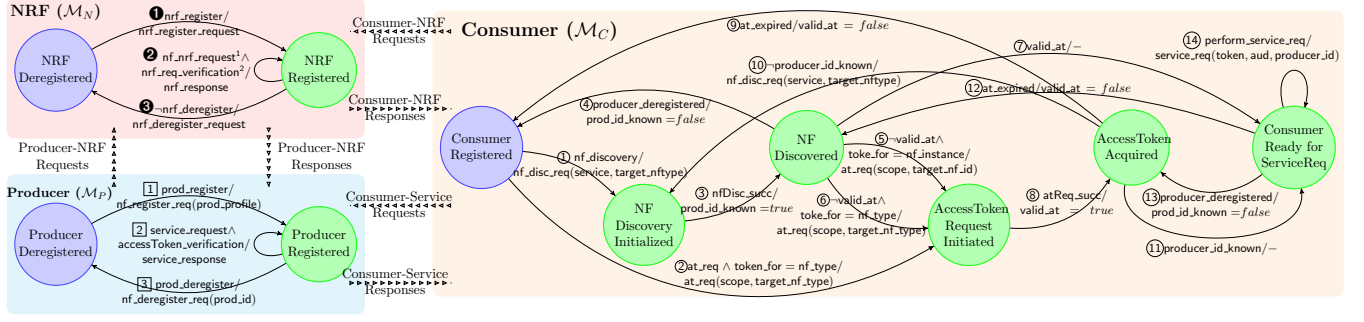


Figure 5: Simplified FSM of the 5G Core access control model. A detailed list of all request types (nf_nrf_request^1) to NRF and the verification performed at NRF ($\text{nf_req_verification}^2$) can be found in Table 3.

Model construction (\mathcal{M}). The initial step of 5GCVerif involves constructing an *abstract model* \mathcal{M}^* , comprising a set of communicating finite state machines (FSMs) as illustrated in Figure 5. \mathcal{M}^* abstracts the access control mechanism of the 5G Core specified by 3GPP Release 17 [27] in four Technical Specifications [22–24, 26]. This abstract model operates at a propositional logic level that encapsulates the generic NF interaction logic (Figure 2). Each NF is abstracted as an FSM, represented by a tuple $\langle \mathcal{I}, \mathcal{O}, \mathcal{V}, \text{Init}, \mathcal{A} \rangle$, where \mathcal{I} represents a finite set of input variables; \mathcal{O} : a finite set of output variables; \mathcal{V} : a finite set of state variables; Init: a set of initial states, and \mathcal{A} : a finite set of variable assignments in \mathcal{V} defining the transition relation of the system.

We represent five NFs as five FSMs in \mathcal{M}^* : two consumers (\mathcal{M}_{C_1} and \mathcal{M}_{C_2}), two producers (\mathcal{M}_{P_1} and \mathcal{M}_{P_2}), and an NRF (\mathcal{M}_N). FSMs interact with each other by transmitting messages through authenticated communication channels. For simplicity, we model the communication channel between any two FSMs, such as \mathcal{M}_{C_1} and \mathcal{M}_{P_1} , with two unidirectional channels: one from \mathcal{M}_{C_1} to \mathcal{M}_{P_1} and another in the reverse direction.

Our threat model considers a single registered consumer NF that has been compromised by a malicious actor. This entity actively seeks unauthorized access to the resources/services provided by benign producer NFs. The malicious NF may send arbitrary requests (e.g., *NFUpdateRequest*, *NFDiscoveryRequest*, *accessTokenRequest*, and *NFServiceRequest*) to any NFs in the 5GC, and can manipulate the input parameters of these messages on behalf of the victim NF. We incorporate such adversary capabilities into a consumer FSM (\mathcal{M}_{C_1} or \mathcal{M}_{C_2}) in \mathcal{M}^* to derive the threat-instrumented model \mathcal{M} . Additionally, \mathcal{M} supports parallel execution of multiple *NFServiceRequests* for the tuple $\langle C, P, \mathcal{R} \rangle$, where $C \in \{C_1, C_2\}$, $P \in \{P_1, P_2\}$, and \mathcal{R} denotes an instance of *NFServiceRequest* from C to P .

Security properties (Φ). We aim to verify the following high-level access control property (also referred to as security property in this paper): *An NF_C can access resources only if it is authorized.* For an NF_C to access sensitive services/resources (denoted as R_P^S) from an NF_P , it must first discover the producer’s NFProfile (denoted as R_N^P), and then obtain an *accessToken* (denoted as R_N^A) from the NRF. Hence, in addition to R_P^S , we consider both R_N^P and R_N^A as sensitive resources. To guarantee the security of 5GC’s access control mechanism, both the NRF and NF_P must collaboratively verify a request. As such, we model security properties to verify

the access request of each of such sensitive resources. A detailed discussion is provided in Section 5.

Model checking process ($\mathcal{M} \models \Phi$). 5GCVerif’s model checking process is inspired by the CEGAR principle [44]. CEGAR-based approaches verify if a concrete system \mathcal{S} satisfies a property Φ by creating an abstract system \mathcal{M}_a from \mathcal{S} , and testing if Φ complies with \mathcal{S} . If compliance is concluded, \mathcal{S} is deemed to satisfy Φ . Otherwise a counterexample π is generated. If π is realizable in \mathcal{S} , a failure in verification is found. Conversely, if it cannot be realized, \mathcal{M}_a is refined to exclude the spurious π , and the process is repeated until either \mathcal{S} satisfies Φ or a realizable counterexample is found.

In our approach, we opt to refine the verification property instead of further refining the threat instrumented model \mathcal{M} . For an access control property Φ , 5GCVerif reformulates it into the form $\sigma \rightarrow \Phi$, where σ denotes a refinement expression, and \rightarrow represents logical entailment. This specific refinement of the property enables the verification process to focus solely on execution traces/paths in \mathcal{M} that satisfy σ . This approach also allows 5GCVerif to filter out previously identified counterexamples from the succeeding iterations of the model checking process.

Manually creating models for every unique core network configuration and then testing them against properties is a combinatorial explosion problem. Additionally, analyzing the counterexamples and refining the properties for each configuration would require significant time and effort. Furthermore, not all configurations are semantically valid. In other words, numerous network configuration parameters, when generated randomly, may not correspond to any logically correct real-world 5GC systems as some parameters are interdependent. To address this challenge, during model checking, we incorporate additional *constraints* (represented with logical formula, e.g., $\Psi = \psi_1 \wedge \psi_2 \wedge \dots$) into the refinement expression σ , i.e., $(\sigma \wedge \Psi) \rightarrow \Phi$. The additional constraints represent the classes of specification-compliant core network configurations. This approach allows us to reason about different configurations of the 5GC system using a single model and guide the model checker to focus on exploring specific types of execution paths in which core network configurations are semantically valid. Additionally, we introduce other constraints to further control (i.e., add/remove) adversary capabilities to pinpoint the minimal adversary capabilities leading to violations. More discussions about such refinements can be found in Section 6.

4 MODEL CONSTRUCTION DETAILS

In this section, we delve into various aspects detailing the design choices and decisions made to address major modeling and implementation challenges.

4.1 Modeling Transitions, States, and Resources

FSMs. The authorization process of 5GC can be represented as communicating FSMs between an NF_C (M_C), and an NF_P (M_P) or the NRF (M_N), each featuring multiple states and transitions (Figure 5). Interactions among M_C , M_P , and M_N constitute the abstract model M^* . The NF_C begins at ConsumerRegistered state, and if an `nf_disc_req` (①) is successfully verified by the NRF (②), NF_C transitions to NFDiscovered state. In the NFDiscovered state, NF_C obtains from NRF the NFProfiles (③) of candidate NF_P s and can request for an `accessToken` with `at_req` (⑤) or (⑥). If the authorization checks by NRF pass (②), the NF_C receives the `accessToken` (⑧) and proceeds to ConsumerReadyForServiceReq state. The NF_C can then request access to NF_P 's services (⑭). The `service_req` will be granted if NF_P verifies the `accessToken` (⑫) provided by NF_C .

