Automated Analysis of OAuth Policy between Network Functions in the 5G Core

Seaver Thorn, K. Virgil English, William Enck
North Carolina State University
{swthorn,kvenglis,whenck}@ncsu.edu
Kevin R.B. Butler
University of Florida
butler@ufl.edu

***Abstract*—The transition to 5G cellular networks will bring disruptive change to both industry and society. Two key disruptive innovations are (1) running the 5G core network on commodity cloud infrastructure and (2) introducing 5G applications that allow third parties to implement value-added services for end users and businesses. While these changes will foster significant innovation, they also open cellular networks—particularly network cores—to new attack surfaces. This paper provides the first security analysis of access control *within* the 5G core. We begin by performing static analysis of a 5G core reference implementation to derive least-privilege access patterns. We contrast the access patterns with the OAuth policy in the 3GPP specifications to identify two new sources of over-privilege, which allow a malicious network function to interrupt user equipment connectivity and extract subscriber information. We then use the access patterns to derive a least-privilege OAuth policy and automatically instrument the 5G core reference implementation to enforce the policy. By adopting our refined policy, 5G core implementations will reduce the attack surface presented by opening the network to new threats.**

I. Introduction

The cellular industry is in the process of migrating to 5G. As with every new generation of cellular technology, 5G will bring disruptive change to both industry and society, promising to connect everything from factories and agriculture to vehicles and smart cities. However, 5G's innovation goes far beyond simply connecting devices: its main appeal is providing *services* for these devices. To support a wide variety of services and customers, the mobile core has been fundamentally rearchitected to disaggregate traditional functional entities (e.g., HSS, MME, PGW) into many different Network Functions (NFs). Each NF is implemented as Cloud Native Functions (CNFs) organized within a Service-based Architecture (SBA) running on both private and public clouds. Customers will run 5G applications that interact with the 5G core, providing service-specific value-added services for end users and businesses.

Existing academic literature on 5G security has been limited. There have been several security studies performing protocol analysis [38], with a specific focus on the 5G Authentication and Key Agreement (5G-AKA) [16], [28]. Beyond these core protocols, other studies examined how a malicious Radio Access Network (RAN) could exploit the 5G system [21]. Researchers have also analyzed the standards documents using natural language processing [25].

No prior work has considered the security of the 5G core. Unlike the mobile core in prior generations, the 5G core is designed to run on public infrastructure where perimeter defenses such as firewalls provide limited protection. Furthermore, a key new feature is the introduction of 5G applications, which allow third-parties to define and run NFs that interact with the 5G core. Insufficient protection between NFs can compromise 5G core functionality, including disclosure of sensitive subscriber information and selective DoS. To combat such attacks, 5G Americas [13] recommends deploying a Zero Trust Architecture (ZTA) that includes the use of OAuth tokens to authorize requests between NFs.

The 3GPP standards include optional specifications for using OAuth tokens to authorize requests between NFs [6]. The standards provide a machine-readable OAuth access control policy as part of OpenAPI specification YAML files. 5G core implementations use the OpenAPI specifications to automatically generate the code for making and handling network requests between NFs. Therefore, one might expect access control between 5G core NFs to be a solved problem.

Studying OAuth in the 5G core, we make two observations. First, a *manual* review of the access control policy 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) [5]. While this flaw has been fixed in 3GPP Release 17 [4], an automated approach is likely to identify additional policy flaws. Second, no existing open source 5G core implementation includes any code for requesting or validating OAuth tokens. In fact, the OpenAPI generator tool for Go (the language used by free5GC) does not even support the OAuth Client Credentials type used by the 3GPP OpenAPI specification.

In this paper, we define and instrument least-privilege access control for requests between NFs in the 5G core. We accomplish this goal by introducing two tools: *5GAC-Analyzer* and *5GAC-Instrumenter*. 5GAC-Analyzer uses static

program analysis of a 5G reference implementation[1] to extract access patterns that represent least-privilege. We use the access patterns in two ways. First, we derive a least privilege policy that optimizes usability by defining permissions at the service granularity whenever possible. Second, we compare the access patterns to the 3GPP defined policy, identifying two additional sources of over-privilege not identified by the 3GPP's manual analysis. The newly identified policy flaws (a) provide an additional attack vector to extract subscriber information, and (b) allow interruption of User Equipment (UE) connectivity.

Next, we modify the 5G reference implementation to enforce an OAuth policy. While we manually extended the Network Repository Function (NRF) to respond to OAuth token requests based on a given policy file, 5GAC-Instrumenter automatically inserted code for 78 OAuth token requests and 64 validation checks into NFs. The OAuth token requests double the number of network calls (and therefore time) required to perform many operations within the core. However, we experimentally demonstrate that these operations infrequently impact UEs. In particular, the addition of OAuth checks has a negligible impact on UE network throughput. Furthermore, when compared to the baseline 3GPP access control policy, our least-privilege policy has a negligible impact on performance throughout the core.

We make the following contributions in this paper:

- *We derive a least-privileged access control policy for the 5G core.* We perform program analysis to identify NF interactions by using a 5G reference implementation. From these interactions, we derive an access control policy that balances usability and security.
- *We identify two previously unknown access control flaws in the 3GPP policy.* In addition to identifying the same over-privilege previously reported in 3GPP TR 33.855 Key Issue #29, our automated analysis identified two additional access tokens definitions that result in over-privilege. We are in the process of disclosing these vulnerabilities to the GSMA.
- *We automatically instrument a 5G core reference implementation to enforce least-privilege.* No open-source 5G core implements access control between NFs. The instrumentation is policy-agnostic, allowing the use of the 3GPP security policy or our optimized least-privileged policy.

We note that our approach is limited by the 5G core functionality implemented in the reference implementation. This paper considers version 3.0.7 of free5GC, which implements 3GPP Release 15. As discussed in Section IV, free5GC includes incomplete stubs for some core functionality. This functionality is not included in our least-privilege policy. As the implementation matures, 5GAC-Analyzer can be re-run to derive a more complete least-privilege policy. We will release the source code of 5GAC-Analyzer and 5GAC-Instrumenter to support these future efforts. Furthermore, we will attempt to upstream both our OAuth enhancements to free5GC and additions to the OpenAPI-generator so that future projects using the 3GPP OpenAPI specifications can define NFs with OAuth checks.

[1]We use free5GC [1] as our reference implementation, which is the base of the Open Networking Foundation (ONF)'s 5G mobile core SD-Core [51].

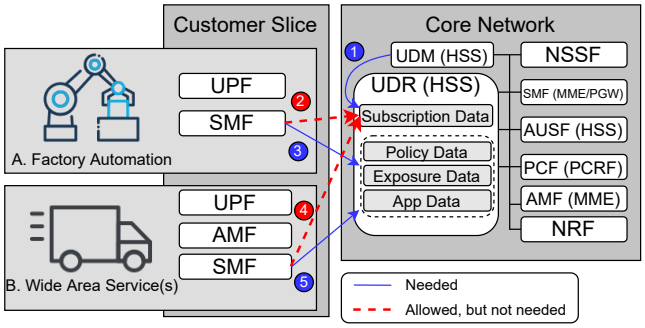

Fig. 1: Example of different 5G slice configurations with annotations describing 3GPP TR 33.855 Key Issue #29. The UDR's subscription data is more sensitive than the other data it exposes. While only the UDM needs to access the subscription data, the 3GPP policy allows all NFs to access it.

The remainder of the paper proceeds as follows. Section II provides background and motivation. Section III provides an overview of our design. Section IV describes our policy analysis. Section V describes how we instrument the 5G reference core with OAuth. Section VI details the discovered security flaws. Section VII describes the performance impact of the implementation. Section VIII discusses limitations. Section IX overviews related work. Section X concludes.

II. 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 SBA. The SBA defines functionality in the network core through a software Application Programming Interface (API) over HTTP. Using these exposed APIs, 3GPP standards define sets of NF to NF interactions. In addition, the 3GPP standards allow some flexibility in NF implementation; for example, developers can implement NFs as stateful or stateless.

Each NF has a purpose and required functionality to comply with the standards. The User Plane Function (UPF) provides the interface between the UE and the Data Network (DN), which the S-GW and P-GW performed in 4G. The Access & Mobility Management Function (AMF) provides registration, connection, and reachability management for a UE, roughly equivalent to the MME in 4G. The Session Management Function (SMF) allocates IP addresses to UEs and handles session management, including tunneling to the UPF. The SMF combines certain portions of the S-GW, P-GW, and the MME in 4G. The Unified Data Repository Function (UDR) is a centralized repository for subscriber, application, exposure, and other data. The Unified Data Management Function (UDM) manages data from the UDR and distributes it to the corresponding NFs. 5G splits the HSS into the UDR and UDM. Some NFs will have an equivalent function or two in 4G, and some are new to 5G. For example, the NRF is new to 5G and has no equivalent component in 4G.

