5GAC-Analyzer: Identifying Over-Privilege Between 5G Core Network Functions

Seaver Thorn
swthorn@ncsu.edu
North Carolina State University
Raleigh, NC, USA

K. Virgil English
kvenglis@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Kevin R. B. Butler
butler@cise.ufl.edu
University of Florida
Gainesville, FL, USA

William Enck
whenck@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

5G technology transitions the cellular network core from specialized hardware into software-based cloud-native network functions (NFs). As part of this change, the 3GPP defines an access control policy to protect NFs from one another and third-party network applications. A manual review of this policy by the 3GPP identified an over-privilege flaw that exposes cryptographic keys to all NFs. Unfortunately, such a manual review is difficult due to ambiguous documentation. In this paper, we use static program analysis to extract NF functionality from four 5G core implementations and compare that functionality to what is permissible by the 3GPP policy. We discover two previously unknown instances of over-privilege that can lead denial-of-service and extract sensitive data. We have reported our findings to the GSMA, who has confirmed the significance of these policy flaws.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**; **Access control**; • **Networks → Mobile networks**.

KEYWORDS

5G Core; OAuth; Access Control

ACM Reference Format:
Seaver Thorn, K. Virgil English, Kevin R. B. Butler, and William Enck. 2024. 5GAC-Analyzer: Identifying Over-Privilege Between 5G Core Network Functions. In *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '24), May 27–30, 2024, Seoul, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3643833.3656134

1 INTRODUCTION

5G technology marks a significant shift in wireless communication, ushering in a new era of connectivity, speed, and innovation. One of the most consequential changes of 5G occurs in the network core, which has been rearchitected to disaggregate specialized *hardware* entities (e.g., HSS, MME, P-GW) into many different *software* Network Functions (NFs) running in public and private clouds. The 5G core adopts a service-based architecture to manage NF communication, which allows application developers to create new services that directly interact with core functionality. User Equipment (UEs) depends on the resource isolation of the core NFs to safeguard their sensitive information (e.g., cryptographic keys and encryption policy) from disclosure and modification.

The 5G core performs two types of access control decisions: (1) checks based on runtime NF identifiers (i.e., NFProfile); and (2) service-level logic (i.e., OAuth policy). Unfortunately, the 3GPP specifications are known for being ambiguous and under-specified [15, 20, 31, 32, 38]. Akon et al. [8] formally modeled the access control in the 5G core based on the 3GPP specifications and found that it fails to sufficiently specify access control checks of runtime identifiers between logically separate networks (i.e., network slices). However, their formal model incorrectly assumes that all access control checks and logic in the specifications are correct.

A *manual* review of the service-level OAuth policy logic by the 3GPP identified an over-privilege access control flaw that unnecessarily exposes read and write access to sensitive subscriber data, including cryptographic keys (3GPP TR 33.855 Key Issue #29) [1]. Essentially, an OAuth scope was specified at a service-level when it should have been more fine-grained and specified at an operation-level. While Akon et al. [8] analyzed 3GPP Release 17 (which fixed this flaw), their model would not have found the over-privilege, because they assume the logic is correct.

In this paper, we study over-privilege in the 5G service-level OAuth policy logic. We overcome the limitations of ambiguous and underspecified 3GPP policy by leveraging how developers have interpreted the standards. Specifically, we use the aggregate of four open-source 5G core implementations to approximate the required functionality. We use static program analysis to extract *access patterns* that define NF calls to privileged operations in other NFs. In doing so, we identify two service-level OAuth scopes that are too coarse-grained. We reported our findings to the GSMA who have issued CVDs and made recommendations to the 3GPP.

Key Results: The newly identified policy flaws (a) allow interruption of UE connectivity and (b) provide an additional attack vector to extract subscriber information. The first policy flaw allows disruption of UE connectivity, which may trigger a re-authentication with the core network. During this time, the UE cannot directly connect to the Internet or may have their traffic relayed through a malicious Access & Mobility Management Function (AMF). The second policy flaw allows an attacker to extract sensitive subscription data for a UE (e.g., subscribed slices, integrity protection, encryption policy, IP addresses, charging, and QoS policies) from the Unified

WiSec '24, May 27–30, 2024, Seoul, Republic of Korea

ACM ISBN 979-8-4007-0582-3/24/05…$15.00
https://doi.org/10.1145/3643833.3656134

Data Management Function (UDM). This flaw is a variation of the flaw described in 3GPP TR 33.855 Key Issue #29.

We make the following contributions in this paper:

- *We automatically extract 5G NF functionality using a collection of 5G implementations.* NF functionality in 3GPP specifications is often ambiguous and underspecified. Developers implementing 5G cores have to interpret the specifications into working systems.
- *We create a program analysis framework to model access control policy for the 5G core.* We define a high-level model that facilitates analysis from 5G core implementations despite vast differences. Our framework uses analysis sub-modules to extract implementation-specific values.
- *We identify two previously unknown access control flaws in the 3GPP policy.* In addition to confirming 3GPP TR 33.855 Key Issue #29, we identify over-privilege policy flaws that enable disruption of UE connectivity and leaking of sensitive subscription data. The GSMA has agreed these are flaws in the OAuth policy and assigned 2 CVDs.

Availability: The source code for 5GAC-Analyzer can be accessed at https://github.com/wspr-ncsu/5GAC

The remainder of the paper proceeds as follows. Section 2 provides background and motivation. Section 3 provides an overview of our design. Section 4 describes our policy analysis. Section 5 details the discovered security flaws. Section 6 describes our proposed policy. Section 7 discusses limitations. Section 8 overviews related work. Section 9 concludes.

2 BACKGROUND & MOTIVATION

5G Core Architecture: The 5G core makes many improvements over the 4G/LTE core. Notably, 5G shifts away from specialized hardware into a software-defined service-based architecture, which defines functionality in the core through a REST-based Application Programming Interface (API) over HTTP. The 3GPP standards allow some flexibility in NF implementation; for example, developers can implement NFs as stateful or stateless.

5G operates in two distinct modes: 5G Non-Standalone (NSA) is a 5G RAN with a 4G core network, and 5G Standalone (SA) is a 5G RAN and 5G core. Since we consider the 5G core's access control policy, we assume 5G SA for the remainder of this paper. Network slicing is a key enhancement in 5G and is only achievable in 5G SA [24]. Network slicing allows distributed networks that achieve specific properties such as low latency or ultra-reliability. Slicing also enables business customers to create a logical network isolated from other slices facilitated by NFs. Network core operators provision and configure slices based on agreements between the business and operators. The most common NFs deployed in slices are the User Plane Function (UPF), followed by the AMF and Session Management Function (SMF), then the Unified Data Repository Function (UDR) and UDM [10, 11, 24].

The 5G core network or a slice can deploy a 5G application. This app may access subscriber information and provide additional content to specific UEs. A sports stadium slice can deploy a 5G app with specific network requirements to provide an improved fan experience to consumers. For example, AR, interactive contests, or other features.

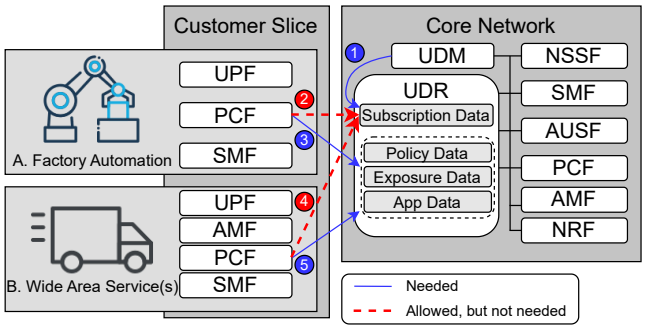

Figure 1: Example of different 5G slice configurations with annotations describing 3GPP TR 33.855 Key Issue #29. The UDR's subscription data is more sensitive than the other data it exposes. While only the UDM needs to access the subscription data, the 3GPP policy allows all NFs to access it.