Table 3: Summary of request verification policies by NRF

nf_nrf_request to NRF	nrf_req_verification by NRF
NFRegister	Primarily OAM mandates which NF is authorized to register on the network. In our model, as we aim to test all NFProfile combinations, we do not enforce any checks for this request.
NFUpdate	OAM determines whether the NF can update its profile successfully. In our model, the attributes allowed to be updated is controlled by a set of control parameters. A detailed discussion can be found in Section 4.3.
NFDiscovery	An NF_C should only be allowed to discover NF_P it is permitted to access. NRF cross checks the authorization parameters of the NF_P with the attributes in the NF_C 's NFProfile.
accessToken Request	An NF_C should only be allowed to acquire <code>accessToken</code> to services provided by an NF_P it is permitted to access. A similar check is performed to that of the NF Discovery.
NFDeregister	NFs are allowed to deregister under all circumstances.

States. States are defined by a combination of their names and associated variables. For instance, Boolean variables `prod_id_known` and `valid_at` in M_C are used to denote whether the NF_C has obtained an NFProfile of its target NF_P , and a valid `accessToken` to access NF_P , respectively. As a participant transitions from a state to another, both the state name and the variables are updated accordingly to reflect the semantics of its current state.

Transitions. Transition labels shown in Figure 5 follow the form of "conditions/actions". Conditions are propositional logic formulas specifying the prerequisites to trigger a transition, while actions denote the sequence of operations that the FSM executes (in the order they appear) once the transition is taken. While actions can be empty (denoted by '-'), conditions can not. Conditions are modeled using state variables, environment variables (random Boolean variables set non-deterministically by the model checker), and input messages. Actions are modeled using assignment operations on state variables, environment variables, and output messages.

Resources. We model NF resources, including NFProfiles, `accessToken`s, and NFServices, as distinct modules using the MODULE construct in the SMV language [40]. These modules resemble struct or class constructs in C/C++ in that they can store and group data fields, define transitions based on those fields, and be instantiated

with different parameters as required. This modular design enhances extensibility and customizability, as it allows creating multiple instances of a particular module type. To populate attribute fields within these modules, we limit the domain of possible values of each attribute to a predefined set of ENUM values of reasonable size. For instance, a real-life network may have numerous slices; however, our model restricts `sNssais`-related attributes to only four values: slice 1–4. Similarly, NFService names provided by NF_P s are defined as arbitrary string values; however, our model chooses their values only from a predetermined pool of strings. As we aim for soundness, these design choices aids in modeling complex details while maintaining scalability. These modeling disciplines also enable us to model a multi-consumer and multi-producer architecture (e.g., used by property Φ_4 in Section 5) while preserving the essential characteristics of a real 5GC.

Messages and communication channels. For tractability, we opt to model only the critical message data/fields related to the access control mechanism (e.g., attributes listed in Table 1). We model the communication channel, the conduit for sending and receiving messages, using a *shared resource architecture*. In this setup, an instance of the message module is allocated to each sender-receiver pair. Once a message is transmitted by the sender via the designated channel, it becomes immediately available to the receiver.

4.2 Threat Instrumentation

5GCVerif takes the abstract model M^* and instruments it to incorporate an adversary to obtain the threat instrumented model M . Based on our threat model (Section 3.1), 5GCVerif models adversary capability (A-1) by randomly selecting one of the consumers M_C in M^* as the adversarial party and introduces the following instrumentation to M_C : first, M_C may break the FSM's stateful nature by sending any request message to NRF or producers at any time; while sending a request message, malicious consumer may set any arbitrary values to the request parameters, and it may change the message data or parameter values. To model adversary capability A-3, 5GCVerif allows M_C to update its NFProfile at any time via `NFUpdateRequest` given OAM approves the update, per the discussion in the following Section 4.3.

4.3 Tackling Underspecifications in Modeling

Due to high complexities of NFs and their services, 3GPP frequently falls short in conveying policies consistently in natural language. This results in ambiguous, conflicting, or underspecified instructions. We broadly categorize these as *underspecifications*.

For instance, (U1) although the specification mandates that NRF is responsible for validating all authorization requests, it is not clearly instructed how NRF should validate the authorization parameters. For instance, on `accessTokenRequest` verification, TS 29.510 [23] states the following.

"An access token request should be rejected if the requester NF is not allowed to access the target NF based on the *authorization parameters* in the NF profile of the target NF. The authorization parameters in NF Profile are those used by NRF to determine whether a given NF Instance/NF Service Instance can be discovered by an NF Service Consumer in order to consume its offered services (e.g. "allowedNFTypes", "allowedNFDomains", etc.)."

Here, the definition of *authorization parameters* is not precise and the specification fails to provide a complete list of such parameters. The specification continues as follows.

"Based on operator's policies, an access token request not including the requester's information necessary to validate the authorization parameters in the target NF Profile *may* be rejected."

The specification advises verifying the requester's information, but it is unclear whether the consumer's NFProfile should be cross-referenced with *the producer's NFProfile authorization parameters*.

(U2) While modeling *NFDiscoveryRequest*, we come across conflicting information about the required parameters in two different specification documents. TS 23.502 [24] asserts that, "For network slicing the NF service, consumer ID is a required input." However, TS 29.510 [23] defines consumer ID as an optional parameter.

(U3) The Operations, Administration, and Maintenance (OAM) system is responsible for configuring and managing network elements in the 5GC, including NFProfiles [24]. However, the precise roles and functionalities of the OAM system remain underdefined. As per 3GPP, the *NFRegister* and *NFUpdate* APIs are accessible to any authenticated NFs. This implies that a compromised NF may also modify its NFProfile without OAM system involvement.

Table 4: NF Update Schemes and their usecase in our findings

NFUpdate Scheme	Attributes Allowed to Update	Usecase in Finding
1	Only parameters that are non-critical for authorization decisions, such as load, balance, priority, and NFStatus, may be updated.	Findings 1, 2, 4, 5
2	Besides Scheme 1, the compromised consumer's own authorization attributes (e.g., allowedNssais, allowedNFtypes, etc.) may also be updated.	No attack
3	Besides Scheme 2, other essential attributes like sNssais and nfDomain, which determine the consumer's core capabilities, can be added, modified, or deleted. More granular control can be achieved using additional constraints in the property.	Finding 3 depends only on the removal of sNssais.
4	All attributes (unrealistic).	Not tested

Address underspecifications. To address U1 and U2, we choose to model the most conservative policy in \mathcal{M} so that we ensure the strongest authorization guarantees. This is because we aim to find vulnerabilities in 5GC system even when the most conservative/strictest policies are chosen by network operators. For instance, to address (U1), we model the NRF to perform not only a cross-profile validation check of every parameters related to authorization in the request message against the NF_P's NFProfile, but also a cross-check between the request message and the NF_C's NFProfile. This is done to prevent the spoofing of request parameters to match the NF_P's NFProfile, which would otherwise render the access control scheme ineffective. This issue has been identified as a vulnerability by others [28], but we do not consider exploits resulting from specification ambiguity. Similarly, for (U2), we consider the consumer's NFInstanceId mandatory.

As the behavior of OAM is not explicitly defined by the specification (U3), we aim to infer and model all possible behaviors of OAM. It enables 5GCVerif to analyze every potential scenario involving NFProfile updates. To achieve this, we introduce two environment variables in \mathcal{M} — *isOAMPresent* and *doesOAMApprove*. These variables represent if OAM exists, and if OAM approves an

NFProfile update request, respectively. If both are True, depending on the importance of each NFProfile attribute, we, further, regulate the update of attributes by another environment variable, *NFUpdateScheme*. We categorize attributes into four groups based on their significance in NFProfile updates. This constitutes four update schemes as summarized in Table 4. Considering that OAM can employ any of these schemes, we model all four possible NF update schemes in \mathcal{M} .

Similarly, for any access control policies for which alternative policies are explicitly or implicitly defined (e.g., optional policies suggested by 3GPP specifications using keywords *may* or *should*), we model all candidate policies in \mathcal{M} , as long as none are evidently weak or insecure. Using environment variable, we ensure that the model checker randomly explores and analyzes all possible behaviors of 5GC system until a counterexample is found. During our experiments, the model checker can non-deterministically set the values of these variables at runtime, or we can also control the variables' values via security properties to focus on investigating specific execution paths.

