A Framework for Detecting and Analyzing DoS and Slicing Attacks in 5G Core Networks

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Anand Keshav
(210101014)

under the guidance of

Moumita Patra

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**A Framework for Detecting and Analyzing DoS and Slicing Attacks in 5G Core Networks**" is a bonafide work of **Anand Keshav (Roll No. 210101014)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Moumita Patra**

Assistant/Associate

Professor,

May, 2025 — Department of Computer Science & Engineering,

Guwahati. — Indian Institute of Technology Guwahati, Assam.

Acknowledgements

I acknowledge the kind guidnace and constant encouragement of **Prof. Moumita Patra**. Her insightful feedback and expertise were instrumental in shaping the direction and execution of my work. I am also deeply thankful to my mentor, **Meenu Rani Dey**, for her consistent support and constructive discussions during our weekly meetings.

Contents

List of Figures **v**

List of Tables **vi**

1 Introduction **1**

1.1 Service-Based Architecture (SBA) Overview 1

1.2 Network Slicing in 5G Core 2

1.3 The Critical Role of NRF in SBA 3

1.4 Motivation 3

1.5 Organization of The Report 4

2 Review of Prior Works **5**

2.1 Limitations of Prior Works 7

3 Threat Model and Attack Scenarios **9**

3.1 Slicing Attack Scenario 9

3.2 NRF-Targeted Attacks and Their Impact on Network Slicing 9

3.2.1 Resource Exhaustion Attacks 10

3.2.2 Replay Attacks 11

4 Attack Simulation and Results: Phase 1 **13**

4.1 Slicing Attack Scenario 13

4.1.1 Execution Steps 14
4.2 NRF-Targeted Attack Scenario 16
4.2.1 Execution Steps 16
4.2.2 Mitigation Strategy: Rate Limiting for Flooding Attacks 18

5 Phase 2: Methodology and Implementation 19
5.1 Motivation for Framework Selection 19
5.2 Logging Enhancements in Free5GC 20
5.3 Simulation Environment and Scenarios 21
5.4 Log Parsing and CSV Generation 22
5.5 Provenance Graph Construction 23
5.6 Anomaly Detection 23
5.6.1 Training on Benign Graphs 23
5.6.2 Execution on All Scenarios 25
5.6.3 Results on Attack Scenarios 26

6 Detection Framework and Results 27
6.1 GraphSAGE Architecture 27
6.1.1 Provenance Graph Formalization 27
6.1.2 Forward Propagation (FP) 28
6.2 Feature Extraction for Inter-NF Communications 28
6.3 Multi-Model Training Algorithm for NF Provenance Graphs 29
6.3.1 Anomaly Detection Using Multi-Model Framework 31
6.3.2 Normalized Entropy-Based Anomaly Score 32
6.4 Results and Evaluation 35
6.4.1 System Specifications and Simulation Setup 35
6.4.2 Performance Overhead of Enhanced Logging 37

7 Conclusion and Future Work 39

References **41**

List of Figures

1.1 5G Service Based Architecture [Cal24] 2

3.1 Illustration of slicing attack [PY24] 10

3.2 NRF-based registration and discovery 11

3.3 Illustration of the NRF targeted DoS attack 12

4.1 Free5gc logs under slicing attack 15

4.2 Spike in CPU resources usage during the attack simulation. 17

5.1 Provenance Graph capturing NF registration and GnB setup 24

6.1 Anomaly scores of various NFs under DoS attack 34

6.2 Anomaly scores of various NFs under slicing attack 34

6.3 Performance comparison of different models across various metrics 36

6.4 CPU usage comparison: With vs Without Logging 38

6.5 Memory usage comparison: With vs Without Logging 38

List of Tables

4.1 Comparison of System Metrics Before and During Flooding Attack Simulation 16

6.1 Comparison of detection performance across different models 36

Abstract

The integration of 5G networks in critical infrastructure has highlighted their vulnerabilities to emerging cyber threats. This work focuses on simulating, analyzing and mitigating Denial-of-Service (DoS) and slicing attacks in the 5G core network, specifically targeting the Network Repository Function (NRF) and inter-slice communications respectively. Using Free5GC and UERANSIM, we demonstrate resource exhaustion and unauthorized access to user data across slices. To detect such anomalous behavior, we construct provenance graphs from enhanced inter-NF logging and apply a multi-model GraphSAGE-based anomaly detection framework. Trained on benign interactions, the system generalizes well to unseen attacks, achieving high accuracy and interpretability through confidence and entropy-based scoring. The approach shows strong potential for zero-day threat detection in 5G service-based architectures.

Chapter 1

Introduction

As 5G networks become a crucial part of modern critical infrastructure, ensuring their reliability and security is vital. Any disruptions or vulnerabilities in these networks can lead to significant societal consequences, ranging from economic instability to threats to public safety. Unlike earlier generations, 5G utilizes a Service-Based Architecture (SBA), which allows network functions to interact in a flexible and on-demand manner, improving scalability and efficiency. The introduction of network slicing and the essential role of the Network Repository Function (NRF) for service discovery also bring unique vulnerabilities that need to be addressed to maintain strong 5G security.

1.1 Service-Based Architecture (SBA) Overview

The SBA in 5G represents a groundbreaking design that enables flexible and scalable interactions among Network Functions (NFs). In contrast to traditional monolithic architectures, SBA specifies NFs that can communicate through a shared framework using HTTP/2-based interfaces. This shift in architecture promotes virtualization, modularity, and a service-oriented approach to network design, allowing individual NFs to scale independently. However, the SBA also presents new security challenges, as these NFs are more exposed and communicate over standardized interfaces, which heightens the risk of unauthorized access and service interruptions.

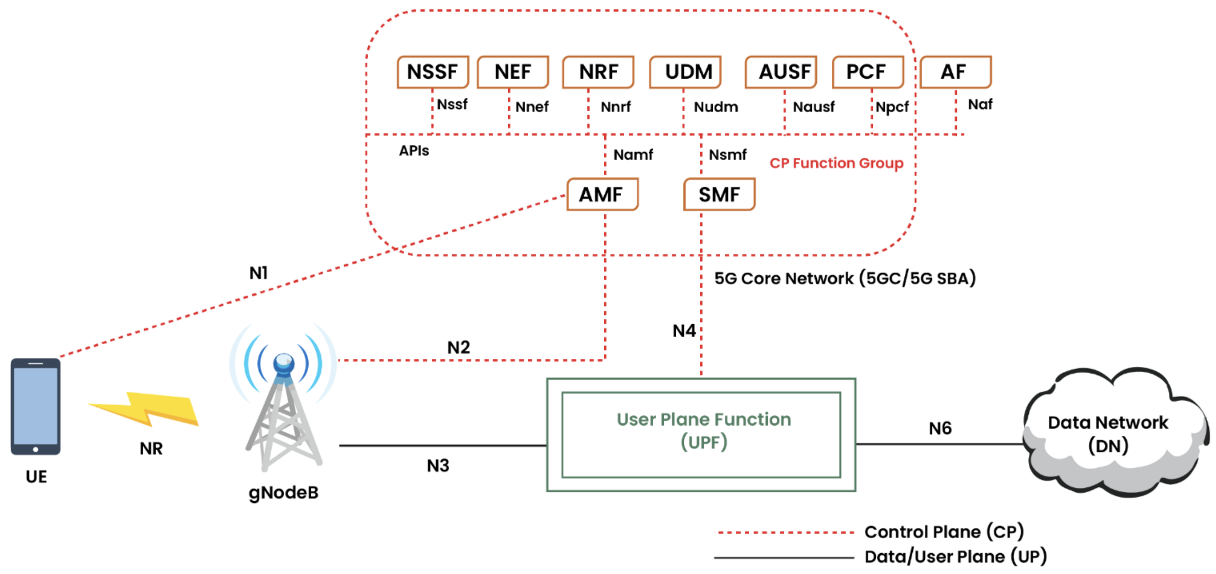


Fig. 1.1 5G Service Based Architecture [Cal24]

1.2 Network Slicing in 5G Core

Network slicing empowers 5G operators to establish multiple virtual networks or **slices** on a shared physical network infrastructure. Each slice is tailored to meet the requirements of specific applications or end users. While this approach offers enhanced security by segregating purposes across slices, any misuse of access privileges by unauthorized entities can jeopardize sensitive internal data and critical functionalities. A **Slicing Attack** occurs when a malicious actor exploits vulnerabilities in slice management or access control mechanisms to gain unauthorized access to resources or data intended for another slice. Consequently, efficient slice management with robust access controls is crucial to safeguarding the integrity of essential 5G services and preserving user privacy.In this work, we explore and simulate a slicing attack scenario where an adversary exploits these vulnerabilities to access sensitive user data, highlighting the importance of secure slice governance.

1.3 The Critical Role of NRF in SBA

The NRF plays a pivotal role in the 5G SBA by acting as the central directory for all Network Functions. It enables seamless NF discovery and interaction, making it a cornerstone of the network slicing framework. However, the NRF's criticality also makes it a prime target for cyberattacks. Frequent registration and discovery requests create opportunities for DoS and replay attacks, which can deplete NRF resources and disrupt the entire network.

In this work, we focus on assessing the NRF's vulnerability to resource exhaustion and cross-slice data access. Understanding these risks is essential for designing mitigation strategies to ensure the reliability of service discovery and the resilience of the 5G core network.

1.4 Motivation

The advancements in 5G technology have introduced both groundbreaking opportunities and significant security challenges. The motivations for this work are as follows:

1. **Emerging Threat Landscape:** With the adoption of network slicing and the SBA, new attack vectors have surfaced. Vulnerabilities such as cross-slice data leakage and resource exhaustion pose significant risks to user privacy and service continuity. This project seeks to address these challenges by bridging the gap in understanding and mitigating these threats.

2. **Requirement for a Unified Security Framework:** There is a critical need to analyze the impact of various attack scenarios on the 5G core network and subsequently develop a unified framework capable of detecting and mitigating these threats. By focusing on diverse attack types, this work aims to provide a comprehensive approach to securing 5G networks against evolving threats.

1.5 Organization of The Report

This chapter provides an introduction to the 5G core network, covering key concepts such as the SBA, network slicing, and the role of the NRF, while emphasizing the importance of addressing security challenges. Chapter 2 reviews existing research on vulnerabilities in the 5G core, including threats to network slicing and NRF operations, and highlights gaps that motivate this study. Chapter 3 focuses on secure network slicing, offering a detailed exploration of its complexities and presenting the background necessary to understand the attack scenarios. Chapter 4 describes the attack simulations, including execution steps, observed implications, and results, providing insights into the impact of these threats and the efficacy of proposed mitigation strategies. In Chapter 5 the report concludes with a summary of findings and outlines future directions for enhancing the security of 5G networks.