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. 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 [29]. Network slicing defines separate logical networks on the same physical network. 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 CNFs. Network core operators provision and configure slices based on agreements between the business and operators. The most common NFs for slices are the UPF, followed by the AMF and SMF, then the UDR and UDM [12], [13], [29].

The 5G core network or a slice can deploy a 5G application. For example, a network operator may provide an additional video streaming service to all customers as a 5G app. This app would have access to subscriber information and can provide additional content to customers who pay more. A sports stadium slice can deploy a 5G app to provide real-time updates to consumers. This 5G app could reveal player positions or additional features, such as AR/VR. These features could be given to customers who provide their phone numbers to the sports stadium and entirely run through a 5G slice. 5G slices can manage all the data for these applications using the UDR.

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 in the core network. Deploying NFs in the slice enables fine-grained control, independence, and isolation from other slices and the core network. Slices typically deploy a UPF as deploying a UPF in specific geographic locations helps to achieve low latency, enforce encryption standards, and avoid user data leaving the slice [29].

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. An AMF may also be deployed if UE control signaling requires the same isolation properties. However, 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 [29].

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). Due to the low-latency requirements, NFs may be more prominently distributed than other slice types. 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 additional mobility. Therefore, it may include a dedicated AMF. This slice will be ideal for low latency, high availability, and mission-critical applications [12].

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. However, the use of OAuth is defined as optional. OAuth is necessary for robust access control in the core network, especially when multiple tenants are involved. Assuming slices agree to share data via the Security Edge Protection Proxy (SEPP), the access control policy determines what NFs among separate slices share which data. Additionally, slices may request data from core NFs about other slices, breaking slice isolation without a least-privileged access control policy.

Based on the 3GPP standards [6], OAuth enforcement should operate as follows. When a consumer NF wants to call an API exposed by a producer NF, it will first ask the NRF for the access token needed for the API. The NRF will validate that the requesting NF is who it says it is by verifying its `clientID`. Each `clientID` is a Universally Unique Identifier (UUID) generated by the NF during registration with the NRF. The NRF validates each NF by checking its internal repository of NFs for a matching `clientID`. The NRF decides if the consumer NF should get the token, returning it if appropriate. 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 correct scope to access the API. If the validation succeeds, the producer NF performs the API operation and returns the appropriate result to the consumer NF.

3GPP TR 33.855 Key Issue #29: The 3GPP published a technical report [5] describing security issues within the 5G core SBA. It found that some data or operations in NF's exposed services are more sensitive than others. For example, the UDR provides policy, exposure, and application data that many NFs access. However, the UDR also includes subscription data, which contains highly-sensitive data, including cryptographic keys and primitives used for UE authentication. Thus, only the UDM should access the subscription data. The subscription data contains many other highly sensitive fields related to 5G-AKA, Steering of Roaming (SoR), and UE Parameter Update cryptographic information. Furthermore, all cryptographic operations become meaningless with write access to the subscription data.

Figure 1 shows two slices interacting with the same UDR in the core network. An SMF 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, including cryptographic keys, are exposed to other slices. The UDM Subscription Data Management service's purpose is to manage this sensitive data and only to provide it to NFs that are allowed to retrieve it. There are two types of connections between NFs in Figure 1. Connections 1 and 3 are allowed by the OAuth policy; however, 2 and 4 should not happen but are permitted by the policy. The SMF could be given subscription information used by network slice B only, thus breaking the isolation between slices.

3GPP Release 17 fixes this specific flaw by introducing additional access tokens on the different data types in the UDR. However, the interaction between NFs in the 5G core is complex. As demonstrated by 3GPP TR 33.855 Key Issue #29, over-privilege is a real problem. This paper seeks to methodologically study over-privilege for the 5G core to recommend a refined OAuth policy to the 3GPP.

Threat model and Assumptions: We assume that each NF

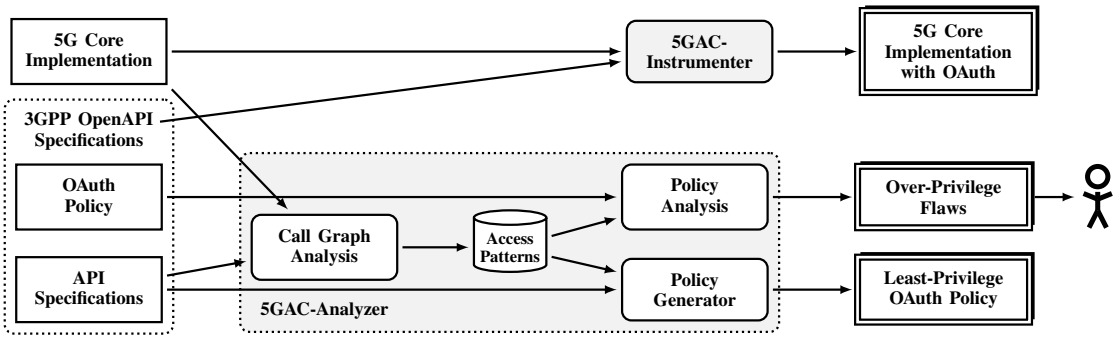

Fig. 2: Overview of our approach consisting of 5GAC-Analyzer for policy analysis and generation and 5GAC-Instrumenter for instrumenting the 5G core reference implementation to enforce OAuth.

requires different levels of access to different data types in the 5G core. For example, the UDM needs to access all data from the UDR, but other NFs should not access subscription data. 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. 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. This distributed architecture of the 5G core could be typical with multiple network slices, each managing its NF and its own UE's. When an NF is compromised, we assume it can create arbitrary requests to any other NF with exposed API interfaces over HTTP. Additionally, we assume that the HTTP interface is encrypted with SSL/TLS, which performs mutual authentication between all NFs.

We assume that the source code for free5GC or other 5G cores is easily obtainable by an adversary, and they can run their instance of a 5G core or slice. The maintainers of 5G core software implementations do not intend for the source code to be used for malicious purposes. We also assume that the optional OAuth 2.0 is enforced in the core as specified in the 5G standards. Free5GC does not enforce optional OAuth 2.0 as specified in the standards, but we instrument the code to add the required checks. Additionally, we consider the cloud provider, physical hardware, UE, and RAN as part of the trusted computing base.

III. Overview

3GPP TR 33.855 Key Issue #29 demonstrated that the 3GPP OAuth policy contains exploitable over-privilege. The goals of this paper are to (a) identify other sources of over-privilege in the 3GPP OAuth policy, and (b) help 5G core implementations achieve least privilege between NFs. Our key insight is identifying functional needs using program analysis on a reference 5G core. However, doing so requires overcoming the following research challenges:

- *The static program analysis must be domain-informed.* The analysis must map the semantics of 5G NF interfaces to the code locations for both making and handling requests.
- *Access token granularity must balance maintainability and security.* Adding more granular access tokens increases maintenance overhead for the standards and the core implementation. The policy needs to balance this concern with security to find an optimal fit.
- *No existing open-source 5G core implements OAuth access control.* Evaluating our proposed policy requires instrumenting a 5G core implementation to enforce OAuth.

We address the first challenge using the 3GPP specifications to identify domain-specific APIs and their corresponding code locations. 3GPP provides OpenAPI specification YAML files that are used to auto-generate code that establishes and handles RESTful HTTP connections between the NFs. In many cases, names within the OpenAPI files correspond to names of functions in the code. However, names are not always defined, and names are not always unique. We overcome these complications by modeling default name construction patterns in the OpenAPI-Generator and extracting package context from source code files.

The static analysis produces a set of *access patterns* that define which NFs need to call which entry points in which services. We use the access patterns in two ways. First, we compare the access patterns to the 3GPP policy and identify two previously unknown over-privilege issues. As described in Section VI, these flaws expose some subscriber information and enable UE DoS. Second, we use the access patterns to propose a new least-privilege policy. In doing so, we address the second challenge by designing an algorithm that optimizes access token permissions at the granularity of services within NFs. If more granularity is needed, we cluster entry points within those services to minimize the number of permissions. Since the 3GPP policy currently defines permissions at the granularity of services, the 3GPP policy only requires minimal changes to adopt our suggestions.

Finally, we overcome the third challenge by automatically instrumenting the free5GC codebase with OAuth token requests and validation in all NFs. While it is relatively straightforward to add logic to the NRF to respond to OAuth token requests, many changes are required throughout the code to request, send, and validate the tokens. Access token requests are inserted after all relevant error checking and included transparently in HTTP headers for each proxy. Each entry point

in each service in each NF must validate access tokens upon receiving requests.