4.4 Modeling Authorization Logic

When the NRF receives an *NFDiscoveryRequest* or *accessTokenRequest*, it checks the input parameters and the NFProfile of the requesting NF_C against those of candidate producers. The verification process for these operations are defined in TS 29.510 [23]. Similarly, when an NF_P receives an *NFServiceRequest*, it validates the input parameters and *accessToken* attributes using verification conditions from TS 33.501 [22]. These conditions are scattered throughout the specifications and can be difficult to compile coherently.

Conceptually, the implementation of a verification logic is not complicated as it is just a collection of conditional statements. However, implementing the logic in SMV language is not trivial. This is because symbolic model checkers typically do not support loops, reference variables, or array of modules. To address this challenge, we unroll the loops in authorization logic and use implication operator (\rightarrow) of propositional logical formula to implement if – else constructs. To illustrate, consider a set of potential producer NF instances, denoted as $\mathcal{P} = [p1, p2, p3]$. Each producer possesses various attributes, like NFInstanceId (id) and NFType (nfType). Should the NRF, during the *NFDiscovery* phase for a NF_C, need to determine if any NF_P in \mathcal{P} has the NFType of AMF, the specific check it performs is presented in the following.

```
output = (i=p1.id & i=p2.id & i=p3.id) & (i=p1.id -> p1.nfType=AMF)
& (i=p2.id -> p2.nfType=AMF) & (i=p3.id -> p3.nfType=AMF)
```

4.5 Further Tackling Scalability

To prevent our model from running into state explosion problems, we leverage the following abstraction techniques.

Data abstraction. We employ data abstraction techniques to simplify complex structures of resources and attributes while retaining essential functional properties of the system. This approach disregards properties that are irrelevant to the verification task at hand. For example, the 3GPP specification defines NFInstanceId, a unique ID for each NFInstance, as a string composed of four universally unique identifiers (UUID) [23, 64]. However, we abstract this attribute as a simple integer value within a finite range.

Table 5: Different classes of security properties tested with 5GCVerif.

Property Class	Authorization Request	Sensitive Resource	Authorizing Agent	Property Target
Φ_1	NFDiscoveryRequest	R_N^P	NRF	Access
Φ_2	accessTokenRequest	R_N^A	NRF	Access
Φ_3	NFServiceRequest	R_P^S	Producer	Access
Φ_4	NFServiceRequest	R_P^S	Producer	Exclusivity

Behavioral abstraction. To streamline the verification process and to create a manageable model, we abstract away unrelated implementation details, focusing solely on the important behaviors. For example, all cryptographically protected messages (e.g., Access Token JWT [58]) are abstracted to their plain text format. Such abstractions do not affect the faithful representation of 5GC access control mechanisms because our threat model assumes all communications are cryptographically secure.

Predicate abstraction. To further reduce the state space, we apply several predicate abstractions. Intuitively, the idea is to model a predicate over a component instead of capturing all its details directly, thereby simplifying the model. Our representation of OAM (Section 4.3) is an example of this technique. We use only three variables to model the complex supervising behavior of an OAM while preserving all possible allowed update schemes. Besides, the expiration of access tokens, a crucial attribute for authorization, is modeled as a simple random Boolean variable *expired*.

5 SECURITY PROPERTIES

The high level security property we aim to verify is: *an NF_C can access resources: (i) NF_P's NFProfile (R_N^P), (ii) NF_P's accessToken (R_N^A), (iii) NF_P's service (R_P^S) only if the NF_C is authorized to.* Authorization for R_N^P and R_N^A is performed by NRF during *NFDiscoveryRequest* and *accessTokenRequest*, respectively. NRF cross-checks the *authorization parameters* in the target producers NFProfiles with that in the *NFDiscoveryRequest* and *accessTokenRequest* messages sent by NF_C. On the other hand, authorization for R_P^S is performed by the producer by validating the *accessToken* provided during *NFServiceRequest*. Therefore, we choose to test security properties on both authorizing agents, NRF and the producer, to verify if any resource grant violates the access control property.

For each property, we generally target to validate that *if a resource is granted to an NF_C, then the NF_C's NFProfile must match the authorization parameters set in the corresponding NF_P's NFProfile*. This results in three classes ($\Phi_1 - \Phi_3$) of security properties as shown in the first three rows of Table 5. For instance, Φ_1 states that *NRF's grant of R_N^P to the NF_C does not violate access control policy only if the NFProfile of the NF_P allows the NF_C to discover it*.

Property classes $\Phi_1 - \Phi_3$ can effectively capture Vertical Privilege Escalations (VPE) [18, 63]. VPE occurs when an attacker with lower-level privileges tries to access higher-level privileges within a system or application. Finding 2 in Section 7.1.2 presents an example of VPE. On the other hand, Horizontal Privilege Escalation (HPE) [18, 63] occurs when a user accesses resources of other users at the same privilege level. Finding 5 in Section 7.1.5 provides an instance of HPE. While $\Phi_1 - \Phi_3$ may capture some HPE instances, we introduce a distinct property class Φ_4 specifically to detect HPEs.

Φ_4 validates the exclusivity of resources: *if r is a sensitive resource meant to be exclusive to a specific group g , then r should not be authorized to access by an NF_C $\notin g$* . To illustrate, if consumer C_1 created a resource r in a producer P_1 that is exclusive to itself, then all other consumers, e.g., C_2 should not be able to access r ; or if r' is a resource exclusive for consumers in slice 1, then C_3 who only serves slice 2 should not obtain access to r' .

An *authorizing agent* (NRF or NF_P) must reject an *authorization request* even when a single attribute of the *authorization parameters* fails to be validated. Building on this insight, we decompose a property (Φ) into multiple simple properties (denoted as ϕ_i , where i is a natural number), each focusing only on a single *authorization parameter* associated with the corresponding sensitive resource. E.g., Φ_1 is broken down into 21 simplified properties. Two examples of the simplified properties of Φ_1 are: (ϕ_1) NRF must reject *NFDiscoveryRequest* if *allowedNssais* fails to match; and (ϕ_2) NRF must reject *NFDiscoveryRequest* if *allowedNfTypes* fails to match. Given the complexity of the generated counterexamples, testing these simplified properties focusing on a single *authorization parameter* simplifies the identification of vulnerabilities and their root causes.

6 MODEL CHECKING PROCESS

In this section, we demonstrate how model checking is performed during our workflow with an illustrated example.

6.1 Property Refinement Strategies

Adding constraints to Φ for generating specification-compliant network configurations. While verifying a security property, i.e., testing if $\mathcal{M} \models \Phi$, 5GCVerif needs to not only find the counterexample that violates the access control policies, but also generate a correct core network configurations for which the violation may happen. However, if the model checker generates network configurations completely at random without any guidance, the majority of generated configurations are unlikely to expose any access control vulnerabilities, and many of which are even semantically invalid with respect to the Technical Specifications. For example, if NFService 1 is a service of AMF, but the network configuration associates NFService 1 to an NF_P of type UDM, then the configuration is invalid as NFs of different NFTypes can not share the same NFService. Testing security properties against this invalid network configuration is therefore meaningless.

To address this challenge, we restrict the model checker to explore counterexamples to only within the specification-compliant NF configurations. We manually analyzed 3GPP specifications and 5G OpenAPIs to identify 6 major categories of constraints on NF configurations (shown in Table 6). For example, category #1 ensures that no NFServices with the same name can be assigned to producers of different NFType. This prevents the inconsistency presented in the example above. We instantiate each constraint type with the values of attributes in NFProfiles and craft 230 constraints in total. To apply a constraint σ_i to the model checker, we refine Φ as $\sigma_i \rightarrow \Phi$ to prevent the model from generating an invalid network configuration. We check each constraint-enforced property against the model by testing if $\mathcal{M} \models (\sigma_i \rightarrow \Phi)$.

