# PROV5GC: Hardening 5G Core Network Security with Attack Detection and Attribution Based on Provenance Graphs

Harsh Sanjay Pacherkar
Department of Computer Science
Binghamton University
Binghamton, New York, USA
hpacher1@binghamton.edu

Guanhua Yan
Department of Computer Science
Binghamton University
Binghamton, New York, USA
ghyan@binghamton.edu

## ABSTRACT

As 5G networks become part of the critical infrastructures whose dysfunctions can cause severe damages to society, their security has been increasingly scrutinized. Recent works have revealed multiple specification-level flaws in 5G core networks but there are no easy solutions to patch the vulnerabilities in practice. Against this backdrop, this work proposes a unified framework called PROV5GC to detect and attribute various attacks that exploit these vulnerabilities in real-world 5G networks. PROV5GC tackles three technical challenges faced when deploying existing intrusion detection system (IDS) frameworks to protect 5G core networks, namely, message encryption, partial observability, and identity ephemerality. The key idea of PROV5GC is to use provenance graphs, which are constructed from the communication messages logged by various 5G core network functions. Based on these graphs, PROV5GC infers the original call flows to identify those with malicious intentions. We demonstrate how PROV5GC can be used to detect three different kinds of attacks, which aim to compromise the confidentiality, integrity, and/or availability of 5G core networks. We build a prototype of PROV5GC and evaluate its execution performance on commodity cluster servers. We observe that due to stateless instrumentation, the logging overhead incurred to each network function is low. We also show that PROV5GC can be used to detect the three 5G-specific attacks with high accuracy.

## CCS CONCEPTS

• **Security and privacy** → **Mobile and wireless security**.

## KEYWORDS

5G networks, provenance graphs, attack detection and attribution

**ACM Reference Format:**
Harsh Sanjay Pacherkar and Guanhua Yan. 2024. PROV5GC: Hardening 5G Core Network Security with Attack Detection and Attribution Based on Provenance Graphs. 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, 11 pages. https://doi.org/10.1145/3643833.3656129

This work is in part supported by the US National Science Foundation under award CNS-1943079.

## 1 INTRODUCTION

As 5G networks are becoming part of the critical infrastructures, it is of crucial importance to harden their security. Although 5G standards released by the 3rd Generation Partnership Project (3GPP) have incorporated enhanced security features to protect user privacy and data security, we have witnessed increased efforts on scrutinizing potential vulnerabilities in 5G core networks recently. Formal analysis has revealed new vulnerabilities in the 5G Authentication and Key Agreement (5G-AKA) protocol which allow impersonation attacks [20, 26]. Multiple slicing attacks unique to 5G networks have been explained in detail in a whitepaper published by AdaptiveMobile Security [16]. Using model checking techniques, Akon *et al.* [17] have discovered five new types of vulnerabilities related to the access control mechanisms in 5G core networks. Moreover, as 5G networks are gradually rolled out worldwide, we expect to see an increasing number of penetration testing or fuzz testing efforts on real-world 5G core networks [1, 41].

As many 5G core network vulnerabilities discovered are due to specification-level flaws, there are no easy solutions to patch them in practice. Therefore, there is an urgent need for a unified framework that can detect and attribute attacks exploiting these vulnerabilities in real-world 5G core networks. To this end, we first attempt to deploy existing Intrusion Detection Systems (IDSes) such as Snort [8] and Zeek [11] to identify the key technical challenges. We observe that three main hurdles affects the effectiveness of these IDS frameworks in protecting 5G core networks. First, communication messages among different 5G core Network Functions (NFs) are encrypted by the Transport Layer Security (TLS) protocol, which renders it difficult for IDSes to perform deep packet inspection for attack detection. Second, the partial observability from a single vantage point within a 5G core network is insufficient for effective attack detection due to lack of 5G call flow semantics. Last but not least, identifiers observed within a 5G core network, such as IP addresses and 5G Global Unique Temporal Identifier (5G-GUTI), can be ephemeral, which complicates attack attribution.

To tackle these challenges, this work proposes the PROV5GC framework to detect and attribute various attacks targeting 5G core networks based on provenance graphs. Provenance graphs have been widely used to characterize causal relationships existing among the events (represented as edges) occurring to various entities (represented as nodes) in a complex system [37, 58]. 5G core networks adopt a Service-Based Architecture (SBA) where call flows are implemented by stitching the services provided by individual network functions (NFs) through service request/response messages exchanged among them [50]. It is thus natural to model these NFs as nodes and their communication messages as edges in a provenance graph. Based on this provenance graph, we can infer 5G call flows to identify various attack activities conducted by either misbehaving NFs or malware-infected mobile devices.

In a nutshell, our contributions in this work can be summarized as follows. **(1)** We analyze the technical challenges in deploying existing IDS frameworks such as Snort and Zeek to protect 5G core security. (§3). **(2)** We design a new framework called PROV5GC to facilitate attack detection and attribution based on provenance graphs, which are constructed from logged messages by 5G core NFs (§4). **(3)** Within the PROV5GC framework, we develop detection algorithms to identify three types of attacks against 5G networks from the confidentiality, integrity, and availability perspectives, respectively. (§5). **(4)** Using the 5G core NFs within an existing 5G network security testbed, we implement a prototype of PROV5GC (§6). We perform experiments to study the execution performance of PROV5GC as well as its detection accuracy. Our results show that the logging overhead is low as the average CPU usage and memory usage increases by less than 15% and 12%, respectively, for the resource-intensive core NFs. For the two signature-based detection schemes developed, we observe a detection accuracy of 100%, while the anomaly detection technique used to identify signaling storm attacks achieves a precision of 100% and a recall of 69.3% (§7).

## 2 BACKGROUND

A typical 5G network is comprised of access networks and a core network. An 5G access network can be a Radio Access Network (RAN), which includes a number of cooperating base stations which are called gNodeBs (gNBs). A User Equipment (UE) (e.g., a mobile phone) communicates with a 5G network through a gNB or other non-3GPP access technologies. The data plane of a 5G network is implemented by User Plane Functions (UPFs). The control plane of a 5G core network follows a SBA, whereby various NFs communicate with each other through well-defined interfaces. Key NFs in a 5G core network include Access and Mobility Management Function (AMF), Session Management Function (SMF), Authentication Server Function (AUSF), Unified Data Management (UDM), NF Repository function (NRF), Network Slice Selection Function (NSSF), and Policy Control Function (PCF).

The security architecture and procedures for 5G networks have been described in 3GPP TS 33.501 [15]. Even with enhanced security features, 5G networks still face various security challenges [40]. This work considers three different kinds of attacks that aim to compromise the availability, integrity, and/or confidentiality of 5G core networks: SMS signaling storm attacks abuse the delivery of SMS messages within the control plane of a 5G core network to cause denial or degradation of service, Packet Forwarding Control Protocol (PFCP) attacks spoof the UDP-based PFCP messages between SMF and UPF to inject malicious control messages, and slicing attacks exploit 5G core's flawed access control mechanisms to obtain user-sensitive data or cause DoS attacks.

**SMS signaling storm attacks:** The SMS-over-NAS feature introduced in 5G networks can be abused to create signaling storm attacks. SMS-over-NAS allows SMS messages to be delivered through the control plane only, which opens the opportunity for overloading 5G AMFs with pulsating SMS message delivery requests within a short period of time. To launch an SMS signaling storm attack, the attacker can use a mobile botnet to send SMS messages simultaneously. Due to the attack, a surging number of Service Request messages and/or Uplink NAS transport messages for SMS delivery can arrive at the AMF within a short period of time. As the AMF is involved in many of the signaling call flows in a 5G network, overloading it can degrade 5G network performances significantly.

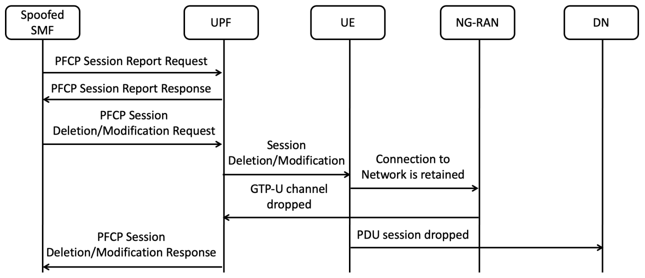

Spoofed SMF
UPF
UE
NG-RAN
DN
PFCP Session Report Request
PFCP Session Report Response
PFCP Session Deletion/Modification Request
Session Deletion/Modification
Connection to Network is retained
GTP-U channel dropped
PDU session dropped
PFCP Session Deletion/Modification Response

**Figure 1: Illustration of PFCP DoS attacks**

**PFCP attacks:** The PFCP protocol is used for communications over the N4 interface between an SMF and a UPF in the 5G core network. Particularly, session-related PFCP messages enable an SMF to establish, modify, or delete a PFCP session at an UPF. However, the PFCP protocol works on UDP over default port number 8805, suggesting that its packets can be easily spoofed. A successful attack only needs to guess correctly the Session Endpoint Identifier (SEID), which is used to uniquely identify a PFCP session at the IP address of a PFCP entity. Motivated by this idea, the work in [18] presents a few PFCP Denial of Service (DoS) attacks, which are illustrated in Figure 1.

