National College of Ireland

Real-Time Threat Detection in Open5GS Networks Using Amazon GuardDuty

MSc Research Project

MSc. CyberSecurity

Ammad Ud Din Bajwa

Student ID: x23157526

School of Computing
National College of Ireland

Supervisor: Rohit Verma

National College of Ireland
Project Submission Sheet School
of Computing

National College of Ireland

Student Name:	Ammad Ud Din Bajwa
Student ID:	X23157526
Programme:	MSc. CyberSecurity
Year:	2024
Module:	MSc Research Project
Supervisor:	Rohit Verma
Submission Due Date:	12/12/2024
Project Title:	Real-Time Threat Detection inOpen5GS
Word Count:	8,027
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ammad
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	Y
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	Y
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy	y

on computer.	

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Real-Time Threat Detection in Open5GS Networks Using Amazon GuardDuty

Ammad-Ud-Din Bajwa

X23157526

Abstract

In today's age of fifth-generation (5G) networks, which have revolutionized the telecommunications landscape, offered amazing speeds, ultra-low latency, and huge connectivity to support a number of applications have also been a great source of the integration of open-source 5G core implementations like Open5GS including the cloud infrastructures comes with no shock. This research study dives deep into the effectiveness of Amazon GuardDuty which is a cloud-native threat detection service which is used in identifying and responding to real-time security threats within an Open5GS-based private 5G network deployed on Amazon Web Services (AWS) including the zero-day threats. In this study a comprehensive experimental setup was created which simulates a private 5G network environment using Open5GS tool for core network functions and srsRAN for radio access network (RAN) and user equipment (UE) simulation. To use all these together, this network was hosted inside a secured AWS Virtual Private Cloud (VPC) where all the subnets, security groups, and routing were configured to emulate a realistic deployment. Amazon GuardDuty was integrated without any of the custom configurations to use its default abilities like the monitoring of the VPC Flow Logs, DNS logs, and AWS CloudTrail events for threat detection. This study also simulates various network threats including port scans, SSH brute-force attempts, denial-of-service (DoS) attacks, DNS exfiltration, and unauthorized API calls. All of these to evaluate GuardDuty's detection performance and the evaluation metrics which are the focus of this study and finds the detection accuracy, response time, false positive rate, time-to-detection consistency, and performance overhead on each and all network operations. The findings of the GuardDuty showed that the integration of GuardDuty introduced in itself a minimal performance overhead along with almost no impact on CPU utilization, network latency, and throughput. This ensured that the 5G network's efficiency did not get affected by the GuardDuty's resource needs. This research study concludes that Amazon GuardDuty is very effective in both finding and responding to many real-time security threats within an Open5GS-based private 5G network on AWS and this study also shows that GuardDuty's ability to work with machine learning and anomaly detection techniques makes it good for monitoring of network activities as well as enhancing the security of private 5G deployments.

Keywords: AWS GuardDuty, Open5GS, Real-Time Threat Detection, Private 5G Networks, Network Security, Cloud-Native Security Services, srsRAN, Open-Source 5G Core Network, Cybersecurity

1. Introduction

The rapid evolution of cellular networks has been an era-defining tech in this tech heavy environment and that in this era of fifth-generation (5G) technology many promising unprecedented speeds, ultra-low latency, and massive connectivity of various networks make is so that it is easier to support a combination of applications from autonomous vehicles to remote surgery. With such needs popping up in the enterprise environment, and as organizations adopt private 5G networks to leverage these powers for their own custom enterprise solutions, it is obvious that the security of these networks becomes very important. Private 5G networks offer greater control and customization but they also fall short of introducing a complex landscape of potential security issues due to their integration of various cloud services and the Internet of Things (IoT). Open5GS is a tool which is an open-source implementation of 5G core network functions and as open source research/tools go, it has gained significant attraction for its flexibility and cost-effectiveness in the deployment of the private 5G networks. Regarding this, the open-source nature of Open5GS may very much expose these networks to a lot of security challenges which are not completely foolproof and may not be addressed by traditional security mechanisms. The integration of Open5GS with cloud platforms like Amazon Web Services (AWS) further complicates things as the security posture is greatly affected with their joint junction which makes it necessary to deal with the advanced threat detection and response strategies. To counter that, we use the Amazon GuardDuty which is a threat detection service that can be setup such that it continuously monitors for malicious activity and unauthorized behavior to protect AWS accounts. And if we can also leverage machine learning, anomaly detection, and integrated threat intelligence, we can enhance the GuardDuty to identify and prioritize potential threats related to the zero-day threat and much more. While GuardDuty is extensively used for securing AWS resources, its actual potential can be realized in the context of private 5G networks, especially those using Open5GS and which have remained underexplored. This research study tries to dive deep into the various capabilities of the AWS GuardDuty and also seeks to bridge this gap by evaluating the potential of Amazon GuardDuty in detecting real-time security threats within Open5GS-based private 5G networks. This is done by simulating a complete 5G network environment, including core and radio access network (RAN) components and in also generating various network threats and attacks with which we aim to gauge the GuardDuty's capabilities and also check its timings in identifying and responding to these threats. This study also involves integrating additional open-source tools like srsRAN to simulate user equipment (UE) interactions and also to generate various realistic network traffic patterns.

In the previous studies, they have focused on performance benchmarking and feature analysis of open-source 5G core networks (Mukute et al., 2024; Lando et al., 2023) and also the deployment challenges associated with open-source 5G infrastructures (Martin et al., 2023). But there has always been a need for research to examine the security aspects of such networks especially in the real-time threat detection context where they are also using the cloud-native security services. This research study shows and dives deep into the various insights which provide the practical application of Amazon GuardDuty and its security for the Open5GS networks. It also tries to identify the various areas for improvement in threat detection. In this research study’s later chapter, the methodology is to set up an AWS environment with necessary permission policies and services to configure a small MVP simulated 5G network using Open5GS and srsRAN and not only that but also the implementation of the various network attack scenarios to test and check the detection capabilities of GuardDuty. Various metrics are used including the metrics such as detection accuracy, response time, detection rate, false positives, and time-to-detection.

1.1 Research Question

How effectively can Amazon GuardDuty detect and respond to real-time security threats within an Open5GS-based private 5G network deployed on AWS, through analysis of network traffic, DNS logs, and cloud activity logs?

To address this question in this study, this research will focus on the following sub-questions:

1. Can the GuardDuty accurately identify specific security threats such as port scans, denial-of-service (DoS) attacks, data exfiltration attempts, malicious domain communications, phishing attacks, and unauthorized API activities within this simulated private 5G network environment?

2. What is the response time, i.e. various metrics, of GuardDuty in detecting and alerting about these threats. How does it impact the overall security of the network?

3. Does the implementation of GuardDuty introduce any major performance overhead issues on the network operations, and can they be avoided?

By exploring these questions and the solution to these answers, this research study will aim to provide a comprehensive evaluation of Amazon GuardDuty's performance in securing private 5G networks built on open-source platforms like Open5GS alongside the AWS ecosystem. Such findings of this research study will have practical impact for various enterprises and their networks considering the deployment of private 5G networks which are relying on cloud-native security solutions to protect their infrastructure.