Table 6: Constraints implemented in 5GCVerif

Constraint Category	Constraint Description	# Constraint Instances
(1) NFType Consistency	For any two NF services that have the same name, they should belong to the same NFType.	6
(2) NF Service Consistency	For any operations that have the same name, they should belong to the same NF Service.	24
(3) Resource-level Scope Consistency	For any two operations that have the same name, they should be assigned resource-level scopes of the same name.	24
(4) Operation Parameters Consistency	For any two operations that have the same name, they should have the same set of operation parameters.	24
(5) Authorization Parameters Consistency	For any operations that have the same resource-level scopes, their operation-level authorization parameters, i.e., <i>allowedNFTypes</i> and <i>allowedNFInstances</i> , should also be the same.	24
6: Parameter Sensitivity Consistency	For any two operation parameters that have the same names, their sensitivity should also be the same.	128

Finding minimal adversary capabilities for a violation. During threat instrumentation, we introduce different adversary capabilities, such as, updating a profile or spoofing access token request and service request messages. In this adversary setting, a counterexample found during model checking may exploit multiple adversary capabilities although not all those adversary capabilities are necessary to realize the attack. Hence, to find out the minimal and sufficient set of adversarial capabilities required to fulfill an attack, we introduce additional controls as refinements into Φ and test \mathcal{M} against each of the refined properties.

Refining Φ to suppress previously discovered counterexamples. Apart from the above controls and constraints, it is necessary to impose additional restrictions in the property to refine the model as part of the CEGAR framework (Section 3.3), and to suppress already discovered counterexamples. We discuss in detail this process with the illustrative example in the following section.

Encoding the refined property. The security property is an implication statement written in Linear Temporal Logic (LTL) [66] formula where the premise consists of all the control variables, constraints and refinement conditions, and the conclusion represents the high level security property we aim to validate.

6.2 Illustration of Model Checking Process

We demonstrate the strength of 5GCVerif in detecting over-privilege in the 5G Core through a running example.

Desired example property. The first property φ_1 we want to verify falls in the property category Φ_3 (discussed in Section 5) and describes: *During an NFServiceRequest, NF_C can access R_N^S only if it is allowed by NF_P's allowedNssais attribute in its NFProfile.*

Verification of φ_1 . Checking $\mathcal{M} \models \varphi_1$ yields a counterexample π_1 encompassing a trace of length 6. Each state in the trace is defined by 598 variables. A careful evaluation of π_1 reveals a novel attack exactly describing the exploit presented in Section 3.2. We name it *Confused Producer Attack*. Counterexample π_1 gives a 5G Core configuration illustrating an interesting scenario where the compromised consumer NF C_1 serving a specific network slice gets unauthorized access to a producer NF P_2 that is not supposed to authorize *NFServiceRequest* from that slice.

Refinement of φ_1 . \mathcal{M} contains a nondeterministic environment variable, *reqForSpecificProducer*, which if set to True, guarantees that the NF_C invokes *accessTokenRequest* for a specific NFInstance instead of for a general NFType. \mathcal{M} generates counterexample π_1 by disabling the control variable. In the refinement process, we manually forced the enabling of *reqForSpecificProducer* while leaving other parameters unchanged and test if \mathcal{M} can find other counterexamples. We denote the refined property φ_2 .

Verification of φ_2 . Does not produce counterexample.

Refinement of φ_2 . Verifying \mathcal{M} against φ_2 leads to no counterexample, suggesting that the constraints are overly strict and too static. We refine φ_2 to permit a single benign update of producers while keeping other conditions unchanged, resulting in φ_3 .

Verification of φ_3 . Upon verifying model \mathcal{M} against φ_3 , a counterexample π_2 emerges. Careful inspection reveals another previously unidentified attack, which we denote as *Token Reuse Attack*. This counterexample illustrates a network configuration where an NF_C C_1 gains unauthorized access to NF_P P_1 . This occurs when C_1 is initially authorized to access P_1 , and despite an update to P_1 's authorization parameter revoking C_1 's access, C_1 can exploit the vulnerability to continuously gain the access to P_1 's services. Further details of this attack are discussed in Section 7.1.2.

Further refinements of φ_3 . Similar to the above refinement examples, we further refine φ_3 by modifying different environment variables, adversary capabilities, malicious and benign update restrictions, etc. We also test other access control properties based on the verification of *accessTokenRequest* and the *NFDiscoveryRequest* messages. Detailed description of all vulnerabilities and exploits we uncovered is presented in Section 7.1.

7 IMPLEMENTATION AND EVALUATION

Implementation. 5GCVerif primarily uses nuXmv [40] symbolic model checker to construct \mathcal{M} and verify if $\mathcal{M} \models \Phi$. 5GCVerif consists of 9 modules. Each instantiated NF_C or NF_P consists of 15 states and 48 transitions. With two NF_P and two NF_C, 5GCVerif consists of 4,767 lines of code. We have tested 55 security properties consisting of 3406 lines of code in total. The model and security properties used in our experiments are available on GitHub [4].

Evaluation setup. We use a laptop with Intel i7-9750H CPU and 16GB DDR4 RAM. We demonstrate the effectiveness of 5GCVerif by illustrations of the vulnerabilities and attacks 5GCVerif uncovered in Section 7.1, followed by a briefly introduction of the time and resource consumption of 5GCVerif in Section 7.2.

7.1 Effectiveness of 5GCVerif

Following the workflow described in Section 6, we aim to demonstrate the effectiveness of 5GCVerif by answering the question: *how effective is 5GCVerif in finding access control violations in 5G?*

Using 5GCVerif, we have identified five classes of previously undiscovered access control vulnerabilities within the 5G Core. The uncovered vulnerabilities (summarized in Table 7) can potentially lead to a range of attacks, such as illegitimate acquisition of sensitive information, unauthorized access to services, and Denial of Service (DoS). For each counterexample produced by 5GCVerif, we manually validate the vulnerability and analyze its root causes.

We also attempt to confirm that the demonstrated attacks are present in open-source implementations of the 5G Core. However,

Table 7: Summary of 5GCVerif's findings

Attack	Vulnerability Description	Adversary Assumption	Notable Implications	Validation
<i>Confused Producer Attack</i>	A compromised NF _C obtains an accessToken for an authorized NF but misuses it to access a different NF of the same NFType in a different slice, where access should not be granted.	Attacker has knowledge of the victim's endpoint address.	Overprivileged access, sensitive information leakage	●
<i>Token Reuse Attack</i>	A compromised NF _C can reuse a previously saved and unexpired accessToken to a victim NF _P which it should no longer be allowed to access due to a policy change.	Attacker needs previous permission to the victim NF _P .	Policy change bypass, overprivileged access, sensitive information leakage.	●
<i>Default Overprivilege Attack</i>	By emptying the sNssais attribute in its NFProfile, a compromised NF _C can exploit 3GPP's flawed "allow by default" policy, accessing NFs in slices it should not be permitted to.	In the presence of OAM, it needs to approve attacker's removal of its sNssais.	Overprivileged access, sensitive information leakage	●
<i>Authorization Bypass Attack</i>	A compromised NF _C discovers NFs it should not be able to by misusing the sNssais attribute and set it as any slice it wishes. This attack exploits the lack of cross-check for this field in NRF.	NRF does not implement the cross-check between NFDISCOVERYRequest and the NFProfile of the consumer.	Sensitive metadata and authorization logic leakage, denial-of-Service	●
<i>Parameter Misuse Attack</i>	Once the compromised NF _C acquires an accessToken to a producer NF in a legitimate slice, it is also implicitly granted access to the same operations in other slices within the same NF. The attacker can retrieve, create, or alter information from slices they should not access, simply by supplying the corresponding query parameters.	-	Overprivileged access, sensitive information leakage, DoS	●

●: Attack is possible in free5GC. However, free5GC did not implement some features described in the attack scenario yet.

●: Attack is fully verified in free5GC.

none of the open-source 5GC projects [14, 15], except free5GC [8], implement the access control system. Free5GC incorporates OAuth 2.0 access control framework and provides the accessToken validation ability for producers, but it is primarily based on 3GPP Release 15, whereas our work adheres to the latest Release 17. As a result, we do not use free5GC as a baseline for validating our findings; instead, we use it to solely demonstrate that the issues identified in specifications exist in real implementations, thereby confirming their real-world implications.