Figure 2 provides a high-level overview of our approach, which consists of two tools. The 5GAC-Analyzer tool consumes the 5G core reference implementation and the 3GPP OpenAPI specification files. It uses call graph analysis to identify which NFs access which service entry points. 5GAC-Analyzer then uses the access patterns to both (a) provide a set of over-privileged policy flaws for a human analyst, and (b) derive a least-privilege OAuth policy. 5GAC-Instrumenter takes a 5G core reference implementation and the 3GPP OpenAPI specification files and produces a modified implementation that enforces the OAuth policy. 5GAC-Instrumenter is policy-agnostic and can consume both the 3GPP OAuth policy and the least-privilege policy created by 5GAC-Analyzer.

Our current implementation uses free5GC [1] as the 5G core reference implementation. The free5GC implementation is primarily written in Go; hence, our analysis is built on top of the `ast` package in the Go standard library. However, the analysis algorithm is lightweight and general enough to be ported to other languages. One advantage of 5GAC-Instrumenter is that it retains all whitespace and comments when it instruments code. Hence, we can automatically produce patches for free5GC, which we plan to contribute as pull requests to the free5GC Github project.

IV. 5GAC-Analyzer

5GAC-Analyzer performs three tasks: (1) identify access patterns between NFs, (2) identify over-privilege in the 3GPP policy, and (3) derive an OAuth policy that captures least privilege between NFs and entry points to services. This section describes how 5GAC-Analyzer extracts access patterns using static analysis and then uses them to accomplish the latter two tasks. Our access control model considers NFs as subjects and individual entry points as objects.

Let $\mathcal{N}$ be the set of all NFs, $\mathcal{S}$ be the set of all services, and $\mathcal{E}$ be the set of all entry points. 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 entry points $E \subseteq \mathcal{E}$. Furthermore, 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 $e \in \mathcal{E}$ be an NF and an entry point respectively. An access pattern $a \in \mathcal{A}$ is a pair (n, e) empirically derived through program analysis.

Definition 2 (Access Rule)**.** The set of all possible access rules $\mathcal{R}$ is a set of triples. Let $n \in \mathcal{N}$, $e \in \mathcal{E}$, and $p \in \mathcal{P}$ be an NF, an entry point, and a permission text string. An access rule $r \in \mathcal{R}$ is a triple (n, e, p). A policy Γ is a set of access rules.

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

The remainder of this section describes our approach for extracting a set of access patterns A from a 5G core implementation and testing if the 3GPP provided policy Γ_{3GPP} is least privilege. As we discuss in Section VI, 5GAC-Analyzer finds Γ_{3GPP} is not least privilege, leading to several vulnerabilities. Therefore, 5GAC-Analyzer generates a least privilege policy that provides a trade-off between policy usability and security. We perform our analysis on free5GC, which is written in Go. However, the approach itself is more generic and can be applied to other 5G core implementations such as Open5GS, which is written in C.

A. Identifying Service Entry Points

To derive the access patterns, 5GAC-Analyzer must determine the location of NFs, services, and service entry points within the code. Fortunately, the code location of NFs and services are relatively straightforward to identify. For example, free5GC and Open5GS use source code directory structure to organize code by NF and service. Entry points are function callbacks within those files.

Service entry points implement the RESTful APIs defined by the 3GPP specifications. 5GAC-Analyzer uses OpenAPI specifications provided by 3GPP to bridge the semantic gap between the APIs and the function callbacks. The specifications provide both the API definition *and* callback function name. Both free5GC and Open5GS use the OpenAPI-Generator [26] tool to create code from the 3GPP specifications. OpenAPI-Generator uses the `operationId` field in the OpenAPI specification to name the function in the source code.

Unfortunately, the 3GPP OpenAPI specification has two limitations: (1) the `operationId` is not defined for every API, and (2) entry points will use the same function names when the `operationId` is the same. Figure 3 shows two separate proxy functions. OpenAPI-Generator creates these functions with the same function name because they have the same `operationId` in the 3GPP specifications.

5GAC-Analyzer generates a call graph from the AST by recursively locating all nodes that call functions starting from the program start. Every NF starts a set of services that define entry points. 5GAC-Analyzer traverses the call graph of each NF to identify each service's startup routine. Next, 5GAC-Analyzer extracts the routing table, which defines each entry point during the service startup routine. However, these entry points need to be mapped to the 3GPP specifications.

5GAC-Analyzer assigns the function to the corresponding API when its name matches the `operationId` from the specifications. To avoid name collisions, 5GAC-Analyzer analyzes each service independently. When the 3GPP specification does not define an `operationId`, 5GAC-Analyzer falls back to the auto-generated name created by OpenAPI-Generator. The fallback name consists of the HTTP request type and API path to the API. This strategy successfully identifies all callbacks in the free5GC code-base. 5GAC-Analyzer identified 64 entry points in free5GC version 3.0.7. Manual verification of these entry points confirmed the mapping was correct in all cases.

B. Extracting access patterns

A key part of 5GAC-Analyzer is extracting *access patterns*: i.e., which NFs call which entry points. A proxy function is a function in the code that has the sole purpose of performing an HTTP request to an entry point in another NF. A naïve solution is to assume that all of the proxy functions defined in the code-base for a NF are used by that NF. However, we found that

```go
func Get(...) (...) {
    var localVarHTTPMethod   = strings.ToUpper("Get")
    ...
    localVarPath := a.client.cfg.BasePath() +
     "/{ueId}/registrations/amf-3gpp-access"
    ...

    r, err := openapi.PrepareRequest(...)
    ...
    localVarHttpResponse, err := openapi.CallAPI(r, ...)

```

```go
func Get(...) (...) {
    var localVarHTTPMethod   = strings.ToUpper("Get")
    ...
    localVarPath := a.client.cfg.BasePath() + "/{supi}"
    ...

    r, err := openapi.PrepareRequest(...)
    ...
    localVarHttpResponse, err := openapi.CallAPI(r, ...)
}

```

Fig. 3: Function name collision to call APIs in two different services. Both services expose different REST APIs and the function to call them has the same name. The `operationId` for both APIs is "`Get`" in the 3GPP standards.

Algorithm 1 Algorithm to find over-permissive access rules based on a policy Γ and set of access patterns A

procedure FINDOVERPRIVILEGE(Γ, A)
  $O := \emptyset$
  for $(n, e, p) \in \Gamma$ **do**
    if $(n, e) \notin A$ **then**
      O.append(n, e, p)
    end if
  end for
  return O
end procedure

OpenAPI-generator includes definitions of *all* proxy functions, regardless of whether or not they are used. Therefore, a call graph is required to determine the access patterns.

5GAC-Analyzer constructs a call graph from the main program entry point of each NF's program. Similar to the entry point function discovery, the call graph construction resolves links to the entry point call back methods by resolving links in the routing table. 5GAC-Analyzer then traverses the call graph to identify all function calls that match the possible `operationId` values in the specifications. Again, 5GAC-Analyzer must account for the two limitations of the OpenAPI specification: missing `operationId` values and name collisions. Missing `operationId` values are handled similar to entry point discovery. However, name collisions require a different approach.

When identifying the proxy functions, 5GAC-Analyzer could assume function names were unique for the service under analysis. However, the access pattern extraction must resolve which service the function name belongs to. To perform this resolution, 5GAC-Analyzer analyzes all `import` statements for the file. Each proxy function invocation references a package from the `import` statements. We extract this package reference from each proxy call to resolve which service the proxy function belongs to.

C. Least-privileged 5G core access control

5GAC-Analyzer uses the access patterns to achieve its primary goals: (1) check an existing access control policy to determine if it is least privileged, and (2) derive a least-privileged access control policy while optimizing for both security and maintainability.

Algorithm 2 Algorithm to create a least privilege policy for a set of services S based on a set of access patterns A

procedure CALCULATEPOLICY(S, A)
  $\Gamma := \emptyset$
  for $s \in S$ **do**
    $G := \emptyset$ ▷ Set of $2^E \times 2^N$ pairs
    for $e \in s$ **do**
      $C := \{n | (n, e) \in A\}$ ▷ All n that call e
      found := **false**
      for $(E, N) \in G$ **do**
        if $N = C$ **then**
          E.append(e) ▷ Also updates E in G
          found := **true**
          continue
        end if
      end for
      if found = **false then**
        G.append$(\{e\}, C)$
      end if
    end for
    for $(E, N) \in G$ **do**
      p := CREATEPERMISSIONNAME(s, E)
      for $e \in E$ **do**
        for $n \in N$ **do**
          Γ.append(n, e, p)
        end for
      end for
    end for
  end for
  return Γ
end procedure

Finding Over-Permissive Policy: Algorithm 1 determines the over privilege in an access control policy Γ with respect to an empirically derived access pattern A. It iterates through Γ checking each policy tuple for a matching pattern pair in A. If a rule allowing a NF to call an entry point exists in the access control policy and not the access patterns, then permission p is too permissive. The corresponding access rule is added to O, which is returned by the algorithm. We discuss policy over-privilege discoveries in Section VI.