Chapter 2

Review of Prior Works

[GLF$^+$24] In this work, the authors examine and evaluate the research on security threats concerning the design of the 5G network slice. It focuses on aspects such as cross-slice information leakage, resource takeover in the slices, and replay attacks on network function entities like NRF. The survey emphasizes the importance of protective measures like slicing isolation and secure resource provisioning to mitigate risks such as inter-slice attacks and replay attacks.

[PY24] assesses the core network in 5G, identifying weaknesses and attack patterns termed as slicing. These attacks exploit access control mechanisms, enabling attackers to access restricted parts of users' private information or disrupt other slices while avoiding detection. The authors simulated slice and DoS attacks on the RET5G testbed and demonstrated how their framework prevents such attacks through causal communication events in the 5G core network using provenance graphs. For my project, I implemented and simulated the Type-I slicing attack, as defined by Free5GC and UERANSIM, where an NF in one slice correlates sensitive information in other slices.

[BEBP21] focuses on microservices-based architectures and their impact on the 5G NRF, addressing confidentiality, integrity, and availability concerns. This paper presents formulas for assessing these effects and evaluates three decomposed structures for the NRF, along with their security analysis. It aims to identify insufficiencies in the NRF system under attack and propose ways to mitigate such risks.

Free5GC and UERANSIM are open-source applications for simulating, testing, and evaluating 5G core networks. I found these tools useful for simulating both slicing and DoS attacks. The following studies highlight the effectiveness of these tools: [MML$^+$24] examines the control plane performance of Free5GC, noting how it meets 3rd Generation Partnership Project (3GPP) specifications and incorporates network slicing. Benchmarking shows scalability issues, with performance dropping after 300 UEs in high-load conditions. Despite these drawbacks, its extensive features and modular design make it suitable for 5G studies. [CH24] evaluates the performance of Free5GC, Open5GS, and OpenAirInterface (OAI) using UERANSIM for UE registration simulations. Free5GC consumes more CPU resources and takes longer to register users under saturation, indicating higher resource usage.

[PY24] (PROV5GC) proposes a provenance-based detection system for the 5G core, targeting three key attack types: SMS signaling storms, PFCP DoS attacks, and slicing attacks. Although the system effectively detects known attacks by observing communication messages at NF endpoints, it relies heavily on predefined detection strategies for each threat type. This reduces its ability to detect zero-day or stealthy attacks unless explicitly modeled.

[ZLZ$^+$23] (MAGIC) addresses key shortcomings of prior APT detection approaches by employing self-supervised masked graph representation learning. Unlike most systems that depend on attack-containing logs or handcrafted rules, MAGIC learns benign patterns and performs outlier detection using graph autoencoders. However, MAGIC primarily focuses on system-level audit logs and process/file-based graphs, which do not directly translate to the NF-level communication patterns in Free5GC-based environments.

[WWZ$^+$22] (THREATRACE) presents a GNN-based threat detection framework that uses a GraphSAGE-based multimodel approach for anomaly detection at the node level.

It avoids prior assumptions about known attack types and trains solely on benign data, enabling detection of unknown or stealthy threats. THREATRACE supports both graph-level and node-level granularity and introduces an online, real-time detection model. Its applicability to provenance graphs generated from Free5GC logs forms the foundation of our detection pipeline.

2.1 Limitations of Prior Works

1. **Limited Focus on Network Function-Level Provenance:** While works like MAGIC and THREATRACE are highly advanced in terms of using system-level audit logs for anomaly detection, they are not tailored to the specific structure and behavior of 5G SBA. These methods model interactions among system processes and files but do not consider the semantics of NF-to-NF communications in the control plane of the 5G core.

2. **Scenario-Specific Detection in PROV5GC:** PROV5GC employs separate detection strategies for different classes of attacks—slicing, PFCP spoofing, and SMS storms—rather than a unified detection mechanism. This makes the framework less flexible in detecting unknown or zero-day attacks, which may not follow a known signature or pattern.

3. **Lack of Unsupervised or Semi-Supervised Adaptability:** Traditional IDS approaches (e.g., Snort, Zeek) and even some provenance-based models rely on labeled datasets or expert-defined rules, making them ineffective for evolving threats and unseen attack patterns. MAGIC addresses this using self-supervised learning, and THREATRACE improves this further by adapting GraphSAGE for role learning across heterogeneous node types.

4. **No Provenance Modeling Based on Free5GC SBI Logs:** All the aforementioned studies rely on host-based system logs or testbed instrumentation but do not

utilize logs extracted from real or simulated 5G core deployments like Free5GC. Our work fills this gap by designing a lightweight logging and extraction mechanism within Free5GC to build an NF-level provenance graph.

5. **Limited Granularity in Detection:** Most prior works either detect anomalies at the graph level (e.g., StreamSpot, Unicorn) or lack the ability to trace specific malicious nodes. THREATRACE overcomes this by performing fine-grained node-level classification, and our work extends this paradigm to NF interactions.

Chapter 3

Threat Model and Attack Scenarios

The slicing attack, illustrated in Fig. 3.1, exploits a vulnerability in the access control mechanisms of network slicing. In this scenario, NF A, which belongs to Slice 2, initiates the attack by sending an `Nnrf_AccessToken_Get` request to the NRF, requesting a token to access the services provided by NF B, specifying Slice Identifier $j = 1$. Since NF B is shared by both Slices 1 and 2, the NRF approves the request and issues a valid token for Slice 1, despite NF A being unauthorized to access it. Using this token, NF A sends a service request to NF B. Upon validating the token, NF B considers the request legitimate and processes the service. It then responds to NF A with the requested service data. This sequence demonstrates how improper validation of slice-specific access permissions allows malicious actors to gain unauthorized access to services and data intended for a different slice, thereby compromising the security and isolation of the network slicing architecture.

3.1 Slicing Attack Scenario

3.2 NRF-Targeted Attacks and Their Impact on Network Slicing

The NRF plays a crucial role in enabling service discovery and registration among network functions within and across 5G network slices. Acting as a central repository, the NRF

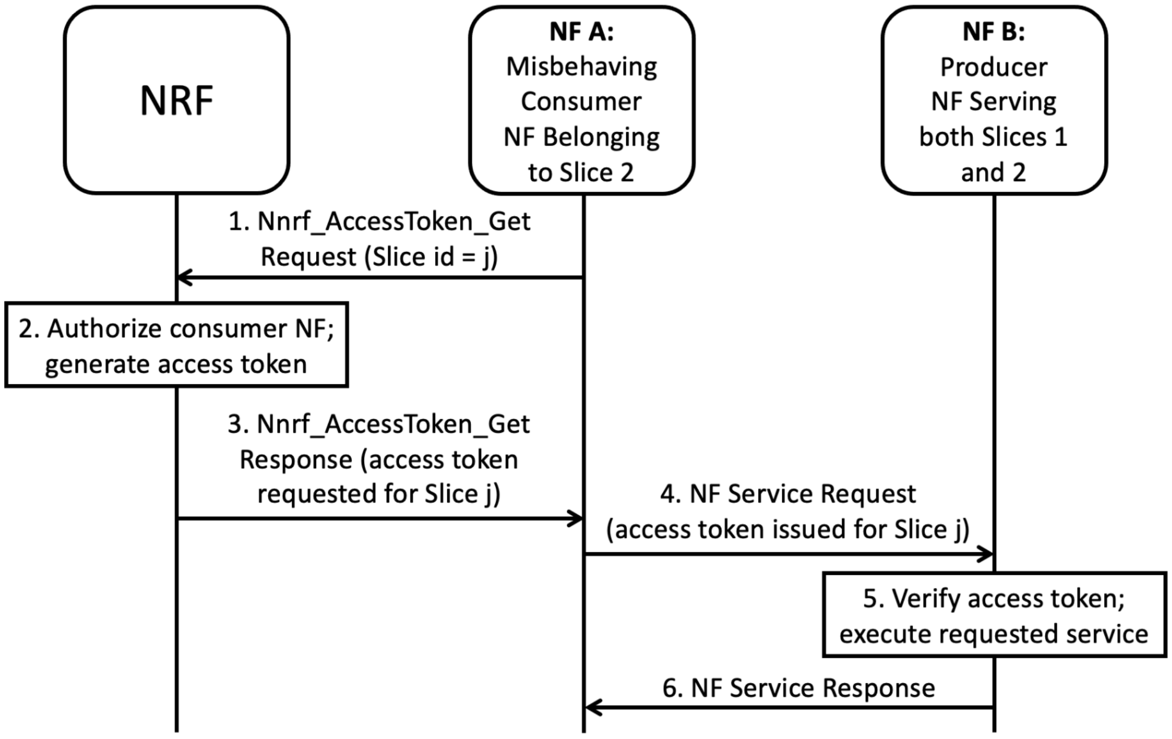

NRF
NF A: Misbehaving Consumer NF Belonging to Slice 2
NF B: Producer NF Serving both Slices 1 and 2
1. Nnrf_AccessToken_Get Request (Slice id = j)
2. Authorize consumer NF; generate access token
3. Nnrf_AccessToken_Get Response (access token requested for Slice j)
4. NF Service Request (access token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

Fig. 3.1 Illustration of slicing attack [PY24]

ensures smooth communication by maintaining an updated directory of available NFs and their capabilities. However, this centrality also makes the NRF a prime target for a variety of attacks. Compromising the NRF not only disrupts service discovery but also poses significant threats to the security and functionality of the entire 5G network slicing architecture. In this work we explore and simulate a NRF targeted attack scenario which is illustrated in the figure 3.3 and explained below.

3.2.1 Resource Exhaustion Attacks

Since, NRF handles the registration and discovery requests of various NFs, the NRF could also be a target of overwhelming malicious traffic, therefore resulting in denial-of-service attacks. This may affect service discovery, inhibiting or even preventing the vital functions and services of slices, which is particularly hard in cases of multi-slice architecture.