The threat model for each attack scenario aligns with the one discussed in Section 3.1. It's important to note that, in all attack scenarios, the malicious NF_C is assumed not to update any authorization attributes ((i.e., NFUpdateScheme = 3 in Table 4)) unless explicitly stated otherwise.

7.1.1 Confused Producer Attack. The model-checking details are already presented in Section 6.2, along with a concrete case illustrating the exploitation of the vulnerability in Section 3.2. Hence, we refrain from discussing the same details here.

Additional adversary assumption. For the malicious or compromised NF_C C_1 , in addition to the threat model outlined in Section 3.1, it is also assumed that C_1 possesses knowledge of the victim NF_P's (P_2 in Figure 3) host/IP address. This information is required to establish a connection to the targeted NF, but the attacker is unable to acquire the endpoint of the targeted producer through a standard NFDISCOVERY process. Methods such as network scanning [32, 49] can be utilized to obtain this information, and we show that it could also be easily attained using the attack described in Section 7.1.4.

Attack verification. In free5GC, the NRF accepts *accessTokenRequests* that provide only the desired NFType. A malicious consumer with the generated accessToken by the NRF is capable of passing all validation checks enforced by producers in free5GC.

7.1.2 Token Reuse Attack. A counterexample (discovered during the running example in Section 6.2) shows that a malicious NF_C can continuously access an NF_P even after its permission is revoked. The interesting components in the counterexample found by the model checker includes a consumer NF C_1 and a producer NF P_1 .

C_1 serves only sNssai 2 (i.e., slice #2) whereas P_1 serves all sNssais and also authorizes consumers from any slices.

C_1 first invokes *accessTokenRequest* (for NFInstance P_1) to the NRF. NRF verifies authorization parameters and concludes that C_1 has permission for P_1 , and grants the consumer an accessToken T containing audience NFInstanceID as P_1 . However, in the mean time, producer NF P_1 invokes *NFUpdateReq* to the NRF to set *allowedNssais* to 1 meaning that only the consumer serving sNssai 1 can access P_1 's resource from now on. Effectively, it revokes C_1 's access to the service of P_1 as C_1 does not participate in sNssai 1. However, given the obsolete accessToken T is not yet expired, when the malicious C_1 uses the previously acquired accessToken T to invoke *NFServiceRequest* to P_1 , P_1 verifies the attributes in T and finds that it still authorizes access to services provided by P_1 , so the *NFServiceRequest* is successful. The accessToken has been granted to the consumer C_1 before its access has been revoked, and NRF cannot stop the old (but unexpired) token T from being used. As the producer can only verify attributes presented in the accessToken, it cannot determine that the token is no longer valid due to a lack of information, e.g., an *issue time* attribute in the token.

Attack verification. We verify that in free5GC, an NF_C can continue to access the producer using an outdated but unexpired accessToken even after NF_C's permission is revoked. Here, we update *allowedNFTypes* instead of *allowedNssais* to revoke NF_C's permission since the later is not yet implemented in free5GC.

7.1.3 Default Overprivilege Attack. The property φ we want to verify is in category Φ_2 (Section 5) and states, *an NF_C can access R_N^A only if it is allowed upon cross-checking allowedNssais attribute of NF_P against NF_C's NFProfile during accessTokenRequest*. We encounter this attack when verifying $\mathcal{M} \models \varphi$ while keeping the adversary update enabled.

5GCVerif provides a counterexample that demonstrates how a malicious NF_C C_1 , despite being restricted from accessing a benign NF_P P_2 due to sNssais restrictions, can acquire service from P_2 without the need to add sNssai 3 to C_1 's sNssais attribute. The critical components in the counterexample includes a compromised consumer C_1 residing in sNssai 1, and a benign producer P_2 serving only sNssai 3, as shown in Figure 6. This attack is a result of a

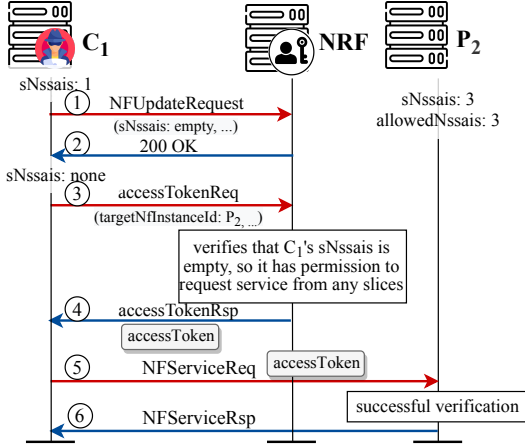


Figure 6: Demonstration of Default Overprivilege Attack

problematic specification in 3GPP's technical documents. In particular, 3GPP employs an "allow by default" principle (TS 29.510 [23]), where the absence of a specific attribute for an NF is interpreted as including all possible values for that attribute. As an example, if the *sNssais* field in the NFProfile of an NF_C is left empty, according to the specification, the NF_C has access to all network slices.

The specification, however, does not produce direct counterexamples. For example, an NF_C with an empty *sNssais* attribute can access NF_Ps in all slices, which may be unintended for network operators but is explicitly allowed in technical documents. The issue arises when a compromised NF_C, such as C₁ in this scenario, has a non-empty *sNssais* attribute at the outset.

The counterexample shows that C₁ initiates an *NFUpdateRequest* to remove sNssai 1 from *sNssais* attribute in its NFProfile (steps ①–② in Figure 6). Even if an OAM component is present in 5GC, this action is likely to be approved as it merely involves removing attribute values rather than adding new ones. However, now that C₁'s *sNssais* is empty, the default rule comes into play, and C₁ suddenly becomes eligible to acquire permission to access NF resources from all slices, provided other checks by NRF, such as *allowedNFTypes*, are successful (③–⑥).

This attack highlights one of the ramifications of the perplexing default strategy endorsed by 3GPP. While 3GPP specifies the "allow by default" behavior, they may not adequately underscore its implications. The attack targeting *accessTokenRequest* is just one example, as *NFDiscoveryRequest* is also vulnerable to analogous attacks. Moreover, aside from *sNssais*, other *authorization parameters* are also susceptible to similar exploitations. Depending on the implementation detailed in Section 4.3, if the OAM system prohibits even attribute removal during *NFUpdateRequest* verification, the aforementioned attack may become infeasible.

Attack verification. In free5GC, we successfully perform the attack where a malicious NF_C removes its *sNssais* through *UpdateNFInstanceRequest* API. However, since free5GC does not enforce slice checks, we cannot validate "allow-by-default" behavior. Nonetheless, such a rule is clearly stated in the Technical Specification, and we expect all implementations to follow it, in which case *Default Overprivilege Attack* is feasible.

7.1.4 Authorization Bypass Attack. The property φ we want to verify falls in category Φ_1 and describes, NF_C can access R_N^P only if it is allowed upon cross-checking *allowedNssais* attribute of NF_P against NF_C's NFProfile during *NFDiscoveryRequest*. As part of the property refinement process, we disabled all adversarial NF updates. This restriction suppresses *Default Overprivilege Attack*, and reveals this new vulnerability. The counterexample generated by 5GCVerif reveals that a malicious NF_C C₁ can discover a benign NF_P P₂ without making any modifications to its NFProfile, even though P₂'s *allowedNssais* field explicitly prohibits C₁'s access. The crucial components in the counterexample are similar to the ones in previous Section 7.1.3, consists of a compromised NF_C C₁ residing in sNssai 1, with a benign NF_P P₂ serving only sNssai 3.

In *NFDiscoveryRequest*, the consumer may set two crucial attributes, *sNssais_{nfDisc}*, which denotes sNssais the NF_C wishes to discover, and *requestersNssais_{nfDisc}*, which refers to sNssais served by the NF_C. NRF follows the following steps to verify an *NFDiscoveryRequest*. First, it finds all target NFs that serve sNssais as appeared in *requestersNssais_{nfDisc}*, then validates if NF_C has access to the those NFs by cross-checking *requestersNssais_{nfDisc}* against *allowedNssais* in the NF_P's NFProfile, and finally, it filters potential producers based on *sNssais_{nfDisc}* [23]. However, as both of the mentioned parameters can be spoofed by the malicious NF_C, it may set these parameters to any values to discover any NF in the 5GC. While the *allowedNssais* of the target NF is checked against *requestersNssais_{nfDisc}* by NRF, *requestersNssais_{nfDisc}* is not cross-checked against *sNssais* in NF_C's NFProfile. In this setup, C₁ can set *requestersNssais_{nfDisc}* to sNssai 3, and obtains the NFProfile of P₂ that includes sensitive metadata of the victim NF. Similar attack can be found for *accessTokenRequest* as well upon verifying similar properties of type Φ_2 focusing on R_N^A .