In a *session deletion* attack, a spoofed SMF sends a *PFCP Session Deletion Request* message to the UPF, requesting to delete a PFCP session with a targeted SEID and its associated rules. If the SEID is guessed correctly, the UPF deletes its session with the UE affected, while the connection between the UE and the gNB is still retained. After the attack, the GTP-U channel between the gNB and the UPF is dropped, and the Packet Data Unit (PDU) session between the UE and the data network is also dropped, effectively causing a DoS attack against the victim UE. Thereafter the UPF sends back a response message to the spoofed SMF. A session modification attack can be performed similarly, with the spoofed SMF sending a *PFCP Session Modification Request* to the UPF.

**Slicing attacks:** AdaptiveMobile Security has published three types of slicing attacks, which can cause leakage of sensitive user data or denial of service against another NF [16]. As illustrated in Figure 2, all these attack scenarios involve the NRF, a misbehaving NF belonging to Slice 2 (NF A), and an NF which is shared by both Slices 1 and 2 (NF B).

*Type-I:* A Type-I slicing attack is performed as follows. (1) NF A sends a Nnrf_AccessToken_Get request to the NRF, requesting a token to access the service provided by NF B, with Slice Identifier $j = 1$. (2) As NF B is shared by both Slices 1 and 2, the NRF approves the request. (3) The NRF replies back to NF A with a valid token for Slice 1 to access NF B. (4) NF A requests a service from NF B, using the access token obtained from the NRF. (5) NF B accepts the token as it is valid and thus executes the service requested. (6) NF B replies to NF A with the service response.

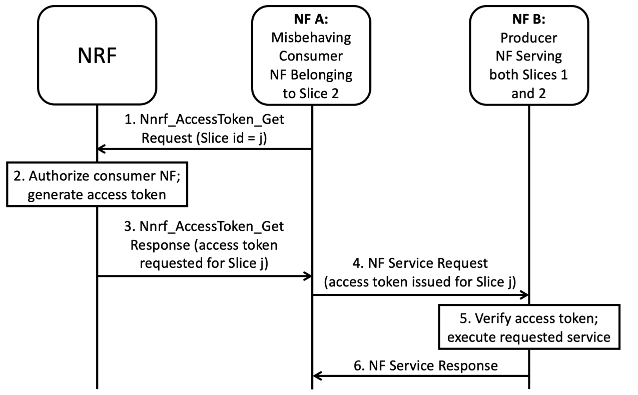

NRF
NF A: Misbehaving Consumer NF Belonging to Slice 2
NF B: Producer NF Serving both Slices 1 and 2
1. Nnrf_AccessToken_Get Request (Slice id = j)
2. Authorize consumer NF; generate access token
3. Nnrf_AccessToken_Get Response (access token requested for Slice j)
4. NF Service Request (access token issued for Slice j)
5. Verify access token; execute requested service
6. NF Service Response

**Figure 2: Illustration of slicing attacks**

*Type-II:* A Type-II slicing attack is aimed at causing denial of NF B's service to Slice 1. Its first three steps are the same as those in a Type-I attack, except that Slice Identifier $j$ in the Nnrf_AccessToken_Get request message is 2. As there is no foul play here, NF A obtains a valid token from the NRF for Slice 2 to access NF B. In Step 4, NF A uses this token to access NF B's service, while the service request includes an Overload Control header (i.e., *3gpp-Sbi-Oci*) for Slice 1. As the token is valid, NF B continues to process the *3gpp-Sbi-Oci* header. Given the misinformation, NF B assumes that Slice 1 was overloaded and thus should not be contacted, effectively leading to a DoS attack.

*Type-III:* A Type-III slicing attack assumes that NF B is an AMF. The first three steps are the same as those in a Type-II slicing attack. In Step 4, NF A uses the token to request a UE's sensitive data from NF B with the UE belonging to Slice 1. As the token is valid, NF B assumes that the NRF has already authorized the access. NF B can leak sensitive UE data from Slice 1 to NF A if it does not check whether the UE belongs to Slice 2 for which the token is valid.

## 3 THREAT MODEL AND CHALLENGES

5G core networks are vulnerable to various attacks due to security flaws in 5G protocol specifications [17, 49], insecure software implementation [3, 4], supply chain attacks [2], and malware-infected UEs [25, 29]. The **threat model** assumed by this work includes active attacks against 5G core networks from either *compromised 5G core NFs* or *malware-infected UEs*. As a compromised NF can always perform passive attacks (e.g., sniffing packets passing through it), its existence is difficult, if not impossible, to detect if it does not deviate from its normal behavior. Our threat model also rules out collusion attacks from multiple 5G core NFs, as it can be difficult for a malicious party to control multiple NFs in practice, but it is assumed that coordinated attacks are possible from a collection of malware-infected UEs. Our work assumes that the NRF is not compromised as it plays a central role in enforcing access control within the 5G core networks. This assumption allows us to develop a detection framework that can be seamlessly integrated into the SBA of an existing 5G core network.

Our goal in this work is to develop a unified framework to detect and attribute attacks that exploit 5G core network vulnerabilities. We encounter the following challenges when applying existing IDS frameworks such as Snort [8] and Zeek [11] for this purpose.

**Message encryption:** In order to detect the slicing attacks illustrated in Figure 2, it is necessary to understand the semantic meanings of the messages, such as message types and slice IDs, transmitted among the NRF, NF A, and NF B. According to 3GPP Technical Specification 33.501 [15], HTTP/2 communications inside the control plane of a 5G network should be protected using the TLS protocol. Snort ignores encrypted traffic for performance purposes and to minimize false alarms. Although Zeek provides limited support to handle encrypted traffic, doing so requires the necessary key materials available and thus complicates key distribution and management in 5G core networks.

**Partial observability:** The PFCP messages between the SMF and the UPF, which are transmitted over UDP packets, are not encrypted by TLS and thus open to analysis by Snort and Zeek. However, observing only PFCP messages is insufficient for these tools to decide whether the packets are sent by legitimate NFs or spoofed by malicious parties. To identify whether the source IP address in a PFCP message corresponds to the Fully Qualified Domain Name (FQDN) of a legitimate NF, an IDS can monitor the Domain Name System (DNS) packets to derive the mappings between FQDNs and IP addresses. However, such DNS packets can also be spoofed by the attacker to poison the mappings obtained by the IDS. Another possibility is to monitor the NF registration procedure, where the FQDN and the IP addresses appear in *nfProfile* Json object of the *Nnrf_NFManagement* service messages. Unfortunately the signaling packets used by this procedure are protected by the TLS protocol as discussed above, suggesting that these fields are invisible to the IDS without proper decryption keys.

**Identity ephemerality:** To detect SMS signaling storm attacks, it may be possible for an IDS to monitor the spike in the traffic volume destined to the AMF. As Non Access Stratum (NAS) messages between the UEs and the AMF are encrypted in 5G networks, the IDS deployed in between cannot recognize the NAS messages used for SMS delivery to detect directly SMS signaling storm attacks. Supposing that the IDS resorts to a coarse-grained detection model, which monitors the volume of NAS messages destined to the AMF to detect flooding attacks, attack attribution is still difficult because of the following reasons. First, the N2 interface between the gNB and the AMF can sometimes be protected by IPSec, whose existence can obscure the true IP addresses of the UEs that participate in the flooding attacks. Second, even if the N2 traffic is not encrypted by IPSec or the IDS is deployed after the packets are decrypted, it can still be difficult to map from the IP addresses to the permanent identities of the UEs (e.g., Subscription Permanent Identifier (SUPI)). In a 5G network IP addresses are dynamically assigned to the UEs by the SMF based on the DHCPv4 protocol. Therefore, a UE may use different IP addresses over time, or the same IP address can be assigned to distinct UEs during different periods.