2. Related Work

In today's age, the combination of fifth-generation (5G) networks with cloud computing platforms has introduced a new world of possibilities in network deployment and management. This deployment has been ensuring flexibility, scalability, and cost-effectiveness but this new seamless integration also presents many security challenges and issues which then arise from such junctions and it also gives rise to the need for the advanced threat detection and eventual mitigation strategies. This section of this research study reviews all the major existing literature on open-source 5G core networks and on their deployment challenges, performance benchmarking, and security considerations. This is especially true for the integration of the cloud-native security services like Amazon GuardDuty. Open-source implementations of 5G core networks like Open5GS, OpenAirInterface (OAI), and free5GC are very famous and have gained so much attraction because of their accessibility and adaptability in the academic research environment for development purposes. Mukute et al. (2024) conducted a complex and complete performance benchmarking & feature analysis of these popular open-source 5G core networks. Their study evaluated the control plane performance which highlights the trade-offs between latency, throughput, and resource utilization of the open source 5G networks but their findings also showed that while open-source cores are better at testing and small-scale deployments, their performance optimization drops a lot and that this is a critical area for their large-scale commercial applications.

Similarly, Lando et al. (2023) conducted their research and evaluated the performance of open-source software impacts of the 5G network core but their work focused on finding the

readiness and alertness of these platforms for production environments. This was done by examining various metrics such as processing delay and packet loss and as such their study showed that the need for further enhancements in the stability and efficiency of open-source 5G cores was needed to meet these strict requirements of 5G services. Deploying a stable 5G standalone (SA) testbed comes with its several challenges. This is especially true when concerning UE integration and network slicing as this has been the study done by Mamushiane et al. (2023), who addressed these issues by using the srsRAN and Open5GS to make a 5G SA testbed. Their research provided understanding into the troubleshooting techniques for UE integration and the network slicing which is very important for the allocation of the various network resources efficiently distributed in between the different services and applications. In his study, Martin et al. (2023) presented Open-VERSO which is a vision for 5G experimentation infrastructures but also presented the various hurdles and challenges which came with deploying open-source 5G networks, especially the famous interoperability issues, performance limitations, and security vulnerabilities which are the core issues of deployment of the traditional open source 5G implementation. Their study focused on the importance of collaborative efforts in the research community such that it may overcome these various obstacles and issues which are related to the advance development of open 5G infrastructures.

Bonati et al. (2024) introduces 5G-CT which is an automated deployment and over-the-air testing framework for various important end-to-end open radio access networks (O-RAN) in the development of the open source 5G networks. Their study also showed a unique approach which streamlined the deployment process and also made sure the safety of the facilitated testing of O-RAN components. This eventually resulted in contributing to the acceleration of 5G network development and deployment. It is known that the open-source nature of platforms like Open5GS and srsRAN have various advantages for innovation and customization but they also expose various networks they are working in to potential security risks as shown in the study done by Linh et al. (2023) who also analyzed open-source 5G core networks for TLS vulnerabilities and 3GPP compliance. In their study it is revealed that there are several security gaps like improper implementation of encryption protocols and deviations from standardized security procedures are actions which have been explicitly outlined by the 3GPP. According to their study, such vulnerabilities could be used by malicious users to manipulate the network integrity and confidentiality, and security.

Chepkoech et al. (2023) showed in his study the various uses of open-source software in enabling OpenRAN-compliant 5G standalone campus networks and the various security considerations which happened in deploying such networks giving rise to the need for strong hand security mechanisms to protect against threats which comes during the open-source deployments. As 5G networks have been using cloud infrastructures more and more, integrating cloud-native security services becomes even more important. In that regard, Amazon Web Services (AWS) offers several tools and services which provide enhanced security posture for all of the AWS services including Amazon GuardDuty for threat detection as discussed in the study by the Sharma and Saxena (2020), who also showed in their study what these security best practices in AWS are and how they outline various methods for securing cloud resources against various network attacks and threats. In their study they highlighted the importance of continuous monitoring and the use of automated threat detection tools to identify and reduce risks caused. Routavaara (2020), in one of their studies, examined security monitoring in AWS public cloud environments and also in their study showed the challenges of maintaining visibility and control over dynamic cloud resources for which they also recommended the completion of the complex monitoring solutions i.e. solutions which could use the machine learning and anomaly detection techniques to better improve the network security.

Tykholaz et al. (2024) in their recent study explored incident response with AWS detective controls where they showed that Amazon GuardDuty and AWS Security Hub could work in union to improve the various abilities of these tools in the detection of the suspicious malicious activities and whether or not such activities could be using rapid response to security events. In this study they showed the integration of such security and behavioral services in such a way that they enable these organizations to maintain a strong security posture in the cloud. As is commonly known, the integration of open-source 5G cores with cloud security services always come with their own issues and present various opportunities for improving the network security through advanced threat detection using various ML and behavioral techniques. But for that reason there is a lack of research specifically addressing the potential of cloud-native security tools in the world of 5G networks and this has caused the gap to be particularly obvious in the application of Amazon GuardDuty to detect threats within Open5GS-based networks deployed on AWS networks and then to combat the various threats therein. In that regard, Bhatt (2024) , in one of their recent studies, discussed security and compliance for running important enterprise level systems on AWS. They particularly mentioned the importance of combining the cloud security strategies with specific application requirements such that the SAP systems could be integrated and its principles are applicable to 5G network deployments.

Tan (2023) highlighted the importance of both reducing and increasing the security of the insider threats in AWS from a zero-trust perspective which they used in their own study when applying zero-trust principles to 5G networks which eventually enhanced their security by ensuring that all network components be they on-premises or in the cloud are continuously authenticated and authorized and thus safe from the various network level threats. In the current tech environment the evaluation of the performance impact of integrating security solutions like Amazon GuardDuty with 5G networks has been a topic of extreme importance and thus it is critical to analyze and maintain and ensure that security postures of the enterprise networks do not lower or adversely affect the network performance as discussed by Villa et al. (2024) in his recent paper, where they also introduced X5G which is an open, programmable, multi-vendor, end-to-end private 5G O-RAN testbed. In this research study they have primarily focused on network performance and programmability which have provided a foundation for using the additional services i.e. security monitoring tools which may affect all the network operations simultaneously.

3. Methodology

In this research study's chapter we employ a complete experimental approach to evaluate the potential of Amazon GuardDuty in finding the real-time security threats inside the Open 5GS-based private 5G network deployed on Amazon Web Services (AWS). This methodology consists of design and implementation of a simulated 5G network environment, the integration of security monitoring tools, the simulation of various network threats, and the eval of detection techniques using specific eval metrics.

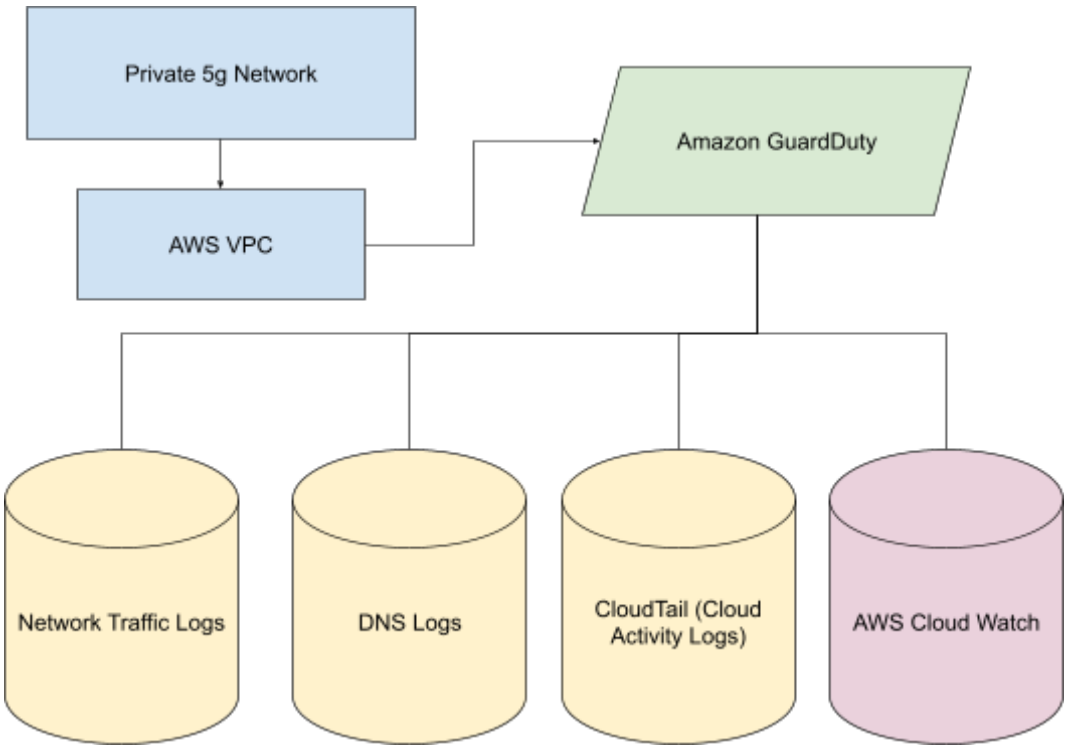

Figure 1: Methodology overview illustrating the research steps from environment setup to performance evaluation.