Attack verification. In free5GC, NRF performs no cross-checks between the *sNssais* in consumer's NFProfile and *requestersNssais_{nfDisc}* field in *NFDiscoveryRequest*, and thus the vulnerability exists.

7.1.5 Parameter Misuse Attack. This attack is uncovered by the following property in category Φ_4 , φ : If access to a sensitive resource r which should only be exposed to sNssais s is granted for NF_C, then the NF_C must serve sNssais s . The crucial NFs in the counterexample produced by 5GCVerif contains a compromised consumer C₁, that serves only sNssai 1; and a benign producer P₂, which resides in both sNssais 1 and 2, that accepts *NFServiceRequests* from sNssais 1 and 2. The counterexample shows that C₁ is able to access resources stored in P₁ not only for sNssai 1 (we denote such resource as r_1), but also for sNssai 2 (we denote such resource as r_2). C₁ initiates an *accessTokenRequest* to obtain an *accessToken* for P₂, and NRF grants the token since sNssai 1 is in P₂'s *allowedNssais*. However, C₁ uses the token to request r_2 by providing the query parameter related to r_2 in its *NFServiceRequest*. P₂ will only validate that the *accessToken* contains the correct scopes and returns the requested data, as it has no way to learn that the consumer should only acquire services related to sNssai 1 (i.e., r_1) using the token it received. As a result, the consumer obtains sensitive information r_2 that it should not have access to as the consumer and r_2 are in separate network slices. In this scenario, the query parameter acts as a fragile authorization parameter that is implicitly used to restrict access of the consumer in addition to the *accessToken*, yet 3GPP underspecifies

the importance of the parameter and does not mention any of its implications. Using this method, malicious consumers can query for or even modify sensitive subscription information or contextual UE information, including user location and other privacy data, for UEs residing in slices the consumer should not have access to, simply by providing the corresponding UE ID.

Attack verification. In free5GC, we confirm the vulnerability in UDR. A malicious NF_C can extract UE authentication and subscription information of any sNssais using operations like *Policy-DataUesUeldAmDataGet* by providing the corresponding UE ID as a parameter, despite the imposed sNssais restrictions.

7.2 Resource Consumption of 5GCVerif

We evaluate the resource consumption of 5GCVerif by answering the following research question: *how does the trace length (i.e., number of states in a trace) explored by the model checker affect the performance of 5GCVerif?* To address this, we tested a simple reachability property against our model under varying state transition lengths, recording both time and memory consumption. For each state transition length, we repeated the experiment 5 times and calculated the average result. The findings are illustrated in Figure 7, and it shows that both time and memory consumption exhibit exponential growth with increasing trace length (i.e., number of state transitions in a trace). During our iterative workflow, we noted that the majority of counterexamples appear within the first 10 state transitions. As a result, the exponential growth in resource consumption has minimal impact while verifying 5GC's access control mechanism.

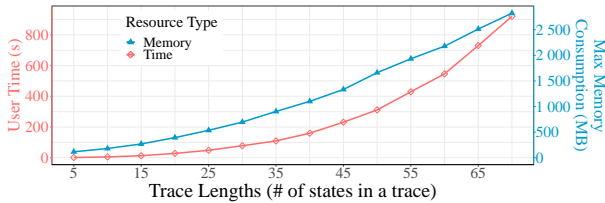


Figure 7: 5GCVerif's resource consumption w.r.t. trace lengths

8 RELATED WORK

Access control in 5G. Existing work explore different aspects related to access control issues in 5G systems (5GS). These include formalizing 5G access control mathematically [70], characterizing access control challenges and speculating new access control frameworks [28, 29, 50] for specific 5G use cases (e.g., IoT, healthcare, etc.) [38, 39, 41, 52, 61, 72]. None of the aforementioned works, however, formally analyzes access control mechanism of 5G systems.

Analysis of cellular networks using formal verification. Multiple attempts were made in the past to formally verify the cellular network protocols [36, 45, 54, 55, 59, 65]. For example, Cremers & Dehnel-Wild [45], and Basin et al. [36] model the Authentication and Key Agreement Protocols (5G-AKA) of 5GS using Tamarin [19] prover, while Hussain et al. [54, 55] analyze several NAS and RRC Layer protocols of 4G and 5G networks. However, all previous works only focus on modeling small parts of the 5G core network interactions, while assuming the rest to be well-protected and secure.

In contrast, 5GCVerif models all major authorization interactions in 5GC faithfully while also allowing for multiple consumer NFs to interact parallelly with producers.

Analysis of other access control systems using formal verification. Several previous works attempted to formally verify the access control policies of different systems [43, 53, 56]. For instance, Jayaraman et al. [56] propose new approach to formally verify AR-BAC based access control system. 5GC's access control is, however, not an ARBAC system. Chen et al. [43] formally analyze the access control configurations from software traces in Windows OS to find attack patterns. However, specific access control configurations of 5GC are private information and not accessible publicly. To address this, 5GCVerif generates semantically valid 5GC network configurations itself to find authorization flaws in the design irrespective of network configurations.

Security analysis on OAuth 2.0 implementations. OAuth 2.0 [46] framework is fundamentally designed for web-based applications. As opposed to analyzing the design, majority of the prior research in this area focuses on analyzing OAuth 2.0 implementations in particular use cases [30, 31, 35, 42, 62, 68, 69, 71, 73, 74]. Even the OAuth 2.0 framework is not flawless. Fett et al. [51] leverage formal methods to analyze the OAuth 2.0 framework and discover 4 vulnerabilities. In contrast, our focus is on faithful modeling and analyzing the design of OAuth 2.0-based access control mechanism of 5GC outlined in the 3GPP specifications.

9 DISCUSSION

Scope of our analysis. Our current analysis covers the most critical access control specifications, excluding the SCP and SEPP Proxies, which are involved in indirect and inter-network-operator communications, respectively. We prioritize security-critical attributes in NFProfile to reduce 5GCVerif's complexity, although there might be other attributes that could lead to unidentified attacks. Future work could explore these directions. Additionally, it is essential to note that 5GCVerif might overlook vulnerabilities stemming from misconfigurations or implementation-level flaws as we only model and analyze the specifications.

Threat to validity. Our manually extracted FSMs from the standard might not fully reflect the behavior of real operational networks. Inaccuracies in the FSMs may induce false positives, although based on our tested properties, we have not observed any. Furthermore, we reported our findings to GSMA [9] and consulted with GSMA's panel of experts who acknowledged the vulnerabilities [10].

Countermeasures. We intentionally refrain from discussing countermeasures for the observed attacks in the main body of the paper. Adding security measures into an existing protocol without thoroughly considering factors like backward compatibility can result in solutions that may lack long-term efficacy or fail under rigorous scrutiny. Instead, we are collaborating closely with the GSMA CVD panel to develop recommendations for updating the technical specifications. However, we do offer some insights into potential short-term patches in Appendix A.1.

10 CONCLUSION

We develop 5GCVerif to formally verify the access control mechanism of 5G core network through model checking. Our framework

is inspired by the CEGAR approach and is capable of automatically analyzing the access control mechanism for valid 5G core network configurations. Our evaluation of 5GCVerif uncovers five categories of previously uncovered vulnerabilities in 5GC. With its modular and highly customizable design, we envision 5GCVerif as a useful tool in continuously identifying and mitigating security threats in 5GC, ultimately contributing to the deployment of more secure and trustworthy 5G systems.