Generating a Least Privilege Policy: 5GAC-Analyzer derives 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 entry points. Conceptually, 5GAC-Analyzer seeks to achieve

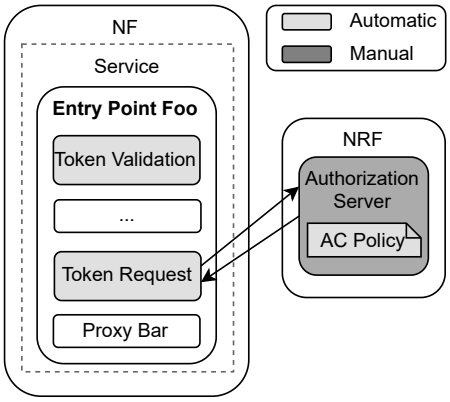

Fig. 4: 5GAC-Instrumenter changes the system behavior of the 5G core by automatically adding token requests and validation to each NF. The authorization server is a one-time manual effort and consumes an access control policy.

least privilege without unusable permission bloat (e.g., one permission per entry point). The existing 3GPP policy defines a permission to access each service. Therefore, 5GAC-Analyzer uses service-level permissions whenever possible. Doing so minimizes the differences between the 3GPP and the least privileged policies.

Let E_s be the set of entry points for service s. If all $e \in E_s$ have the same set of accessing NFs in A, then 5GAC-Analyzer defines the permission at the granularity of the service. Otherwise, we cluster each e by NF callers and create new permissions based on the set of callers for each s. This strategy is intuitive. The 3GPP policy already uses service names for permission strings. For finer-grained names, 5GAC-Analyzer creates names based on the prefixes of the entry point names.

Algorithm 2 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 5-18, we create groups of entry points G, which is a tuple of a set of entry points and a NF (E, N). In lines 19-26, we create a permission for each group of entry points which ensures a new permission for each access pattern. The result of the algorithm is an access control policy that is consumed by 5GAC-Instrumenter to instrument the 5G core.

V. 5GAC-Instrumenter

The 3GPP specifies that when authorization between NFs is needed, the 5G core should use OAuth 2.0 with Client Credentials, which is a machine-to-machine authorization type. Enforcement of OAuth with Client Credentials requires instrumentation in three ways. (1) In every proxy, the NF must request an OAuth access token from the NRF. (2) The NRF needs to determine the identity of the NF and whether it is allowed the access token based on the access control policy. If the access control policy allows this interaction, then the NRF signs the access token and returns it. (3) Every entry point must validate the access token and check its permissions.

5GAC-Instrumenter instruments NFs in free5GC to request, send, and validate OAuth tokens. It inserts access token requests to all proxies in free5GC and token validation at all entry points. When doing so, 5GAC-Instrumenter can use either access control policy from 5GAC-Analyzer or the 3GPP policy. We also manually added an authorization server into the NRF, including the logic to handle OAuth token requests.

A. Access Control Architecture

Figure 4 details our authorization architecture for enforcing OAuth in the 5G core. We now discuss the three key components of the architecture.

Token Request: Each time an NF wants to call an entry point in another NF, it requests a short-lived access token for that entry point from the authorization server. That token is then included in the subsequent call. Alternatively, the authorization could assign a validity period and allow the NF to cache and reuse the access token. While caching access tokens reduces the number of network connections required to perform an operation, it increases the attack surface, as tokens may leak during their validity period. We show in Section VII that requesting a new token for each call does not significantly impact UE network performance.

Authorization Server: The 3GPP specifications define the NRF as the authorization server. The authorization server determines which access tokens are granted to which NFs. When a consumer NF initiates an access token request, it includes a client ID, which is a UUID that the NF gave to the NRF instance when it initially registered to the NRF. The authorization server uses the NF instance ID and maps this to an internal NF repository, validating the NF in the process. It then checks if the access control policy allows the NF to access the entry point. The authorization server creates an access token, embedding its identity, the consumer NF's identity, the producer NF's identity, access token scope, and expiry time. The 3GPP standards describe either a pre-shared secret key between the NRF and all NFs or TLS certificates to sign and verify access tokens. However, they do not standardize any key agreement protocol for signing and verifying the access tokens, instead leaving it up to the implementations. After signing the access token, the authorization server returns the generated access token to the consumer NF. We manually implement the authorization server in the free5GC NRF, which is a one-time effort.

Cross-slice access control policies are supported between NFs of separate slices. Each slice's NRF manages the access token's requests. 5GAC-Instrumenter determines the NRF location dynamically based on each NF's configuration. The SEPP acts as a firewall between slices and manages authentication of NFs. However, authorization to specific data types is outside the scope of the SEPP and is managed by OAuth even between slices.

Token Validation: Each entry point performs token validation to verifies the caller has sufficient permissions to perform the intended operation. When the entry point receives a request, it validates the access token by checking its signature is valid, that it is not expired, and that it trusts the signature. It also checks that it trusts the consumer and producer NF identities.

```
func AMFStatusChangeSubscribe(...) (...) {
    var localVarHttpMethod   = strings.ToUpper("Post")

    localVarPath := a.client.cfg.BasePath() + "/
     subscriptions"

    ....
+   scopes := []string{"namf-comm"}
+   tokenUrl := fmt.Sprintf("%v/oauth2/token"},
+                               authorization_server)
+   conf := &clientcredentials.Config{Scopes: scopes,
+                               TokenURL: tokenUrl, ...}
+   token := conf.Token(...)

+   r  := openapi.PrepareRequest(localVarPath, token, ...)
-   r  := openapi.PrepareRequest(localVarPath, ...)
    ...
    localVarHttpResponse, err := openapi.CallAPI(r, ...)

```

Fig. 5: Example client-side proxy function that calls the 5G core API listed in Figure 6. The code additions instrumented by 5GAC-Instrumenter are shown in green.

```
var routes = Routes{
...
{
    "AMFStatusChangeSubscribe",
    strings.ToUpper("Post"),
    "/subscriptions",
    HTTPAMFStatusChangeSubscribe,
},
...


func HTTPAMFStatusChangeSubscribe}(...) {
+    scopes := []string{"namf-comm"}
+    _, oauth_err := openapi.CheckOAuth(..., scopes)
+    if oauth_err != nil {
+        c.JSON(http.StatusUnauthorized, ...)
+        return
+    }

    var subscriptionData models.SubscriptionData

    requestBody, err := c.GetRawData()

    ...
}

```

Fig. 6: Entry point that handles the 5G core API called from Figure 5.

Finally, the entry point checks the scope of the access token is the scope of the entry point. After all validation, the entry point performs its operation and returns results to the requesting NF.

B. Instrumenting Token Requests

The token request instrumentation must address the following criteria: (1) when the authorization server changes location, access token requests should query the correct authorization server; (2) access token requests must be inserted in the proxy's function before the HTTP request; and (3) access token requests should be inserted after most or all error checking to eliminate unnecessary requests to the authorization server. We note that unrelated errors, such as missing required parameters by the proxy caller, will be rejected by the entry point regardless of permission. In this case, the 5G core must not unnecessarily query the authorization server when a proxy call fails. 5GAC-Instrumenter minimizes the additional network overhead by only contacting the authorization server when all other error checking is complete.

Figure 5 shows an example proxy definition that includes our token request instrumentation on lines 7–14. The `scope` defines the permission that the NF will request from the authorization server, the `TokenURL` defines the location of the authorization server, and `conf.Token` is the function call that initiates the token request. Finally, the proxy includes the token in the HTTP header to call the API.

5GAC-Instrumenter inserts token requests immediately before the proxy's HTTP request. Proxies are identified to instrument token requests in the 5G core as described in Section IV. Proxies must insert each token directly into the HTTP request sent to the entry point. 5GAC-Instrumenter modifies the content of the HTTP request to include the access token in all proxy functions transparently.

Finally, the location of the authorization server may change and scale to changes in network load dynamically. Each NF maintains a configuration structure for the IP address of the authorization server. 5GAC-Instrumenter examines each NF's AST to extract the configuration structure. Once identified, 5GAC-Instrumenter inserts the NRF IP address variable and the NF instance ID variable into all token requests. We reference the NRF IP address as the base location to request the access token for an HTTP request, adding the constant "/oauth2/token".