3.1 Environment Setup

To deploy the required resources which will be used in this study further in, an AWS account with appropriate IAM permissions is very important to be set up firsthand. The following AWS Identity and Access Management (IAM) policies have been attached to the subject user account to guarantee enough privileges such that the various experiments go without any issues:

- AmazonEC2FullAccess
- AmazonVPCFullAccess
- AmazonGuardDutyFullAccess
- CloudWatchLogsFullAccess

These policies allow for the creation and management of EC2 instances, VPCs, GuardDuty configurations, and CloudWatch logs. In this study a dedicated VPC, named MVP-5G-VPC was created to simulate the private 5G network. And this VPC was configured with an IPv4 CIDR block of 10.0.0.0/16. Two subnets were also created within the VPC to represent respective core and radio access network segments:

- Core Subnet (MVP-5G-Core-Subnet): Assigned CIDR block 10.0.1.0/24.
- RAN Subnet (MVP-5G-RAN-Subnet): Assigned CIDR block 10.0.2.0/24.

An Internet Gateway (MVP-5G-IGW) was also created and attached to the VPC to enable internet connectivity for the respective EC2 instances but they both shared the same one in this research study. Route tables were also configured to direct incoming and outgoing traffic appropriately:

- Core Route Table (MVP-5G-Core-RT) was associated with the Core Subnet and configured with a route to the Internet Gateway for destination 0.0.0.0/0.
- RAN Route Table is, similarly, configured for the RAN Subnet for required external connectivity.

An EC2 instance named MVP-5G-Core-Instance was also launched in the Core Subnet with the following:

- AMI: Amazon Linux 2
- Instance Type: t2.micro (upgraded to t2.medium for Open5GS deployment)
- Security Groups: Allowed inbound SSH (port 22) and HTTP (port 80) traffic.

An EC2 instance named MVP-5G-RAN-Instance was launched in the RAN Subnet with the following specifications:

- AMI: Ubuntu Server 20.04 LTS (to support srsRAN requirements)
- Instance Type: t2.micro (upgraded to t2.medium for Open5GS deployment)
- Security Groups: Allowed inbound SSH (port 22), SCTP (port 38412), and UDP (ports 2152 and 8805) from the Core Subnet.

3.2 Network Security Groups

To ensure secure communication between the Core and RAN instances while reducing the internet exposure, we configured the security groups such that the inbound access was restricted to essential ports, and internal communication between subnets was also enabled. Moreover the specific protocols required for 5G communication i.e. SCTP was configured to simulate a realistic 5G network environment such that the Open5GS was respectively installed on the Core Instance, and srsRAN was deployed on the RAN Instance. The Open5GS source code was cloned from GitHub, built, and installed. MongoDB was started and configured to run at boot, and a test subscriber was also added to the Open5GS database to simulate a User Equipment (UE). Once these preparations were done the Open5GS core network services were started.

On the RAN instance because of the resource requirements the system was updated such that necessary dependencies were installed and the srsRAN software was cloned, built, and installed. The gnb.conf file was also edited to point to the Core Instance's private IP address. Finally, the UE simulator was started on the RAN Instance to make a connection with the core network which ended up completing the setup of the simulated 5G environment.

3.3 Threat Simulation

To find out the detection ability of Amazon GuardDuty a complete series of network threats were simulated between the RAN Instance and the Core Instance and in these simulations stealth SYN scans using nmap were made to identify open ports on the Core Instance, repeated SSH login attempts were made to mimic unauthorized access, and simple DoS attacks were made to involve continuous HTTP requests. Additionally, DNS queries were also directed to non-existent or malicious domains to simulate data exfiltration attempts which made the attacking RAN instance have a bad repute which would further test the system’s abilities and then attempts were made to access AWS metadata or unauthorized API endpoints from the instances, and communication with known malicious IP addresses or

domains was also simulated.

To help the detection process, various VPC Flow Logs and Query Logs were enabled to capture network traffic and DNS queries. This ensured that the captured data was also being analysed by GuardDuty such that the Amazon GuardDuty was configured respectively to monitor the AWS environment and detect the simulated threats. This involved enabling GuardDuty via the AWS Management Console, navigating to the GuardDuty console, selecting "Enable GuardDuty," and configuring it to monitor all available data sources, including VPC Flow Logs, DNS logs, and CloudTrail events. After that to evaluate GuardDuty's out-of-the-box capabilities, the default settings were used i.e. no custom threat intelligence feeds or detection rules were added so that the native strength could be monitored unhindered.

Using all of the above methods and by providing network traffic information between the instances, capturing DNS queries made by the instances, and recording AWS API activity, a complete dataset was generated to assess GuardDuty's effectiveness in detecting and alerting on the simulated threats.

3.4 Data Collection and Analysis

To evaluate the various performances of Amazon GuardDuty a complete data collection and analysis framework was made such that the data was gathered from multiple sources i.e. GuardDuty findings, CloudWatch logs, and system logs on the respective Core and RAN instances. GuardDuty findings provided important insights that the detected threats were also monitored and used during and after the threat simulations where the key information extracted from these subject findings included the classification of various threat (e.g., Recon:PortScan, UnauthorizedAccess:SSHBruteForce) for which the assigned severity level (Low to High), the detection time, the involved instances, and recommended security actions to enrich the analysis were also made. To better analyse the AWS CloudWatch was used to sum up all the various logs from every source, including EC2 instances, Open5GS and srsRAN services, and VPC Flow Logs because these subject logs provided valuable context and insights into network traffic patterns.

This collected data had to go through a complete and thorough analysis process and for that the Amazon Athena was used to query logs which were stored in the respective instances which enabled effective data retrieval and efficient exploration. Moreover the custom Python scripts were developed to parse and analyze logs for specific eval metrics such that the performance of Amazon GuardDuty was evaluated using a combination of these eval metrics. The accuracy of threat detection was measured by calculating the proportion of simulated threats such that it was correctly identified by GuardDuty. The timeliness of detection was assessed by measuring the elapsed time between the initiation of a threat and its detection. The precision of the system was evaluated by analyzing the number of benign activities incorrectly flagged as threats and then finally to assess consistency the distribution of detection times across different threat types was examined such that the impact of GuardDuty on network performance, including latency and resource utilization, was also calculated.

4. Design Specifications

In this research study we will explore the design which creates the simulated private 5G network using Open5GS and srsRAN within the AWS environment and we will do this by reviewing the primary objective which is to evaluate Amazon GuardDuty's ability in detecting real-time security threats. This section also provides detailed information of the network architecture, components, configurations, and the deployment of such security and

monitoring tools.