ACKNOWLEDGMENT

We appreciate the anonymous reviewers' feedback and GSMA's support during the vulnerability disclosure process. This study was supported by the NSF under grants 2145631, 2215017, and 2226447 and by the Defense Advanced Research Projects Agency (DARPA) under contract number D22AP00148.

REFERENCES

- [1] 3GPP Standard. www.3gpp.org.
- [2] 5G Networks Are Worryingly Hackable. <https://spectrum.ieee.org/5g-virtualization-increased-hackability>. [Online; accessed July 25, 2023].
- [3] 5G OpenAPIs. https://forge.3gpp.org/rep/all/5G_APIs.
- [4] 5GCVerif. <https://github.com/SyNSec-den/5GCVerif>.
- [5] 5GRadar: Discover how MVNOs can make the most of 5G. <https://www.5gradar.com/features/discover-how-mvnos-can-make-the-most-of-5g>. [Online; accessed July 25, 2023].
- [6] Bluetooth Specification. <https://www.bluetooth.com/specifications/bluetooth-core-specification/>.
- [7] CVE-2016-1906. <https://www.cvedetails.com/cve/CVE-2016-1906>. [Online; accessed July 25, 2023].
- [8] Free5GC. www.free5gc.org.
- [9] GSMA Coordinated Vulnerability Disclosure Programme. <https://www.gsma.com/security/gsma-coordinated-vulnerability-disclosure-programme/>.
- [10] GSMA Mobile Security Research Acknowledgements (previously known as GSMA Mobile Security Hall of Fame). <https://www.gsma.com/security/gsma-mobile-security-research-acknowledgements/>.
- [11] Hildegard: New TeamTNT Cryptojacking Malware Targeting Kubernetes. <https://unit42.paloaltonetworks.com/hildegard-malware-teamtnt/>. [Online; accessed July 25, 2023].
- [12] Kubernetes RBAC Used By Attackers To Deploy Persistent Backdoor. <https://phishingtrackle.com/articles/kubernetes-rbac-used-by-attackers-to-deploy-persistent-backdoor/>. [Online; accessed July 25, 2023].
- [13] Malicious Kubernetes Helm charts can be used to steal sensitive information from Argo CD deployments. <https://apiiro.com/blog/malicious-kubernetes-helm-charts-can-be-used-to-steal-sensitive-information-from-argo-cd-deployments/>. [Online; accessed July 25, 2023].
- [14] Open5GS. open5gs.org.
- [15] OpenAirInterface. <https://www.openairinterface.org/>.
- [16] OpenRAN – 5G hacking just got a lot more interesting. <https://media.ccc.de/v/mch2022-273-openran-5g-hacking-just-got-a-lot-more-interesting>. [Online; accessed July 25, 2023].
- [17] Over 900,000 Kubernetes instances found exposed online. <https://www.bleepingcomputer.com/news/security/over-900-000-kubernetes-instances-found-exposed-online/>. [Online; accessed July 25, 2023].
- [18] OWASP: Testing for privilege escalation. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/03-Testing_for_Privilege_Escalation.
- [19] Tamarin Prover. <https://tamarin-prover.github.io/>.
- [20] Telecoms: How 5G will revolutionise the MVNO market. <https://telecoms.com/opinion/how-5g-will-revolutionise-the-mvno-market/>. [Online; accessed July 25, 2023].
- [21] The OpenAPI Initiative. www.openapis.org.
- [22] 3GPP. 5G; Security architecture and procedures for 5G System. TS 33.501. 17.5.0.
- [23] 3GPP. 5G System; Network function repository services; Stage 3. TS 29.510. 17.7.0.
- [24] 3GPP. Procedures for the 5G System (5GS). TS 23.502. Version 17.6.0.
- [25] 3GPP. Study on security aspects of the 5G Service Based Architecture (SBA). TR 33.855. Version 16.1.0.
- [26] 3GPP. System architecture for the 5G System (5GS). TS 23.501. Version 17.6.0.
- [27] 3GPP. Release 17 Description; Summary of Rel-17 Work Items. TR 21.917, 2022. 1.0.0.
- [28] AdaptiveMobile Security. White Paper: A slice in time: Slicing security in 5G Core Networks. Technical report. [Online; accessed December 1, 2022].
- [29] Ahmad, Ijaz and Kumar, Tanesh and Liyanage, Madhusanka and Okwuibe, Jude and Ylianttila, Mika and Gurtov, Andrei. Overview of 5G security challenges and solutions. *IEEE Communications Standards Magazine*, 2(1):36–43, 2018.
- [30] Al Rahat, Tamjid and Feng, Yu and Tian, Yuan. Oauthlint: an empirical study on oauth bugs in android applications. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 293–304. IEEE, 2019.
- [31] Al Rahat, Tamjid and Feng, Yu and Tian, Yuan. Cerberus: Query-driven Scalable Security Checking for OAuth Service Provider Implementations. *arXiv preprint arXiv:2110.01005*, 2021.
- [32] Allman, Mark and Paxson, Vern and Terrell, Jeff. A brief history of scanning. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 77–82, 2007.
- [33] Aminof, Benjamin and Koteek, Tomer and Rubin, Sasha and Spegni, Francesco and Veith, Helmut. Parameterized model checking of rendezvous systems. *Distributed Computing*, 31:187–222, 2018.
- [34] Arons, Tamarah and Pnueli, Amir and Ruah, Sitvanit and Xu, Ying and Zuck, Lenore. Parameterized verification with automatically computed inductive assertions? In *Computer Aided Verification: 13th International Conference, CAV 2001 Paris, France, July 18–22, 2001 Proceedings 13*, pages 221–234. Springer, 2001.
- [35] Bansal, Chetan and Bhargavan, Karthikeyan and Delignat-Lavaud, Antoine and Maffei, Sergio. Discovering concrete attacks on website authorization by formal analysis. *Journal of Computer Security*, 22(4):601–657, 2014.
- [36] Basin, David and Dreier, Jannik and Hirschi, Lucca and Radomirovic, Saša and Sasse, Ralf and Stettler, Vincent. A Formal Analysis of 5G Authentication. In *CCS '18*.
- [37] Basin, David and Dreier, Jannik and Hirschi, Lucca and Radomirovic, Saša and Sasse, Ralf and Stettler, Vincent. A Formal Analysis of 5G Authentication. *CCS '18*, 2018.
- [38] Behrad, Shanay and Bertin, Emmanuel and Tuffin, Stéphane and Crespi, Noel. 5G-SSAAC: slice-specific authentication and access control in 5G. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 281–285. IEEE, 2019.
- [39] Behrad, Shanay and Bertin, Emmanuel and Tuffin, Stéphane and Crespi, Noel. A new scalable authentication and access control mechanism for 5G-based IoT. *Future Generation Computer Systems*, 108:46–61, 2020.
- [40] Cavada, Roberto and Cimatti, Alessandro and Dorigatti, Michele and Griggio, Alberto and Mariotti, Alessandro and Micheli, Andrea and Mover, Sergio and Roveri, Marco and Tonetta, Stefano. The nuXmv symbolic model checker. In *International Conference on Computer Aided Verification*, pages 334–342. Springer, 2014.
- [41] Chen, Baozhan and Qiao, Siyuan and Zhao, Jie and Liu, Dongqing and Shi, Xiaobing and Lyu, Minzhao and Chen, Haotian and Lu, Huimin and Zhai, Yunkai. A security awareness and protection system for 5G smart healthcare based on zero-trust architecture. *IEEE Internet of Things Journal*, 8(13):10248–10263, 2020.
- [42] Chen, Eric Y and Pei, Yutong and Chen, Shuo and Tian, Yuan and Kotcher, Robert and Tague, Patrick. OAuth demystified for mobile application developers. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 892–903, 2014.
- [43] Chen, Hong and Li, Ninghui and Gates, Christopher S and Mao, Ziqing. Towards analyzing complex operating system access control configurations. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, pages 13–22, 2010.
- [44] Clarke, Edmund and Grumberg, Orna and Jha, Somesh and Lu, Yuan and Veith, Helmut. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*, pages 154–169. Springer, 2000.
- [45] Cremers, Cas and Dehnel-Wild, Martin. Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. 2019.
- [46] D. Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, RFC Publisher, October 2012.
- [47] R. de Nicola and M. C. B. Hennessy. *Testing equivalences for processes*, pages 548–560. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.
- [48] Dolev, Danny and Yao, Andrew. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.
- [49] Durumeric, Zakir and Wustrow, Eric and Halderman, J Alex. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *USENIX Security Symposium*, volume 8, pages 47–53, 2013.
- [50] Dutta, Ashutosh and Hammad, Eman. 5G security challenges and opportunities: a system approach. In *2020 IEEE 3rd 5G World Forum (5GWF)*, pages 109–114. IEEE, 2020.
- [51] Fett, Daniel and Küsters, Ralf and Schmitz, Guido. A comprehensive formal security analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1204–1215, 2016.
- [52] Gujja, Daniel and Siddiqui, Muhammad Shuaib. Identity and access control for micro-services based 5G NFV platforms. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–10, 2018.
- [53] Hughes, Graham and Bultan, Tevfik. Automated verification of access control policies using a SAT solver. *International journal on software tools for technology*

