1. Introduction

This manual provides a step-by-step guide to setting up and configuring a simulated private 5G network for evaluating the real-time threat detection capabilities of Amazon GuardDuty. The setup is based on Open5GS for core network functions, srsRAN for radio access simulation, and Amazon Web Services (AWS) for hosting the environment. This configuration ensures a secure and realistic testbed for exploring the integration of cloud-native security services with private 5G networks. The guide will also demonstrate how to deploy threat simulation scenarios and evaluate GuardDuty's detection accuracy, response time, and performance overhead.

2. System Requirements and Libraries

Hardware Requirements:

Component	**Instance Type**	**Storage**	**Operating System**	**Network CIDR**	**Subnets**
Core Instance	t2.medium	20 GiB (gp2)	Ubuntu Server 20.04 LTS	10.0.0.0/16	Core: 10.0.1.0/24
RAN Instance	t2.medium	20 GiB (gp2)	Ubuntu Server 20.04 LTS	10.0.0.0/16	RAN: 10.0.2.0/24

Software Requirements:

Component	**Required Software**
Core Instance	Open5GS, MongoDB, build-essential, meson, ninja-build
RAN Instance	srsRAN, libuhd-dev, cmake, gcc

Dependencies:

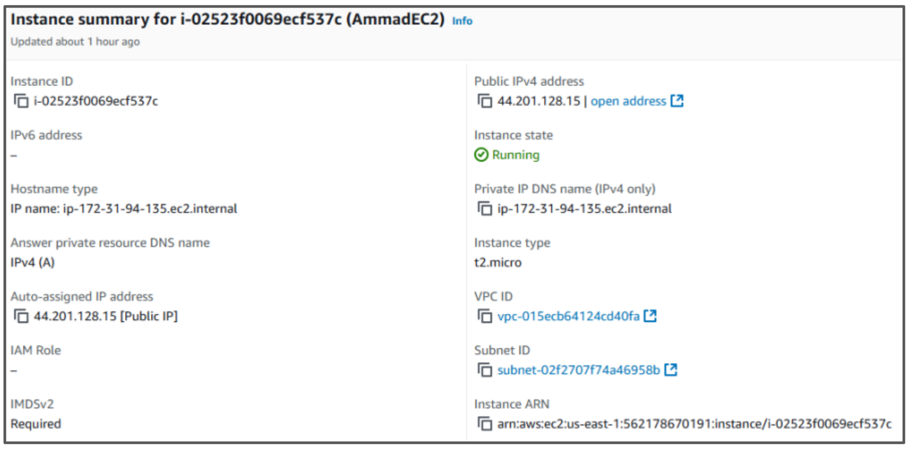
Component	**Required Tools**
Core Instance	net-tools, curl
RAN Instance	nmap, iperf3

AWS Services and Policies:

AWS Resource	**Details**
IAM Policies	AmazonEC2FullAccess, AmazonVPCFullAccess, AmazonGuardDutyFullAccess, CloudWatchLogsFullAccess
Networking	VPC with Internet Gateway, Security Groups for Core and RAN Subnets