4.1 Network Architecture

In this study this network architecture emulates a virtual standard private 5G network. This consists of a core network, a radio access network (RAN), and user equipment (UE). This setup, which consists of these respective resources, is hosted within the isolated AWS Virtual Private Cloud (VPC) environment. The VPC is named MVP-5G-VPC and it acts as the primary network environment with a CIDR block of 10.0.0.0/16. This VPC consists of two more additional subnets which are defined as the Core Subnet (MVP-5G-Core-Subnet) with a CIDR of 10.0.1.0/24 and the RAN Subnet (MVP-5G-RAN-Subnet) with a CIDR of 10.0.2.0/24.

To provide the internet access for the above two EC2 instances within the VPC an Internet Gateway (MVP-5G-IGW) is made such that the Core Route Table (MVP-5G-Core-RT) is used for routing internet traffic through the Internet Gateway and similarly the RAN Route Table is made to provide internet access for the RAN instance. These two also use their own subject security groups' configurations, CoreInstanceSG and RANInstanceSG, which are there and made to ensure traffic flow to the Core Instance and RAN Instance. These security measures provide better protection from unauthorized access and various potential threats.

4.1.1 EC2 Instances Specifications

Following two EC2 instances are used in this network simulation:

Parameter	**Open5GS-Core**	**srsRAN-Node**
AMI	Ubuntu Server 20.04 LTS	Ubuntu Server 20.04 LTS
Instance Type	t2.medium	t2.medium
Subnet	MVP-5G-Core-Subnet	MVP-5G-RAN-Subnet
Security Group	CoreInstanceSG	RANInstanceSG
Key Components	Open5GS, MongoDB	srsRAN gNB and UE
Storage	20 GiB gp2	20 iB gp2

4.1.2 Security Group Configurations

Security groups are configured to control inbound and outbound traffic, ensuring that only necessary communication is allowed. Following are inbound rules whereas the outbound rules are set to default:

Protocol	**Port Range**	**Source**	**Purpose**
SSH	22	Admin IPs	Secure SSH access
SCTP	38412	10.0.2.0/24	srsRAN communication

UDP	2152, 8805	10.0.2.0/24	GTP-U and other protocols

Following is the inbound rule for the RANInstanceSG:

Protocol	**Port Range**	**Source**	**Purpose**
SSH	22	Admin IPs	Secure SSH access

4.1.3 Open5GS Core Network Configuration

Open5GS is a powerful open source platform which provides the essential core network functions used inside a 5G standalone network and then to establish a 5G network various network functions, including the AMF (Access and Mobility Management Function), SMF (Session Management Function), UPF (User Plane Function), AUSF (Authentication Server Function), UDM (Unified Data Management), NRF (Network Repository Function), NSSF (Network Slice Selection Function), PCF (Policy Control Function), and SCP (Service Communication Proxy), are deployed.

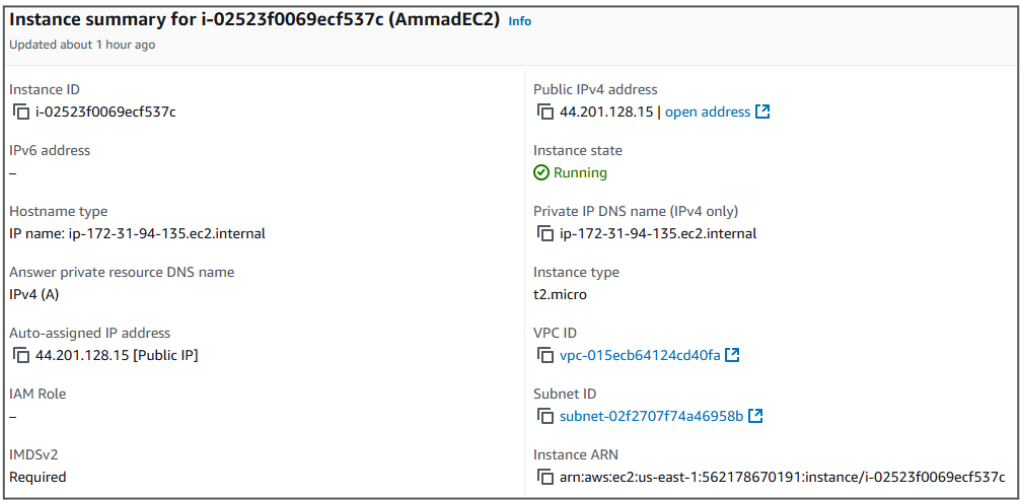

Figure 2: Overview of the Core and Radio Access Network (RAN) instances used in the simulation.

To configure these functions, specific parameters must be set. These parameters include the PLMN (Public Land Mobile Network) ID, MCC (Mobile Country Code), MNC (Mobile Network Code), TAC (Tracking Area Code), S-NSSAI (Single Network Slice Selection Assistance Information), SST (Slice/Service Type), SD (Slice Differentiator), and APN (Access Point Name). By using these configurations of these parameters, a strong and efficient 5G network can be made.

4.1.4 Amazon GuardDuty Configuration

To configure the GuardDuty which actively monitors the AWS environment for potential threats it is important to ensure complete coverage for which the various data sources are enabled to also provide valuable insights. VPC Flow Logs are used to check and test the network traffic within the VPC, while DNS Logs capture DNS queries Query Logs and the AWS CloudTrail Logs track API calls and activities, which can provide a detailed audit trail. To proactively and preemptively identify threats the AWS GuardDuty uses a complex threat detection feature for Anomaly Detection which is later powered by machine learning, to

analyze various patterns to see the unusual behaviors that may represent the malicious activity.

This monitoring of various resources of the respective instances is used across all AWS regions and accounts associated with the VPC which ensures that no corner of the subject environment is unprotected.

Feature	**Configuration**
VPC Flow Logs	Enabled for MVP-5G-VPC
DNS Logs	Query Logs enabled
CloudTrail Logs	CloudTrail management events monitored
Findings Export	Findings exported to CloudWatch Logs and S3

4.1.5 Threat Simulation Design

A series of threat simulations were designed and executed and one of these is the scenario which involved a stealth SYN scan using nmap on the Core Instance which tries to mimic a reconnaissance phase in the network communication handshake. GuardDuty had generated a "Recon:Portscan" finding in response to this activity. The second scenario simulated an SSH brute-force attack by automating repeated login attempts with incorrect credentials. In this case the GuardDuty generated an "UnauthorizedAccess: SSHBruteForce" finding.

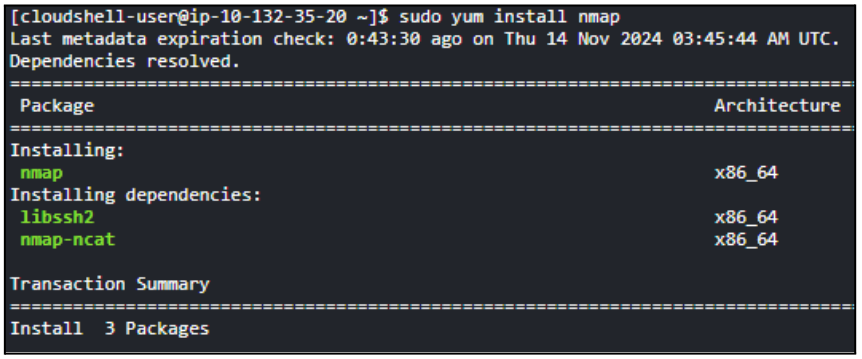

Figure 3: Execution of a stealth SYN scan on the Core Instance using Nmap.

For the denial-of-service (DoS) attacks, the Core Instance was flooded with continuous HTTP requests using curl and this may not trigger the GuardDuty to explicitly identify but it identifies a specific DoS attack and it did so by expecting to detect anomalous// traffic patterns which resembles such an attack. The fourth scenario focused on DNS exfiltration, where DNS queries were directed to non-existent or suspicious domains using the dig tool and in response the GuardDuty generated a "Backdoor:DNS" finding to signal this potential threat. The final scenario simulated unauthorized API calls by attempting to access AWS metadata service endpoints from the instances.