- transfer, 10(6):503–520, 2008.
- [54] Hussain, Syed and Chowdhury, Omar and Mehnaz, Shagufta and Bertino, Elisa. LTEInspector: A systematic approach for adversarial testing of 4G LTE. In *Network and Distributed Systems Security (NDSS) Symposium*, 2018.
 - [55] Hussain, Syed Rafiul and Echeverria, Mitziu and Karim, Intiaz and Chowdhury, Omar and Bertino, Elisa. 5GReasoner: A Property-Directed Security and Privacy Analysis Framework for 5G Cellular Network Protocol. In *CCS '19*.
 - [56] Jayaraman, Karthick and Ganesh, Vijay and Tripunitara, Mahesh and Rinard, Martin and Chapin, Steve. Automatic error finding in access-control policies. ACM Press, 2011.
 - [57] Jones, M. and Bradley, J. and Sakimura, N. JSON Web Signature (JWS). Technical report, RFC Editor, May 2015.
 - [58] Jones, M. and Bradley, J. and Sakimura, N. JSON Web Token (JWT). Technical report, RFC Editor, May 2015.
 - [59] Karim, Intiaz and Hussain, Syed and Bertino, Elisa. ProChecker: An Automated Security and Privacy Analysis Framework for 4G LTE Protocol Implementations. In *Proceedings of the 41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021*.
 - [60] Krzysztof R. Apt and Dexter C. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307–309, 1986.
 - [61] Li, Qi and Xia, Bin and Huang, Haiping and Zhang, Yinghui and Zhang, Tao. TRAC: traceable and revocable access control scheme for mHealth in 5G-enabled IIoT. *IEEE Transactions on Industrial Informatics*, 18(5):3437–3448, 2021.
 - [62] Li, Wapeng and Mitchell, Chris J. Security issues in OAuth 2.0 SSO implementations. In *International Conference on Information Security*, pages 529–541. Springer, 2014.
 - [63] Maliheh Monshizadeh, Prasad Naldurg, and VN Venkatakrishnan. Mace: Detecting privilege escalation vulnerabilities in web applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 690–701, 2014.
 - [64] P. Leach and M. Mealling and R. Salz. A Universally Unique Identifier (UUID) URN Namespace. Technical report, July 2005.
 - [65] Peltonen, Aleksis and Sasse, Ralf and Basin, David. A comprehensive formal analysis of 5G handover. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, Jun 2021.
 - [66] Pnueli, Amir. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, 1977.
 - [67] Richer, J. OAuth 2.0 Token Introspection. Technical report, RFC Editor, oct 2015.
 - [68] Shehab, Mohamed and Mohsen, Fadi. Towards enhancing the security of oAuth implementations in smart phones. In *2014 IEEE International Conference on Mobile Services*, pages 39–46. IEEE, 2014.
 - [69] Sherman, Ethan and Carter, Henry and Tian, Dave and Traynor, Patrick and Butler, Kevin. More guidelines than rules: CSRF vulnerabilities from noncompliant OAuth 2.0 implementations. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 239–260. Springer, 2015.
 - [70] Suárez, Luis and Espes, David and Cuppens, Frédéric and Bertin, Philippe and Phan, Cao-Thanh and Le Parc, Philippe. Formalization of a security access control model for the 5G system. In *2020 11th International Conference on Network of the Future (NoF)*, pages 150–158. IEEE, 2020.
 - [71] Sun, San-Tsai and Beznosov, Konstantin. The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 378–390, 2012.
 - [72] Wang, Qixu and Chen, Daijiang and Zhang, Ning and Qin, Zhen and Qin, Zhiguang. LACS: A lightweight label-based access control scheme in IoT-based 5G caching context. *IEEE Access*, 5:4018–4027, 2017.
 - [73] Yang, Ronghai and Lau, Wing Cheong and Chen, Jiongyi and Zhang, Kehuan. Vetting Single {Sign-On} {SDK} Implementations via Symbolic Reasoning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1459–1474, 2018.
 - [74] Yang, Ronghai and Li, Guanchen and Lau, Wing Cheong and Zhang, Kehuan and Hu, Pili. Model-based security testing: An empirical study on oAuth 2.0 implementations. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 651–662, 2016.

A APPENDIX

A.1 Outlines of Potential Fixes

In what follows, we discuss potential temporary fixes to discovered vulnerabilities. For long-term countermeasures, we are actively collaborating with GSMA through GSMA CVD Programme.

A.1.1 Confused Producer Attack. An accessToken already contains an attribute, *producerSnsaiList*, which specifies a list of sNsais that the consumer is authorized to access. The NF_P can use this list to validate whether the consumer is allowed to access its services to prevent *Confused Producer Attack*. However, the 3GPP specifies *producerSnsaiList* to be optional and hence the NF_P cannot rely on this attribute to validate the consumer's authorizations. We propose to make *producerSnsaiList* mandatory in the accessToken.

A.1.2 Token Reuse Attack. Currently, no revocation mechanism is described by 3GPP for OAuth 2.0 tokens. One plausible solution is to enable an NF_P to check if an NF_C is using an obsolete accessToken. For this, NF_P can query NRF through a new API call, *TokenVerificationRequest (accessToken)*, upon receiving *NFServiceRequest* from NF_C. A similar solution is also discussed in RFC 7662 [67]. However, introducing an additional network interaction between NF_P and NRF for each *NFServiceRequest* can significantly impact the performance of both NRF and NF_P, and defeats the purpose of caching accessToken. Another solution is to introduce a new attribute, *timestamp*, which represents the time of issuance, to the accessToken. Additionally, the NF_P should maintain an attribute, *lastUpdateTime*, to track the most recent critical NFProfile update. NF_P will deny *NFServiceRequest* if *timestamp* in the accessToken is earlier than NF_P's *lastUpdateTime*.

A.1.3 Authorization Bypass Attack. Enforcing a cross-check between *requestersNsais_{nfDisc}* message and *sNsais* of NF_C's NFProfile during the verification of *NFDiscoveryRequest* by NRF will address this vulnerability.

A.1.4 Default Overprivilege Attack. To mitigate this attack, OAM must verify an update to critical NFProfile attributes before the profile update is granted. Additionally, during each *accessTokenRequest*, NRF should verify the authorization parameters from NF_P's NFProfile against relevant attributes in the NF_C's NFProfile, as discussed in Appendix A.1.3. It is also essential to avoid the *allow-by-default* policy for all critical attributes, including *sNsais*. Instead, *deny-by-default* policy should be enforced. However, implementing this fix may cause interoperability issues if not all parties adopt the fix.

A.1.5 Parameter Misuse Attack. Section 7.1.5 underscores the necessity of strict verification for crucial input parameters in an *NFServiceRequest*. However, implementing such measures poses challenges. One approach involves NRF checking the input parameters during *accessTokenRequest* verification, and appending only the verified values to the accessToken. However, this approach requires significant modifications to the existing accessToken design and may result in increased system overhead. It also limits accessToken caching and may create communication bottlenecks. Alternatively, if the NF_P undertakes the verification, the NF_P would require additional information about the NF_C (e.g., NFProfile) and might require additional queries to the NRF, incurring substantial delays.