## 4 PROV5GC ARCHITECTURE

To overcome the aforementioned challenges, we propose the PROV5GC framework, whose architecture is shown in Figure 3. Each 5G core NF is instrumented to selectively log key information contained within communication messages that it is going to send or has just received. As an outgoing message is logged by the sending NF before its encryption by the TLS protocol and an incoming one is logged by the receiving NF after it has been decrypted by

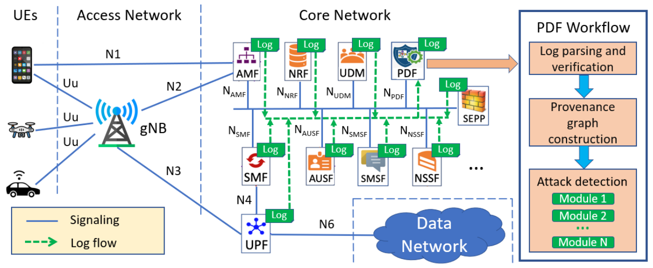


**Figure 3: Architecture for attack detection and attribution based on provenance graphs in 5G networks**

the TLS protocol, it overcomes the message encryption challenge faced by the existing IDSes like Snort [8] and Zeek [11].

All logged messages are transmitted to a new SBA-compliant NF called Provenance-Based Detection Function (PDF) for attack detection and attribution. By constructing a provenance graph from the logged messages, the PDF is able to achieve a global view of all the call flows that have occurred in the 5G core network. The rich semantics contained with the provenance graph enables the PDF to detect various activities that deviate from the normal behavioral patterns of the core NFs or the UEs, which can be difficult to recognize based on the observations from a single vantage point.

Within a 5G core network, the same UE can be referred by different identifiers, such as SUPI, Subscription Concealed Identifier (SUCI), 5G-GUTI, IP address, and so on. A core NF can also be identified by its IP address and FQDN. Some identifiers may have limited lifespans, which complicates attack attribution. By analyzing the call flows within the provenance graph, the PDF can track how these identifiers are created and used to establish the correct correspondences among the various identifiers observed over time.

In our design PROV5GC should be deployed on a Public Land Mobile Network (PLMN) basis, as the collaborative relationships established among each PLMN's core NFs can be naturally extended for cooperative attack detection in practice. Moreover, as PDF itself is implemented as an NF in the 5G SBA, its deployment can leverage the existing security enforcement mechanisms in 5G core networks, which include mutual authentication and transport security between NFs based on the TLS protocol and token-based authorization for access of NF services using OAuth 2.0 [39].

In order for an NF to report its logs to the PDF, it needs access tokens from the NRF. As it is assumed that the NRF is not compromised in our threat model, the attacker cannot arbitrarily inject spoofed logs to the PDF. However, it is possible that an existing NF compromised by the attacker intentionally adds, deletes, or modifies the logs reported to the PDF. PROV5GC catches such abnormal behaviors by comparing the logs of the same message from the respective sender and receiver NFs.

## 4.1 Stateless NF instrumentation for logging

To minimize performance overhead, each NF is instrumented in a *stateless* manner, which means that the NF does not need to add extra states in its implementation for logging. For each message sent or received by an NF, the NF is instrumented to log the following information: *TimeStamp* (the *local* time at which the message is sent or received), *Role* (whether the message is logged by the consumer or the provider NF), *SenderID* (the identifier of the sender), *SenderType* (the type of the sender), *ReceiverID* (the identifier of the receiver), *ReceiverType* (the type of the receiver), *MessageType* (the type of the message as described in 5G specifications), *MessageData* (important data included in the message stored as key-value pairs), *MessageHash* (the hash value of the message body) and *RequestHash* (the hash value of the request message).

Due to stateless logging, not all the above fields are known when a message is logged. We thus consider the following different cases when a message is logged by NF A. **(i) NF A is an AMF and it is receiving a NAS message:** The *SenderID* is set to be the 32-bit `AMF-UE-NGAP-ID` IE (Information Element) in the message, which uniquely identifies the UE association over the NG interface within the AMF [14] and the *SenderType* is `UE`. The *ReceiverID* is set to be the 128-bit NF instance ID of the AMF, while *ReceiverType* is *AMF*. **(ii) NF A is sending a message:** *SenderID* is set to the 128-bit NF instance ID of NF A [13] and the *SenderType* is NF A's type. Both the *ReceiverID* and *ReceiverType* fields are filled with their respective values if they are known, or special bits (e.g., all 1's) otherwise. **(iii) NF A is receiving a message:** the *ReceiverID* is set to the 128-bit NF instance ID of NF A and the *SenderType* is its type. Both the *SenderID* and *SenderType* fields are filled with their respective values if they are known, or special bits (e.g., all 1's) otherwise.

While the *MessageHash* field is logged by both the sender and the receiver of the same message to ensure that neither behaves anomalously, the *RequestHash* field allows to establish the correspondence between the request and the response messages. Due to the SBA nature of the 5G core network, communications among its NFs follow a HTTP/2-based client-server model where two typical types of messages are used: request/response and subscribe/notify. For the same pair of request/response messages, the hash value of the request message is logged as its *RequestHash* field for both messages, which can thus be used to establish their correspondence. For subscribe/notify messages, the *RequestHash* field can be ignored.

## 4.2 Discovery and negotiation with the PDF

When the PDF starts, it first registers itself at an NRF where other NFs can discover its service. When an NF other than the PDF starts, it contacts the NRF to obtain an JSON Web Token (JWT) to access the service provided by the PDF. Using this token, the NF further contacts the PDF and requests to upload its logs to it. After the NF acquires the access token from the NRF, it sends a Connection Request to the PDF along with the token. The PDF checks the validity of the token and, if the token is valid, responds with an OK message. Thereafter the NF sends a Negotiation Request message to the PDF with the mode of log transmission, indicating whether the logs should be transmitted in a streaming fashion or in a batch manner. The request data also contains a randomly generated integrity key $k_{int}^{NF}$, which will be used later to ensure the integrity of logs. The corresponding response message contains a public key $k_{pub}^{PDF}$, which later will be used to encrypt logs during their transmissions.

In a distributed environment, there may exist clock skews among different NFs. During the negotiation phase, the PDF also estimates the clock skew of the requesting NF, say, NF A. We borrow the idea from the Network Time Protocol (NTP) [45] as follows. The PDF first sends Message 1 to NF A at its local time $T_1^A$. When the

NF receives this request, it records its local time as $T_2^A$ and sends back a response to the PDF at local time $T_3^A$, where the response message contains both $T_2^A$ and $T_3^A$. When the PDF receives the response message at its local time $T_4^A$, it estimates the clock offset as $((T_2^A - T_1^A) + (T_3^A - T_4^A))/2$, assuming that the reverse path (from NF A to the PDF) takes the exactly the same time as the forward one (from the PDF to NF A). The round trip time between the PDF and NF A can be estimated as $(T_4^A - T_1^A) - (T_3^A - T_2^A)$.

## 4.3 Log transmission and commitment

Logs are encrypted with the PDF's public key $k_{pub}^{PDF}$ to achieve confidentiality and integrity-protected with $k_{int}^{NF}$. The NF sends the logs to the PDF in a streaming or batch mode, as negotiated between it and the PDF in the previous phase. For each NF A, we denote its batch period (i.e., how long the logs should be buffered in a batch before being transmitted) as $\gamma^A$. If the streaming mode is used, $\gamma^A$ can be simply set to be 0.

For each logged message $m$ received, the PDF first checks its validity before committing it to the provenance graph $\mathcal{G}_{prov}$, which keeps all logged information extracted so far. (1) If *m.ReceiverType = AMF* and *m.MessageType* indicates that this is a NAS message, logged message $m$ should be transmitted from the receiving AMF. Hence, the PDF simply verifies whether the NF whose instance Id is *m.ReceiverID* is indeed an AMF according to $\mathcal{G}_{prov}$. The information that this NF should be an AMF can be derived from earlier messages when the AMF requests an access token from the NRF to access the PDF's service, where the request message must include the NF type of the consumer NF. (2) For each message transmitted between two NFs, its validity is ensured only after two logged messages received from both the consumer and producer NFs, respectively, are matched based on their TimeStamp, SenderID, SenderType, ReceiverID, ReceiverType, MessageType, and MessageHash fields.

Uncommitted logged messages are sorted based on their adjusted time stamps and wait in queue $Q_{uc}$ to be matched later. Detection modules can be developed to scan this queue for those that have not been committed for an excessively long period.