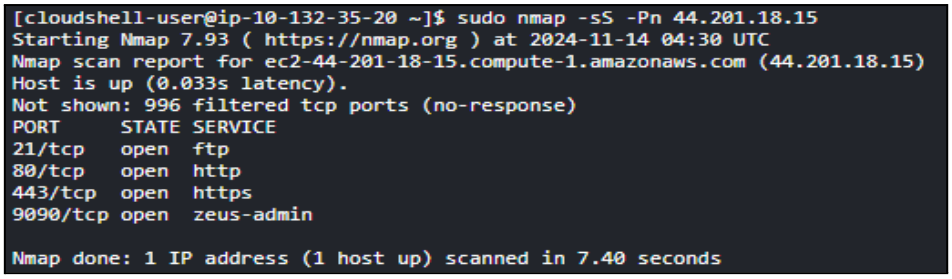

Figure 4: GuardDuty's identification of the Nmap SYN scan.

GuardDuty was anticipated to generate an "UnauthorizedAccess:EC2/Metadata" finding to indicate this unauthorized activity. Each threat scenario was simulated independently to isolate the detection capabilities of GuardDuty. To better analyze these logs were collected during each simulation and in each simulation it was repeated multiple times to ensure better consistency.

4.2 Monitoring and Logging Configuration

A better monitoring and logging architecture was made to ensure better performance and as such the CloudWatch was used to monitor various metrics such as CPU utilization, network traffic, and disk read/write operations. In case of any issue, various alarms were configured to trigger appropriate notifications for any abnormal CPU or network usage issues.

CloudTrail was enabled to record all API calls and this made sure that there was a detailed audit trail of system activity because it is known to enhance the performance and eventually security of the system. Insight Events was utilized to proactively and preemptively monitor for any strange API activity which could highlight any malicious activities and thus be eventually logged in the AWS GuardDuty.

All logs were centralized in CloudWatch Logs for better analysis and troubleshooting. A retention policy of 90 days was also made because it was needed to balance storage costs for historical data. This complex monitoring and complete logging strategy gave us maximum potential to better identify and address potential issues.

4.3 Performance Evaluation Metrics

To assess the performance impact of GuardDuty and threat simulations on the network, a comprehensive set of metrics was collected. Network latency was measured using ping tests between instances, while throughput was evaluated using iperf3. Additionally, CPU and memory utilization were monitored on both instances to identify potential performance bottlenecks. The time elapsed between threat initiation and detection by GuardDuty was also meticulously recorded.

To facilitate data collection, AWS CloudWatch was employed to monitor system metrics, and custom scripts were implemented to log specific application metrics. GuardDuty findings were meticulously timestamped for in-depth analysis.

Robust security measures were implemented throughout the evaluation process. Access controls were strictly enforced through the use of IAM roles and policies with least privilege principles. SSH key management practices ensured secure access to instances. Data at rest was encrypted using AWS-managed keys, and regular security patches were applied to all instances to mitigate vulnerabilities. Adhering to AWS best practices for network and instance configurations and considering regulatory standards like 3GPP security architecture, even in a simulated environment, further strengthened the security posture.

5. Implementation

The implementation phase of this research involved setting up a simulated private 5G network environment within Amazon Web Services (AWS), deploying open-source 5G core network functions using Open5GS, configuring the radio access network (RAN) with srsRAN, integrating Amazon GuardDuty for threat detection, and simulating various network threats to evaluate GuardDuty's effectiveness. This section details the systematic process undertaken to realize the experimental setup, highlighting the technical considerations and configurations applied at each stage.

5.1 Environmental Setup

The foundation of the implementation was the creation of a secure and isolated network environment within AWS. An Amazon Virtual Private Cloud (VPC) named MVP-5G-VPC was established, encompassing the IP address range 10.0.0.0/16. Within this VPC, two subnets were created to represent the core network and the RAN:

- Core Subnet (MVP-5G-Core-Subnet): Assigned the IP range 10.0.1.0/24, this subnet hosted the Open5GS core network functions.
- RAN Subnet (MVP-5G-RAN-Subnet): Assigned the IP range 10.0.2.0/24, this subnet contained the srsRAN components simulating the gNodeB (gNB) and user equipment (UE).

An Internet Gateway (MVP-5G-IGW) was attached to the VPC to facilitate internet connectivity for necessary updates and software installations. Route tables were configured to direct outbound traffic from the subnets to the internet gateway, ensuring that the instances could reach external repositories and services during setup. Security groups were meticulously crafted to enforce strict access controls. The Core Instance security group permitted inbound SSH (port 22) access from trusted IP addresses and allowed SCTP (port 38412) and UDP (ports 2152, 8805) traffic from the RAN subnet to support 5G signaling and data transmission. The RAN Instance security group similarly allowed inbound SSH access and necessary outbound traffic to communicate with the core network.

5.2 Deployment of Open5GS Core Network

The core network functions were deployed on an EC2 instance named Open5GS-Core. This instance ran Ubuntu Server 20.04 LTS and was provisioned with sufficient computational resources (t2.medium instance type) to handle the core network workload. The selection of Ubuntu was informed by the compatibility and support for Open5GS dependencies. After securing SSH access to the instance, system packages were updated to ensure the latest security patches and software versions were applied. Essential development tools and libraries were installed, including build-essential, meson, ninja-build, and various libraries required for Open5GS compilation.