3. Cloud Execution

a. EC2 instance details

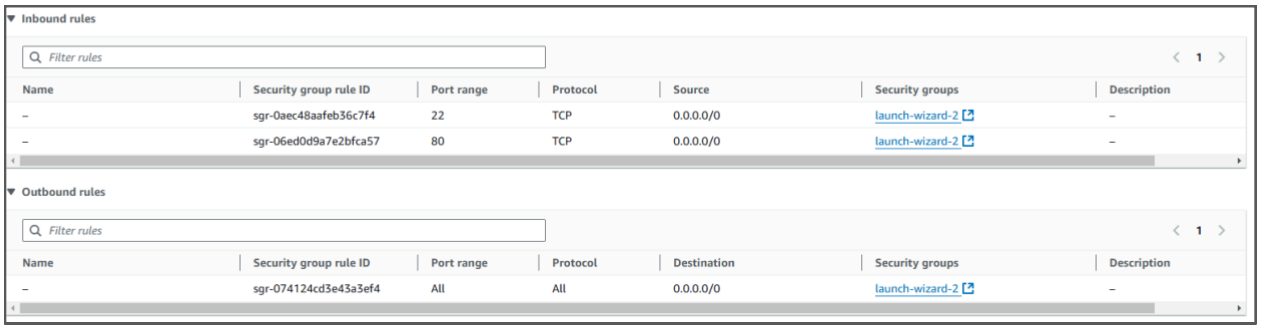


Instance summary for i-02523f0069ecf537c (AmmadEC2) Info

Updated about 1 hour ago

Instance ID i-02523f0069ecf537c	Public IPv4 address 44.201.128.15 | open address
IPv6 address –	Instance state Running
Hostname type IP name: ip-172-31-94-135.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-94-135.ec2.internal
Answer private resource DNS name IPv4 (A)	Instance type t2.micro
Auto-assigned IP address 44.201.128.15 [Public IP]	VPC ID vpc-015ecb64124cd40fa
IAM Role –	Subnet ID subnet-02f2707f74a46958b
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:562178670191:instance/i-02523f0069ecf537c

Figure 1: Summary of the EC2 instance and its details



Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
–	sgr-0aec48aafeb36c7f4	22	TCP	0.0.0.0/0	launch-wizard-2	–
–	sgr-06ed0d9a7e2bfca57	80	TCP	0.0.0.0/0	launch-wizard-2	–

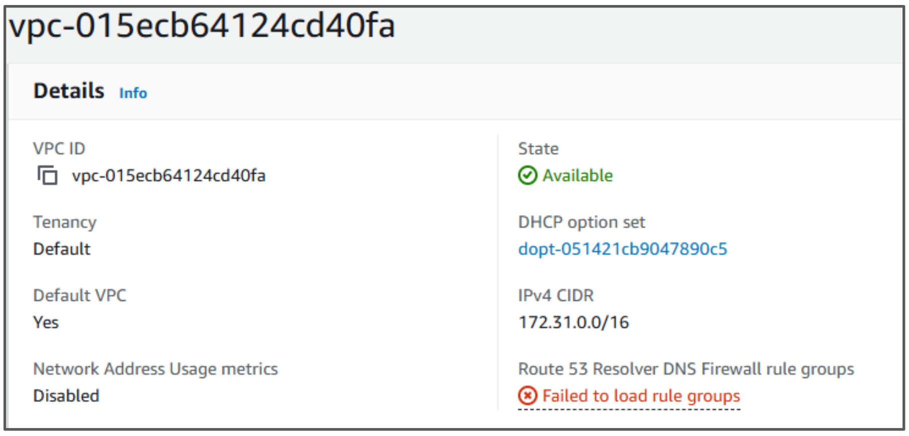
Outbound rules

Name	Security group rule ID	Port range	Protocol	Destination	Security groups	Description
–	sgr-074124cd3e43a3ef4	All	All	0.0.0.0/0	launch-wizard-2	–

Figure 2: The inbound and the outbound rules for the EC2 instance

- EC2 Instance: t2.micro running Amazon Linux 2.
- VPC: ID vpc-015ecb64124cd40fa in us-east-1.
- Security Groups: Configured for SSH (port 22) and HTTP (port 80) access.
- CloudTrail: Active multi-region trail AmmadTrail.
- GuardDuty: Enabled and active in us-east-1.
- VPC Flow Logs: AmmadFlowLog (inactive) and AmmadTestFlowLog (active).