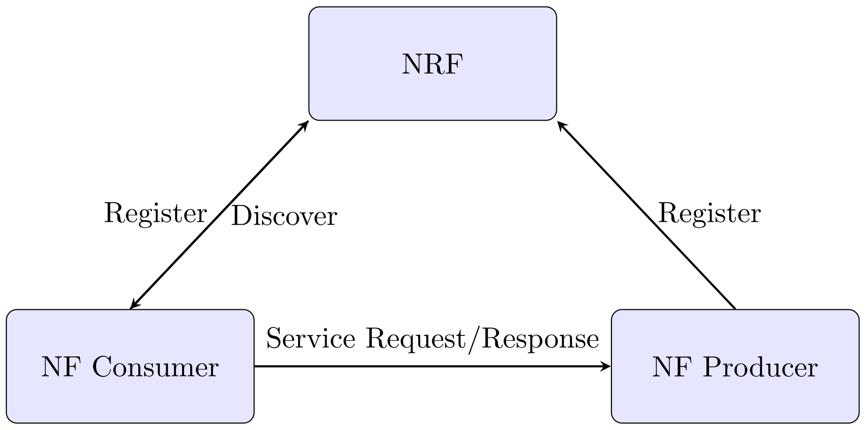


Fig. 3.2 NRF-based registration and discovery

3.2.2 Replay Attacks

Figure 3.3 illustrates how a replay attack on the NRF is carried out. In this scenario, the attacker which is a malicious NF repeatedly sends a discovery request, overwhelming the NRF and disrupting its ability to differentiate between genuine and malicious traffic. Broadcasting repeat requests to the network function repository for registration and discovery of services can lead to the depletion of the available network function repository resources, thus affecting the ability to service all the request across slices.

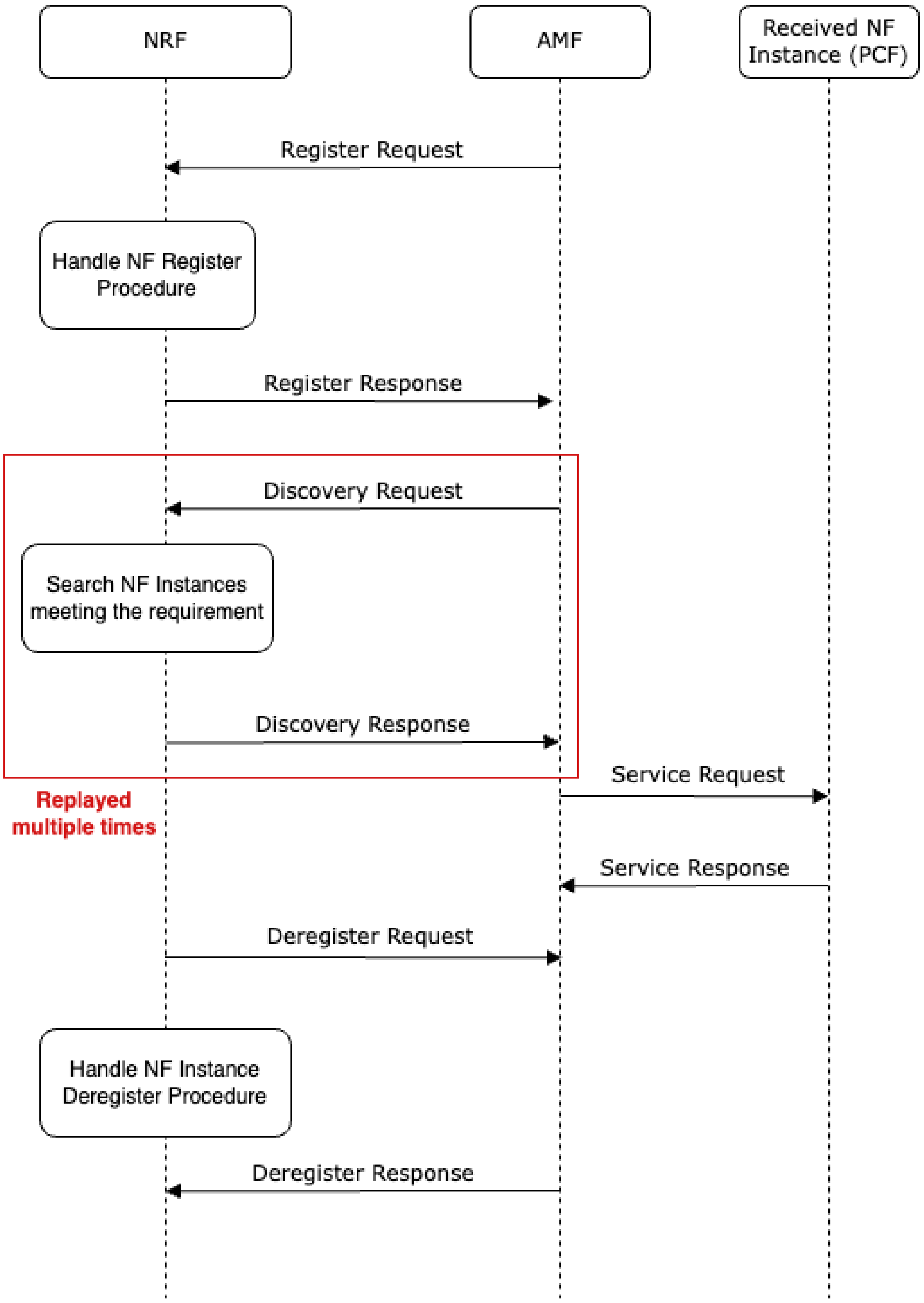


Fig. 3.3 Illustration of the NRF targeted DoS attack

Chapter 4

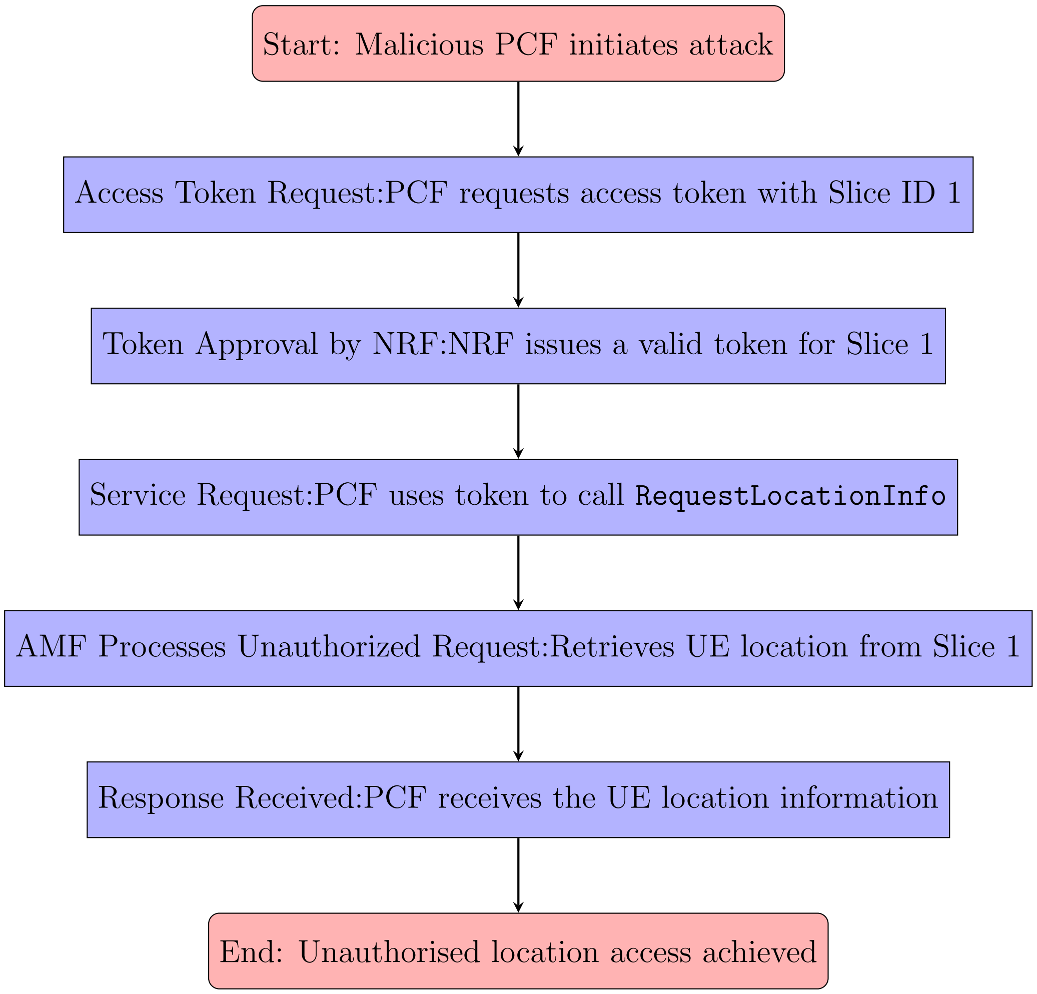
Attack Simulation and Results: Phase 1

The following attacks on 5G core were simulated using free5gc and UERANSIM. Free5gc is a open-source framework which is used to simulate 5G core network efficiently. UERANSIM is used to simulate the various user equipmets under the attack scenario. HTOP and some other linux tools were used to analyse the memory usage, CPU utilisation etc under the attack scenario.

4.1 Slicing Attack Scenario

The purpose of this simulation of blocking an attack between network slices is to confirm the feasibility of unlawful cross-slice access in a multi-slice environment, particularly the cross-slice authorization mechanism. Below configuration and execution steps explain how a misbehaving Network Function (which is PCF in our case) within one slice is able to gain service-based access into unauthorized resources(location information of a user) of another slice.

4.1.1 Execution Steps


Start: Malicious PCF initiates attack
Access Token Request:PCF requests access token with Slice ID 1
Token Approval by NRF:NRF issues a valid token for Slice 1
Service Request:PCF uses token to call RequestLocationInfo
AMF Processes Unauthorized Request:Retrieves UE location from Slice 1
Response Received:PCF receives the UE location information
End: Unauthorised location access achieved

1. **Access Token Request**: The malicious NF A, which is a policy control function (PCF) of slice 2, SD 112233, makes a request to the NRF, through its `Nnrf_AccessToken_Get` function, with slice identifier $j = 1$ which is slice one SD: 010203.