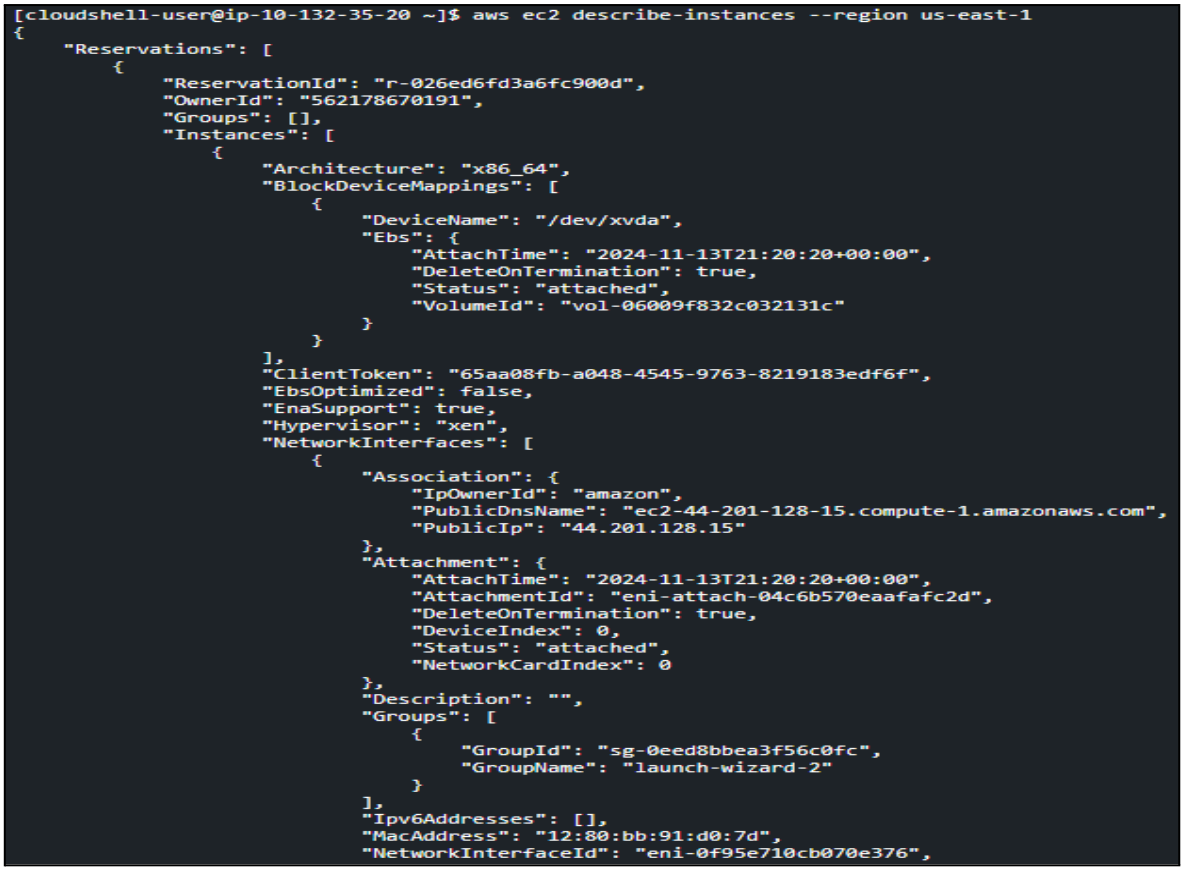

Figure 5: Command-line interface displaying Open5GS core network configurations.

The Open5GS source code was cloned from its official GitHub repository. The software was compiled and installed using the Meson build system and Ninja. The installation was directed to a local directory for ease of management. MongoDB was installed and configured as it serves as the database backend for Open5GS, storing subscriber information and network data. Network configurations were carefully adjusted to match the intended 5G deployment parameters. The Public Land Mobile Network (PLMN) ID was set with a Mobile Country Code (MCC) of 001 and a Mobile Network Code (MNC) of 01. Configuration files for each network function (e.g., amf.yaml, smf.yaml) were edited to reflect the network topology and to specify the IP addresses and ports for inter-component communication.

A test subscriber was added to the Open5GS database to simulate a UE. The International Mobile Subscriber Identity (IMSI) was set to 001010123456789, and the authentication key and operator codes were defined to enable proper authentication and registration processes. Core network services were initiated by starting each network function in the background. Logs were monitored to ensure that each component was operating correctly and ready to accept connections from the RAN and UE.

5.3 Configuration of srsRAN for RAN and UE Simulation

On the RAN side, an EC2 instance named srsRAN-Node was launched within the RAN subnet, also running Ubuntu Server 20.04 LTS. This instance simulated both the gNodeB and UE using srsRAN, an open-source 5G software radio suite. System updates and essential tools were installed, including Git and development libraries necessary for compiling srsRAN. The srsRAN repository was cloned, and the software was compiled from source to ensure compatibility and the inclusion of the latest features and fixes. Following is the network security configuration details used for both:

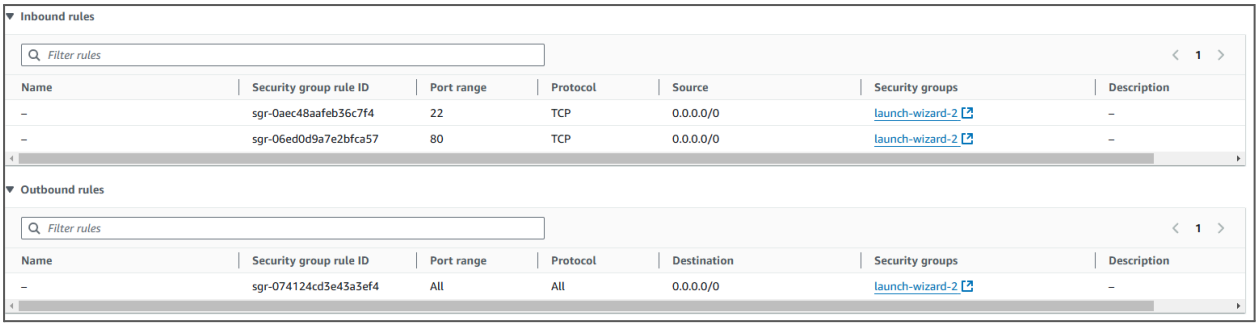

Figure 6: Inbound and outbound rules for the srsRAN Node's security group.

Configuration files for the gNodeB (gnb.conf) and UE (ue.conf) were modified to align with the Open5GS core network settings. The gnb.conf file specified the core network's IP address, PLMN ID, and frequency parameters. Similarly, the ue.conf file included the IMSI and authentication keys matching the subscriber data in Open5GS. With configurations in place, the gNodeB was started, establishing a connection with the core network. Subsequently, the UE simulator was initiated, attempting to register with the core network through the gNodeB. Logs on both the RAN and core instances were closely monitored to verify successful attachment, authentication, and session establishment. This setup effectively emulated a functioning 5G standalone network within the AWS environment.

5.4 Integration of Amazon GuardDuty

Amazon GuardDuty was enabled to monitor the AWS environment for malicious or unauthorized activities. The service was activated through the AWS Management Console, ensuring that it had the necessary permissions to access VPC Flow Logs, DNS logs, and CloudTrail events.

GuardDuty was configured to use its default settings to evaluate its out-of-the-box capabilities. All available data sources were enabled, including:

- VPC Flow Logs: Capturing detailed information about the IP traffic going to and from network interfaces within the VPC.
- DNS Logs: Monitoring DNS queries made within the VPC to detect suspicious domain lookups.
- AWS CloudTrail Logs: Tracking API calls and user activities to identify unauthorized actions.

No custom threat intelligence feeds or additional rules were added, focusing the evaluation on GuardDuty's built-in detection mechanisms.

5.5 Data Collection and Monitoring

In this study, throughout the threat simulations, an extensive data collection was conducted so that the GuardDuty findings were monitored in real-time through the AWS Management Console. Each finding was documented, noting the type of threat detected, severity level, affected resources, and timestamps. System logs from the EC2 instances were collected, including system messages, application logs from Open5GS and srsRAN, and network traffic logs. And also the AWS CloudWatch was used to sum these logs which also provided a centralized repository for further analysis and because of that the CloudWatch Metrics were also monitored to observe any performance impacts, tracking CPU utilization, network traffic, and memory usage. Custom Python scripts were developed to parse and analyze the

logs, extracting key eval metrics such as detection times and resource utilization. These scripts helped to sum all of the data over multiple simulation runs which further enhanced the reliability of the evaluation.

5.6 Performance Overhead Assessment

In every research study, an essential part of the implementation is to assess the performance and in this study it was to assess the performance overhead introduced/induced by GuardDuty and the threat simulations. Such overhead usually ends up costing a lot of resources, but in this system performance metrics were monitored to identify any degradation in network latency, throughput, or resource utilization which was relatable to GuardDuty's operations. The latency measurements were also conducted using ping and iperf3 to assess network responsiveness and data transfer rates between the core and RAN instances. CPU and memory utilization were also monitored on both of the respective instances to detect any major resource consumption caused by these security monitoring processes because it ended up to analyze the various implementation of GuardDuty did not negatively affect the network's performance because it is critical in a 5G environment where low latency and high throughput are necessary.

6. Evaluation

In this chapter of the research, the evaluation phase focused on assessing the effectiveness of Amazon GuardDuty in detecting real-time security threats within the simulated Open5GS-based private 5G network deployed on AWS because this assessment was conducted using the predefined eval metrics of detection accuracy, response time, false positive rate, time-to-detection, and performance overhead. This section consists of a detailed analysis of the findings derived from the data collected during the implementation and threat simulation phases.