## 4.4 Provenance graph construction

Once a logged message is committed, all its unknown fields should be resolved. Its information is then added to the provenance graph $\mathcal{G}_{prov}$. A node in $\mathcal{G}_{prov}$ represents a *continuous* state of a UE or an NF recognized by the core network. Its *type* attribute indicates whether it is a UE or a particular type of NF nodes (e.g., AMF). Its *idmap* attribute maps one of the node's identity names to a list of $(value, startTime, endTime)$ triples, where $value$ is the identity value while $startTime$ and $endTime$ are the identity's start time and end time recorded by the core network, respectively. **(1)** A **UE node** is first created when a message of type *Initial_UE_Message* is received from an AMF that carries a Registration Request from the UE. A UE's possible identity names include SUPI, SUCI, 5G-GUTI, S-TMSI (S-Temporary Mobile Subscriber Identity), PEI (Permanent Equipment Identifier), IP_ADDR (IP address), RAN_UE_NGAP_ID, AMF_UE_NGAP_ID, and SNSSAI (slice identifier). The correspondences among the same UE's identities can be established by analyzing the relevant call flows such as UE registration. The *idmap* is updated by parsing the MessageType fields as well as the information stored in the MessageData fields in the log. **(2)** An **NF node** is created when it registers itself with the NRF through the *Register_NF* service request message. An NF node's identity names include InstanceID (instance ID), IPAddr (IP address), FQDN, and SNSSAI (slice identifier), and their values are extracted from *Register_NF* service request messages. Each identity's start (end) time is set to the time when the NF is registered (deregistered) at the NRF.

For each committed message, a directed edge is added to $\mathcal{G}_{prov}$ from the node representing its sender to the one its receiver. Each edge includes three attributes, *time*, *type* and *data*. The *time* attribute gives the receiving time of the message, adjusted based on the PDF's local clock, and the latter two are copied from the *MessageType* and *MessageData* fields of the logged message, respectively. For each directed edge $e$, we let $e.tail$ and $e.head$ denote its tail and head node, respectively. Figure 4 shows an example provenance graph.

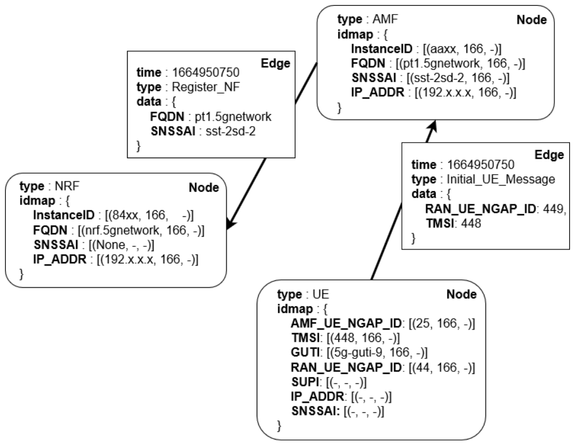

type : AMF
Node
idmap : {
InstanceID : [(aaxx, 166, -)]
FQDN : [(pt1.5gnetwork, 166, -)]
SNSSAI : [(sst-2sd-2, 166, -)]
IP_ADDR : [(192.x.x.x, 166, -)]
}
Edge
time : 1664950750
type : Register_NF
data : {
FQDN : pt1.5gnetwork
SNSSAI : sst-2sd-2
}
type : NRF
Node
idmap : {
InstanceID : [(84xx, 166, -)]
FQDN : [(nrf.5gnetwork, 166, -)]
SNSSAI : [(None, -, -)]
IP_ADDR : [(192.x.x.x, 166, -)]
}
Edge
time : 1664950750
type : Initial_UE_Message
data : {
RAN_UE_NGAP_ID: 449,
TMSI: 448
}
type : UE
Node
idmap : {
AMF_UE_NGAP_ID: [(25, 166, -)]
TMSI: [(448, 166, -)]
GUTI: [(5g-guti-9, 166, -)]
RAN_UE_NGAP_ID: [(44, 166, -)]
SUPI: [(-, -, -)]
IP_ADDR: [(-, -, -)]
SNSSAI: [(-, -, -)]
}

**Figure 4: Provenance graph example. The hyphen symbol (i.e., '-') means that the value is unknown yet.**

## 4.5 Periodic persistence of provenance graph

There can be so many messages logged in a 5G network that $\mathcal{G}_{prov}$ becomes too big to be stored entirely in the memory. By dividing time into *epochs*, we can partition the provenance graph temporally to subgraphs, each containing only the edges whose *time attributes* fall into a certain epoch. At the end of each epoch, its corresponding provenance subgraph is dumped from the memory onto the disk.

When an epoch ends, a logged message whose time attribute falls into this period may not have been received by the PDF yet due to its *transmission latency* from the sender NF and the possible *batching latency* at the sender NF. To circumvent this issue, the PDF always waits for $\Delta_{any}$ extra time units after the end of an epoch before dumping the current provenance subgraph onto the disk, where $\Delta_{any}$ can be configured as follows. Recall that the round trip time between the PDF and NF A can be estimated as $(T_4^A - T_1^A) - (T_3^A - T_2^A)$ during the negotiation phase. We define:

$$\Delta_x = \max_{A \in \mathcal{F}: A.type = x} \frac{(T_4^A - T_1^A) - (T_3^A - T_2^A)}{2} + \gamma^A + \epsilon, \qquad (1)$$

where $\mathcal{F}$ is a set including all NFs in the 5G SBA, $\epsilon$ provides sufficient cushion for latency jitter from any NF to the PDF, and $\gamma^A$ is the batching period of NF $A$. Hence, $\Delta_{any}$ provides an upper bound on the latency of each logged message from all NFs.

## 5 DETECTION MODULES

In this section we describe the detection modules for the three attacks discussed in Section 2. Let $G_i$ be the provenance subgraph for the $i$-th *logging period*. We use $G_i^m = (V_i^m, E_i^m)$ and $G_i^d = (V_i^d, E_i^d)$ to denote $G_i$ stored in memory and on the disk, respectively.

### 5.1 SMS signaling storm attack detection

We consider the non-parametric CUSUM method [23] as it does not need to know the distribution of the random process before or after the change occurs. A non-parameter CUSUM scheme uses two key parameters: *drift* $r$ and *threshold* $h$. Let $X = (x_1, x_2, ...)$ denote the observed random sequence with mean $\alpha(X)$ under its normal operation. To detect if there is an abrupt change in $X$'s mean value, we transform it to a new sequence $Z = (z_1, z_2, ...)$ where $z_i = x_i - r$. Drift $r$ should be chosen in such a way that $Z$ has a negative mean before the change point but has a positive mean after it (i.e., $r > \alpha(X)$). The non-parametric CUSUM method can run in an online fashion:

$$y_0 = 0 \tag{2}$$

$$y_k = \max\{0, y_{k-1} + z_k\}, \tag{3}$$

where an alarm is raised at step $k$ if $y_k$ exceeds threshold $h$. After an alarm is raised, we do not reset $y_k$ to 0 in order to achieve high detection accuracy [31].

We consider online detection of SMS signaling storm attacks where the CUSUM-based detection algorithm runs periodically every $\theta$ time units. The detection scheme needs only logged messages from AMFs. As these messages may vary in their arrival times at the PDF, for each detection interval $[t, t + \theta)$ we postpone the single-step execution of the online CUSUM detection method until $t + \theta + \Delta_{amf}$, where $\Delta_{amf}$, as shown in Eq. (1), provides the upper bound on the latency of each logged message from any AMF node.

Therefore, online detection of SMS signaling storm attacks can be performed as in Algorithm 1. Due to different latency, logged messages whose timestamps belong to different detection intervals can arrive at the PDF at the same time. Hence, two different sets of states are used: $x$ and $w$ are used for the current detection interval while $\widehat{x}$ and $\widehat{w}$ are for the next one. Both $x$ and $\widehat{x}$ are initialized to be 0. Both $w$ and $\widehat{w}$ are initialized to be an empty set. Variable $t_{last}$ stores the last time a single step of CUSUM is executed.

The `Detect_Single_Step` method in Algorithm 1 is invoked every $\theta$ time units. It applies a single step of the CUSUM method to detect abrupt changes (Line 2) and then reset $x$, $w$, and $t_{last}$ (Line 3). If an alarm is raised by CUSUM, the method marks the last detection period as an attack interval (Line 5) and perform attack attribution to identify UEs participating in the attack (Line 6).