5G Slice Architecture: Figure 1 provides an example configuration of a 5G core with multiple slices for different business uses. Each slice can deploy its own NFs or use the NFs exposed by the core network to the slice. Deploying NFs in the slice enables fine-grained control, independence, and isolation from other slices and the core network. Slices typically deploy a UPF to help achieve low latency, enforce encryption standards, and avoid user data leaving the slice [24].

Slice A in Figure 1 represents a typical design for an industrial automation slice deployed at a single location. This slice configuration enables robotics and surveillance of a factory by deploying a UPF and SMF to handle network sessions. Most of the AMF's mobility functionality is unnecessary as the devices in the factory are not moving large distances. Therefore, an AMF deployment in an industrial automation slice may be more applicable when the automation is highly mobile, such as in outdoor farm equipment [24].

Slice B in Figure 1 shows a slice for a large-scale fleet of vehicles, such as delivery trucks or drones. Cellular-based Vehicle-to-Everything (V2X) use cases require Ultra-Reliable Low-Latency Communication (URLLC). Some NFs may be duplicated on the network edge and in the core for redundancy and load balancing. This slice will have similar requirements as Slice A but with a dedicated AMF for mobility. This slice is ideal for low latency, high availability, and mission-critical applications [10]. The NFs in a slice may communicate with other NFs in the core network or another slice depending on how the network is configured.

Access Control in the 5G core: The 3GPP has standardized the use of OAuth 2.0 for access control between NFs in the 5G core. While the use of OAuth is defined as optional, it is necessary for robust access control in the core network, especially when multiple tenants are involved. Assuming slices agree to share data, the access control policy determines what NFs among separate slices share which operations. Additionally, slices may request data from the core's NFs about other slices or UEs on no slices. A least-privileged access control policy is needed to ensure resource isolation among multiple tenets.

In the 3GPP standards, an access token can be *scoped* to allow only a group of operations for a service. Specifically, they define

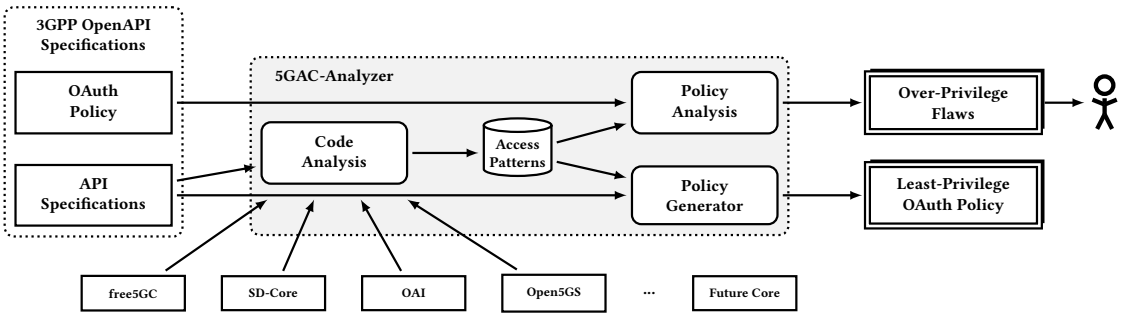

Figure 2: 5GAC-Analyzer uses software diversity to identify over-privilege and proposes a revised policy for the 3GPP specifications.

two types of scopes: service-level and operation-level. When an access token is assigned a service-level scope, it authorizes access to any operation in that service. An operation-level scope is more fine-grained and overrides service-level scopes where they are defined. Operation-level scopes can still be associated with multiple operations in a service, but they are less-privileged than a service-level scope (which allows access to all operations). The 3GPP uses service-level scopes in most cases (e.g., nudm-ee); however, it uses operation-level scopes for highly privileged operations (e.g., nudm-ee:subscription:create).

Based on the 3GPP standards [3], OAuth enforcement should operate as follows. When a consumer NF wants to call an operation exposed by a producer NF, it will first ask the Network Repository Function (NRF) for the access token needed. The NRF will validate the requesting NF's identity by using TLS mutual authentication and grant the token with the appropriate OAuth scopes. The client NF then sends the access token with the request to the producer NF. When the producer NF receives the request, it will check and validate that the access token has the OAuth scope required by the API. If the validation succeeds, the producer NF performs the operation and returns the appropriate result to the consumer NF.

3GPP TR 33.855 Key Issue #29: The 3GPP published a technical report [1] describing security issues within the 5G core service-based architecture. It found that some operations in an NF's exposed services are more sensitive than others. In the report, Key Issue #29 describes the UDR's data-repository service as providing policy, exposure, and application data that many NFs access and modify. However, this service also includes highly-sensitive subscription data, including cryptographic keys and primitives used for UE authentication.

Figure 1 shows two slices interacting with the same UDR in the core network. A Policy Control Function (PCF) in one network slice can access or modify the core network's subscription data in the UDR under the 3GPP access control policy. Furthermore, if network slices do not operate their own UDR, their highly sensitive data are exposed to other slices. Figure 1 shows two types of connections between NFs. The OAuth policy allows connections 1, 3, and 5; however, 2 and 4 are permitted by the policy but should never happen in practice. The PCF in slice A could be given subscription information used by network slice B only, thus breaking the isolation between slices. The specific problem with the subscription data in 3GPP TR 33.855 Key Issue #29 was that a service-level scope was assigned to all operations when some were more privileged. The 3GPP concluded that only the UDM should access the subscription data and created a new operation-level scope for all of the subscription data operations.

Threat Model and Assumptions: We assume that each NF requires different access levels to different data types in the 5G core. We consider two types of adversaries that have compromised NFs. First, we consider adversaries that have compromised the container or virtual machine that runs an NF by exploiting a vulnerability in that NF. We assume the NRF cannot be compromised. The NRF performs most access control checks and grants scoped access tokens to other NFs [2]. If the NRF is compromised, access control in the core fundamentally breaks down. Second, we consider a malevolent or curious business customer that may operate its slice in a way that harms the core network. We assume that every instance of an NF is on a different physical network than every other NF, and the 5G core may be distributed across many different locations. When an NF is compromised, we assume it can create arbitrary requests to any other NF with exposed API interfaces over HTTP. We assume that the HTTP interface is encrypted with TLS and performs mutual authentication between all NFs. Additionally, we consider the cloud provider, physical hardware, UE, and RAN as part of the trusted computing base.

3 OVERVIEW

The 3GPP standards define all NFs, services, and operations, including parameters and return data types. However, they ambiguously define the service-level logic of where and when operations should be called from which NFs. This lack of detail is intentional, as the 3GPP wants to allow developers to design systems in flexible and creative ways. Our insight is to use static program analysis to extract access patterns that represent which NFs use which privileged operations. However, each implementation implements functionality in its own way. Therefore, it is insufficient to consider only one implementation. As such, we take the union of access patterns across multiple cores to extract the functionality intended by the standards.

Figure 2 provides a high-level overview of our approach called 5GAC-Analyzer. Specifically, 5GAC-Analyzer is a collection of CodeQL queries and Python scripts that extracts access patterns and compares those against OAuth rules in the 3GPP specifications.

Table 1: A unified analysis model is needed to handle how 5G cores implement similar functionality differently.

Core	Static API Routes	State Machine	Function per API
free5GC (R15)	✓	✗	✓
SD-Core (R15)	✓	✗	✓
OAI (R15)	✗	✗	✓
Open5GS (R16 & R17)	✗	✓	✗

We use these access patterns to (a) provide a set of potential over-privileged policy flaws for a human analyst and (b) derive a least-privilege OAuth policy. Section 4 details our analysis framework. Section 6 describes the generation of a least-privilege policy.

Challenge: *5G cores differ vastly in implementation.* Developers make a range of software design choices when implementing the 5G core. Table 1 shows the high-level differences between the 5G core implementations relevant to our analysis. In addition to the choice of programming language, implementations use different abstractions to call APIs and handle requests. For example, free5GC statically defines strings for each HTTP REST API and creates a dedicated function to handle them. In contrast, Open5GS creates a state machine for each service and demultiplexes HTTP REST APIs using conditional logic. The 3GPP standards do not generally specify access patterns; developers need to read and interpret the standards to determine which ones are necessary. Additionally, the 3GPP standards define alternative methods of implementing the same functionality. For example, OpenAirInterface (OAI) registers callbacks between NFs by passing HTTP REST strings as arguments to other APIs.