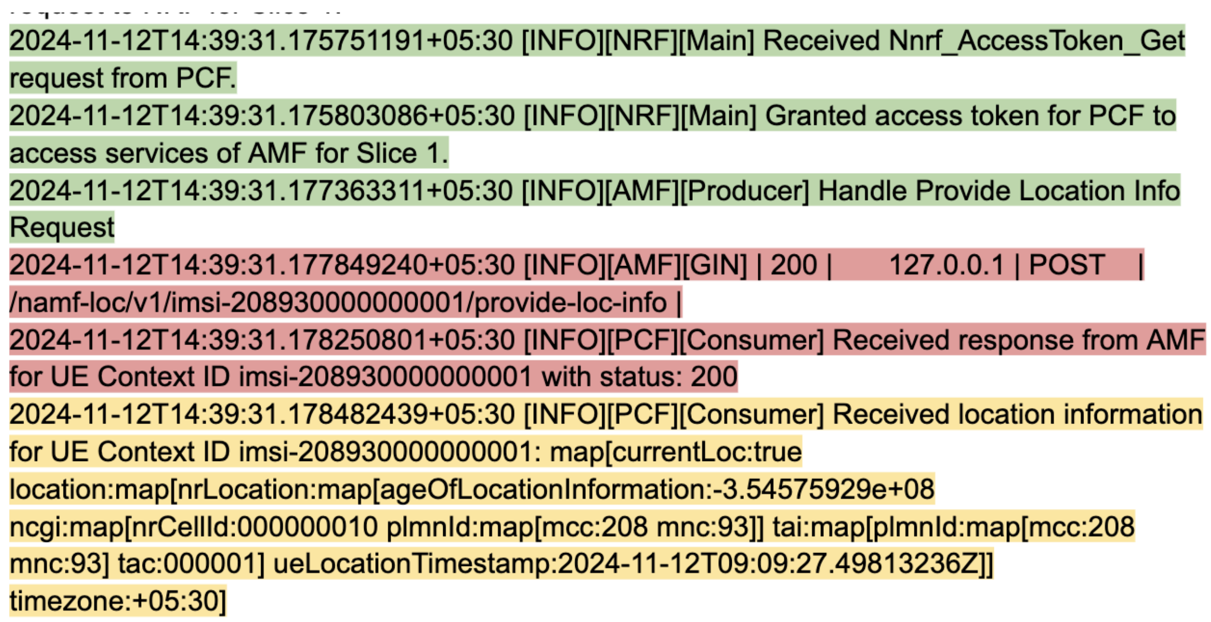
2. **Token Approval in Case of NRF**: In this case, the NF B (represented as an AMF) service is shared with Slices 1 and 2, allowing the crushed token request from NF A. Thus, even for this case the NRF provides a valid access token for Slice 1 that allows NF A (PCF) to reach NF B (AMF).

3. **Service Request Using Access Token**: NF A (PCF) applies the access token and makes a function call `RequestLocationInfo` to NF B (AMF) using an HTTP POST

request. The `amfUri` and `ueContextId` included in the request are for locating the UE whose location is being requested, however, this UE is part of Slice 1 which NF A is not authorized access.

4. **AMF Processes Unauthorized Request**: After receiving a valid token NF B (AMF), it considers the request to be legitimate and processes it, for instance, by retrieving the location information of the mentioned UE in Slice 1. The response is formulated and sent with a 200 OK status, thus concluding the service request as if it was authorized.

5. **Response is Received and Processed**: NF A (PCF) obtains and deserializes the location information stored in the AMF and log this information allowing further unauthorized access to certain information residing in Slice 1.



```
2024-11-12T14:39:31.175751191+05:30 [INFO][NRF][Main] Received Nnrf_AccessToken_Get
request from PCF.
2024-11-12T14:39:31.175803086+05:30 [INFO][NRF][Main] Granted access token for PCF to
access services of AMF for Slice 1.
2024-11-12T14:39:31.177363311+05:30 [INFO][AMF][Producer] Handle Provide Location Info
Request
2024-11-12T14:39:31.177849240+05:30 [INFO][AMF][GIN] | 200 |      127.0.0.1 | POST    |
/namf-loc/v1/imsi-208930000000001/provide-loc-info |
2024-11-12T14:39:31.178250801+05:30 [INFO][PCF][Consumer] Received response from AMF
for UE Context ID imsi-208930000000001 with status: 200
2024-11-12T14:39:31.178482439+05:30 [INFO][PCF][Consumer] Received location information
for UE Context ID imsi-208930000000001: map[currentLoc:true
location:map[nrLocation:map[ageOfLocationInformation:-3.54575929e+08
ncgi:map[nrCellId:000000010 plmnId:map[mcc:208 mnc:93]] tai:map[plmnId:map[mcc:208
mnc:93] tac:000001] ueLocationTimestamp:2024-11-12T09:09:27.49813236Z]]
timezone:+05:30]
```

Fig. 4.1 Free5gc logs under slicing attack

This scenario illustrates how NFs can be victims of malicious cross-slice usage in case of insufficiency of data authorization control mechanisms. While demonstrating this cross-slice data leakage, we point out the drawbacks in the fit-for-purpose access control and justify for the implementation of more effective management concerning sub-slices.

4.2 NRF-Targeted Attack Scenario

In this scenario, we simulate a denial-of-service (DoS) attack on the NRF to exploit its role as a critical service directory for Network Function (NF) registration and discovery, ultimately degrading service performance across slices.

4.2.1 Execution Steps

1. **Setup of Attack Environment**: The attack initiates with the AMF sending repetitive registration and discovery requests targeting the NRF. Each registration request uses a unique NF Identifier (NFID), ensuring that the NRF treats each instance as a distinct request to prevent simple filtering of duplicate requests. And the discovery requests replay *SendSearchNFInstances* function with the following parameters NRFUri, PCF(target NF), AMF(requesting NF), param(SearchNFInstancesParamOpts).

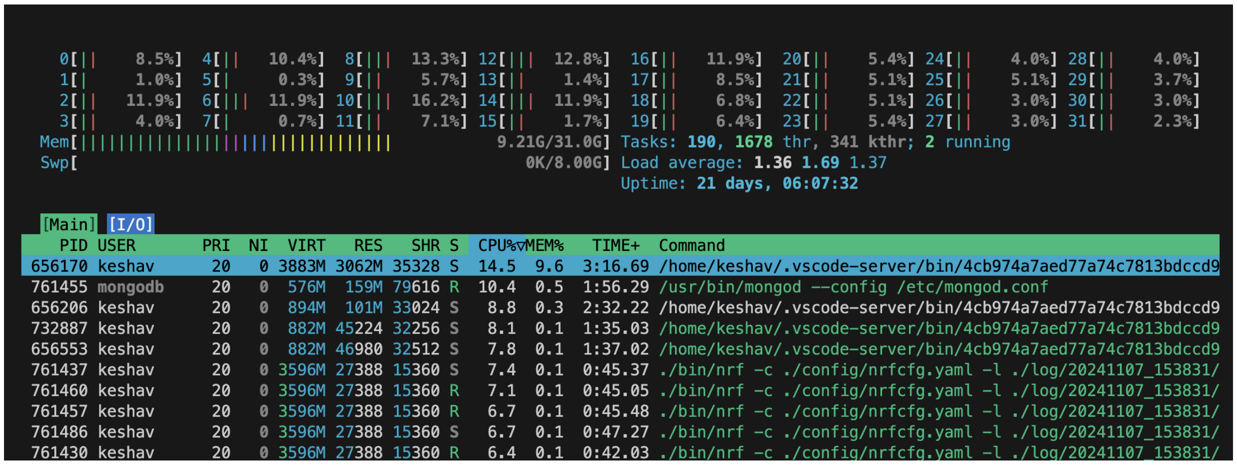
2. **Resource Exhaustion**: These repeated registration and discovery requests to the NRF, consume its memory and processing resources.The flooding of the NRF with these requests leads to delays or outright failures in processing legitimate service discovery and registration requests from other NFs. We observed the following metrics using htop for 100000 replays of discovery requests.

Metric	**Before Attack**	**During Attack**
RAM Usage	5.97 GB	9.21 GB
CPU Util.	1–3%	16%
Load Average	0.22, 0.16, 0.13	1.36, 1.69, 1.37
Active Processes	Light load	Multiple NRF processes

Table 4.1 Comparison of System Metrics Before and During Flooding Attack Simulation

3. **Impact on Service Discovery**: As the attack progresses, the NRF's response times degrade, affecting the discovery process for all connected NFs. This leads to service

interruptions, delays, and potentially failed connections for NFs attempting legitimate service discovery.



```
   0[||     8.5%]  4[||    10.4%]  8[|||   13.3%] 12[|||   12.8%]  16[||    11.9%]  20[||     5.4%] 24[||     4.0%] 28[||     4.0%]
   1[|      1.0%]  5[|      0.3%]  9[||     5.7%] 13[||     1.4%]  17[||     8.5%]  21[||     5.1%] 25[||     5.1%] 29[||     3.7%]
   2[||    11.9%]  6[|||   11.9%] 10[|||   16.2%] 14[|||   11.9%]  18[||     6.8%]  22[||     5.1%] 26[||     3.0%] 30[||     3.0%]
   3[||     4.0%]  7[|      0.7%] 11[||     7.1%] 15[||     1.7%]  19[||     6.4%]  23[||     5.4%] 27[||     3.0%] 31[||     2.3%]
 Mem[|||||||||||||||||||||||||||||||||          9.21G/31.0G] Tasks: 190, 1678 thr, 341 kthr; 2 running
 Swp[                                             0K/8.00G] Load average: 1.36 1.69 1.37
                                                            Uptime: 21 days, 06:07:32

 [Main] [I/O]
    PID USER       PRI  NI  VIRT   RES   SHR S  CPU%▽MEM%   TIME+  Command
 656170 keshav      20   0 3883M 3062M 35328 S  14.5  9.6  3:16.69 /home/keshav/.vscode-server/bin/4cb974a7aed77a74c7813bdccd9
 761455 mongodb     20   0  576M  159M 79616 R  10.4  0.5  1:56.29 /usr/bin/mongod --config /etc/mongod.conf
 656206 keshav      20   0  894M  101M 33024 S   8.8  0.3  2:32.22 /home/keshav/.vscode-server/bin/4cb974a7aed77a74c7813bdccd9
 732887 keshav      20   0  882M 45224 32256 S   8.1  0.1  1:35.03 /home/keshav/.vscode-server/bin/4cb974a7aed77a74c7813bdccd9
 656553 keshav      20   0  882M 46980 32512 S   7.8  0.1  1:37.02 /home/keshav/.vscode-server/bin/4cb974a7aed77a74c7813bdccd9
 761437 keshav      20   0 3596M 27388 15360 S   7.4  0.1  0:45.37 ./bin/nrf -c ./config/nrfcfg.yaml -l ./log/20241107_153831/
 761460 keshav      20   0 3596M 27388 15360 R   7.1  0.1  0:45.05 ./bin/nrf -c ./config/nrfcfg.yaml -l ./log/20241107_153831/
 761457 keshav      20   0 3596M 27388 15360 R   6.7  0.1  0:45.48 ./bin/nrf -c ./config/nrfcfg.yaml -l ./log/20241107_153831/
 761486 keshav      20   0 3596M 27388 15360 S   6.7  0.1  0:47.27 ./bin/nrf -c ./config/nrfcfg.yaml -l ./log/20241107_153831/
 761430 keshav      20   0 3596M 27388 15360 R   6.4  0.1  0:42.03 ./bin/nrf -c ./config/nrfcfg.yaml -l ./log/20241107_153831/
```

Fig. 4.2 Spike in CPU resources usage during the attack simulation.