b. Details about VPC Logs

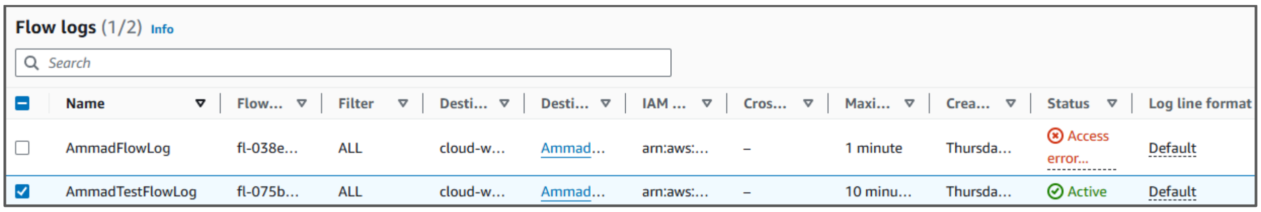


vpc-015ecb64124cd40fa

Details Info

VPC ID vpc-015ecb64124cd40fa	State Available
Tenancy Default	DHCP option set dopt-051421cb9047890c5
Default VPC Yes	IPv4 CIDR 172.31.0.0/16
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups Failed to load rule groups

Figure 3: Details of the VPC logs showcasing the current state and the VPC ID

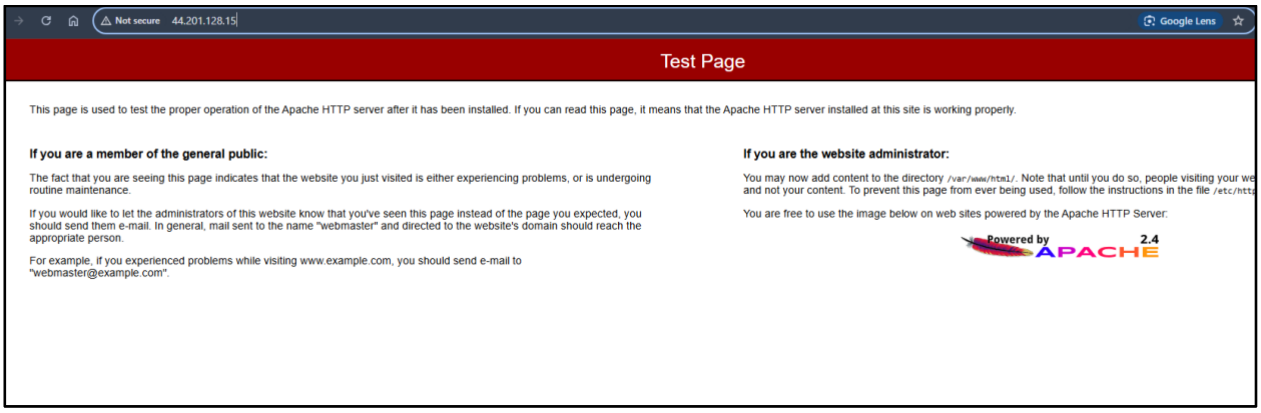


Flow logs (1/2) Info

Search

Name	Flow…	Filter	Desti…	Desti…	IAM …	Cros…	Maxi…	Crea…	Status	Log line format
AmmadFlowLog	fl-038e…	ALL	cloud-w…	Ammad…	arn:aws:…	–	1 minute	Thursda…	Access error…	Default
AmmadTestFlowLog	fl-075b…	ALL	cloud-w…	Ammad…	arn:aws:…	–	10 minu…	Thursda…	Active	Default

Figure 4: Status of the Flow Logs used to detect the network activity of the EC2 instance



Not secure 44.201.128.15

Test Page

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

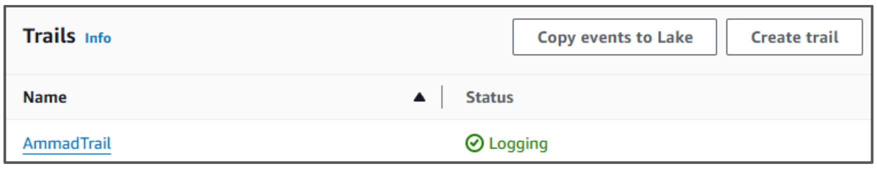
For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