C. Instrumenting Token Validation

5GAC-Instrumenter instruments token validation at each entry point following two rules: (1) token validation and requests must agree on the access control policy and (2) to minimize additional code, we instrument most access control checks in a shared file between all entry points. Therefore, access token validation instrumentation only varies in a single function call that specifies the intended permission to minimize code overhead and maintenance.

Figure 6 shows an entry point definition with token validation instrumented. The route definition identifies `HTTPAMFStatusChangeSubscribe` as the entry point for the *POST /subscriptions* API. Line 13 specifies the valid `scopes` for the entry point based on the access control policy during instrumentation. On line 14, the access token is validated according to the expected permission. An HTTP Unauthorized response is returned if this access token is invalid or does not contain the expected permission. If the entry point successfully validates the access token, it continues normally.

5GAC-Instrumenter must decide the access control policy and instrument token validation, the authorization server, and token requests with the same policy. If there is a mismatch in the access control policy, token validation may reject the NF when it should not. 5GAC-Instrumenter instruments token requests and validation simultaneously, ensuring the access control policy is consistent.

Token validation is designed to be as generic as possible to ensure it is compatible with all entry points. Every entry point must validate the access token's signature, and the entry point must accept one or more of the token's scope. The signature

validation can be shared for every entry point, however, the scope validation must change for each entry point since it may accept different scopes. Additionally, entry points may enforce multiple required scopes, each requiring validation. Token signature validation occurs by verifying an HMAC with a shared key from the NRF and all NFs. For our implementation, the secret key is a hardcoded key that we pre-define. We discuss the key agreement protocol in Section VIII.

5GAC-Instrumenter supports enforcing multiple scopes within the access control policy for an entry point. Each scope needs to be validated independently of other scopes in this case. The scopes are an array of strings, with each element in the array is validated for the entry point. When all scopes in the access token are validated, the entry point continues normal execution.

D. Enhancing Future Projects

OpenAPI-Generator supports generating code for an OpenAPI specification but lacks OAuth generation for many languages. We have created a set of changes to OpenAPI-Generator that generates OAuth token requests and validation for all future projects in Go. However, improving OAuth generation in OpenAPI-Generator will not fix free5GC's OAuth enforcement as regenerating the code creates many merge conflicts. We intend to help all future use of OpenAPI-Generator by creating a pull request to add in OAuth generation.

5GAC-Instrumenter can re-instrument the same code base multiple times, enforcing a new access control policy. When the 3GPP standards change and new functionality is added into free5GC, 5GAC-Instrumenter can enforce a new access control policy. Even when free5GC is already instrumented with additional OAuth, 5GAC-Instrumenter detects this and only adjusts the code locations to change the policy.

VI. 3GPP Policy Flaws

This section describes the policy flaws we discovered in the 3GPP access control policy and how our new policy addresses them. We describe the possible attacks due to the access control policy being over-privileged. The flaws were discovered in 3GPP Release 15 and have not been fixed in 3GPP Release 17 unless explicitly stated. We have disclosed our findings to the GSMA and have not received a response confirming our findings yet.

A. Experimental Setup

We ran our experiments on a VM with an Intel Xeon E5-2620 @ 2.4 GHz and ten vCPUs and 60 GBs RAM running Ubuntu 20.04 on Linux kernel 5.4. We ran 5GAC-Analyzer on 3GPP Release v15.2.0, which contains 78 files defining the access control policy. Notably, this version is used by free5GC to generate code, which is why we chose it. We obtained the policy files from 3gpp.org, which is the 3GPP working area. 5GAC-Analyzer only uses policy files defining access control policy and additional structures for NFs present in free5GC. Free5GC does not implement every NF standardized in 3GPP Release 15, which we discuss further in Section VIII. 5GAC-Analyzer filters out all unused policy files in free5GC, leaving a total of 38 policy files. 5GAC-Analyzer and 5GAC-Instrumenter completed execution in 1.56 seconds and 1.78 seconds, respectively.

```
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   /ue-contexts/{ueContextId}/assign-ebi: ...
22   ...
23
```

Fig. 7: Example OAuth policy in the AMF Communication service [6]. Line 10 indicates that an OAuth token is optional for all endpoints in the service [3].

B. Newly Discovered Policy Flaws

When running 5GAC-Analyzer on free5GC, we discovered two additional flaws in the access control policy. In total, 5GAC-Analyzer found 3 out of 13 access tokens in free5GC to be over-privileged. One of these findings 5GAC-Analyzer independently re-discovers the example described in 3GPP TR 33.855 Key Issue #29, which was discussed Section II. Note that free5GC does not implement every NF in 3GPP Release 15, which limits our analysis. We expand on this limitation in Section VIII.

Finding 1 (Negated OAuth Policy): *The 3GPP policy includes null access rules for each entry point.* 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.

Figure 7 shows an example service's YAML security definition. In an OpenAPI security definition, the empty braces on line 10 means 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.

We believe this flaw results from a misunderstanding of the empty braces feature in OpenAPI. The OpenAPI specifications [3] 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 need to manually remove all of the empty braces, which

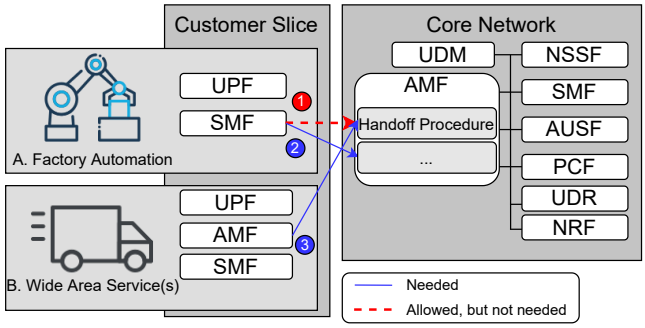

Fig. 8: Only the AMF should be allowed to perform an AMF handover. However, the PCF and SMF are also allowed under the 3GPP access control policy. This can cause an insecure AMF handover which may switch UE's AMF to another slice. The only information needed to perform this operation is a UE's SUPI.

is an error prone process.

Finding 2 (AMF Re-Allocation): *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. An AMF handover is similar but not identical to when a handover occurs in the RAN. The AMF serving the UE is the only entity that should initiate the handover to another AMF.

With current policy, an SMF or Policy Control Function (PCF) may also initiate this handover. This over privilege can be exploited by malicious slices. Figure 8 shows how two separate slices can exploit the permissive access control policy. Previous work [21] has studied handover flaws that occur in the RAN, but they did not 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 can retrieve 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 9 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. Initially, 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.

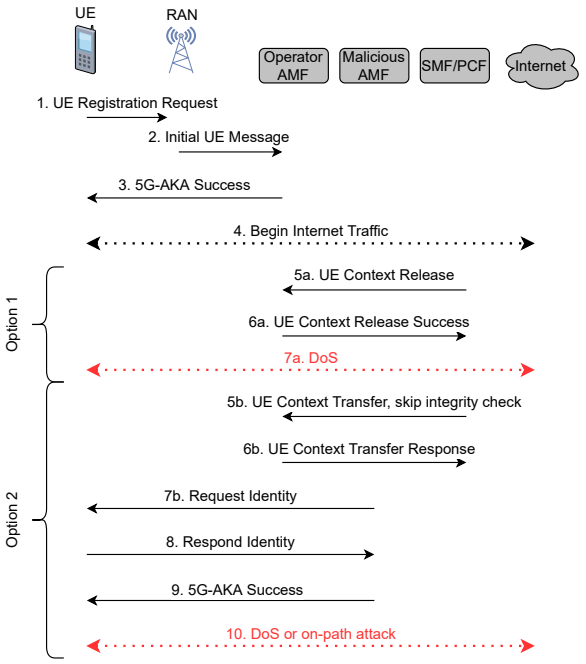

Fig. 9: The AMF Re-Allocation attack allows the SMF or PCF to initiate an AMF switch for a UE. The attacker may switch the UE to a new slice that the attacker controls, causing DoS or on-path attacks.

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 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 in the meantime.

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 [7]. When the request succeeds in step 6b, the malicious AMF needs to 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. If any failure occurs after step 6b is successful, DoS for the UE occurs.

Finding 3 (Subscription Data Management Exposure): *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's only purpose is to read the UE subscription data from the UDR and provide portions of it to consumer NFs. the nudm-sdm service does not achieve its designed purpose without a least-privileged policy. The access control policy for this service allows any of the consumer NFs to read any of 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. 5GAC-Analyzer determined that the SMF only needs to access data exposed by the *GET /{supi}/sm-data* endpoint.

To validate that this attack works, we set up 5GAC-Instrumenter to instrument the 3GPP policy. Then, we request the nudm-sdm access token from the SMF and query the *POST /{supi}/am-data* API, which should not be allowed. We confirmed that we obtained valid data.