To address this challenge, we define a CodeQL model to query for access patterns similar to how Prolog is used in prior work. We chose CodeQL because it can analyze multiple programming languages with the same abstractions. Therefore, defining a CodeQL model allows us to transfer much of the analysis between implementations. Using this model, we simplify the static analysis to creating subqueries to identify key features, such as identifying service entry points and extracting proxy function calls.

Our CodeQL implementation covers four open-source 5G core implementations including two versions of one 5G core targeting two different 3GPP release versions: (1) free5GC version 3.3.0 (R15); (2) ONF SD-Core version 1.3 (R15), which is a fork of free5GC; (3) Open5GS versions 2.5.4 (R16) and version 2.6.4 (R17); and (4) OAI version 1.5.1 (R15). When new access patterns are implemented into open source cores, we can analyze new portions of the access control policy.

4 5G CORE ANALYSIS WITH CODEQL

Our CodeQL model translates formal definitions of access patterns into CodeQL queries. An access pattern represents an API call for an operation from one NF to another. Extracting access patterns involves capturing all API calls associated with an NF. Any reachable API calls are added to the list of access patterns for that NF. We now formally describe the mathematical notation that our CodeQL model is based on.

```
from ServiceCall s, string handleName
where (exists(Handler hh | isDescendant(s, hh.asFunction()) |
      handleName = hh.getName())
or (exists(TerminatingFunction f | isDescendant(s, f) |
      handleName = f.getName())
        and not exists(Handler hh | isDescendant(s, hh.
      asFunction()))
        ))
select s.getHttpType() as http_type,
handleName as serviceCallCFSource,
s.getSourceNF() as sourceNF,
s.getName() as serviceCallFunction,
s.getTargetNF() as targetNF,
s.getColonizedPath() as targetPath,
s.getPathRoot() as targetRootPath
```

Figure 3: CodeQL Entry Point Extraction. This query structure applies to all analyzed cores, even though they are written in Go and C.

Let $\mathcal{N}$ be the set of all NFs, $\mathcal{S}$ be the set of all services, and $\mathcal{O}$ be the set of all operations. Each NF $n \in \mathcal{N}$ is a set of services $S \subseteq \mathcal{S}$. Each service $s \in \mathcal{S}$ is a set of operations $O \subseteq \mathcal{O}$. Finally, we assume a set of permissions $\mathcal{P}$ is text strings used to define access control rules.

Definition 1 (Access Pattern)**.** The set of all possible access patterns $\mathcal{A}$ is a set of pairs. Let $n \in \mathcal{N}$ and $o \in \mathcal{O}$ be an NF and an operation respectively. An access pattern $a \in \mathcal{A}$ is a pair (n, o) empirically derived through program analysis. Conceptually, an access pattern captures one NF calling an operation in another NF. As such, each access pattern consists of a calling NF, receiving NF, service, and an operation. For simplicity, our formal notation only includes the calling NF and operation.

Definition 2 (Access Control Policy)**.** Let the set of all possible access rules $\mathcal{R}$ be a set of triples (n, o, p), where $n \in \mathcal{N}$, $o \in \mathcal{O}$, and $p \in \mathcal{P}$ are an NF, an operation, and a permission text string. A policy Γ is a set of access rules.

Conceptually, an access control policy represents the allowed interactions between NFs and an operation. The policy consists of many rules, where each rule describes a single interaction between an NF and an operation. The OpenAPI standard used by the 3GPP allows specifying permissions (a) for each operation or (b) for a service, which includes many operations. Each service groups operations by functionality; however, as previously discussed, these groups do not consider the exposed data.

CodeQL Implementation: For each core implementation, 5GAC-Analyzer requires a query to identify: (1) the calling NF, (2) an HTTP type of an API path (e.g: POST, GET, PUT), (3) the API path itself, and (4) the name of the function that calls the API, or applicable code location. We implement a query in the model for each of these pieces of information, shown in Figure 3. Each 5G core needs a submodule that includes implementation-specific details to identify an access pattern. To determine reachability, 5GAC-Analyzer begins at the program's main entry point and recursively searches for the corresponding code that calls the API.

Each submodule needs to implement the queries from Figure 3. Section 4.1 describes how to identify service entry points in the called NF, which are the `getTargetNF()` and `handleName` CodeQL queries. Section 4.2 describes the `getHttpType`, `getTargetNF`, `getColonizedPath`, and `getRootPath` queries. Reachability analysis requires

no additional queries and is included by implementing the above queries.

4.1 Identifying Service Entry Points

5GAC-Analyzer needs to identify all API entry points in the implementations to determine which 3GPP specification APIs are supported. In some cases, the specifications alone do not provide sufficient detail to identify all entry points in a implementation. All 5G cores that we analyzed use the OpenAPI-Generator [19] tool to create code from the 3GPP specifications. Therefore, our analysis uses the specifications and name construction patterns to identify the code implementations for entry points.

Free5GC, SD-Core & OAI: OpenAPI-Generator's template uses the `operationId` field in the OpenAPI specification to name the function in the source code. Unfortunately, the 3GPP OpenAPI specification and OpenAPI-Generator have two limitations: (1) the `operationId` is not defined for every operation, and (2) entry points will use the same function names when the `operationId` is the same. OpenAPI-Generator creates these functions with the same function name because they have the same `operationId` in the 3GPP specifications.

We use CodeQL to identify each service's routing table, which matches APIs to functions. Importantly, this describes each APIs implementation for a particular operation in the specifications. CodeQL identifies the routing table by searching all global variables for ones that match the routing table type. The routing table is a list of API paths and a function pointer. In OAI, a C++ class is defined with function pointers mapping to APIs rather than a routing table, however, it serves the same purpose.

5GAC-Analyzer assigns the function to the corresponding API endpoint based on its unique URI that is defined in a routing table structure for the HTTP service. A backwards call graph analysis verifies the target service URI from a generated Router object or function call in OAI. This strategy successfully identifies the entry points in the free5GC, SD-Core, and OAI codebases.

Open5GS: Open5GS entry points have manually-generated function names. Therefore we can not use the `operationId` to identify these functions. Instead, we extract these functions from the state machines Open5GS uses to handle incoming requests. To identify these functions, 5GAC-Analyzer needs to: (1) collect all functions that take the API request structure as a parameter; (2) perform a call graph analysis to verify that this function is called somewhere in the call graph; and (3) extract the URI path from the state machine.

4.2 Extracting Proxy Function Calls and Access Patterns

5GAC-Analyzer extracts the API calls that cross microservice boundaries, which we call *access patterns*. These cross-microservice API calls typically occur via a proxy function that wraps the HTTP request. 5GAC-Analyzer's goal is to identify the proxy functions, the URI, and receiving NF for every HTTP request. Finally, 5GAC-Analyzer maps each access pattern to the 3GPP OAuth policy to extract the access token that is necessary to call all identified APIs.

For all cores, 5GAC-Analyzer starts the process by identifying the common function that crafts and sends the HTTP request. For `C`-based cores, this is some variation of a `curl` API. For `Go`-based cores, this is the `CallAPI` function, generated by the OpenAPI-Generator. 5GAC-Analyzer uses these common HTTP request functions to identify the proxy functions in each core.

```
1 CoreServiceCall() {
2     exists(Function f, CallExpr c |
3         c.getCalleeName() = "CallAPI" and
4         c.getEnclosingFunction() = f
5         | this = f and apicall = c )}
```

Figure 4: Lower-level CodeQL query to identify proxy functions in free5GC & SD-Core. Additional checks are made in other queries to identify a complete list of proxy functions.

Free5GC & SD-Core: Figure 4 shows the CodeQL query identifying all functions in the code base that call the `CallAPI` function. The target URI in the HTTP request is extracted to map the proxy function to a target service endpoint. 5GAC-Analyzer then performs call graph analysis to connect proxy functions to calling entry points. This analysis achieves two goals: identifying all auto-generated proxy functions implemented and connecting API calls with their calling NF entrypoint. The identified connections create our access pattern (n, o) tuples.