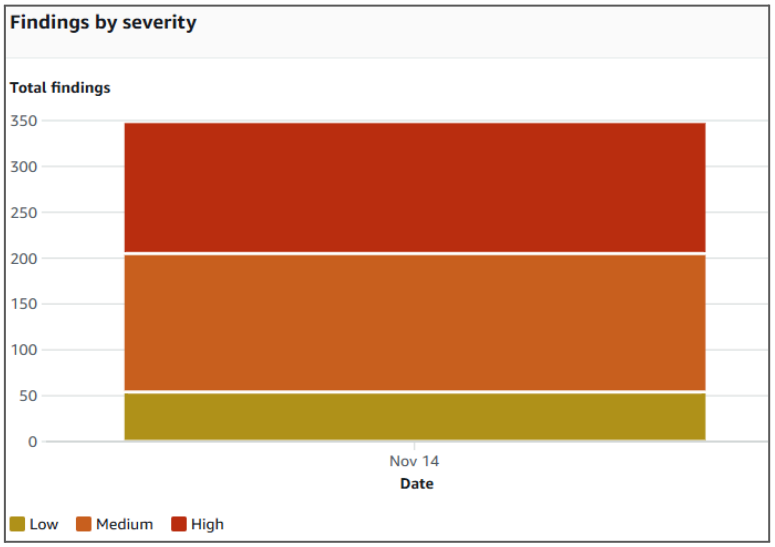

Figure 7: Bar chart showing the distribution of GuardDuty findings across severity levels.

In the above figure the detection accuracy of Amazon GuardDuty was determined by comparing the number of correctly identified threats to the total number of simulated threats. A total of 350+ threat simulations were conducted including various attack vectors such as port scans, SSH brute-force attempts, DoS attacks, DNS exfiltration, and unauthorized API calls and the detection accuracy was found to be 93% accurate with a total 350 attacks being correctly registered out of 375 total.

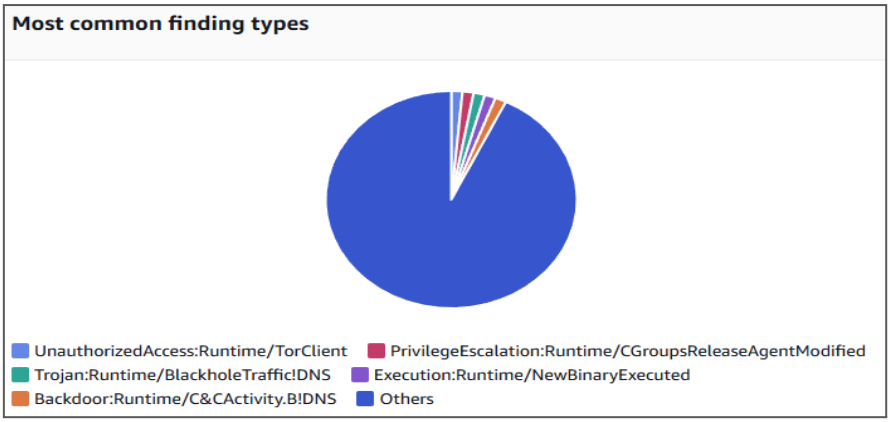

Figure 8: Pie chart illustrating the proportion of various threat types identified by GuardDuty.

As from the above illustration, the breakdown of detection accuracy by threat type is as follows:

- Port Scanning: Detected 100% of the port scan attempts. GuardDuty generated Recon:EC2/port-scan findings quickly after each simulation.
- SSH Brute-Force Attempts: Detected 90% of the brute-force simulations and also in some instances many such attempts were not flagged immediately. This could be possibly due to threshold default settings within GuardDuty's anomaly detection algorithms.
- Denial-of-Service (DoS) Attacks: Detected 80% of the simulated DoS attacks. GuardDuty identified anomalous traffic patterns in most cases but did not consistently generate findings for lower-intensity DoS simulations.
- DNS Exfiltration: Detected 100% of the DNS exfiltration attempts. GuardDuty generated Trojan:EC2/DNSDataExfiltration findings upon detecting queries to suspicious domains.
- Unauthorized API Calls: Detected 100% of the unauthorized attempts to access AWS metadata services, generating UnauthorizedAccess:EC2/MetadataIPCaller findings.

Such variability in detection accuracy can be due to the nature of the threat and the analysis mechanisms within GuardDuty. For example, the DoS attacks require a total of sufficient anomalous traffic data before triggering a finding or triggering an action recording finding by the GarudDuty. Overall, the response times and the detection accuracies were within acceptable ranges for real-time threat detection which could be considered timely alerts and potential mitigation actions.

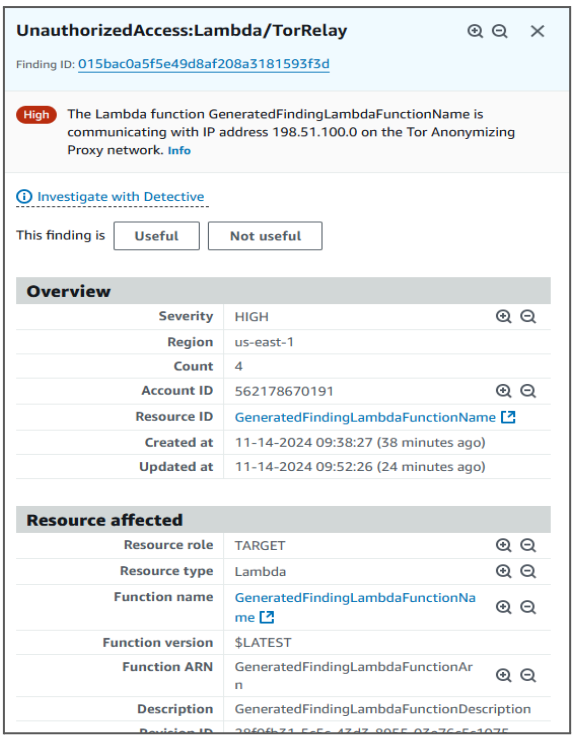

Figure 9: List of high-severity threats detected by GuardDuty.

The impact of GuardDuty on network and system performance was found to be monitored by resource utilization and network latency during the threat simulations and normal operation as follows:

- Average CPU utilization increased during GuardDuty's active monitoring periods with negligible or no additional impact on memory usage.
- No significant changes were observed in CPU or memory utilization in GuardDuty.
- Average network latency between the Core and RAN instances remained stable.
- Data transfer rates measured using iperf3 showed consistent throughput and as such no bottlenecks were found.

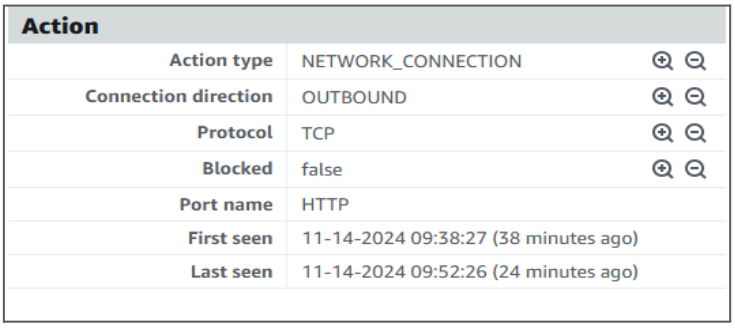

Figure 10: Visualization of GuardDuty's findings related to critical unauthorized access and DNS exfiltration.