C. Updated Access Tokens

We propose six new access tokens to address the overprivileged policy. Table I 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 2. Three new access tokens separate the exposed APIs into NF-specific groups, negating the scenarios discussed in Section VI-B. These tokens are shown in rows 2-4 of Table I.

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 I. Finally, the changes in the `nudr-dr` service were discovered independently by 5GAC-Analyzer, so we include them in our findings. However, changes to the access control policy for `nudr-dr` were implemented in 3GPP Release 17 [4].

VII. Performance Evaluation

This section evaluates the performance of free5GC before and after 5GAC-Instrumenter modifies the code base. Specifically, we answer the following questions:

- **Q1** How much additional latency is there in the core network by adding additional access control checks?
- **Q2** How much overhead is introduced by changing the access control policy?
- **Q3** How does OAuth in the core affect individual UE's?

A. Experimental Setup

We ran our experiments on a VM with an Intel Xeon E5-2620 @ 2.4 GHz and ten vCPUs and 60 GBs RAM running Ubuntu 20.04 on Linux kernel 5.4. We evaluated free5GC under three different configurations.

- **free5GC**: The default, unmodified free5GC 3.0.7 available on GitHub. This configuration does not enforce any OAuth access control between the core functions.
- **free5GC with OAuth**: The **free5GC** configuration with code added for OAuth. This configuration uses the default 3GPP Release 15 access control policy.
- **free5GC with Improved OAuth**: The free5GC with OAuth configuration but enforcing our least-privilege policy (Table I).

We used free5GC v3.0.7 run on Docker. We installed the gtp5g v0.5.4 Linux kernel module required by the free5GC UPF, which enables the network stack for a UE. For gNodeB and UE simulation, we used UERANSIM 3.2.6. We used iPerf 3.7 to measure throughput for the UE between the docker container for UERANSIM and the host machine over the network interface created by the UPF [2].

B. Overhead in the Core

To evaluate the performance impact on the core network, we simulated registering 210 UEs. Registration involves the 5G-AKA process, PDU Session Establishment, UPF tunneling, PCF policy enforcement, and network slice selection. The 210 UEs are run in batches of 35 UEs using UERANSIM. When using batches larger than 35 UEs, free5GC's UPF would run out of memory. Therefore, we selected this batch size due to memory allocation issues within the UPF of free5GC.

We recorded the start time as the time that the RAN first receives a UE Registration_Request, after which the RAN sends this information to the relevant AMF. A UE registration is one of the most common operations in the core and triggers a series of inter-NF communication among the NFs. The end time is when the RAN establishes the data-plane connection to the UPF for the UE. Between these times, the core is responsible for authenticating a UE and establishing the necessary structures in the core to support it.

Our results report the arithmetic mean over all 210 UE registrations. Each UE registration executes around 40 API calls in the NRF, NSSF, SMF, AUSF, AMF, PCF, UDM and UDR. These API calls nearly double when OAuth is enforced, as OAuth requires an access token request for each API call (except for NF registrations to the NRF). The core initialized all NFs for each configuration before the experiments began. The whiskers on the figure indicate the 95% confidence interval.

Adding OAuth: Figure 10 shows the overhead in the core for all three configurations of free5GC. The free5GC configuration takes 3.21 seconds to establish a UE's connection while adding OAuth nearly doubles this setup time to 6.20 seconds (93.27% increase). While there is room for improvement, additional access control causes the number of API requests in the core to nearly double as each API call requests an access token. We believe improvements in 5GAC-Instrumenter will increase performance. For example, each access token is given an expiry time in which it is valid, which we do not consider. Instead

TABLE I: Proposed changes to the access control policy in 3GPP Release 15.

Service	Consumer NF	Proposed Access Token	APIs	Finding
namf-comm	SMF	namf-comm:n1n2-messages	POST /ue-contexts/{ueContextId}/n1-n2-messages	Finding 2
namf-comm	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 2
namf-comm	PCF	namf-comm:subscriptions	POST /subscriptions	Finding 2
nudm-sdm	SMF	nudm-sdm:sm-data	GET /{supi}/sm-data	Finding 3
nudm-sdm	UDM	nudm-sdm:shared-data	GET /shared-data POST /shared-data-subscriptions DELETE /shared-data-subscriptions/{subscriptionId}	Finding 3
nudm-sdm	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 3
nudr-dr	UDM	nudr-dr:subscription-data	GET, POST, DELETE, PATCH /subscription-data/*	Fixed in 3GPP R17
nudr-dr	PCF	nudr-dr:policy-data	GET, POST, DELETE, PATCH /policy-data/*	Fixed in 3GPP R17

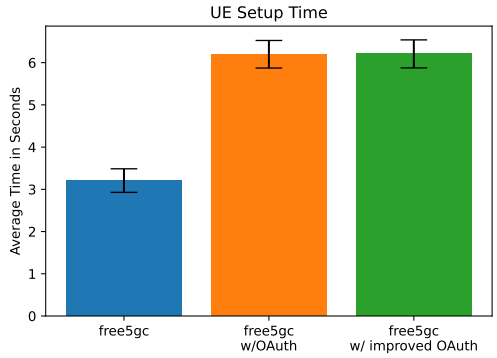

Fig. 10: Average time for a single UE to connect and set up the core for operation.

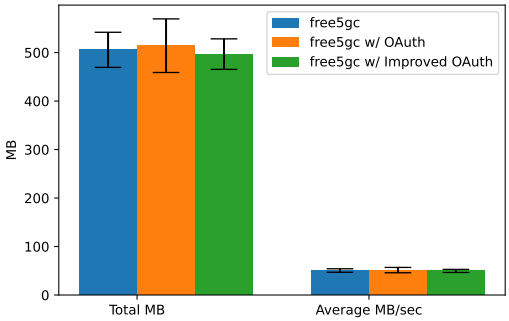

Fig. 11: UE Throughput.

of caching the tokens for the expiry period, we request a new access token when it is necessary.

Policy Overhead: Figure 10 shows the performance overhead of changing the policy in the core. We see no significant difference between the enforcement of the 3GPP policy and one generated by the Policy Generator. Enforcing the least-privilege policy causes an additional overhead of 8 ms (0.14%) compared to the 3GPP policy. This difference is within the expected experimental error.

C. UE Overhead

The 5G core network operates on the control plane, and each UE connects to the Internet on the data plane. The UE establishes an Internet connection through a direct link to the RAN and UPF. Generally, overhead in the core network should not affect a UE's connection throughput due to the separation in the control and data plane. To measure UE performance impact, we used the iPerf client in the docker container and the iPerf server on the host machine in 10-second batches for 100 seconds. We present the results as the average total MB over 10 seconds and the average MB/sec for each configuration. The whiskers on the figure indicate the 95% confidence interval.

Figure 11 shows the throughput of a UE's connection to the Internet as measured by iPerf. In the free5GC configuration, the average total MB transferred over 10 seconds is 505.70 MB, averaging 50.51 MB/sec. Introducing OAuth into free5GC shows negligible difference, with the average MB transferred over 10 seconds to 514.20 MB (1.6% increase). We believe the difference is due to network conditions outside our control and generally unaffected by additional access control checks in the core.

VIII. Discussion

Limitations: Free5GC has not yet implemented every NF and API defined in 3GPP Release 15, but it is a fully working 5G core with some unused or not implemented APIs. Therefore, our code analysis is limited to what code is implemented and used in free5GC; however, since our approach is automated, this process can be re-run when new functionality becomes available. We identify NF interactions by examining HTTP calls, therefore, NFs that do not implement a SBA such as the UPF or Non-3GPP Interworking Function (N3IWF) are not included. We have not applied our improved access control policy to another 5G core, such as Open5GS.

We know two other open-source 5G cores: Open5GS (C++) and the OpenAirInterface 5G core (C++). Since 5GAC-Analyzer and 5GAC-Instrumenter were built around Golang, they cannot be directly applied to these implementations. However, based on a manual investigation of the 3GPP specifications, the described security weaknesses apply to the spec-

ifications themselves. For example, since OAuth is optional between the NFs, 5G cores adhering to the standard do not have to implement OAuth at all. We believe that any code implementation strictly following the specs will implement this weakness; however, we leave it to future work to further evaluate this.

Key agreement: The 3GPP standards specify a key agreement protocol such that each NF can validate an access token is from the authorization server. They leave the exact protocol for distributing signing keys to all NFs left to the implementations. We do not establish a specific key agreement protocol and only investigate the access control policy. Instead, we assume hard-coded keys for signing access tokens. We leave it to future work to implement TLS certificate signing and validation of access tokens.

IX. Related Work

Static Analysis for Access Control: Static analysis has been used in prior work to identify a security policy [22], [44], [47], [60] or confine system calls within an OS or container [31], [52]. Li et al. [47] 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 [6].