The `Update_Edge` method is called whenever there is a new edge is added to $G_n^m$. If the corresponding message is used for SMS delivery over NAS, its timestamp is checked to see if it belongs to the current detection interval $[t_{last} - \Delta_{amf}, t_{last} + \theta - \Delta_{amf})$, or the next one $[t_{last} + \theta - \Delta_{amf}, t_{last} + 2\theta - \Delta_{amf})$. In the former case, states in

**Algorithm 1:** Online detection of SMS signaling storm attacks from the in-memory provenance subgraph

```
Input: $G_n^m$, $x$, $\widehat{x}$, $w$, $\widehat{w}$, $t_{last}$, $r$, $h$
Output: Alarms for SMS signalling storm attacks
1  Function Detect_Single_Step($X$, $W$, $r$, $h$):
2      Treating $x$ as $x_k$, apply a single step of CUSUM;
3      $t_{prev} \leftarrow t_{last}$, $t_{last} \leftarrow$ current wallclock time;
4      if an alarm is raised by CUSUM then
5          Mark $[t_{prev}, t_{last}]$ as an attack interval;
6          Perform attack attribution based on edges in set $w$;
7      $x \leftarrow \widehat{x}$, $w \leftarrow \widehat{w}$;
8      return;
9  Function Update_Edge($e$):
10     if $(e.head.type = \text{“}AMF\text{''}) \wedge (e.type = \text{“}UE\_Init\_NAS\text{''}) \wedge (e.data[\text{“}msgType\text{''}] = \text{“}SMS\text{''})$ then
11         if $(t_{last} - \Delta_{amf} \leq e.time < t_{last} + \theta - \Delta_{amf})$ then
12             $x \leftarrow x + 1$, $w \leftarrow w \cup \{e\}$;
13         else if $(t_{last} + \theta - \Delta_{amf} \leq e.time < t_{last} + \theta)$ then
14             $\widehat{x} \leftarrow \widehat{x} + 1$, $\widehat{w} \leftarrow \widehat{w} \cup \{e\}$;
15         else
16             Report logging time error in $e$;
17     return;
```

$x$ and $w$ are updated, while in the latter one, states $\widehat{x}$ and $\widehat{w}$ are updated. Note that in the latter case, the timestamp in the logged message should not exceed the next time when `Detect_Single_Step` is called (i.e., $t_{last} + \theta$). When a previous detection period ends, states $\widehat{x}$ and $\widehat{w}$ are carried over to $\widehat{x}$ and $\widehat{w}$ for the next detection period, respectively (Line 7 in the `Detect_Single_Step` method).

**Attack attribution:** Each edge $e$ in set $w$ is an SMS-over-NAS message, which uses `AMF_UE_NGAP_ID` to identify the UE that sends the SMS. It also includes a timestamp (i.e., *e.time*) indicating the time when the message is observed. Although the `AMF_UE_NGAP_ID` can be ephemeral over time, we can query the provenance graph for the SUPI of the UE that uses `AMF_UE_NGAP_ID` at time *e.time*.

**Evasion robustness:** To evade the CUSUM-based detection, the attacker can limit the flooding rates of SMS-over-NAS messages from malware-infected devices. Although such stealthy attacks can still drain the resources in the targeted 5G core networks, their attack effects become less obvious to the legitimate users.

### 5.2 Slicing attack detection

Algorithm 2 presents how to detect the three types of slicing attacks discussed in Section 2 from on-disk provenance subgraphs. Without loss of generality, we assume that subgraphs $(G_1^d, ..., G_n^d)$ are considered. The algorithm checks every edge in the current provenance subgraph to identify slicing attacks. The helper logic function, `Find`$(J, L, t)$, is defined as:

$$\exists j \in J, l \in L : j = l.value \wedge l.startTime \leq t \leq l.endTime.$$

For Type-I slicing attacks, Algorithm 2 examines each edge destined to an NRF node (Line 8); if the edge type indicates an Nnrf_Access_Token_Get message, it checks whether the slice IDs

**Algorithm 2:** Offline detection of slicing attacks

```
Input: $\mathcal{G} = (G_1^d, ..., G_n^d)$
Output: Instance IDs of misbehaving NFs
1  Function Detect_Slicing_Attacks($\mathcal{G}$):
2      $Q_{amf} \leftarrow$ Service requests to AMF returning UE information;
3      foreach $i \in 1..n$ do
4          Load $G_i^d$ into memory as $G_i^m$;
5          $sbi \leftarrow$ "3gppSbiOci";
6          foreach $e \in E_i^m$ do
7              $w \leftarrow e.tail.idmap[``SNSSAI'']$;
8              if $e.head.type = ``NRF''$ then
9                  if $(e.type = ``Nnrf\_AccessToken\_Get'') \wedge (\texttt{Find}(e.data[``SliceIDs''], w, e.time) = False)$ then
10                     Report a Type-I slicing attack;
11             else if $(sbi \in (e.data.keys())) \wedge (\texttt{Find}(e.data[sbi][``SliceIDs''], w, e.time) = False)$ then
12                 Report a Type-II slicing attack;
13             else if $(e.head.type = ``AMF'') \wedge (e.type \in Q_{amf})$ then
14                 foreach $u \in V_i^m$ do
15                     foreach $(y, t_s, t_e) \in u.idmap[e.data[``UE\_ID\_Type'']]$ do
16                         if $(e.data[``UE\_ID\_Value''] = y) \wedge (t_s \leq e.time \leq t_e)$ then
17                             $J \leftarrow \emptyset$;
18                             foreach $s \in u.idmap[``SNSSAI'']$ do
19                                 if $s.startTime \leq e.time \leq s.endTime$ then
20                                     $J \leftarrow J \cup \{s.value\}$;
21                             if $\texttt{Find}(J, w, e.time) = False$ then
22                                 Report a Type-III slicing attack;
23         Remove $G_i^m$ from memory;
24     return
```

requested (i.e., $e.data[``SliceIDs'']$) match any of the slice IDs that the requesting NF (i.e., $e.tail$) belongs to at the request time (i.e., $e.time$) (Line 9). If the check fails, a Type-I slicing attack is reported.

For Type-II slicing attacks, the algorithm checks if there is any 3gppSbiOci data in the logged message data. If so, it checks whether the slice IDs requested for overload control in the message match any of the requesting NF's slice IDs that are valid at the request time (Line 11). If none is found, a Type-II slicing attack is reported.

For Type-III slicing attacks, the algorithm considers only edges destined to AMF nodes; these edges should also correspond to service request messages that are likely to reveal sensitive UE information (i.e., $Q_{amf}$ at Line 13). For each such edge discovered, the algorithm uses the UE's identifier type and value, which are given by UE_ID_Type and UE_ID_Value in the message data, to search for the UE node in the provenance subgraph. During the search, each UE identity's lifetime is used to ensure that at the time of service request (i.e., $e.time$) the UE identity should be valid (Lines 14-16). When such a UE node is found, the algorithm further obtains all its slice IDs which are valid when the service request occurs (Lines 17-20). The algorithm checks if any of these slice IDs of the UE matches the requesting NF's Slice IDs at the service request time; if none is found, a Type-III slicing attack is reported (Lines 21-22).

**Evasion robustness:** The slicing attack detection algorithm exploits the mismatches between the attacking NF's slice IDs and those requested by it. To evade the detection, the attacker can request to join the victim NFs' slices using NFUpdate service requests, which, however, need approval from the Operations, Administration, and Maintenance (OAM) in a 5G network.

## 5.3 PFCP attack detection

PFCP DoS attacks as described in Section 2 are powerful because they only need to *spoof* a legitimate SMF instead of compromising and controlling it. However, they can be easily detected within the PROV5GC framework, because the spoofed SMF is unable to send logged PFCP Session Deletion/Modification Request messages that match those reported from the UPF.

**Algorithm 3:** PFCP attack detection Algorithm

```
Input: $Q_{uc}$ (waiting queue of uncommited messages), $t$ (current local time)
Output: PFCP attack attempts
1  Function Detect_PFCP_Attacks($Q_{uc}, t$):
2      foreach $m \in Q_{uc}$ do
3          if $(m.ReceiverType = ``UPF'') \wedge ((m.MessageType = ``PFCP\_Session\_Deletion\_Request'' \vee (m.MessageType = ``PFCP\_Session\_Modification\_Request''))) \wedge (t - m.TimeStamp > \Delta_{smf})$ then
4              Report a PFCP attack;
5      return
```

Algorithm 3 presents the algorithm to detect PFCP attack attempts illustrated in Figure 1. Its `Detect_PFCP_Attacks` method is called periodically, which scans for uncommitted messages of type either *PFCP_Session_Deletion_Request* or *PFCP_Modification_Request* reported by UPFs in the waiting queue $Q_{uc}$. If such a message has stayed in the waiting queue for more than $\Delta_{smf}$ time units, a PFCP attack is reported. Note that $\Delta_{smf}$ provides an upper bound on the latency of each logged message received from any SMF.

**Evasion robustness:** The effectiveness of the PFCP attack detection scheme builds upon the observation that spoofed PFCP request messages from the SMF to the UPF are logged by the UPF but not the SMF, leading to uncommitted messages in the waiting queue.

To evade the detection, the adversary has to compromise the UPF and prevent it from reporting the spoofed PFCP request messages to the PDF. However, the adversary who is able to hijack the UPF can cause the same kind of attack damages without using spoofed PFCP requests. Another way of evading detection is to spoof the logged PFCP request messages from the SMF to the PDF, which, however, is hard to achieve as a valid authorization token issued by the NRF is needed for an NF to report its logs to the PDF.