4. **Denial of Service and Performance Degradation**: The sustained flood of replayed and varied requests ultimately overburdens the NRF, causing service slowdowns and affecting overall network performance. The simulation highlights the vulnerability of NRF to resource exhaustion attacks and showcases the need for robust rate-limiting and detection mechanisms.

4.2.2 Mitigation Strategy: Rate Limiting for Flooding Attacks

The following algorithm limits the discovery requests each NF type can make within a specified time window, thus protecting the NRF from being overwhelmed by flooding attacks.

Algorithm 1: Check if a NF discovery request has exceeded the rate limit

```
1  Function isNFDiscoveryRateLimited($nfType$):
       Data: nfType (string), requests (map of string to slice of time)
       Result: True if rate limited, False otherwise
2      nfDiscoveryRateLimiter.Lock();
3      nfDiscoveryRateLimiter.Unlock();
4      now ← time.Now();
5      timeWindow ← time.Minute;
6      maxRequests ← 20;
7      if $nfDiscoveryRateLimiter.requests$ $exists$ then
8          validRequests ← []time.Time;
9          for $t$ in $nfDiscoveryRateLimiter.requests[nfType]$ do
10             if $now - t < timeWindow$ then
11                 validRequests.append($t$);
12         nfDiscoveryRateLimiter.requests[$nfType$] ← validRequests;
13     if $len(nfDiscoveryRateLimiter.requests[nfType]) \geq maxRequests$ then
14         logger.DiscLog.Warnf("Discovery rate limit exceeded for NF type
            %s:  too many requests", nfType);
15         return $True$;
16     nfDiscoveryRateLimiter.requests[$nfType$] ←
        nfDiscoveryRateLimiter.requests[$nfType$].append(now);
17     return $False$;
```

Chapter 5

Phase 2: Methodology and Implementation

5.1 Motivation for Framework Selection

While prior works have demonstrated promising results in using provenance graphs for threat detection, most approaches—including [ZLZ+23] and [WWZ+22]—are designed for host-based Advanced Persistent Threat (APT) detection and operate on system-level audit logs. These graphs model file accesses, process interactions, and system calls within a single host, which do not translate directly to NF interactions within a 5G core. As a result, such methods are unsuitable for modeling the dynamic, inter-service communication patterns that occur in real-world 5G SBA deployments.

Moreover, PROV5GC [PY24] proposes a provenance-based security framework tailored for 5G networks, but it primarily focuses on detecting known attack types like PFCP DoS, SMS storms, and specific slicing attacks through separate detection logic. It does not support zero-day or stealthy anomaly detection in a unified manner. The reliance on manually crafted rules or attack-specific algorithms limits its adaptability to evolving threats.

To address these limitations, we adopt a Graph Neural Network (GNN)-based anomaly detection framework inspired by THREATRACE. Unlike prior systems, our framework:

- **Leverages Free5GC logs**: We enhance the Free5GC logging infrastructure to capture detailed SBI communications, enabling us to build provenance graphs that represent actual inter-NF behavior in simulated 5G core deployments.
- **Supports zero-day detection**: By training solely on benign graph data, our approach generalizes to detect deviations without requiring prior knowledge of attack signatures or behaviors.
- **Performs node-level classification**: We identify the exact NF instance or UE session responsible for anomalous behavior, supporting fine-grained tracing and analysis.

This motivation guided the architectural and implementation decisions outlined in the following sections.

5.2 Logging Enhancements in Free5GC

- **Motivation:** Ensure every Service-Based Interface (SBI) exchange among NFs is captured for end-to-end provenance.
- **Changes made:**
  1. Unified log format across AMF, SMF, PCF, NRF, UPF modules.
  2. Added request and response identifiers (timestamp, source NF, destination NF, category) to each log entry.
  3. Modified key SBI function calls in both producer and consumer NFs so that each side logs every request and response pair, enabling cross-correlation of logs between interacting NFs.

5.3 Simulation Environment and Scenarios

- **Tools:** Free5GC (core network), UERANSIM gNodeB & UE (radio access), Linux `htop` for resource monitoring.
- **Scenarios:**
  1. *Benign baseline.*
     - **NF registrations:** AMF, SMF, PCF, AUSF and other NFs each register themselves with the NRF via `Nnrf_NFRegister` calls.
     - **SCTP NGSetup:** The gNodeB establishes an SCTP association with the AMF, then issues `NGSetup` to negotiate service capabilities.
     - **UE attach:** UEs perform NAS `RegistrationRequest` to the AMF (via UERANSIM), triggering Authentication (`Nausf_UEAuthenticationGet`), Security Mode Command, and finally `RegistrationAccept`.
     - **Session establishment:** AMF selects SMF (`Nnrf_NFDiscovery`) and sends `PduSessionCreate` (via `Nsmf_PDUSession` APIs), followed by UPF setup.
     - **Service requests:** Regular policy (PCF) and charging (CHF) lookups via `Nnrf_NFDiscovery` and `Nnccir_PolicyControlRequest`.
  2. *Slicing attack.*
     - **Cross-slice token misuse:** A malicious PCF in Slice 2 sends `Nnrf_AccessToken_Get` to the NRF, requesting a token for Slice 1's PCF services.
     - **Token approval:** Because the target AMF instance is shared across slices, the NRF (mis)issues a valid Slice 1 token back to the attacker PCF.
     - **Unauthorized location request:** Attacker PCF uses the token in an HTTP POST `RequestLocationInfo` to the AMF, specifying a UE context belonging to Slice 1.

    - **Data leakage:** AMF processes the request, returns the UE's location in `LocationInfoResponse`, which the malicious PCF logs and correlates.

3. *NRF DoS.*
    - **Flooding discovery:** A script replays `Nnrf_SearchNFInstances` calls at high rate, each time using a unique fake NF-ID so the NRF cannot dedupe requests.
    - **Resource exhaustion:** Memory and CPU usage spikes as the NRF allocates new registration contexts and processes discovery.
    - **Service disruption:** Legitimate NF discovery (`Nnrf_SearchNFInstances`) and NF registration (`Nnrf_NFRegister`) begin to timeout or are rejected, impairing slice operation.

- **Data collection:**
    - System metrics via `htop`: CPU utilization, memory footprint, load average.
    - Free5GC logs: SBI request/response entries instrumented per Section 5.1.
    - Volume: ∼10 000 total SBI calls per scenario.

5.4 Log Parsing and CSV Generation

- **Parser script:**
    1. Reads raw Free5GC logs.
    2. Filters and extracts all relevant SBI interactions, including inter-NF communications (e.g., entries with `Nnrf_`, `Nsmf_`) as well as UE-to-NF signaling exchanges captured in the logs.
    3. Extracts columns: Timestamp, source NF, target NF, category(consumer/producer), message data.

- **Output:** CSV files where each row is an NF-to-NF communication event, ready for provenance graph ingestion.

5.5 Provenance Graph Construction

- **Data source:** CSV from previous stage.
- **Tool:** Python + NetworkX.
- **Node types:** NF instances, UE contexts, GnB.
- **Edge types:** `NFRegister`, `NFDiscover`, `AccessToken`, `UEAuthentication`, `PDUSessionCreate`, `RequestLocationInfo`, `SessionModification`, `SessionRelease`
- **Result:** Directed provenance graph per scenario. Figure 5.1 shows the provenance graph depicting interactions during the registration of various NFs and the gNB. Each edge contains the message data, timestamp, and category (Consumer/Producer).

5.6 Anomaly Detection

5.6.1 Training on Benign Graphs

The anomaly detection framework is based on the THREATRACE architecture, which leverages Graph Neural Networks (GNNs) to learn behavioral embeddings of nodes from provenance graphs. In our case, the graph is constructed from SBI-level inter-NF and UE-to-NF interactions recorded during Free5GC simulations.

1. **Feature Extraction:** For each node in the provenance graph (representing an NF or UE context), a fixed-dimensional feature vector is computed. This vector captures the count of incoming and outgoing edges grouped by communication type. This effectively encodes the interaction pattern of each node and allows

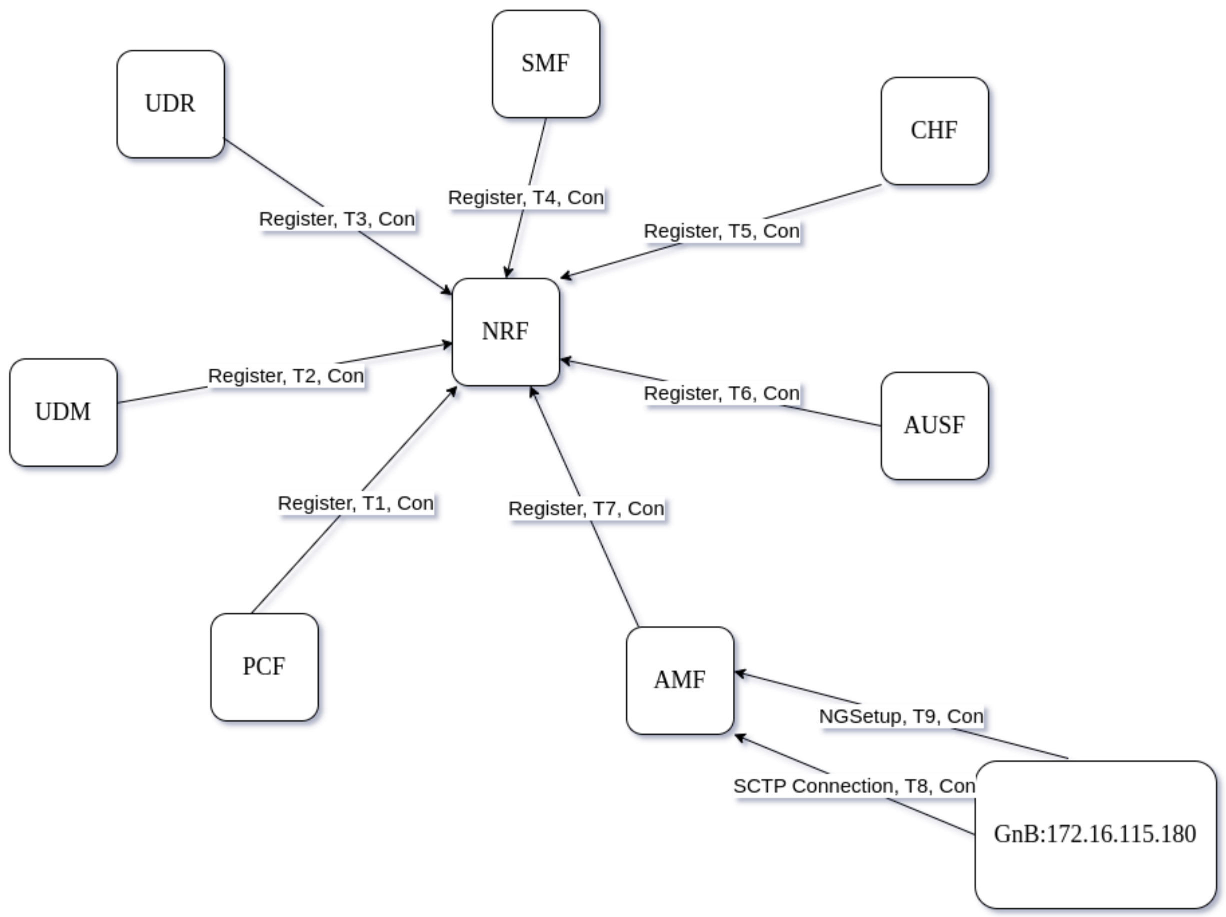


Fig. 5.1 Provenance Graph capturing NF registration and GnB setup

the model to distinguish between consumer and producer roles, as well as typical communication behavior of different NF types (e.g., AMF, SMF, NRF).