Open5GS: Open5GS entry points are organized differently than free5GC and SD-Core. As mentioned previously, Open5GS manages an internal state machine to handle requests and responses to proxies and stubs. It dispatches requests into separate functions stemming from the main state machine function. Additional program analysis is necessary to identify what API these functions implement, as they are not named consistently.

API request functions are identified as the callers of common HTTP request functions containing a service endpoint as the target. Target HTTP endpoints are encoded inside a message structure using constant strings in each request function. We use CodeQL to extract these strings to identify data flows from these message strings to HTTP network requests. 5GAC-Analyzer maps each function call to an API in the specification using the HTTP type and path information. Finally, we identify call paths from the program start or state machine start that lead to the network request functions.

OAI: OAI presented unique challenges compared to the other cores. Similar to the other cores, 5GAC-Analyzer starts by identifying calls to common HTTP request functions and the proxies calling them. However, OAI does not consistently use constant strings to identify the target URI of the call. Instead, combinations of string literals, function calls, and dynamic variables containing both are used to craft the target URI. Many of the target URI's are determined at runtime, and are therefore not directly obtainable with static analysis.

String literals and function calls that return static variables are directly resolved. Some URIs depend on configuration or other dynamic information only determined at runtime. In such cases, HTTP calls are still identified, and we output them for manual investigation. Functions with dynamic returns are resolved as the function name, and that name is used in manual post-processing to identify the target URI path. Variables are recursively resolved to strings and function names based on the same criteria. Two experts verified all manual components, mapping function names to 3GPP

Table 2: Analysis statistics for the analyzed 5G cores. We include the 3GPP release for each 5G core target. We consider the union of all cores in our analysis.

Open Source Core	NFs	Services	APIs	Access Patterns	NF Interactions	Stub APIs	Dynamic Access Patterns
free5GC (R15)	10	13	84	66	18	10	0
SD-Core (R15)	8	13	81	68	14	10	0
OAI (R15)	9	12	43	47	16	9	13
Open5GS (R16 & R17)	10	14	48	60	22	12	10
Union	12	19	100	83	22	31	23

specifications by examining the OAI source code. In cases where the URI could not be determined by both 5GAC-Analyzer or manual investigation, the result was discarded.

OAI's implementation of functionality as callbacks also poses challenges for static analysis. Unlike other cores, OAI relies heavily on callback URLs to receive information, making it difficult to determine the exact endpoints that will be called statically. For example, callback URLs often include the URL of the API to call in the response of registering a callback. Therefore, it is not possible to determine the endpoint that will get called statically.

5 3GPP OAUTH POLICY ANALYSIS RESULTS

This section details the results of our policy analysis. 5GAC-Analyzer identified three instances of over-privilege in the 3GPP OAuth policy. Two of these instances are new; the third confirms 3GPP TR 33.855 Key Issue #29, which was fixed in Release 17. We manually confirmed the two new instances and reported them to the GSMA. Independent of 5GAC-Analyzer, we discover a flaw with how the security requirements are encoded in OpenAPI. This flaw nullifies all access control scopes when OAuth is enabled. After a detailed discussion, the GSMA issued three CVDs and recommended changes to the 3GPP, which are currently under consideration. The remainder of this section details the results of 5GAC-Analyzer and describes the discovered policy flaws in detail.

5.1 Experimental Setup

We ran our analysis on 3GPP Releases 15-17, depending on which core we were analyzing. This analysis contains 79 files per 3GPP release defining the access control policy and API calls. The changes introduced between 3GPP versions did not impact our findings, as there were minimal changes in the access control policy. We obtained the policy files from the 3GPP working area website [5]. We use the policy files defining access control policy and additional structures for NFs present in free5GC, SD-Core, Open5GS, and OAI. The open-source 5G cores do not implement every NF standardized in 3GPP Release 15-17, which we discuss further in Section 7. Finally, we filter out all unused policy files for APIs and services that are not implemented in any of the cores, leaving a total of 39 policy files per 3GPP release.

5.2 Static Analysis Results

5GAC-Analyzer uses static analysis to extract access patterns and therefore may have both false positives and false negatives. For example, 5GAC-Analyzer cannot detect access patterns that depend on data determined at runtime. We begin by detailing the access patterns reported by 5GAC-Analyzer for each of the cores. We then describe the impact of false positives and false negatives on our results. In cases where we encountered false positives and false negatives, we resolved them manually to ensure they did not impact our findings.

Access Pattern Extraction: Table 2 shows the results of 5GAC-Analyzer. When considering the union of all 5G cores, we identified 83 access patterns. There was a large overlap between all the cores' access patterns. However, each core had at least one unique access pattern. For example, free5GC implemented more access patterns related to non-3GPP access and many additional APIs between the UDR and UDM. SD-Core forked an earlier version of free5GC and has additional access patterns for NF deregistration. Open5GS and OAI both had unique subscription-callback APIs that no other core implemented. We did not find any difference in the access patterns between Open5GS Releases 16 and 17, so we do not include separate rows in Table 2.

Missing Access Patterns: A dynamic access pattern is an access pattern that depends on runtime variables. CodeQL cannot fully resolve dynamic access patterns because it uses static analysis. False negatives in the extraction of access patterns leads to false positives in the subsequent policy analysis.

We performed a simple experiment to understand the impact of dynamic access patterns. Let C be the set of our initial conservative results. We then obtained a result set L with filters removed or default values for unresolved attributes. We then examine $L - C$ for APIs. In this expanded set, there are some unresolved attributes, but enough information to determine the access pattern manually. In these cases, our analysis identifies interesting code that requires further human analysis. We found 23 dynamic access patterns in all cores and all were redundant with respect to the other cores automated results. All identified dynamic access patterns are manually added to our access pattern set before the least privilege analysis.

Erroneous Access Patterns: API stubs are APIs that exist in the code and are reachable; however, they are not implemented. Most API stubs include a single print statement indicating it is not implemented. Unimplemented API stubs caused false positives in the extraction of access patterns, which can cause false negatives in the policy analysis. We identified 31 stub APIs manually, which we removed from our list of APIs before considering access patterns.

Summary: We find there are generally three sources of false results in our analysis. First, there is not enough information available statically to determine an access pattern. For example, in subscription-notification APIs, the endpoint called to the consumer is returned from a previous API call. Second, 5GAC-Analyzer overapproximates results as a precaution to handle dynamic access patterns, even if it occasionally results in an erroneous access pattern. Third, an access pattern exists in the calling NF, but the API does not exist in the called NF. This may occur when an implementation has partially implemented functionality.

5.3 Newly Discovered Policy Flaws

5GAC-Analyzer alerted that 3 of 19 services in the 3GPP specifications are over-privileged. None of these alerts were false positives. One of the alerts is for 3GPP TR 33.855 Key Issue #29, whereas the other two were previously unknown. We also discuss a third policy problem that was unrelated to our policy analysis which we reported.

Finding 1 - CVD-2022-0061 (AMF Re-Allocation): *A malicious UE release command may cause DoS or an on-path attack.* UE handover APIs within the `namf-comm` service are exposed to the SMF and PCF despite being AMF specific. The AMF directly connects to the RAN and the UE to provide control-plane messages to both. The AMF handles initial authentication with the core and ensures the UE is always reachable. NFs can be scaled up and down at any time depending on network conditions or specific requirements from slices. Additionally, when a UE moves geographically to a new RAN, it may make a certain AMF more suitable to serve a UE. In this scenario, an AMF handover will occur in the core network. The AMF serving the UE is the only entity that should initiate the handover to another AMF.

With the current policy, an SMF or PCF may initiate this handover. This over privilege can be exploited by malicious slices. We assume the attacker may take advantage of other APIs within the SMF or PCF when performing this attack. For example, a SMF or PCF can retrieve UE identifiers that are necessary to perform a handover. Previous work [16] has studied handover flaws that occur in the RAN, but they have yet to look at how an NF handover in the core may occur by an access control policy flaw. Slice A, which hosts an SMF, can access the core network's AMF to request a handover for a UE. Slice A can initiate an AMF handover to an AMF in another slice if Slice A has a UE identifier of any UE in the core network. Depending on the characteristics of the slices available, Slice A may move the target UE to an AMF under the attacker's control to eavesdrop or deny service.