Pailoor et al. [52] automatically sandbox applications to a limited set of system calls determined by a static analysis of the code base. The system calls extracted from static analysis represent an access control policy for an application. Ghavamnia et al. [31] apply a similar approach, but target applications running inside of a container.

NLP is also common to extract the access control policy from standards documents [11], [64]. These techniques are often too coarse-grained and miss important information due to the limitations in NLP. Fortunately, in the 5G specifications, we are given machine-readable YAML descriptions of all APIs. Therefore, NLP-based approaches are not the most efficient for the 5G specifications.

Related to the 5G core, static analysis has been used to study access control checks within mobile operating systems. PScout [14] uses static call graph analysis to identify permission requirements on system service calls in Android. Kratos [57] identifies missing permission checks at entry points within Android's system services. ACMiner [34] is a framework for evaluating authorization check enforcement within the Android OS. Kratos and ACMiner detect authorization flaws by performing consistency analysis across a large code base containing millions of lines of code. ARF [32] builds on ACMiner and identifies permission re-delegation in Android by creating an interconnected graph of all entry points in Android's system services. FReD [33] builds on ACMiner, and ARF moves beyond permissions to identify file re-delegation from entry points within Android's system services.

OAuth: OAuth 2.0 is a widely used authorization framework that typically encodes access tokens as a JSON Web Token [36], [40]. However, OAuth is often incorrectly used as an authentication protocol [24]. Most security efforts towards OAuth have focused on errors in specific implementations [15], [24], [46], [58], [59], [61], [65]. A study of over 600 mobile applications found that around 60% of them incorrectly implement OAuth and are vulnerable [24]. OAuth is often misunderstood and thus incorrectly implemented, and a poor OAuth implementation can lead to additional vulnerabilities [59].

The OAuth standard is an English text-based protocol description [36] An analysis of the OAuth protocol revealed four previously unknown vulnerabilities [30]. They found these vulnerabilities to be exploitable in practice and considered all four grant types of OAuth, including client credentials. The research on OAuth shows that analysis of protocols and implementations is essential to discover security weaknesses.

Cellular Network Security: Access control in telecommunications has been a security concern for decades. "phreakers" can be highlighted as the first instances of failed access control in telecommunications [54]. These "phreakers" exploited hardware vulnerabilities to make free calls using targeted tones. In the early days of IP-based telecommunications, vulnerabilities were discovered within the SS7 protocol [48]. SS7, a fundamental signaling protocol, was not improved for future telecom generations, resulting in authentication vulnerabilities. Cellular devices were particularly affected.

Cellular generations have their own access control challenges. Works have fuzzed and analyzed various LTE protocols, discovering spoofing, authentication bypass, and identifier leakage vulnerabilities [42], [53]. 5G is not immune. Shortcomings in multiple access control specifications have been identified with the TAMARIN [50] prover [16], [28]. Discoveries include spoofing and authentication bypass race conditions. 5G introduces novel access control challenges from NF virtualization and slicing. Multiple works [9], [20], [56] highlight access control challenges from multi-tenancy and NF virtualization. However, unlike this work, all focus on specification analysis and take a broad, high-level viewpoint.

Works have more broadly studied cellular security. Symbolic model checkers for LTE and 5G standards have been developed to verify various protocols. LTEInspector and 5GReasoner are notable checkers [37], [38]. Numerous works have analyzed specific portions of 5G standards and have suggested various improvements [8], [27], [39]. Others have provided surveys and mitigations of theoretical 5G security threats based on the standards [9], [10], [41], [49], [55], [63]. All of these 5G works study specifications, not implementations.

Specifications have been studied specifically for access control. Primarily, works have proposed a new access control framework for specific 5G use cases, such as industrial IoT or networked health devices [18], [19], [23], [35], [45]. Some have looked at the improvements 5G has made to address access control issues in LTE [17]. Access control between the RAN and Core has been analyzed, and new mechanisms proposed [43], [62]. However, these works involve potential access control mechanisms and proposed use cases. None focus on access control as it currently exists in 5G Core implementations or specifications.

X. Conclusion

The 5G core makes significant advances over the 4G core by switching from specialized hardware to a SBA 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 a 5G core reference implementation to define and instrument a least privilege access control policy for requests between 5G core NFs. It also used the least privilege policy to identify two new over-privilege flaws in the OAuth policy defined in the 3GPP specifications. These flaws allow a malicious NF to interrupt UE network connectivity and extract subscriber information. By adopting our refined policy, 5G core implementations can reduce their attack surface by opening the network to these new threats.

References

[1] "free5GC." [Online]. Available: https://www.free5gc.org/

[2] "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool." [Online]. Available: https://iperf.fr/

[3] "OpenAPI Specification - Version 3.0.3 | Swagger." [Online]. Available: https://swagger.io/specification/#security-requirement-object

[4] 3GPP, "Release 17." [Online]. Available: https://www.3gpp.org/release-17

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

[6] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; 5G System; Access and Mobility Management Services;," {3rd Generation Partnership Project (3GPP)}, Technical Standard (TS) 29.518, Jun. 2022. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3339

[7] ——, "3rd Generation Partnership Project; Technical Specification group Services and System Aspects; Security architecture and procedures for 5G system," {3rd Generation Partnership Project (3GPP)}, Technical Standard (TS) 33.501, Jun. 2022. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169

[8] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov, "5g security: Analysis of threats and solutions," in *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2017, pp. 193–199.

[9] ——, "Overview of 5g security challenges and solutions," *IEEE Communications Standards Magazine*, vol. 2, no. 1, pp. 36–43, 2018.

[10] I. Ahmad, S. Shahabuddin, T. Kumar, J. Okwuibe, A. Gurtov, and M. Ylianttila, "Security for 5g and beyond," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3682–3722, 2019.

[11] M. Alohaly, H. Takabi, and E. Blanco, "A Deep Learning Approach for Extracting Attributes of ABAC Policies," in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, ser. SACMAT '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 137–148, event-place: Indianapolis, Indiana, USA. [Online]. Available: https://doi-org.prox.lib.ncsu.edu/10.1145/3205977.3205984

[12] G. Americas, "Private & Enterprise Networks," Bellevue, WA, Tech. Rep., Aug. 2021. [Online]. Available: https://www.5gamericas.org/private-and-enterprise-networks/

[13] ——, "Security for 5G," Bellevue, WA, Tech. Rep., Dec. 2021. [Online]. Available: https://www.5gamericas.org/security-for-5g/

[14] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "PScout: analyzing the Android permission specification," in *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*. Raleigh, North Carolina, USA: ACM Press, 2012, p. 217. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2382196.2382222

[15] C. Bansal, K. Bhargavan, A. Delignat-Lavaud, and S. Maffeis, "Discovering concrete attacks on website authorization by formal analysis," *Journal of Computer Security*, vol. 22, no. 4, pp. 601–657, 2014, publisher: IOS Press.

[16] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5g authentication," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1383–1396.

[17] S. Behrad, E. Bertin, and N. Crespi, "A survey on authentication and access control for mobile networks: from 4g to 5g," *Annals of Telecommunications*, vol. 74, no. 9, pp. 593–603, 2019.

[18] S. Behrad, E. Bertin, S. Tuffin, and N. Crespi, "5g-ssaac: slice-specific authentication and access control in 5g," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 281–285.

[19] ——, "A new scalable authentication and access control mechanism for 5g-based iot," *Future Generation Computer Systems*, vol. 108, pp. 46–61, 2020.

[20] P. Bisson and J. Waryet, "5g ppp phase1 security landscape," *5G PPP Security Group White Paper*, 2017.

[21] E. Bitsikas and C. Pöpper, "Don't hand it Over: Vulnerabilities in the Handover Procedure of Cellular Telecommunications," in *Annual Computer Security Applications Conference*, ser. ACSAC. New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 900–915. [Online]. Available: http://doi.org/10.1145/3485832.3485914

[22] P. Centonze, R. J. Flynn, and M. Pistoia, "Combining Static and Dynamic Analysis for Automatic Identification of Precise Access-Control Policies," in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, 2007, pp. 292–303.

[23] B. Chen, S. Qiao, J. Zhao, D. Liu, X. Shi, M. Lyu, H. Chen, H. Lu, and Y. Zhai, "A security awareness and protection system for 5g smart healthcare based on zero-trust architecture," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 248–10 263, 2020.