2. **GraphSAGE-based Role Learning:** A GraphSAGE model with 2-hop aggregation is used in an inductive training mode to learn the structure and semantics of node behaviors in the benign graph. Unlike transductive GCNs, GraphSAGE supports the generalization to unseen nodes, which is critical for detecting anomalies in evolving graph streams. To handle the diversity of node behaviors (e.g., AMF vs. PCF), a *multi-model strategy* is adopted: multiple GraphSAGE submodels are trained on overlapping partitions of benign nodes to specialize in different "roles." A node is only considered benign if at least one submodel classifies it with high confidence above a tunable threshold R. This approach allows for specialization of submodels and reduces false negatives for role-specific benign behaviors

5.6.2 Execution on All Scenarios

Once the submodels are trained on benign-only data, the system proceeds to analyze interaction graphs from other scenarios (slicing attack, DoS attack, etc.) in streaming mode.

1. **Sliding Subgraph Buffer:** To scale to high-frequency interaction logs, the system maintains an in-memory subgraph consisting of recently active nodes and their 2-hop neighbors. This ensures that context for newly observed interactions is preserved while bounding memory usage.

2. **Anomaly Classification:** For each new node (or interaction involving a new edge), the system computes the node's feature vector and feeds it to all trained GraphSAGE submodels. If no model returns a classification with confidence exceeding the threshold R, the node is flagged as *anomalous*, indicating a deviation

from learned benign behaviors.

3. **Alert Batching with Time Windows:** To avoid overreacting to transient deviations, a time window T and a tolerance threshold $\widehat{T}$ are applied. An alert is raised only if the number of persistent anomalies in the sliding window exceeds $\widehat{T}$, which helps filter out noise and reduce false positives.

5.6.3 Results on Attack Scenarios

– **Slicing Attack:** During the Type-I slicing attack simulation, our detection module successfully identifies the malicious PCF node as anomalous due to its unauthorized service request pattern, which deviates from the normal role of a PCF in the benign training graph. This is reflected in the anomaly score exceeding the defined threshold, as shown in Figure 6.2.

– **NRF DoS Attack:** In this scenario, the model detects the AMF node as anomalous, due to its abnormally high frequency of discovery requests that do not conform to any known benign behavior. The anomaly score for AMF crosses the detection threshold, as shown in Figure 6.1, highlighting the model's ability to respond proactively.

The following chapter presents a detailed explanation of the underlying system architecture, as well as a comprehensive analysis of the model's performance across various test scenarios.

Chapter 6

Detection Framework and Results

This chapter explains the two key components of the detection framework. **Algorithm 2** performs training of multiple submodels by learning the local communication patterns of each NF and capturing its structural and behavioral characteristics through the beingn dataset. **Algorithm 3** detects anomalies using the trained ensemble of models, where multiple NF-specific models vote to flag a node as anomalous if its behavior deviates from learned norms. This design ensures accurate, NF-aware anomaly detection across the 5GC. More details about the algorithms are provided in the following sections.

6.1 GraphSAGE Architecture

6.1.1 Provenance Graph Formalization

We model the inter-NF interactions from Free5GC logs as a provenance graph

$$G = (V, E, X_v, X_e, T_e),$$

where V is the set of NF-related entities (e.g. NF instances, UE contexts, sessions), E the directed SBI calls between them, $X_v : V \to \Sigma_v$ labels each node with its type, $X_e : E \to \Sigma_e$ labels each edge with its operation and T_e timestamps each edge.

6.1.2 Forward Propagation (FP)

We use GraphSAGE in inductive mode to learn node embeddings by aggregating local structure over K hops. Denote by $\mathbf{h}_v^{(k)} \in R^d$ the representation of node v at layer k. Initialization:

$$\mathbf{h}_v^{(0)} = \mathbf{x}_v,$$

where $\mathbf{x}_v$ is the feature vector (see Section 2). At each layer $k = 1, \ldots, K$:

$$\mathbf{m}_v^{(k)} = \mathrm{MEAN}\big(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}\big),$$

$$\mathbf{h}_v^{(k)} = \sigma\big(W_k \,\|\, [\,\mathbf{h}_v^{(k-1)} \,\|\, \mathbf{m}_v^{(k)}]\big),$$

where $\|$ denotes concatenation, σ is a nonlinearity (ReLU), and W_k are learned weights. After K hops, the final embedding is normalized:

$$\mathbf{z}_v = \frac{\mathbf{h}_v^{(K)}}{\|\mathbf{h}_v^{(K)}\|_2}.$$

A softmax over $\mathbf{z}_v$ produces class probabilities during training and detection.

6.2 Feature Extraction for Inter-NF Communications

To train GraphSAGE in supervised mode without attack labels, we assign each node v a feature vector $F(v) \in N^{2N_e}$ that captures its local SBI call distribution

$$F(v) = [\,a_0, a_1, \ldots, a_{N_e-1}, a_{N_e}, \ldots, a_{2N_e-1}\,],$$

where for $i = 0, \ldots, N_e - 1$:

$$a_i = \bigl|\{\, e \in \mathrm{In}(v) \colon M_e(X_e(e)) = i \}\bigr| \quad \text{and} \quad a_{i+N_e} = \bigl|\{\, e \in \mathrm{Out}(v) \colon M_e(X_e(e)) = i \}\bigr|.$$

Here $N_e = 8$ is the number of distinct SBI operation types used in our Free5GC implementation. These include {`NFRegister`, `NFDiscover`, `AccessToken`, `UEAuthentication`, `PDUSessionCreate`, `RequestLocationInfo`, `SessionModification`, `SessionRelease`}. The mapping function M_e assigns a unique integer index to each of these operation labels.

This vector succinctly encodes how often each NF entity participates in each type of SBI request or response, enabling the GNN to learn "roles" of benign NFs and detect anomalies when deviations occur.

6.3 Multi-Model Training Algorithm for NF Provenance Graphs

To train our anomaly detection framework on the NF-level provenance graphs extracted from Free5GC logs, we adopt the multi-model strategy described in the THREATRACE architecture. The key idea is to iteratively train multiple GraphSAGE-based submodels on subsets of the benign graph data. Each submodel specializes in learning a subset of node behaviors with high confidence, based on structural and semantic features extracted from the graph.

Unlike traditional classifiers trained end-to-end, this approach avoids overfitting to specific node types and improves generalization to unseen or rare benign behaviors. Nodes are only removed from the training pool once a submodel classifies them correctly and with high confidence (above a tunable threshold R_t). This process continues until all benign nodes are assigned to a confidently trained submodel.

- **Benign Classes:** The model is trained to classify each node into one of 11 benign classes representing the various NFs and user equipments, namely: AMF, SMF, PCF, UDM, NRF, AUSF, CHF, UDR, NSSF, gNB, and UE.

Algorithm 2: Multi-Model Training for 5GC Provenance Graphs

```
Input: Subgraph $G = (V, E, F, L)$: nodes $V$, edges $E$,
Feature map $F : V \to R^d$, Label map $L : V \to \{0, \dots, C-1\}$
Confidence threshold $R_t$, Hop number $K$, Epochs $E$
Output: Trained submodels $\{M_0, M_1, \dots, M_{cnt-1}\}$
1  Initialize: $X \leftarrow$ all active nodes in $V$, $cnt \leftarrow 0$ ;
2  while $X$ not empty do
3      $V' \leftarrow$ nodes within $K$-hop neighborhood of all $v \in X$ ;
4      for $e = 1$ to $E$ do
5          $G' \leftarrow (V', E|_{V'}, F, L)$ ;
6          $z \leftarrow$ GraphSAGE_ForwardProp$(G', F, K)$ ;
7          $z_X \leftarrow z[v], \forall v \in X$ ;
8          loss $\leftarrow$ CrossEntropy$(z_X, L(X))$ ;
9          Update model weights using backpropagation ;
10     $M \leftarrow$ Softmax$(z_X)$ ;                    // Prediction probabilities
11     foreach $v \in X$ do
12         $C_v \leftarrow \arg\max M[v]$ ;
13         $C'_v \leftarrow \arg\max_{j \neq C_v} M[v][j]$ ;
14         if $C_v = L(v)$ and $M[v][C_v]/M[v][C'_v] > R_t$ then
15             Remove $v$ from $X$ ;    // High-confidence correct classification
16     $M_{cnt} \leftarrow$ current trained submodel ;
17     $cnt \leftarrow cnt + 1$ ;
18 return $\{M_0, M_1, \dots, M_{cnt-1}\}$
```

6.3.1 Anomaly Detection Using Multi-Model Framework

Once the benign training process is complete using Algorithm 1, the system maintains a set of trained submodels $\{M_0, M_1, \ldots, M_k\}$, each of which specializes in classifying a subset of benign NF behaviors (e.g., AMF, NRF, UDM). These submodels are used during the execution phase to detect anomalous network function behaviors on new graph data derived from Free5GC logs.

Each incoming node v in the runtime provenance graph is processed as follows:

1. **Feature Vector Computation:** For each node v, a feature vector $F(v) \in R^{2N}$ is generated, where $N = 8$ is the number of distinct SBI message types used in our implementation. The feature vector captures the count of incoming and outgoing messages of each type, resulting in a 16-dimensional representation that encodes the interaction pattern of each NF node.

2. **Submodel Evaluation:** Each submodel M_i produces a class probability distribution over benign node types:

$$M_i(v) = \text{softmax}(z_v^{(i)}) \quad \text{where } z_v^{(i)} \in R^C$$

   where C is the number of benign classes and $z_v^{(i)}$ is the logit output from model i.