Figure 5 shows the network connections for an attack exploiting this policy flaw. The operator AMF is one that the core normally runs, whereas the malicious AMF is one that an attacker will stand up on a separate slice of which they have control. In steps 1–3, the victim UE will begin normal operation with the core to authenticate and connect to the Internet. In step 4, a UE can make phone calls and connect to the Internet normally. Sometime later, either the PCF or SMF will launch one of two attacks, shown in steps 5a and 5b, respectively.

The attacker aims to take a victim UE connected to the core network offline for the first attack. As shown in step 5a, the attack

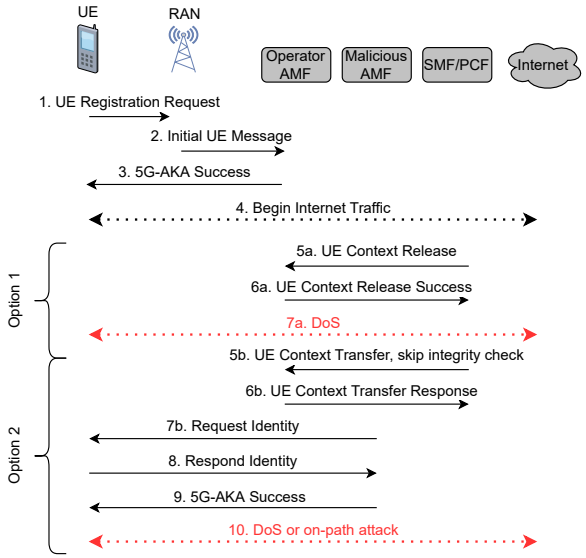

Figure 5: The AMF Re-Allocation attack allows the SMF or PCF to initiate an AMF switch for a UE. The attacker may switch the UE to a new slice that the attacker controls, causing DoS or on-path attacks.

has the PCF or SMF from one slice send a malicious request to the AMF on another slice to release any UE context. Releasing a UE context should only occur after an AMF handover procedure is successful, and the initial AMF is to delete data on the transferred UE. However, releasing UE context is allowed by the PCF and SMF in the 3GPP access control policy at any time. If the UE context release request succeeds, shown in step 6a, then DoS will occur for that UE, shown in step 7a. The UE will try to keep communicating with the AMF, but the AMF will not have enough information to serve the UE. At this point, the core network will need to initialize the UE context causing intermittent connectivity.

The second attack is similar to the first and starts in step 5b. Instead of requesting a UE context release, a rogue PCF or SMF can instead request a UE context transfer which begins an AMF handover procedure, shown in step 5b. If the request reason is specified as MOBI_REG_UE_VALIDATEED, then the AMF shall not perform an integrity check on the request, allowing the attack to proceed [2]. When the request succeeds in step 6b, the malicious AMF must perform a 5G-AKA with the UE. If 5G-AKA succeeds, the malicious AMF can perform an on-path attack on the UE's Internet traffic shown in step 10. DoS for the UE occurs if any failure occurs after step 6b is successful.

Finding 2 - CVD-2022-0062 (Subscription Data Management Exposure): *Sensitive UE subscription information is exposed to NFs besides the UDM.* APIs handling subscriber data in `nudm-sdm` are exposed to the SMF and AMF. We found an issue similar to 3GPP TR 33.855 Key Issue #29 in the "nudm-sdm" service. This service aims to read the UE subscription data from the UDR and provide portions of it to consumer NFs. The nudm-sdm service only achieves its designed purpose with a least-privileged policy. The access control policy for this service allows any of the consumer NFs to read any of

```
1  openapi: 3.0.0
2  info:
3    version: 1.0.8
4    title: Namf_Communication
5    description: |
6      AMF Communication Service
7      © 2022, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI,
         TSDSI, TTA, TTC).
8      All rights reserved.
9  security:
10   - {}
11   - oAuth2ClientCredentials:
12       - namf-comm
13 externalDocs:
14   ...
15 servers:
16   - url: '{apiRoot}/namf-comm/v1'
17     ...
18 paths:
19   /ue-contexts/{ueContextId}: ...
20   /ue-contexts/{ueContextId}/release: ...
21   ...
```

Figure 6: Example OAuth policy in the AMF Communication service [3]. Line 10 indicates that an OAuth token is optional for all endpoints in the service [52].

the exposed UE's subscription data; however, cryptographic information is not exposed. For example, the endpoint *POST /{supi}/am-data* exposes access and mobility data intended for the internal operation of the AMF. However, the SMF also has access to this endpoint and can retrieve information. Additionally, the SMF can delete a subscription, which will cause instability in the AMF or another NF that utilizes this service's subscriptions. The SMF can create a subscription to data it should not access in the nudm-sdm service, causing a potential privacy violation. Our analysis determined that the SMF only needs to access data exposed by the *GET /{supi}/sm-data* endpoint.

5.4 Additional Findings

During the process of extracting and modeling the 3GPP OAuth policy, we identified a problem with how the OAuth policy was encoded in OpenAPI. Specifically, we found that the 3GPP policy includes null access rules for each operation. While the 3GPP specification states that OAuth is optional, an implementation enforcing OAuth will interpret the null access rules as allowing all callers. Hence, these rules negate the OAuth policy entirely.

Finding 3 - CVD-2022-0060 (Negated OAuth Policy): Figure 6 shows an example service's YAML security definition. In an OpenAPI security definition, the empty braces on line 10 mean that OAuth is optional. The security requirements object in OpenAPI accepts a list of security requirements, and only one list item needs to be met for the request to be allowed. Therefore, even though additional OAuth requirements exist on an API on lines 11–12, all rules following an empty brace are negated. Note that line 10 defines the security requirements for the entire service. If the policy defines security requirements for individual APIs, the requirements for the API override the global requirement. However, we found that the 3GPP specification uses empty braces for all security requirements, both global and per-API.

```
1  Procedure CalculatePolicy(S, A)
2      Γ ← ∅;
3      for s ∈ S do
4          G ← ∅ // Set of 2^O × 2^N pairs
5          for o ∈ s do
6              C := {n|(n, o) ∈ A} // All n that call o
7              found ← false;
8              for (O, N) ∈ G do
9                  if N = C then
10                     E.append(e) // update O in G
11                     found ← true;
12                     continue;
13             if found = false then
14                 G.append({o}, C);
15         for (O, N) ∈ G do
16             p ← CreatePermissionName(s, O);
17             for o ∈ O do
18                 for n ∈ N do
19                     Γ.append(n, o, p);
20     return Γ;
```

Algorithm 1: Algorithm to create a least-privilege policy for a set of services S based on a set of access patterns A

This flaw results from a misunderstanding of the empty braces feature in OpenAPI. The OpenAPI specifications [52] give an example of optional *policy* using empty braces. However, we think optional *enforcement* is the intended purpose in the 3GPP specifications. Given the current policy specification, 5G core implementations that wish to enforce OAuth must manually remove all empty braces, which is error prone.

When we reported this flaw to the GSMA they in turn contacted OpenAPI about the semantics of the specification. After multiple rounds of email exchange, the GSMA concluded that the existing policy was ambiguous and requires modification.

6 PROPOSED POLICY FIXES

To address the over-privilege, 5GAC-Analyzer generates a least-privilege policy. Our policy analysis suggests defining six operation-level scopes in two services that only define service-level scopes. Each of the six scopes has a lesser privilege than the two currently existing access tokens. Note that our proposed changes to the OAuth policy do not prevent any functionality and instead require each implementation to request a different access token.

6.1 Generating Least-Privilege Policy

We derive a least-privilege policy based on a set of services S and a set of access patterns A. Recall that each $s \in S$ is a set of operations. Conceptually, our analysis seeks to achieve least-privilege without unusable permission bloat (e.g., one permission per operation). The existing 3GPP policy defines most permissions at the service-level. Therefore, our analysis uses service-level permissions whenever possible.