## 6 IMPLEMENTATION DETAILS

PROV5GC is developed based on the 5G core NFs in VET5G [55], a testbed focused on 5G network security. The testbed implemented the essential features of core NFs shown in Figure 3. To validate that their implementations follow 5G specifications, we tested their interoperability with three UE/RAN configurations: (1) modified Android emulator and OAI-emulated gNB [7]; (2) UEs and gNBs emulated by UERANSIM [10]; and (3) Google Pixel 6 and srsRAN-emulated gNB with aid of USRP B210 [9].

As all VET5G NFs except UPF (part of UPF is written in the P4 programming language) are implemented in Rust, we use Rust to instrument each NF for message logging. The PDF is also implemented in Rust. The communications between the PDF and other NFs are implemented based on the ZeroMQ messaging library [12]. We use the ZeroMQ PUSH/PULL sockets for many-to-one message transfer because they allow the PDF to receive the logs from the queue in order. PROV5GC uses the Curve25519 Elliptic Curve Cryptography (ECC) scheme provided by ZeroMQ for data encryption, while log integrity is ensured by the keyed-hash message authentication code (HMAC) scheme based on SHA256.

We choose the UERANSIM tool [10] to emulate both gNBs and UEs in our experiments because it allows us to simulate large-scale attack scenarios. To simulate user activities in UERAMSIM, we use an ON/OFF model based on Pareto distributions. When a UE starts up and performs registration, the UE starts with an ON period. During the ON period the UE is in a CM_CONNECTED state with the AMF. After this ON period the UE transitions to an OFF period, during which the UE stays in a CM_CONNECTED state with the AMF until the gNB deems the UE to be idle for too long, which thus invokes the call flow to transition the UE into a CM_IDLE state. After the inactivity period the UE transitions back to an ON period by performing a service request. To randomize the UE registration time, we let each UE start initially after a random period whose length follows an exponential distribution.

An established PDU session can be used for applications such as video streaming and web surfing. We do not model user activities during this stage as the user data are transmitted in the data plane. They have limited impact on the operations of the control plane except that some usage reporting events may be triggered by the UPF to notify the SMF through the N4 interface.

## 7 EXPERIMENTS

Our experiments use two machines. (1) Machine 1 is a Linux workstation running Ubuntu 22.04 LTS. It has an Intel i9-10850K CPU with 10 hyper-threaded cores and 32GB RAM. (2) When we emulate a large number of UEs in an experiment, we use Machine 2, which runs Debian 11. The machine has an Intel Xeon Gold 6338 CPU including 64 hyper-threaded cores and a total RAM of 512GBs.

### 7.1 Logging overhead

In the first set of experiments, we measure the logging overhead for each NF instrumented. For each NF type, only one instance is used. Each experiment simulates a 5G network with 50 thousand UEs. Each UE is simulated by an ON/OFF model with Pareto distributions. The mean of each ON or OFF period is 500 seconds. When there is a transition from an OFF state to an ON one, a UE randomly picks another one as the destination to send an SMS message. Machine 2 is used for these experiments. We allocate 16 CPUs to all the NFs in the core network, *excluding* the PDF. We use the Docker stats [5] to measure the resources used by each NF.

We run the experiment 10 times. Figure 5 shows the average CPU and memory usage with and without logging enabled by the core NFs. The NF with the highest CPU and memory usage is AMF. Without logging, the mean CPU usage is 5.4% and the mean memory usage is 1.7GBs. By contrast, with logging enabled, the mean CPU usage is 6.1%, an increase of 13.0%, and the mean memory usage is 1.9GBs, an increase of 11.8%.

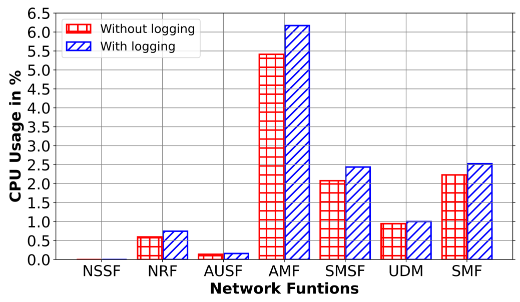
Comparing CPU and memory usages of different NFs with and without logging in Figure 5, we can conclude that logging overhead is low. For all those resource-intensive NFs (i.e., AMF, SMF, SMSF, and UDM), the average CPU usage increases by less than 15% while the average memory usage increases by less than 12%.

### 7.2 Signalling storm attack detection

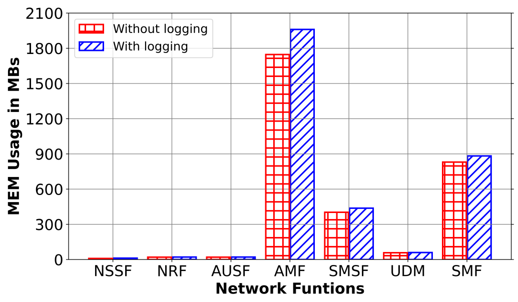
Using Machine 1, we simulate 10,000 UEs based on the ON/OFF user activity model. Each UE's initial waiting time before its registration request follows an exponential distribution with a mean time of 1000 seconds. The ON/OFF user activity of the UE is simulated with a Pareto distribution with a mean of 400 seconds. When UE transitions from an OFF to an ON state, it sends an SMS message to another randomly picked UE. 500 UEs are infected by the bot malware. Each bot-infected UE is commanded to send five SMS messages in a spurt to another random UE at 1800 second in the experiment. Each experiment lasts an hour and is repeated 10 times. The `Detect_Single_Step()` function in Algorithm 1 is invoked every 30 seconds, while the Update_Edge() function is executed whenever a new edge is added to the provenance graph.

We fix threshold $h$ and drift $r$ in the CUSUM algorithm to be 100 and 500, respectively. The detection performance is measured as follows. At each step, a true positive (false positive) is the case when an alarm is raised and the interval does (does not) have attack SMS messages from bot-infected UEs. On the other hand, when no alarm is raised for the current interval, if there are no (at least one) bot-infected UE sending attack SMS messages in this interval, it is a true negative (false negative). Assuming that the numbers of true positives, false positives, true negatives, and false negatives are $TP$, $FP$, $TN$, and $FN$, respectively, we have $precision = TP/(TP + FP)$ and $recall = TP/(TP + FN)$. We observe a precision of 100% and a recall of 69.3% from our experiments. The missed attack intervals, which occur during the ramp-up phase of the SMS signaling storm attack, are understandable as it takes time for the attack to produce an accumulative effect detectable by the CUSUM algorithm.

Recall that the `Detect_Single_Step` method in Algorithm 1 detects attack intervals. For attack attribution, we consider all UEs that send SMS-over-NAS messages during the attack intervals detected as suspicious. A false positive occurs if a normal UE happens



(1) CPU usage



(2) Memory usage

**Figure 5: Logging overhead incurred by each NF instrumented by PROV5GC**

to send such messages during any of the attack intervals, while a false negative happens when a malware-infected UE participates in the SMS signaling storm attack during the attack periods but is not flagged as suspicious. We observe a precision of 68.8% and a recall of 100% by this attack attribution method. The relatively high false alarm rates are understandable as the detection method treats all UEs sending SMS-over-NAS messages during an attack interval as suspicious. Due to its high recall, however, this method can be used as a first line of defense to narrow down the list of UEs that may have been infected by malware for signaling storm attacks, while other complementary methods (e.g., host-based IDSes) can be used to detect which apps installed on these UEs may be malware.

## 7.3 Slicing attack detection

We use Machine 1 to simulate 10,000 UEs for 6000 seconds. Each UE has two sets of activities. An exponential distribution with a mean duration of 500 seconds is used to model both its registered and deregistered states. During a registered state, each UE switches between active and idle states, whose durations follow a Pareto distribution with a mean interval of 100 seconds. We dump the provenance graph into a persistent storage every 600 seconds. During each interval we simulate a hybrid attack scenario. First, the attacker NF uses fake slice information to request access token from the NRF, which is the Type-I slicing attack. It next uses this access token to request a $provide\_location\_information$ service provided by the AMF, which requests the location information of a UE which belongs to a different slice. This leads to a Type-III slicing attack. Finally, the HTTP request for $provide\_location\_information$ has a *3gpp-Sbi-Oci* header with fake information, which makes a Type-II slicing attack.

Slicing attack detection in Algorithm 2 is executed on the on-disk provenance subgraph immediately after it is dumped from the memory. For each call to the algorithm, we measure its execution time. For further in-depth analysis, we measure the size of the graph file dumped to the disk, as well as the numbers of nodes and edges in the provenance subgraph stored in it. To the reduce the effects of random noise, we repeat the experiment ten times to obtain the mean as well as the standard deviation.