3. **Confidence Check:** A node is considered *benign* if any model M_i classifies it correctly with a confidence ratio above the threshold R_t. Let:

$$\widehat{c} = \arg\max M_i(v) \quad \text{and} \quad \widehat{c}' = \arg\max_{j \neq \widehat{c}} M_i(v)[j]$$

The node is accepted as benign if:

$$\frac{M_i(v)[\widehat{c}]}{M_i(v)[\widehat{c'}]} > R_t$$

4. **Anomaly Flagging:** If no model meets the confidence threshold for v, then the node is labeled as *anomalous*.

6.3.2 Normalized Entropy-Based Anomaly Score

In addition to the confidence-ratio-based detection, we define a normalized entropy-based anomaly score that captures the overall uncertainty of the softmax output. Entropy provides a robust way to measure how "confused" a model is while predicting the class of a node.

For each node v, let $M_i(v) = [p_1^{(i)}, p_2^{(i)}, \ldots, p_C^{(i)}]$ denote the softmax output from the ith submodel M_i, where C is the number of benign classes. The entropy of this distribution is:

$$H_i(v) = -\sum_{j=1}^{C} p_j^{(i)} \log p_j^{(i)}$$

We select the model M_{i^*} that is most confident, i.e., has the lowest entropy:

$$i^* = \arg\min_i H_i(v)$$

To normalize this entropy value to the range $[0, 1]$, we divide by the maximum possible entropy $\log C$, obtaining the final anomaly score:

$$\text{AnomalyScore}_{\text{entropy}}(v) = \frac{H_{i^*}(v)}{\log C}$$

This score evaluates to:

– **0** when at least one submodel is completely confident (i.e., softmax output is near one-hot),

– **1** when all submodels are maximally uncertain (i.e., uniform distribution).

This formulation ensures that even if most submodels are unsure, the presence of a confident submodel can prevent false alarms—an important design choice in the multi-model framework.

Algorithm 3: Anomaly Detection via Multi-Model Inference

```
Input: Runtime graph $G = (V, E, F)$, submodels $\{M_0, \ldots, M_k\}$, threshold $R_t$
Output: Set of anomalous nodes $A$
1  $A \leftarrow \emptyset$ ;                                  // Initialize anomaly set
2  foreach $v \in V$ do
3      $flagged \leftarrow$ true ;
4      foreach $M_i$ in submodels do
5          $P \leftarrow M_i(F(v))$ ;
6          $\widehat{c} \leftarrow \arg\max P$ ;
7          $\widehat{c}' \leftarrow \arg\max_{j \neq \widehat{c}} P[j]$ ;
8          if $\frac{P[\widehat{c}]}{P[\widehat{c}']} > R_t$ then
9              $flagged \leftarrow$ false ;
10             break ;                                     // Node accepted as benign
11     if $flagged$ then
12         $A \leftarrow A \cup \{v\}$ ;                   // Flag as anomalous
13 return $A$
```

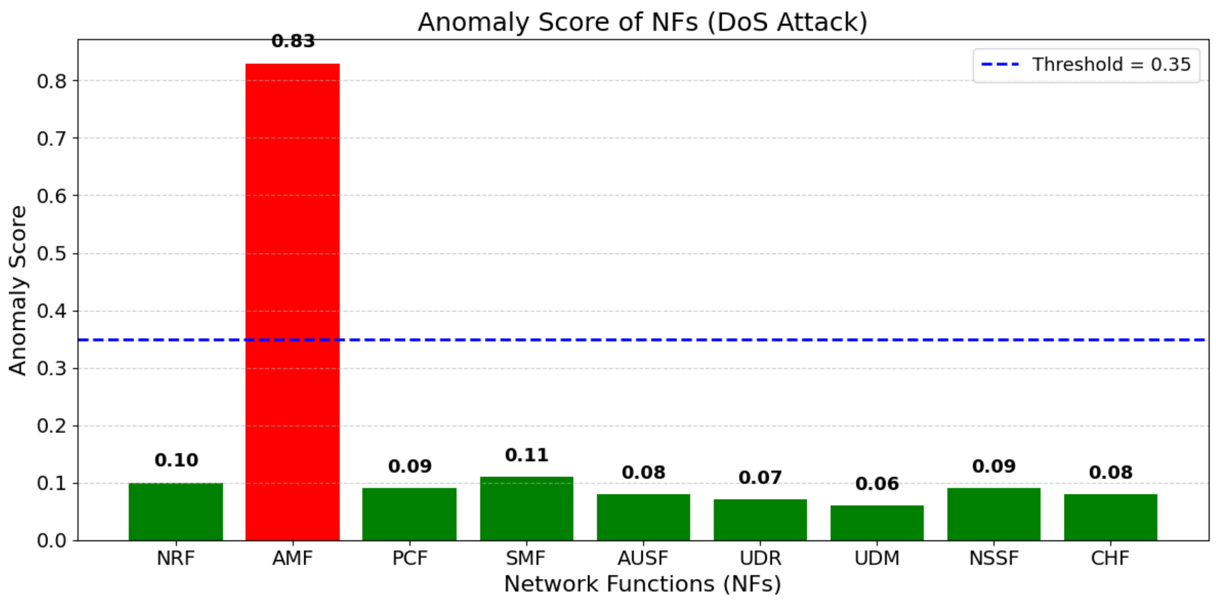


Fig. 6.1 Anomaly scores of various NFs under DoS attack

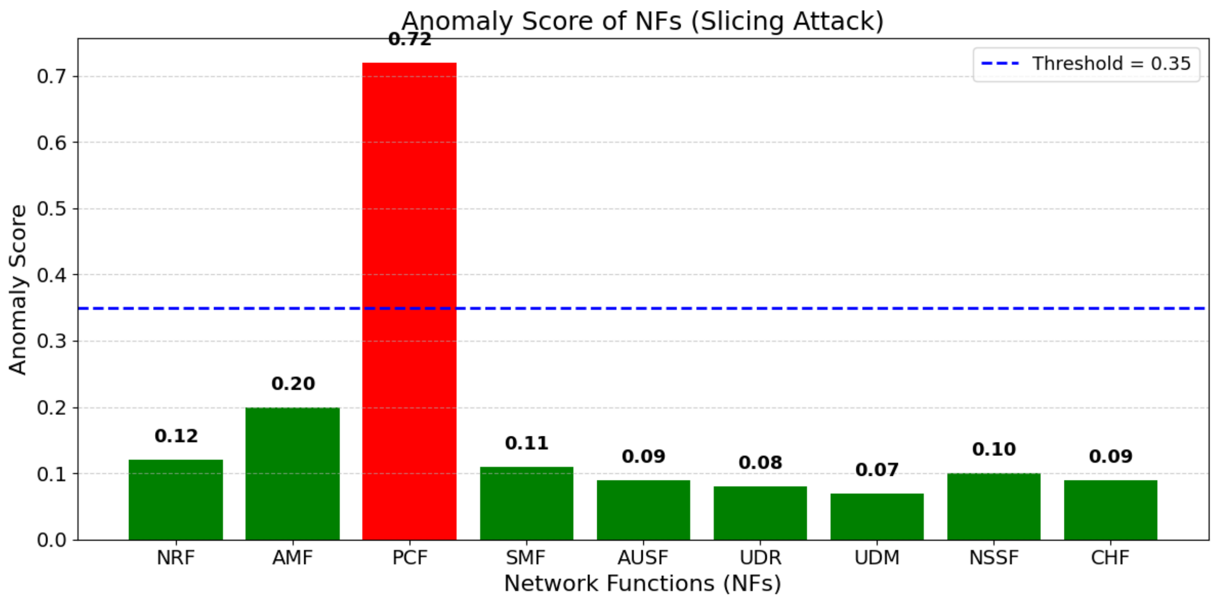


Fig. 6.2 Anomaly scores of various NFs under slicing attack

6.4 Results and Evaluation

6.4.1 System Specifications and Simulation Setup

All experiments were conducted on a system with an Intel Core i9 processor (32 cores), 32 GB RAM, running Ubuntu 24.04. The provenance logs used for training and evaluation were collected from simulated 5GC scenarios using free5GC and UERANSIM. The benign dataset contained approximately 11,000 log entries representing normal SBI interactions, while the slicing attack and DoS attack scenarios togather generated close to 23,000 entries. These logs were used to construct provenance graphs for both model training and anomaly detection evaluation.

To evaluate the performance of our proposed anomaly detection framework, we compared it against three baseline models: a single GNN trained on all benign data, a traditional Random Forest classifier, and Logistic Regression. For a fair comparison, the same node-level feature vectors extracted from the provenance graph were used as input for both Random Forest and Logistic Regression as well.

– **Accuracy:** The Multi-GNN approach achieved the highest accuracy of 91%, significantly outperforming the Random Forest, Logistic Regression, and Single GNN baselines.

– **Precision and Recall:** The proposed model maintained a strong balance between precision (0.92) and recall (0.87), indicating both low false positives and a high detection rate.

– **F1 Score:** With an F1 score of 0.90, the Multi-GNN framework demonstrates robust classification performance across attack types.

Figure 6.3 illustrates the bar plots comparing different models across various performance metrics. Table 6.1 summarizes these results numerically.

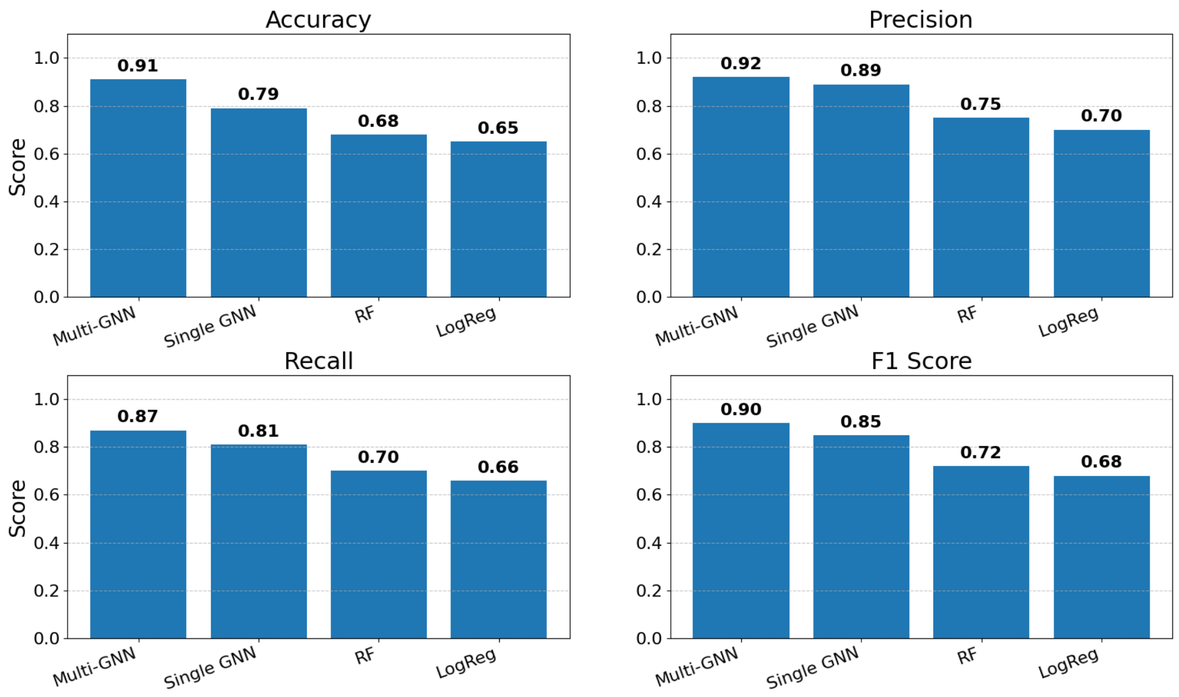


Fig. 6.3 Performance comparison of different models across various metrics

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Proposed (Multi-GNN)	0.91	0.92	0.87	0.90
Single GNN	0.79	0.89	0.81	0.85
Random Forest	0.68	0.75	0.70	0.72
Logistic Regression	0.65	0.70	0.66	0.68