Let O_s be the set of operations for service s. If all $o \in O_s$ have the same set of accessing NFs in A, then we define the permission at the granularity of the service. Otherwise, we cluster each o by NF callers and create new permissions based on the set of callers for each s. The 3GPP policy already uses service names for the scope in an access token. For finer-grained names, we create names based on the prefixes of the operation names.

Table 3: Proposed changes to the access control policy in the 5G core.

Service	Consumer NF	Proposed Access Token	APIs	Finding
namf-comm	SMF	namf-comm:n1n2-messages	POST /ue-contexts/{ueContextId}/n1-n2-messages	Finding 1
	AMF	namf-comm:transfer	POST /ue-contexts/{ueContextId}/release POST /ue-contexts/{ueContextId}/transfer POST /ue-contexts/{ueContextId}/transfer-update PUT /ue-contexts/{ueContextId}	Finding 1
	PCF	namf-comm:subscriptions	POST /subscriptions	Finding 1
nudm-sdm	SMF	nudm-sdm:sm-data	GET /{supi}/sm-data	Finding 2
	UDM	nudm-sdm:shared-data	GET /shared-data POST /shared-data-subscriptions DELETE /shared-data-subscriptions/{subscriptionId}	Finding 2
	AMF	nudm-sdm:amf-data	POST, DELETE /{ueId}/sdm-subscriptions/* GET /{supi}/nssai PUT, GET /{supi}/am-data/* GET {supi}/smf-select-data GET {supi}/ue-context-in-smf-select-data	Finding 2
nudr-dr	UDM	nudr-dr:subscription-data	GET, POST, DELETE, PATCH /subscription-data/*	Fixed in 3GPP R17
	PCF	nudr-dr:policy-data	GET, POST, DELETE, PATCH /policy-data/*	Fixed in 3GPP R17

Definition 3 (Least-Privilege Policy)**.** Let a Γ be a policy and A be a set of access patterns. Γ is least-privilege with respect to A if $\forall(n, o, p) \in \Gamma$, $(n, o) \in A$ (security requirement) and $\forall(n, o) \in A$, $(n, o, \_) \in \Gamma$ (functionality requirement), where _ represents any permission.

Conceptually, a least-privilege policy is an access control policy where each access rule corresponds to an access pattern. Algorithm 1 creates the least-privilege policy based on a set of services S and access patterns A. It starts with an empty policy Γ and builds it up by going through every service $s \in S$, filling in the structure. In lines 4-14, we create groups of operations G, which is a tuple of a set of operations and an NF (O, N). In lines 16-19, we create a permission for each group of operations, which ensures a new permission for each access pattern. The result of the algorithm is an access control policy that can be used to further refine the specifications.

Policy Changes: We propose six new access tokens to address the overprivileged policy. Table 3 shows the proposed tokens. These six access tokens prevent the consumer NF from accessing every API in the service and limit them to only the APIs described in the table. The first changes in the access control policy relate to `namf-comm` and Finding 1. Three new access tokens separate the exposed APIs into NF-specific groups, negating the scenarios discussed in Section 5.3. These tokens are shown in rows 2-4 of Table 3.

The next changes are in the `nudm-sdm` service, exposed by the UDM. This service consumes the subscription data from the UDR and provides other NFs with only what they need. Three new tokens prevent consumer NFs from receiving more data than they require. These tokens are described in rows 5-7 of Table 3. Finally, the changes described in 3GPP TR 33.855 Key Issue #29 were discovered independently by our analysis, so we include them in our findings (`nudr-dr`). However, the access control policy changes for `nudr-dr` were implemented in 3GPP Release 17 [4].

7 DISCUSSION

Incomplete Implementations: Cellular infrastructure always lags the 3GPP specifications, and even production implementations often fail to implement all features [38, 43]. The open source cores we analyzed have not yet implemented every NF and API defined by the 3GPP; however, they are fully working 5G cores. Infact, OAI and Open5GS have commercial backing for enterprises [26, 47]. Our analysis is limited to what code is implemented and used in the union of all analyzed 5G cores. We expect our analysis to continue working in future 3GPP releases of the open-source implementations we have analyzed. This is because these implementations will continue adding new HTTP calls between the NFs in the same way as they have previously.

We note that if an implementation has incorrectly added access patterns that should not exist, our analysis will not identify this as an instance of over-privilege. We believe all of our results are representative of the current functionality of the 5G cores, as discussed in Section 5. Our analysis identifies NF interactions by examining HTTP calls, therefore, NFs that do not implement a service-based architecture such as the UPF, or Non-3GPP Interworking Function (N3IWF) are not included.

OAuth Enforcement in Implementations: While we used the 5G implementations to discover flaws in the 3GPP policy, we did not discover vulnerabilities in the implementations themselves. In fact, none of the analyzed implementations currently enforce OAuth between NFs. To test our least-privileged OAuth policy, we created 5GAC-Instrumenter, which automatically instruments NFs in free5GC to request, send, and validate OAuth tokens based on a policy. All OAuth procedures can be turned on or off with a configuration option compatible with the existing configuration in free5GC. 5GAC-Instrumenter inserts access token requests to all proxies in free5GC and token validation at all entry points. We did not encounter any problems running free5GC with the least-privileged OAuth policy. We created pull requests in free5GC with the OAuth-enforced code base which are under an active discussion. free5GC only began to implement OAuth functionality when we created pull requests to add it. The work to implement OAuth in free5GC is still ongoing [27].

8 RELATED WORK

Static Analysis for Access Control: Static analysis has been used in prior work to identify a security policy [17, 37, 40, 53] or confine system calls within an OS or container [29, 44]. Authorization verification has been widely studied for at least two decades [23, 28, 60]. Li et al. [40] analyzed microservice code using static analysis to extract interactions between the microservices and make a security policy. They enforced the security policy at the container orchestration level in Kubernetes. This architecture does not apply to a 5G core as OAuth is the access control mechanism in the 5G core [3].

NLP is also common to extract the access control policy from standards documents [9, 57]. Xiao et al. [57] introduce Text2Policy which parses and annotates words in phrases. Then these annotations are used to extract entities, actions and objects of the access control policy. These are then converted into a machine-readable format and implemented. Text2Policy identifies between 80 and 90 percent of access control policy-relevant sentences and actions. NLP-based often has many limitations and misses important information. Fortunately, in the 5G specifications, we are given machine-readable descriptions of all APIs.

OAuth: OAuth 2.0 is a widely used authorization framework that is used across many platforms, such as web, mobile, and IoT [30, 34, 56]. Most security efforts towards OAuth have focused on errors in specific implementations [12, 18, 39, 50, 51, 54, 59]. A study of over 600 mobile applications found that around 60% of them incorrectly implement OAuth and are vulnerable [18]. A more recent study performed on Android applications found that this number has decreased to only 32% of applications implementing an OAuth security mistake [45]. An analysis of OAuth libraries identified most of them do not implement Cross-Site Request Forgery protection [51]. Some studies [46, 58] analyzed OAuth provider implementations to identify flaws in the OAuth protocol. They did not examine particular access control policies checking for over privilege, but only that OAuth implementations correctly enforce OAuth.

The OAuth standard is written in natural language [30]. An analysis of the OAuth protocol revealed four previously unknown vulnerabilities [25]. They found these vulnerabilities to be exploitable in practice and considered all four grant types of OAuth, including client credentials.

Cellular Network Security: All cellular generations have their own access control challenges. "phreakers" can be highlighted as the first instances of failed access control in telecommunications [48]. After the transition to SS7, multiple vulnerabilities were discovered within the protocol [41]. However, cellular networks have gotten significantly more complex since 2G. SMS was found to be vulnerable to large-scale botnet attacks [55]. More recently introduced protocols also have shortcomings. For example, issues in 5G-AKA have been identified with Tamarin by modeling the protocol formally [13, 22]. 5G-AKA was found to be vulnerable to spoofing and authentication bypass race conditions. Multiple works [7, 14, 49] highlight access control challenges from newly introduced 5G features, such as multi-tenancy, and NF virtualization. Concurrent to our paper, Akon et al. [8], formally analyzed the 3GPP specifications to identify potential problems in how OAuth is enforced; however, we expand the analysis to consider the OAuth policy directly.