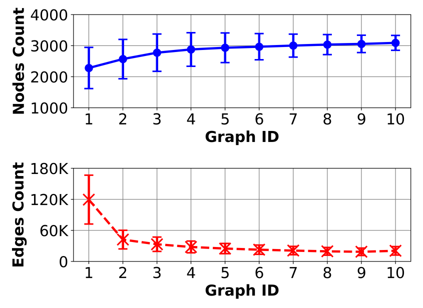
In all the experiments the three types of slicing attacks have always been detected by Algorithm 2. The execution performance results are summarized in Figure 6. From Figure 6(1), we observe that the number of nodes in each provenance subgraph grows over time until it stabilizes in the fifth one, while the number of edges in each provenance subgraph peaks in the first epoch and gradually decreases until the fifth provenance subgraph. The number of edges is affected by the 5G core network events. The initial spike is caused by the exponential waiting times of UEs whose mean is 500 seconds. The probablity density function of the exponential distribution $p(x)$ monotonically decreases with $x$, suggesting that a large fraction of UEs starts their registration procedures within the first 600 seconds. By contrast, the majority of the nodes are registered UEs in the network. Hence, unless a UE is deregistered from the network, it always has a corresponding node in the provenance subgraph.

Figure 6(2) shows that the storage size is highly correlated with the number of edges in each provenance subgraph. This is unsurprising because the number of edges dominates over the number of nodes by $3 \sim 50$ times. Similarly, Figure 6(3) reveals that the execution time of slicing attack detection is also dominated by the number of edges in each graph. This is because the detection algorithm needs to traverse every edge to find unique patterns of three types of slicing attacks (see Algorithm 2). Importantly, less than one second is needed to detect slicing attacks in each epoch, which lasts 600 seconds, suggesting that the execution performance of slicing attack detection is feasible for real-world deployments.

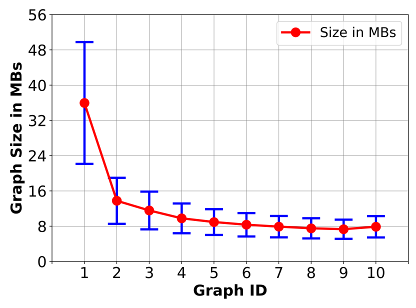
## 7.4 PFCP attack detection

We use Machine 1 to simulate 10,000 UEs. Each experiment runs for an hour. For background traffic generation, each UE behaves similarly as those to evaluate signaling storm attack detection and slicing attack detection. Before the initial registration request, each UE waits for a duration following an exponential distribution with a mean of 1000 seconds. The provenance graphs are dumped from memory every 600 seconds. During each 600-second interval, the following PFCP attack is simulated: the attacker sends a precrafted UDP packet with a random session ID to the UPF at a time randomly chosen within the interval. Algorithm 3 is executed after a current provenance subgraph is dumped onto the disk and the pending queue $Q_{uc}$ is checked to clean up uncommitted communication messages. We repeat the experiment 10 times.

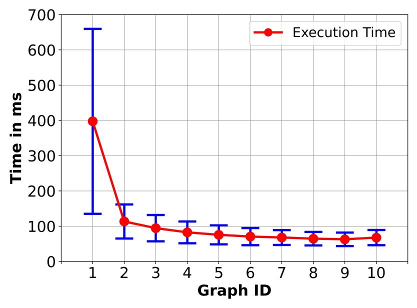
In all our experiments, the PFCP attacks are detected perfectly without any false alarms. This is expected as our detection algorithm shown in Algorithm 3 leverages the unique signature of PFCP attacks for their detection. The execution time used by the



(1) Numbers of nodes and edges



(2) Graph storage size



(3) Execution time

**Figure 6: Execution performances of slicing attack detection. Error bars show standard deviations.**

`Detect_PFCP_Attacks` method is short with its mean varying between 0.08 and 0.14 milliseconds. This is because its time complexity is linear with the number of uncommitted messages in queue $Q_{uc}$, whose mean varies between only 22 and 24 in our experiments.

## 8 RELATED WORK

**Cellular network security.** There have been many efforts on cellular network security, leading to important findings about their security vulnerabilities, such as denial-of-service attacks [27, 28, 41, 52], man-in-the-middle attacks [24, 51], linkability and traceability attacks [19, 33, 42, 51], and data charging bypass attacks [48]. There have been several recent works aimed at finding security vulnerabilities of 5G networks. Basin *et al.* discovered vulnerabilities 5G AKA protocols due to lack of integrity protection on the serving network's identifier [20]. Cremers and Dehnel-Wild used a fine-grained component-based formal analysis of the 5G AKA protocol to find possible attacks of impersonating an honest user to a 5G network [26]. 5GReasoner uses model checkers and a cryptographic protocol verifier to find possible attacks either between the mobile device and the base station or between the device and the core network [36]. Model checkers were also used to reveal 5G core networks' access control mechanisms [17]. Bitsikas *et al.* developed a new framework based on open5GS [6] and srsRAN [9] to evaluate the security vulnerabilities of 5G standalone UEs [22]. Yang *et al.* proposed a protocol-independent fuzzing system to identify 5G RRC vulnerabilities [56]. While these previous works aim to identify potential vulnerabilities in 5G networks, our work focuses on detection and attribution of attacks targeting 5G core networks.

**Attack detection and attribution based on provenance graphs.** For host-based intrusion detection, provenance graphs can be extracted from system event logs collected by either the built-in audit systems of modern operating systems (e.g., Linux Auditd and Windows ETW [44]) or third-party event collectors [30, 46]. To trace sophisticated cyber attacks such as advanced persistent threats (APTs) within an opaque computer system, researchers have been developing various fine-grained system-level provenance graphs methods [21, 34, 35, 38, 43, 47, 53]. Provenance graphs have also been used to detect or explain abnormal behaviors in various networked and/or distributed systems [32, 54, 57]. Although our work has been motivated by these previous efforts, it aims to harden the security of 5G core networks based on provenance graphs.

## 9 CONCLUSIONS AND DISCUSSIONS

In this work, we have proposed the PROV5GC framework for attack detection and attribution within 5G core networks based on provenance graphs. We have identified three key technical challenges in applying existing IDSes for attack detection and attribution in 5G core networks. We have discussed the architecture and workflow of PROV5GC and presented the algorithms to detect three 5G-specific attacks within this framework. We have implemented a prototype of PROV5GC based on an existing 5G network security testbed and evaluated both its logging overhead and its detection performances for the three 5G-specific attack scenarios.

In our future work we plan to address the following limitations of PROV5GC. First, PROV5GC requires cooperative logging efforts by various NFs in a 5G core network to catch abnormal behaviors. If both sender and receiver NFs are compromised and they collude to manipulate the logged messages transmitted between them, PROV5GC is incapable of detecting such attacks. Also, the NAS messages transmitted between the UEs and the AMFs are logged only by the AMFs in PROV5GC as it is infeasible to require individual UEs to log these messages. Hence, if a compromised AMF manipulates the logged NAS messages, such behavior cannot be detected by PROV5GC.

Second, we have built a prototype of PROV5GC and evaluated its performances based on the VET5G testbed [55]. Although we have validated its implementations of 5G core NFs using different UE/RAN configurations, it lacks some features that exist in real-world 5G core networks. Therefore, the relative logging overheads seen in our experiments may differ from those in a real-world deployment. Moreover, to simulate a large number of UEs in our experiments, we used the UERAMSIM tool [10], which ignores the low-level protocol layers between the UEs and the gNBs. Given that SMS signaling storm attacks can also affect the RANs, such attack effects were not taken into consideration in our experiments.

Third, PROV5GC requires instrumentation of all NFs in a 5G core network to log its communications messages. Both logging and log transmissions incur extra workloads to the NFs.

Last but not least, this work considered only three 5G-specific attacks. We plan to extend PROV5GC to detect other attacks against 5G core [17, 20, 26] as well as those targeting their operational environments (e.g., the cloud where the 5G core network runs).