Table 6.1 Comparison of detection performance across different models

Observations

The results clearly indicate that GNN-based models outperform traditional classifiers like Random Forest and Logistic Regression. This improvement is largely due to the ability of Graph Neural Networks to capture complex inter-node relationships and structural patterns in the provenance graph, which are not effectively modeled by flat, feature-based classifiers.

Furthermore, the Multi-GNN approach surpasses the Single-GNN model by leveraging multiple graphs trained on distinct benign contexts, allowing it to generalize better across diverse behavior patterns. This multi-model strategy enhances sensitivity to subtle deviations and ensures robustness against both known and novel attack behaviors.

6.4.2 Performance Overhead of Enhanced Logging

A natural concern with detailed SBI-level logging is the potential performance degradation it may cause—particularly in terms of CPU usage and memory consumption during high-load scenarios. To assess the impact, we conducted a controlled benchmark with and without logging enabled across $\sim 10,000$ SBI requests.

The results, shown in Figures 6.4 and 6.5, indicate that the overhead introduced by logging is minimal and well within acceptable operational bounds. Specifically:

- The average CPU utilization increased by only $\sim 3\%$, demonstrating efficient log writing and minimal contention on logging I/O.
- The memory usage grew by less than 100 MB, which is insignificant considering modern NF deployments run on servers with multiple gigabytes of RAM.

These results affirm that our logging enhancements are scalable and practical for real-time NF monitoring and provenance graph construction.

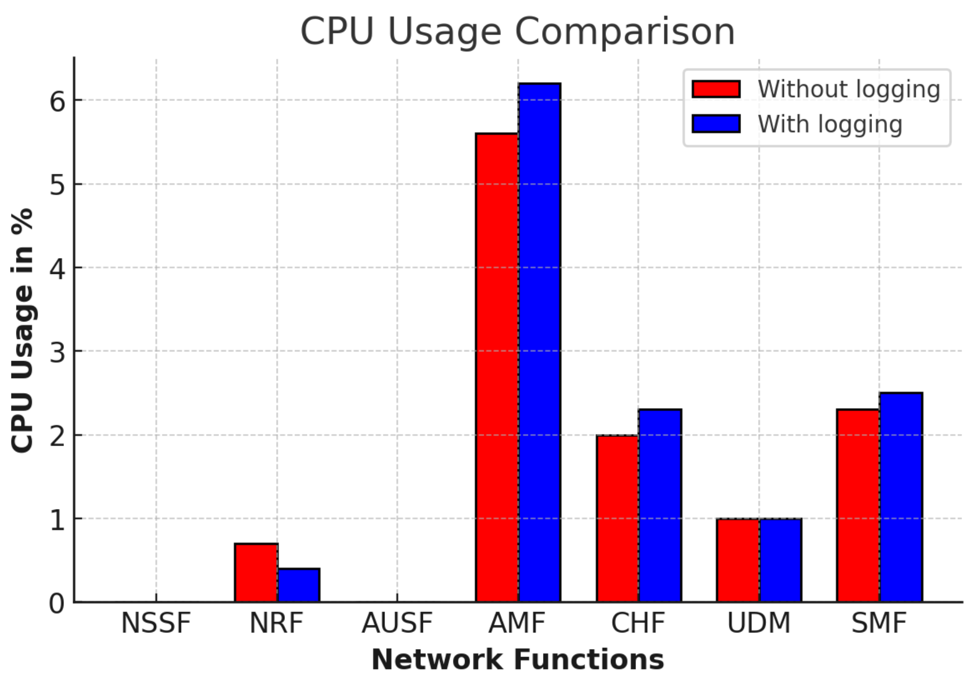


Fig. 6.4 CPU usage comparison: With vs Without Logging

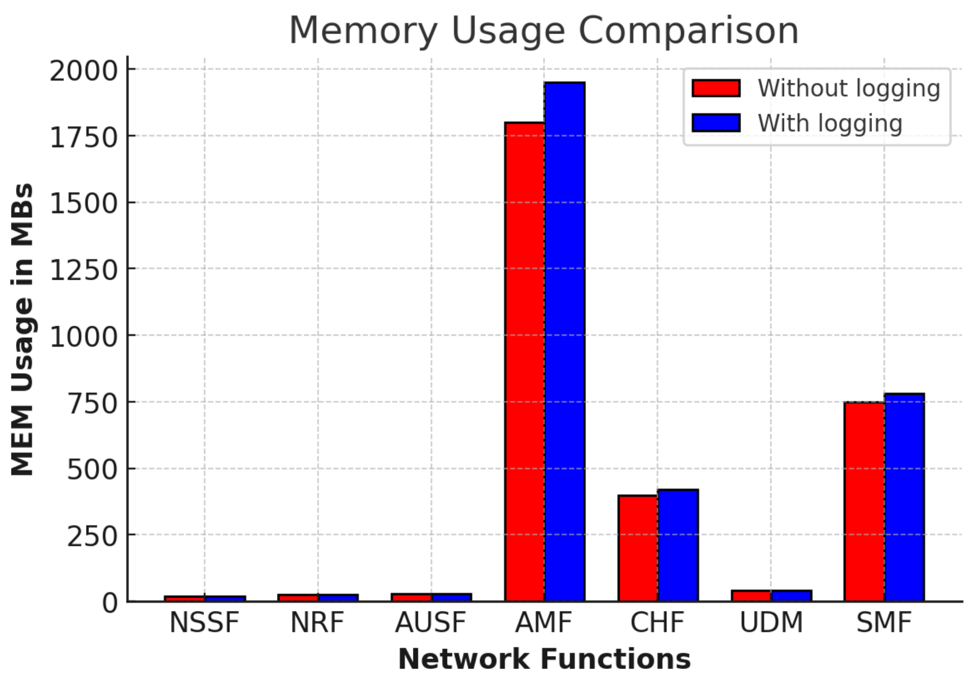


Fig. 6.5 Memory usage comparison: With vs Without Logging

Chapter 7

Conclusion and Future Work

This work provides valuable insights into the vulnerabilities of the 5G core network and demonstrates initial approaches to addressing its security challenges.

The following tasks were completed in **Phase 1** of the BTP:

- Successfully set up Free5GC and UERANSIM to simulate the 5G core network environment, enabling the evaluation of DoS and slicing attacks on the 5G core.
- Analysed key vulnerabilities in the NRF and cross-slice communication, which could compromise network performance and user data confidentiality.
- Implemented a rate-limiting mechanism for the NRF to mitigate flooding attacks, demonstrating its functionality as an initial defense against resource exhaustion.

Phase 2 extended the work toward a graph-based anomaly detection framework:

- Enhanced Free5GC's logging structure to record all service-based interface (SBI) communications among network functions (NFs), including both NF-to-NF and UE-to-NF interactions.

- Simulated multiple operational contexts—benign traffic, slicing attacks, and DoS floods—to generate structured logs capturing diverse behavior.
- Developed log parsers to extract structured interaction data and construct provenance graphs encoding inter-NF communication flows.
- Adapted the THREATRACE architecture for 5G core networks by training on benign Free5GC graph data, enabling zero-day anomaly detection using a multi-model GraphSAGE framework with high performance across test scenarios.

Future enhancements to this work can further improve detection robustness and operational insight:

- Incorporate additional complex attack scenarios such as signaling storms, session hijacking, and inter-slice privilege escalation to broaden the scope of anomalous behavior the framework can detect.
- Perform probabilistic analysis of critical metrics (e.g., anomaly score distributions, model confidence ratios) to derive statistical guarantees and improve threshold tuning strategies.

References

[BEBP21] Shanay Behrad, David Espes, Philippe Bertin, and Cao-Thanh Phan. Impacts of service decomposition models on security attributes: A case study with 5g network repository function. pages 470–476, 06 2021.

[Cal24] Calsoft Inc. 5g service-based architecture (sba). `https://www.calsoftinc.com/blogs/5g-service-based-architecture-sba.html`, 2024. Accessed: 2024-11-20.

[CH24] Peng-Yu Chen and Chin-ya Huang. Evaluation of ue registration performance in open-source 5g core networks. pages 1–5, 08 2024.

[GLF$^+$24] Shujuan Gao, Ruyan Lin, Yulong Fu, Hui Li, and Jin Cao. Security threats, requirements and recommendations on creating 5g network slicing system: A survey. *Electronics*, 13:1860, 05 2024.

[MML$^+$24] Tariro Mukute, Lusani Mamushiane, Albert Lysko, Ramona Modro, and Joyce Mwangama. Control plane performance benchmarking and feature analysis of popular open-source 5g core networks: Openairinterface, open5gs, and free5gc. *IEEE Access*, 12:113336–113360, 08 2024.

[PY24] Harsh Pacherkar and Guanhua Yan. Prov5gc: Hardening 5g core network security with attack detection and attribution based on provenance graphs. pages 254–264, 05 2024.

[WWZ$^+$22] Su Wang, Zhiliang Wang, Tao Zhou, Hongbin Sun, Xia Yin, Dongqi Han, Han Zhang, Xingang Shi, and Jiahai Yang. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security*, 17:3972–3986, 2022.

[ZLZ$^+$23] Yu Zhang, Zhenguang Liu, Tianwei Zhang, Dinghao Wu, and Wei Zou. Magic: Masked graph modeling improves the generalizability of intrusion detection on provenance graph. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1199–1213. ACM, 2023.