LTEInspector and 5GReasoner are symbolic model checkers for LTE and 5G [31, 32]. Numerous works have analyzed specific portions of 5G standards and have suggested various improvements [6, 21, 33]. All of these 5G works study specifications, and do not consider implementations. Various LTE protocols have been fuzzed in real implementations, discovering DoS, spoofing, and eavesdropping on user traffic [36]. BaseSAFE fuzzes baseband firmware embedded in phones by targeting individual functions of interest [42]. BASESPEC compares structures embedded in baseband firmware with the same structures in the specification, uncovering 0-day remote execution attacks [35]. Importantly, both works focus on targeting UE devices rather than the RAN or core network.

Closest to our work, Akon et al. [8], formally modeled 5G's access control between NFs and found many issues related to 5G's slicing capability. They found that a malicious NF could obtain an access token to a NF of a certain type (i.e., AMF) and then use that token to access an AMF on a slice they should not. Another flaw they found was that if no network slice was specified in the access token request, they would be granted access to all slices in the access token. Among these and other identified flaws, they tested these attacks work on free5GC. To identify the flaws, they analyze the NFProfile structure of a NF and consider combinations of possible values described by the specifications. However, they do not consider whether the specifications have flaws, which was highlighted by 3GPP TR 33.855 Key Issue #29.

9 CONCLUSION

The 5G core makes significant advances over the 4G core by switching from specialized hardware to a service-based architecture running on commodity cloud infrastructure and supporting third-party NFs. These changes introduce new attack surfaces that require careful consideration. This paper used static analysis of open-source 5G core implementations as a source of knowledge to identify flaws in the 3GPP's OAuth access control policy. In doing so, we identified two previously unknown policy flaws. These flaws allow a malicious NF to interrupt UE network connectivity and extract subscriber information. We further proposed policy fixes to address overprivilege in the specification which will benefit all implementations. By adopting our refined policy, the 3GPP can reduce the attack surface of the 5G core from new threats introduced by changes in 5G.

ACKNOWLEDGMENTS

This work is supported in part by NSF grants CNS-2054911 and CNS-2055014. Any findings and opinions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. We also thank Patrick Traynor and Roger Brown for their feedback during the writing of this paper.

REFERENCES

[1] 3GPP. 2020. *Study on security aspects of the 5G Service Based Architecture (SBA).* Technical Report (TR). 3rd Generation Partnership Project (3GPP). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3481

[2] 3GPP. 2022. *3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; 5G System; Unified Data Management Services; Stage 3.* Technical Standard (TS) 29.503. 3rd Generation Partnership Project (3GPP).

https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3342

[3] 3GPP. 2022. *3rd Generation Partnership Project; Technical Specification group Services and System Aspects; Security architecture and procedures for 5G system.* Technical Standard (TS) 33.501. 3rd Generation Partnership Project (3GPP). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169

[4] 3GPP. 2022. Release 17. https://www.3gpp.org/specifications-technologies/releases/release-17

[5] 3GPP. 2023. 3GPP Portal > Home. https://portal.3gpp.org/#/

[6] Ijaz Ahmad, Tanesh Kumar, Madhusanka Liyanage, Jude Okwuibe, Mika Ylianttila, and Andrei Gurtov. 2017. 5G security: Analysis of threats and solutions. In *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 193–199.

[7] Ijaz Ahmad, Tanesh Kumar, Madhusanka Liyanage, Jude Okwuibe, Mika Ylianttila, and Andrei Gurtov. 2018. Overview of 5G security challenges and solutions. *IEEE Communications Standards Magazine* 2, 1 (2018), 36–43.

[8] 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)*. Association for Computing Machinery, Copenhagen, Denmark.

[9] Manar Alohaly, Hassan Takabi, and Eduardo Blanco. 2018. A Deep Learning Approach for Extracting Attributes of ABAC Policies. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies (SACMAT '18)*. Association for Computing Machinery, New York, NY, USA, 137–148. https://doi.org/10.1145/3205977.3205984 event-place: Indianapolis, Indiana, USA.

[10] 5G Americas. 2021. *Private & Enterprise Networks.* Technical Report. 5G Americas, Bellevue, WA. 39 pages. https://www.5gamericas.org/private-and-enterprise-networks/

[11] 5G Americas. 2021. *Security for 5G.* Technical Report. 5G Americas, Bellevue, WA. 39 pages. https://www.5gamericas.org/security-for-5g/

[12] Chetan Bansal, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, and Sergio Maffeis. 2014. Discovering concrete attacks on website authorization by formal analysis. *Journal of Computer Security* 22, 4 (2014), 601–657. Publisher: IOS Press.

[13] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1383–1396. https://doi.org/10.1145/3243734.3243846

[14] P Bisson and J Waryet. 2017. 5G PPP phase1 security landscape. *5G PPP Security Group White Paper* (2017), 66 pages.

[15] Evangelos Bitsikas, Syed Khandker, Ahmad Salous, Aanjhan Ranganathan, Roger Piqueras Jover, and Christina Pöpper. 2023. UE Security Reloaded: Developing a 5G Standalone User-Side Security Testing Framework. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23)*. Association for Computing Machinery, New York, NY, USA, 121–132. https://doi.org/10.1145/3558482.3590194 event-place: Guildford, United Kingdom.

[16] Evangelos Bitsikas and Christina Pöpper. 2021. Don't hand it Over: Vulnerabilities in the Handover Procedure of Cellular Telecommunications. In *Annual Computer Security Applications Conference (ACSAC)*. Association for Computing Machinery, New York, NY, USA, 900–915. https://doi.org/10.1145/3485832.3485914

[17] Paolina Centonze, Robert J. Flynn, and Marco Pistoia. 2007. Combining Static and Dynamic Analysis for Automatic Identification of Precise Access-Control Policies. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*. 292–303. https://doi.org/10.1109/ACSAC.2007.39

[18] Eric Y. Chen, Yutong Pei, Shuo Chen, Yuan Tian, Robert Kotcher, and Patrick Tague. 2014. OAuth Demystified for Mobile Application Developers. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Scottsdale Arizona USA, 892–903. https://doi.org/10.1145/2660267.2660323

[19] William Cheng, Jim Schubert, Christopher Bornet, Jeremie Bresson, sunn, and Justin Black. 2024. Hello from OpenAPI Generator. https://openapi-generator.tech/

[20] Merlin Chlosta, David Rupprecht, and Thorsten Holz. 2021. On the challenges of automata reconstruction in LTE networks. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 164–174. https://doi.org/10.1145/3448300.3469133

[21] Merlin Chlosta, David Rupprecht, Christina Pöpper, and Thorsten Holz. 2021. 5G SUCI-catchers: still catching them all?. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 359–364.

[22] Cas Cremers and Martin Dehnel-Wild. 2019. Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion. (2019).

[23] Antony Edwards, Trent Jaeger, and Xiaolan Zhang. 2002. Runtime Verification of Authorization Hook Placement for the Linux Security Modules Framework. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*. Association for Computing Machinery, New York, NY, USA, 225–234. https://doi.org/10.1145/586110.586141 event-place: Washington, DC, USA.

[24] Ericsson. 2021. *The essential building blocks of E2E network slicing.* Technical Report. Ericsson, Stockholm, Sweden. 11 pages.

[25] Daniel Fett, Ralf Küsters, and Guido Schmitz. 2016. A Comprehensive Formal Security Analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1204–1215. https://doi.org/10.1145/2976749.2978385 event-place: Vienna, Austria.

[26] Firecell. 2023. Orion 5G - firecell.io. https://firecell.io/orion-5g/

[27] free5gc 2024. free5GC. https://www.free5gc.org/

[28] Vinod Ganapathy, Trent Jaeger, and Somesh Jha. 2005. Automatic Placement of Authorization Hooks in the Linux Security Modules Framework. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*. Association for Computing Machinery, New York, NY, USA, 330–339. https://doi.org/10.1145/1102120.1102164 event-place: Alexandria, VA, USA.