## REFERENCES

[1] https://www.p1sec.com/corp/2021/12/31/pentesting-5g-core-networks/.
[2] https://www.softeq.com/blog/how-to-ensure-5g-supply-chain-security.
[3] CVE-2021-45462. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-45462.
[4] CVE-2022-43677. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-43677.
[5] Docker Stats: Endpoint for live stream of a container's resource usage statistics. https://docs.docker.com/engine/api/v1.41/tag/Container/operation/ContainerStats.
[6] open5gs. https://www.open5gs.org/.
[7] Openairinterface. https://www.openairinterface.org/.
[8] Snort. https://www.snort.org/.
[9] srsRAN - Your own mobile network. https://www.srsran.com/.
[10] UERANSIM: An Open Source State-of-the-Art 5G UE and RAN (gNodeB) Simulator . https://github.com/aligungr/UERANSIM.
[11] Zeek. https://zeek.org/.
[12] ZeroMQ: An open-source universal messaging library. https://zeromq.org.
[13] 3GPP. 5G; 5G System; Common Data Types for Service Based Interfaces; Stage 3 (3GPP TS 29.571 version 16.6.0 Release 16). https://www.etsi.org/deliver/etsi_ts/129500_129599/129571/16.06.00_60/ts_129571v160600p.pdf.
[14] 3GPP. 5G; NG-RAN; NG Application Protocol (NGAP) (3GPP TS 38.413 version 16.7.0 Release 16). https://www.etsi.org/deliver/etsi_ts/138400_138499/138413/16.07.00_60/ts_138413v160700p.pdf.
[15] 3GPP. 5G;Security architecture and procedures for 5G System (3GPP TS 33.501 version 16.3.0 Release 16). https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/16.03.00_60/ts_133501v160300p.pdf.
[16] AdaptiveMobile Security. A slice in time: Slicing security in 5G core networks. https://info.adaptivemobile.com/5g-network-slicing-security.
[17] M. Akon, T. Yang, Y. Dong, and S. R. Hussain. Formal analysis of access control mechanism of 5G core network. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 666–680, 2023.
[18] G. Amponis, P. Radoglou-Grammatikis, T. Lagkas, W. Mallouli, A. Cavalli, D. Klonidis, E. Markakis, and P. Sarigiannidis. Threatening the 5G core via PFCP DoS attacks: the case of blocking uav communications. *EURASIP Journal on Wireless Communications and Networking*, 2022(1):1–27, 2022.
[19] M. Arapinis, L. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 205–216, 2012.
[20] 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*, pages 1383–1396, 2018.
[21] A. Bates, D. J. Tian, K. R. Butler, and T. Moyer. Trustworthy whole-system provenance for the Linux kernel. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 319–334, 2015.
[22] E. Bitsikas, S. Khandker, A. Salous, A. Ranganathan, R. Piqueras Jover, and C. Pöpper. Ue security reloaded: Developing a 5G standalone user-side security testing framework. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 121–132, 2023.
[23] E. Brodsky and B. S. Darkhovsky. *Nonparametric methods in change point problems*, volume 243. Springer Science & Business Media, 1993.
[24] M. Chlosta, D. Rupprecht, T. Holz, and C. Pöpper. LTE security disabled: misconfiguration in commercial networks. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile networks*, pages 261–266, 2019.
[25] E. Cozzi, P.-A. Vervier, M. Dell'Amico, Y. Shen, L. Bilge, and D. Balzarotti. The tangled genealogy of IoT malware. In *Proceedings of Annual Computer Security Applications Conference*, pages 1–16, 2020.
[26] C. Cremers and M. Dehnel-Wild. Component-based formal analysis of 5G-AKA: channel assumptions and session confusion. In *Proceedings of Network and Distributed System Security Symposium*. Internet Society, 2019.
[27] W. Enck, P. Traynor, P. McDaniel, and T. La Porta. Exploiting open functionality in SMS-capable cellular networks. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 393–404, 2005.
[28] K. Fang and G. Yan. Paging storm attacks against 4G/LTE networks from regional Android botnets: rationale, practicality, and implications. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 295–305, 2020.
[29] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14, 2011.
[30] A. Gehani and D. Tariq. SPADE: Support for provenance auditing in distributed environments. In *Proceedings of the 13th ACM/IFIP/USENIX International Middleware Conference*, pages 101–120. Springer, 2012.
[31] L. M. Hall and J. P. French. A modified CUSUM test to control postoutbreak false alarms. *Statistics in Medicine*, 38(11):2047–2058, 2019.
[32] W. U. Hassan, L. Aguse, N. Aguse, A. Bates, and T. Moyer. Towards scalable cluster auditing through grammatical inference over provenance graphs. In *Network and Distributed Systems Security Symposium*, 2018.
[33] B. Hong, S. Bae, and Y. Kim. GUTI reallocation demystified: Cellular location tracking with changing temporary identifier. In *Proceedings of Network and Distributed System Security Symposium*, 2018.
[34] M. N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. D. Stoller, and V. Venkatakrishnan. SLEUTH: Real-time attack scenario reconstruction from COTS audit data. In *Proceedings of USENIX Security Symposium*, pages 487–504, 2017.
[35] M. N. Hossain, S. Sheikhi, and R. Sekar. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In *IEEE Symposium on Security and Privacy*, pages 1139–1155. IEEE, 2020.
[36] 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*, pages 669–684, 2019.
[37] M. A. Inam, Y. Chen, A. Goyal, J. Liu, J. Mink, N. Michael, S. Gaur, A. Bates, and W. U. Hassan. SoK: History is a vast early warning system: Auditing the provenance of system intrusions. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 307–325. IEEE Computer Society, 2023.
[38] H. Irshad, G. Ciocarlie, A. Gehani, V. Yegneswaran, K. H. Lee, J. Patel, S. Jha, Y. Kwon, D. Xu, and X. Zhang. Trace: Enterprise-wide provenance tracking for real-time APT detection. *IEEE Transactions on Information Forensics and Security*, 16:4363–4376, 2021.
[39] C. Jost and B. Smeets. Security for 5G service-based architecture: What you need to know. https://www.ericsson.com/en/blog/2020/8/security-for-5g-service-based-architecture, 2020.
[40] R. P. Jover and V. Marojevic. Security and protocol exploit analysis of the 5G specifications. *IEEE Access*, 7:24956–24963, 2019.
[41] H. Kim, J. Lee, E. Lee, and Y. Kim. Touching the untouchables: Dynamic security analysis of the LTE control plane. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 1153–1168. IEEE, 2019.
[42] K. Kohls, D. Rupprecht, T. Holz, and C. Pöpper. Lost traffic encryption: fingerprinting LTE/4G traffic on layer two. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 249–260, 2019.
[43] S. Ma, X. Zhang, and D. Xu. ProTracer: Towards practical provenance tracing by alternating between logging and tainting. In *Proceedings of Network and Distributed System Security Symposium*, 2016.
[44] Microsoft. Event Tracing for Windows (ETW). https://learn.microsoft.com/en-us/windows-hardware/drivers/devtest/event-tracing-for-windows--etw-, 2021.
[45] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network time protocol version 4: Protocol and algorithms specification. https://datatracker.ietf.org/doc/html/draft-ietf-ntp-ntpv4-algorithms-01, 2010.
[46] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 405–418, 2017.
[47] T. Pasquier, X. Han, T. Moyer, A. Bates, O. Hermant, D. Eyers, J. Bacon, and M. Seltzer. Runtime analysis of whole-system provenance. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 1601–1616, 2018.
[48] C. Peng, C.-Y. Li, H. Wang, G.-H. Tu, and S. Lu. Real threats to your data bills: Security loopholes and defenses in mobile data charging. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 727–738, 2014.
[49] Positive Technologies. 5G standardalone core security research. https://positive-tech.com/knowledge-base/research/5g-sa-core-security-research/.
[50] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan. *5G Core Networks: Powering Digitalization*. Academic Press, 2019.
[51] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Breaking LTE on layer two. In *Proceedings of IEEE Symposium on Security and Privacy*. IEEE, 2019.
[52] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert. Practical attacks against privacy and availability in 4G/LTE mobile communication systems. *Proceedings of the Network and Distributed System Security Symposium*, 2015.
[53] X. Shu, F. Araujo, D. L. Schales, M. P. Stoecklin, J. Jang, H. Huang, and J. R. Rao. Threat intelligence computing. In *Proceedings of ACM SIGSAC conference on computer and communications security*, pages 1883–1898, 2018.
[54] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter. Fear and logging in the Internet of things. In *Proceedings of Network and Distributed Systems Symposium*, 2018.
[55] Z. Wen, H. S. Pacherkar, and G. Yan. VET5G: A virtual end-to-end testbed for 5G network security experimentation. In *Proceedings of the Workshop on Cyber Security Experimentation and Test (CSET'22)*, 2022.
[56] J. Yang, Y. Wang, T. X. Tran, and Y. Pan. 5G RRC protocol and stack vulnerabilities detection via listen-and-learn. In *Proceedings of the IEEE Consumer Communications & Networking Conference*, pages 236–241. IEEE, 2023.
[57] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr. Secure network provenance. In *Proceedings of ACM Symposium on Operating Systems Principles*, pages 295–310, 2011.
[58] M. Zipperle, F. Gottwalt, E. Chang, and T. Dillon. Provenance-based intrusion detection systems: A survey. *ACM Computing Surveys*, 55(7):1–36, 2022.