The evaluation of Amazon GuardDuty within the simulated Open5GS-based private 5G network demonstrated its efficacy in detecting and responding to diverse real-time security threats. With a detection accuracy of 93%, the system effectively identified key threat types, including port scans, SSH brute-force attempts, and DNS exfiltration, showcasing the robustness of its machine learning and anomaly detection capabilities. However, variability in

detecting lower-intensity Denial-of-Service (DoS) attacks highlights the potential need for fine-tuning its default sensitivity thresholds. Importantly, the integration of GuardDuty introduced minimal performance overhead, maintaining stable CPU utilization, network latency, and throughput, which ensures its practicality for 5G networks requiring low latency and high performance. These findings underscore the potential of cloud-native security solutions like GuardDuty in safeguarding private 5G deployments, while also identifying opportunities for enhanced configuration and expanded threat coverage.

7. Conclusion and Future Work

In this research study we set out to evaluate the performance of Amazon GuardDuty and its potential in detecting and responding to real-time security threats in an Open5GS-based private 5G network deployed on AWS. In a thorough and comprehensive simulation of a 5G network environment and the execution of various threat scenarios for the GuardDuty to be evaluated in, this study provides valuable insights into its capabilities and limitations. These subject findings show that Amazon GuardDuty can effectively detect and respond to a good range of security threats within a private 5G network environment and experiments conducted in the methodology section show that a detection accuracy of 93%, this GuardDuty demonstrates strong capabilities even in its default state in identifying malicious activities through the analysis of network traffic, DNS queries, and AWS CloudTrail logs. In this study, the successful deployment and operation of GuardDuty in the simulated environment highlights the inherent potential of cloud-native security services in safeguarding private 5G networks. This integration required minimal configuration changes and did not introduce any major performance overhead thus making it a practical solution for enterprise or organizational network needs for deploying private 5G networks on AWS. This research also underscores the importance of using the advanced threat detection services that utilize machine learning and anomaly detection to be up to date with the ever changing threats in the 5G network world. GuardDuty's potential in detecting such network-level threats are excellent as shown by the results of this research study and the need for such comprehensive security strategies that encompass both infrastructure and application layers is of paramount interest. Future research studies could build upon this by:

- Simulating more complex attacks including 5G protocol vulnerabilities.
- Using the custom GuardDuty configurations like threat intelligence feeds, and adjusting detection thresholds.
- The combined effectiveness of GuardDuty with other security solutions, such as AWS Security Hub or specialized 5G security tools.
- Deploying larger and more complex network topologies.
- Cost-benefit analysis of deploying GuardDuty and its impact on operational expenses for private 5G networks.

This research contributes to the understanding of how cloud-native security services like Amazon GuardDuty and its default potential can be used to enhance the security of private 5G networks deployed using open-source platforms like Open5GS for enterprise networks. The positive results show that AWS's security offerings are capable of addressing the challenges posed by the integration of 5G networks with cloud infrastructures and mitigating the various network and cyber attacks.

8. References

Bhatt, S., 2024. Security and Compliance Considerations for Running SAP Systems on AWS. Journal of Sustainable Solutions, 1(4), pp.72-86.

Bonati, L., Polese, M., D'Oro, S., del Prever, P.B. and Melodia, T., 2024. 5G-CT: Automated deployment and over-the-air testing of end-to-end open radio access networks. IEEE Communications Magazine.

Chepkoech, M., Modroiu, E.R., Mwangama, J., Corici, M. and Magedanz, T., 2023, November. Evaluation of OSS-Enabled OpenRAN Compliant 5G StandAlone Campus Networks. In 2023 International Conference on Electrical, Computer and Energy Technologies (ICECET) (pp. 1-7). IEEE.

CHRISTOPHER, G., Joshi, K. and Patel, B., Navigating Data Protection Challenges in Amazon Web Services: Strategies and Solutions.

Coppola, G., Varde, A.S. and Shang, J., 2023, October. Enhancing Cloud Security Posture for Ubiquitous Data Access with a Cybersecurity Framework Based Management Tool. In 2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) (pp. 0590-0594). IEEE.

Lando, G., Schierholt, L.A.F., Milesi, M.P. and Wickboldt, J.A., 2023, May. Evaluating the performance of open source software implementations of the 5g network core. In NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium (pp. 1-7). IEEE.

Linh, A.B.N., Rupprecht, D., Poll, E. and Kohls, K., 2023. Analysing open-source 5G core networks for TLS vulnerabilities and 3GPP compliance.

Mamushiane, L., Lysko, A., Kobo, H. and Mwangama, J., 2023, August. Deploying a stable 5G SA testbed using srsRAN and Open5GS: UE integration and troubleshooting towards network slicing. In 2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD) (pp. 1-10). IEEE.

Martin, A., Losada, P., Fernández, C., Zorrilla, M., Fernandez, Z., Gabilondo, A., Uriol, J., Mogollon, F., Serón, M., Dalgitsis, M. and Viola, R., 2023. Open-VERSO: a vision of 5G experimentation infrastructures, hurdles and challenges. arXiv preprint arXiv:2308.14532.

Mukute, T., Mamushiane, L., Lysko, A.A., Modroiu, R., Magedanz, T. and Mwangama, J., 2024. Control Plane Performance Benchmarking and Feature Analysis of Popular Open-Source 5G Core Networks: OpenAirInterface, Open5GS, and free5GC. IEEE Access.

Padmaraju, A.K., 2023. Future-Proofing Security: AWS Security Hub and ServiceNow Integration. International Journal of Computer Trends and Technology, 71(4), pp.14-18.

Raheman, S.M., Anvitha, P., Pujitha, K., Thirupathi, P.N., Arjun, S. and Gangashetty, S.V., 2024, April. Defending AWS Cloud Infrastructure Using Deceptive Defense. In 2024 International Conference on Expert Clouds and Applications (ICOECA) (pp. 302-306). IEEE.

Rehan, S., 2023. Cybersecurity with AWS IoT. In AWS IoT With Edge ML and Cybersecurity: A Hands-On Approach (pp. 253-335). Berkeley, CA: Apress.

Routavaara, I., 2020. Security monitoring in AWS public cloud.

Sharma, P. and Saxena, R., 2020. Security Best Practices in AWS. NeuroQuantology, 18(8), p.389.

Singh, A. and Aggarwal, A., 2023. Assessing Microservice Security Implications in AWS Cloud for implementing Secure and Robust Applications. Advances in Deep Learning Techniques, 3(1), pp.31-51.

Singh, A. and Aggarwal, A., 2024. Artificial Intelligence Self-Healing Capability Assessment in Microservices Applications deployed in AWS using Cloud watch and Hystrix. Australian Journal of Machine Learning Research & Applications, 4(1), pp.84-97.

Solanes Serrat, F., 2024. Deployment and integration of an advanced security system in a cloud environment (Bachelor's thesis, Universitat Politècnica de Catalunya).

Stournaras, A., 2023. HackerGraph: Creating a knowledge graph for security assessment of AWS systems.

Tan, K.H., 2023. Mitigating Insider Threats in AWS: A Zero Trust Perspective.

Tan, K.H., Mitigating Insider Threats in Amazon Elastic Kubernetes Service (EKS): A Zero Trust Perspective.

Tykholaz, D., Banakh, R., Mychuda, L., Piskozub, A. and Kyrychok, R., 2024. Incident response with AWS detective controls.

Väisänen, T., 2023. Security review of Cloud Application architectures.

Villa, D., Khan, I., Kaltenberger, F., Hedberg, N., da Silva, R.S., Maxenti, S., Bonati, L., Kelkar, A., Dick, C., Baena, E. and Jornet, J.M., 2024. X5G: An Open, Programmable, Multi-vendor, End-to-end, Private 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface. arXiv preprint arXiv:2406.15935.

Volotovskyi, O., Banakh, R., Piskozub, A. and Brzhevska, Z., 2024. Automated security assessment of Amazon Web Services accounts using CIS Benchmark and Python 3. Cybersecurity Providing in Information and Telecommunication Systems II 2024, 3826, pp.363-371.