[29] Seyedhamed Ghavamnia, Tapti Palit, Azzedine Benameur, and Michalis Polychronakis. 2020. Confine: Automated System Call Policy Generation for Container Attack Surface Reduction. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. USENIX Association, San Sebastian, 443–458. https://www.usenix.org/conference/raid2020/presentation/ghavanmnia

[30] Dick Hardt. 2012. *The OAuth 2.0 Authorization Framework.* Request for Comments RFC 6749. Internet Engineering Task Force. https://doi.org/10.17487/RFC6749 Num Pages: 76.

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

[32] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 2019. 5GReasoner: A Property-Directed Security and Privacy Analysis Framework for 5G Cellular Network Protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, London United Kingdom, 669–684. https://doi.org/10.1145/3319535.3354263

[33] Madhusanka Liyanage IjazAhmad, Shahriar Shahabuddin, Mika Ylianttila, and Andrei Gurtov. 2018. Design principles for 5G security. *A Comprehensive Guide to 5G Security* (2018), 75.

[34] Michael Jones, John Bradley, and Nat Sakimura. 2015. JSON Web Token (JWT). https://doi.org/10.17487/RFC7519 Issue: 7519 Num Pages: 30 Series: Request for Comments Published: RFC 7519.

[35] Eunsoo Kim, Dongkwan Kim, CheolJun Park, Insu Yun, and Yongdae Kim. 2021. BaseSpec: Comparative Analysis of Baseband Software and Cellular Specifications for L3 Protocols. In *Proceedings 2021 Network and Distributed System Security Symposium*. Internet Society, Virtual. https://doi.org/10.14722/ndss.2021.24365

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

[37] Sven Lachmund. 2010. Auto-generating access control policies for applications by static analysis with user input recognition. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems - SESS '10*. ACM Press, Cape Town, South Africa, 8–14. https://doi.org/10.1145/1809100.1809102

[38] Oscar Lasierra, Gines Garcia-Aviles, Esteban Municio, Antonio Skarmeta, and Xavier Costa-Pérez. 2023. European 5G Security in the Wild: Reality versus Expectations. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23)*. Association for Computing Machinery, Guildford, Surrey, United Kingdom, 6. https://doi.org/10.1145/3558482.3581776

[39] Wanpeng Li and Chris J Mitchell. 2014. Security issues in OAuth 2.0 SSO implementations. In *International Conference on Information Security*. Springer, 529–541.

[40] Xing Li, Yan Chen, Zhiqiang Lin, Xiao Wang, and Jim Hao Chen. 2021. Automatic Policy Generation for Inter-Service Access Control of Microservices. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3971–3988. https://www.usenix.org/conference/usenixsecurity21/presentation/li-xing

[41] G Lorenz, T Moore, G Manes, J Hale, and S Shenoi. 2001. Securing ss7 telecommunications networks. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, Vol. 2. Workshop on Information Assurance and Security, West Point, NY, 273–278.

[42] Dominik Maier, Lukas Seidel, and Shinjo Park. 2020. BaseSAFE: Baseband SAnitized Fuzzing through Emulation. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, Linz Austria, 122–132. https://doi.org/10.1145/3395351.3399360

[43] Shiyue Nie, Yiming Zhang, Tao Wan, Haixin Duan, and Song Li. 2022. Measuring the Deployment of 5G Security Enhancement. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22)*. Association for Computing Machinery, New York, NY, USA, 169–174. https://doi.org/10.1145/3507657.3528559

[44] Shankara Pailoor, Xinyu Wang, Hovav Shacham, and Isil Dillig. 2020. Automated policy synthesis for system call sandboxing. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (Nov. 2020), 135:1–135:26. https://doi.org/10.1145/3428203 Number: OOPSLA.

[45] Tamjid Al Rahat, Yu Feng, and Yuan Tian. 2019. OAUTHLINT: An Empirical Study on OAuth Bugs in Android Applications. In *2019 34th IEEE/ACM International*

Conference on Automated Software Engineering (ASE). IEEE, San Diego, CA, USA, 293–304. https://doi.org/10.1109/ASE.2019.00036

[46] Tamjid Al Rahat, Yu Feng, and Yuan Tian. 2022. Cerberus: Query-Driven Scalable Vulnerability Detection in OAuth Service Provider Implementations. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, New York, NY, USA, 2459–2473. https://doi.org/10.1145/3548606.3559381 event-place: Los Angeles, CA, USA.

[47] RAKwireless. 2023. RAK All-in-One. 5G – Build Your Own Private 5G Network. https://www.rakwireless.com/en-us/5g

[48] Ron Rosenbaum. 1971. Secrets of the little blue box. *Esquire Magazine* 76 (1971), 117–125.

[49] AdaptiveMobile Security. 2021. *White Paper: A Slice in Time: Slicing Security in 5g Core Networks*. Technical Report. https://info.adaptivemobile.com/5g-network-slicing-security

[50] Mohamed Shehab and Fadi Mohsen. 2014. Towards enhancing the security of oauth implementations in smart phones. In *2014 IEEE International Conference on Mobile Services*. IEEE, 39–46.

[51] Ethan Shernan, Henry Carter, Dave Tian, Patrick Traynor, and Kevin Butler. 2015. More guidelines than rules: CSRF vulnerabilities from noncompliant OAuth 2.0 implementations. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 239–260.

[52] SmartBear Software. 2020. OpenAPI Specification - Version 3.0.3 | Swagger. https://swagger.io/specification/#security-requirement-object

[53] Fangqi Sun, Liang Xu, and Zhendong Su. 2011. Static Detection of Access Control Vulnerabilities in Web Applications. In *20th USENIX Security Symposium (USENIX Security 11)*. USENIX Association, San Francisco, CA. https://www.usenix.org/conference/usenix-security-11/static-detection-access-control-vulnerabilities-web-applications

[54] San-Tsai Sun and Konstantin Beznosov. 2012. 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 (CCS '12)*. Association for Computing Machinery, New York, NY, USA, 378–390. https://doi.org/10.1145/2382196.2382238 event-place: Raleigh, North Carolina, USA.

[55] Patrick Traynor, Michael Lin, Machigar Ongtang, Vikhyath Rao, Trent Jaeger, Patrick McDaniel, and Thomas La Porta. 2009. On cellular botnets: measuring the impact of malicious devices on a cellular network core. In *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, Chicago Illinois USA, 223–234. https://doi.org/10.1145/1653662.1653690

[56] Hui Wang, Yuanyuan Zhang, Juanru Li, Hui Liu, Wenbo Yang, Bodong Li, and Dawu Gu. 2015. Vulnerability Assessment of OAuth Implementations in Android Applications. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC '15)*. Association for Computing Machinery, New York, NY, USA, 61–70. https://doi.org/10.1145/2818000.2818024 event-place: Los Angeles, CA, USA.

[57] Xusheng Xiao, Amit Paradkar, Suresh Thummalapenta, and Tao Xie. 2012. Automated Extraction of Security Policies from Natural-Language Software Documents. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE '12)*. Association for Computing Machinery, New York, NY, USA, 11. https://doi.org/10.1145/2393596.2393608 event-place: Cary, North Carolina.

[58] Ronghai Yang, Wing Cheong Lau, Jiongyi Chen, and Kehuan Zhang. 2018. Vetting Single Sign-On SDK Implementations via Symbolic Reasoning. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1459–1474. https://www.usenix.org/conference/usenixsecurity18/presentation/yang

[59] Ronghai Yang, Guanchen Li, Wing Cheong Lau, Kehuan Zhang, and Pili Hu. 2016. 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 (ASIA CCS '16)*. Association for Computing Machinery, New York, NY, USA, 651–662. https://doi.org/10.1145/2897845.2897874

[60] Xiaolan Zhang, Antony Edwards, and Trent Jaeger. 2002. Using CQUAL for Static Analysis of Authorization Hook Placement. In *Proceedings of the 11th USENIX Security Symposium*. USENIX Association, USA, 33–48.