[24] E. Y. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher, and P. Tague, "OAuth Demystified for Mobile Application Developers," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. Scottsdale Arizona USA: ACM, Nov. 2014, pp. 892–903. [Online]. Available: https://dl.acm.org/doi/10.1145/2660267.2660323

[25] Y. Chen, D. Tang, Y. Yao, X. Wang, X. Liu, H. Tang, and D. Zhao, "Seeing the Forest for the Trees: Understanding Security Hazards in the 3GPP Ecosystem through Intelligent Analysis on Change Requests," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/chen-yi

[26] W. Cheng, J. Schubert, C. Bornet, J. Bresson, sunn, and J. Black, "Hello from OpenAPI Generator." [Online]. Available: https://openapi-generator.tech/

[27] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz, "5g suci-catchers: still catching them all?" in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, pp. 359–364.

[28] C. Cremers and M. Dehnel-Wild, "Component-based formal analysis of 5g-aka: Channel assumptions and session confusion," 2019.

[29] Ericsson, "The essential building blocks of E2E network slicing," Stockholm, Sweden, Tech. Rep., Jun. 2021.

[30] D. Fett, R. Küsters, and G. Schmitz, "A Comprehensive Formal Security Analysis of OAuth 2.0," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1204–1215, event-place: Vienna, Austria. [Online]. Available: https://doi-org.prox.lib.ncsu.edu/10.1145/2976749.2978385

[31] S. Ghavamnia, T. Palit, A. Benameur, and M. Polychronakis, "Confine: Automated System Call Policy Generation for Container Attack Surface Reduction," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. San Sebastian: USENIX Association, Oct. 2020, pp. 443–458. [Online]. Available: https://www.usenix.org/conference/raid2020/presentation/ghavanmnia

[32] S. A. Gorski and W. Enck, "ARF: identifying re-delegation vulnerabilities in Android system services," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks - WiSec '19*. Miami, Florida: ACM Press, 2019, pp. 151–161. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3317549.3319725

[33] S. A. Gorski, S. Thorn, W. Enck, and H. Chen, "FReD: Identifying File Re-Delegation in Android System Services," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston,

MA: USENIX Association, Aug. 2022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/gorski

[34] S. A. Gorski III, B. Andow, A. Nadkarni, S. Manandhar, W. Enck, E. Bodden, and A. Bartel, "ACMiner: Extraction and Analysis of Authorization Checks in Android's Middleware," *arXiv:1901.03603 [cs]*, Jan. 2019, arXiv: 1901.03603. [Online]. Available: http://arxiv.org/abs/1901.03603

[35] D. Guija and M. S. Siddiqui, "Identity and access control for micro-services based 5g nfv platforms," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–10.

[36] D. Hardt, "The OAuth 2.0 Authorization Framework," Internet Engineering Task Force, Request for Comments RFC 6749, Oct. 2012, num Pages: 76. [Online]. Available: https://datatracker.ietf.org/doc/rfc6749

[37] S. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino, "Lteinspector: A systematic approach for adversarial testing of 4g lte," in *Network and Distributed Systems Security (NDSS) Symposium 2018*, 2018.

[38] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, and E. Bertino, "5GReasoner: A Property-Directed Security and Privacy Analysis Framework for 5G Cellular Network Protocol," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. London United Kingdom: ACM, Nov. 2019, pp. 669–684. [Online]. Available: https://dl.acm.org/doi/10.1145/3319535.3354263

[39] M. L. IjazAhmad, S. Shahabuddin, M. Ylianttila, and A. Gurtov, "Design principles for 5g security," *A Comprehensive Guide to 5G Security*, p. 75, 2018.

[40] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," May 2015, issue: 7519 Num Pages: 30 Series: Request for Comments Published: RFC 7519. [Online]. Available: https://www.rfc-editor.org/info/rfc7519

[41] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A survey on security and privacy of 5g technologies: Potential solutions, recent advancements, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 196–248, 2019.

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

[43] Z. Kotulski, W. Niewolski, T. W. Nowak, and M. Sepczuk, "New security architecture of access control in 5g mec," in *International Symposium on Security in Computing and Communication*. Springer, 2020, pp. 77–91.

[44] S. Lachmund, "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*. Cape Town, South Africa: ACM Press, 2010, pp. 8–14. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1809100.1809102

[45] Q. Li, B. Xia, H. Huang, Y. Zhang, and T. Zhang, "Trac: traceable and revocable access control scheme for mhealth in 5g-enabled iiot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3437–3448, 2021.

[46] W. Li and C. J. Mitchell, "Security issues in OAuth 2.0 SSO implementations," in *International Conference on Information Security*. Springer, 2014, pp. 529–541.

[47] X. Li, Y. Chen, Z. Lin, X. Wang, and J. H. Chen, "Automatic Policy Generation for Inter-Service Access Control of Microservices," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3971–3988. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/li-xing

[48] G. Lorenz, T. Moore, G. Manes, J. Hale, and S. Shenoi, "Securing ss7 telecommunications networks," in *Workshop on Information Assurance and Security*, vol. 2, 2001, p. 1115.

[49] A. S. Mamolar, Z. Pervez, J. M. A. Calero, and A. M. Khattak, "Towards the transversal detection of ddos network attacks in 5g multi-tenant overlay networks," *Computers & Security*, vol. 79, pp. 132–147, 2018.

[50] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The tamarin prover for the symbolic analysis of security protocols," in *International conference on computer aided verification*. Springer, 2013, pp. 696–701.

[51] ONF, "SD-Core." [Online]. Available: https://opennetworking.org/sd-core/

[52] S. Pailoor, X. Wang, H. Shacham, and I. Dillig, "Automated policy synthesis for system call sandboxing," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 135:1–135:26, Nov. 2020, number: OOPSLA. [Online]. Available: https://doi.org/10.1145/3428203

[53] S. P. Rao, B. T. Kotte, and S. Holtmanns, "Privacy in lte networks," in *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, 2016, pp. 176–183.

[54] R. Rosenbaum, "Secrets of the little blue box," *Esquire Magazine*, vol. 76, pp. 117–125, 1971.

[55] D. Rupprecht, A. Dabrowski, T. Holz, E. Weippl, and C. Pöpper, "On security research towards future mobile network generations," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2518–2542, 2018.

[56] A. Security, *White Paper: A Slice in Time: Slicing Security in 5g Core Networks*, AdaptiveMobile Security. [Online]. Available: https://info.adaptivemobile.com/5g-network-slicing-security

[57] Y. Shao, J. Ott, Q. Alfred Chen, Z. Qian, and Z. Morley Mao, "Kratos: Discovering Inconsistent Security Policy Enforcement in the Android Framework," in *Proceedings 2016 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2016. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2017/09/kratos-discovering-inconsistent-security-policy-enforcement-android-framework.pdf

[58] M. Shehab and F. Mohsen, "Towards enhancing the security of oauth implementations in smart phones," in *2014 IEEE International Conference on Mobile Services*. IEEE, 2014, pp. 39–46.

[59] E. Shernan, H. Carter, D. Tian, P. Traynor, and K. Butler, "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, 2015, pp. 239–260.

[60] F. Sun, L. Xu, and Z. Su, "Static Detection of Access Control Vulnerabilities in Web Applications," in *20th USENIX Security Symposium (USENIX Security 11)*. San Francisco, CA: USENIX Association, Aug. 2011. [Online]. Available: https://www.usenix.org/conference/usenix-security-11/static-detection-access-control-vulnerabilities-web-applications

[61] S.-T. Sun and K. Beznosov, "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*, 2012, pp. 378–390.

[62] Y. Sun, G. Feng, L. Zhang, M. Yan, S. Qin, and M. A. Imran, "User access control and bandwidth allocation for slice-based 5g-and-beyond radio access networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[63] M. Wazid, A. K. Das, S. Shetty, P. Gope, and J. J. Rodrigues, "Security in 5g-enabled internet of things communication: issues, challenges, and future research roadmap," *IEEE Access*, vol. 9, pp. 4466–4489, 2020.

[64] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie, "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*, ser. FSE '12. New York, NY, USA: Association for Computing Machinery, 2012, event-place: Cary, North Carolina. [Online]. Available: https://doi-org.prox.lib.ncsu.edu/10.1145/2393596.2393608

[65] R. Yang, G. Li, W. C. Lau, K. Zhang, and P. Hu, "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*, ser. ASIA CCS '16. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 651–662. [Online]. Available: http://doi.org/10.1145/2897845.2897874

Acronyms

AMF Access & Mobility Management Function.

N3IWF Non-3GPP Interworking Function.

NF	Network Functions.
NRF	Network Repository Function.
PCF	Policy Control Function.
RAN	Radio Access Network.
SBA	Service-based Architecture.
SEPP	Security Edge Protection Proxy.
SMF	Session Management Function.
UDM	Unified Data Management Function.
UDR	Unified Data Repository Function.
UE	User Equipment.
UPF	User Plane Function.