If you are the website administrator:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your we and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/htt`

You are free to use the image below on web sites powered by the Apache HTTP Server:

Figure 5: The conformation site showing the public EC2 instance ip successful status

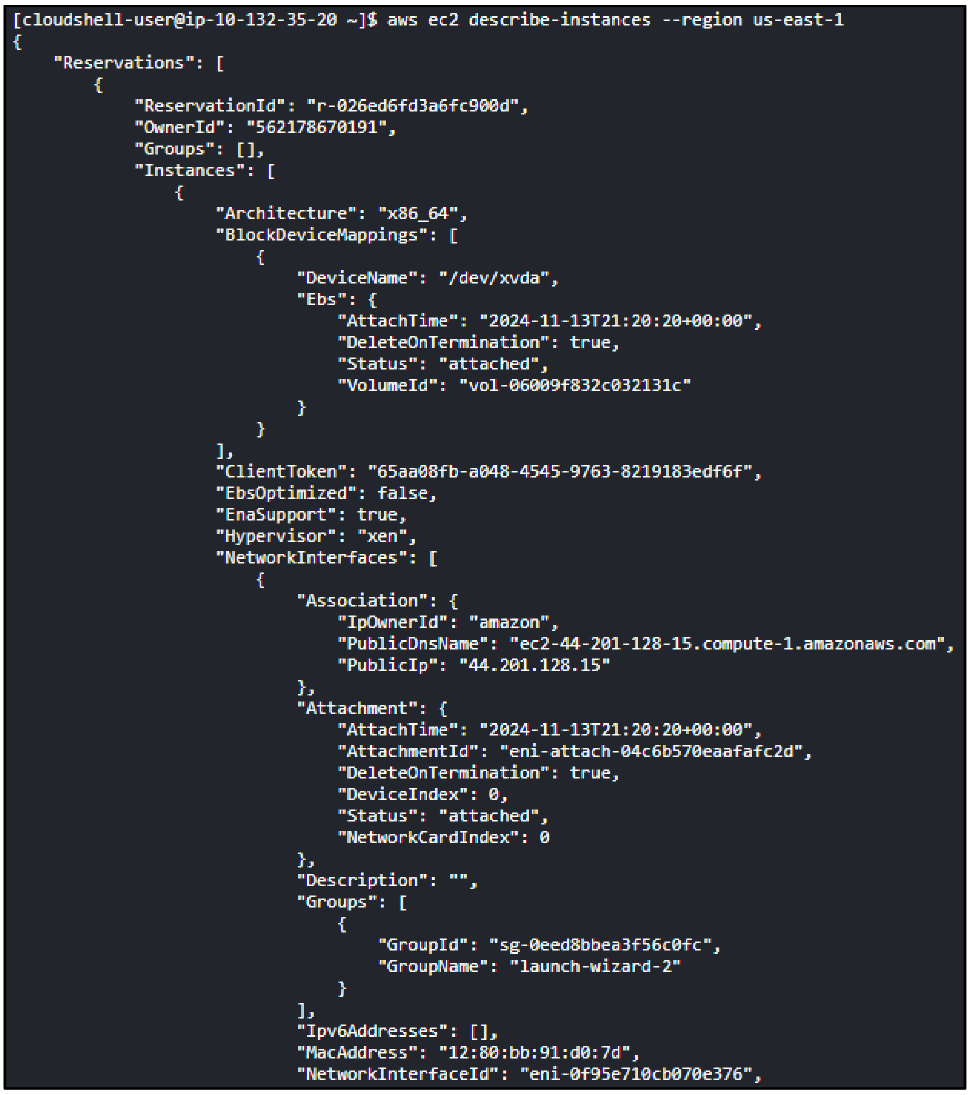


Trails Info

Name	Status
AmmadTrail	Logging

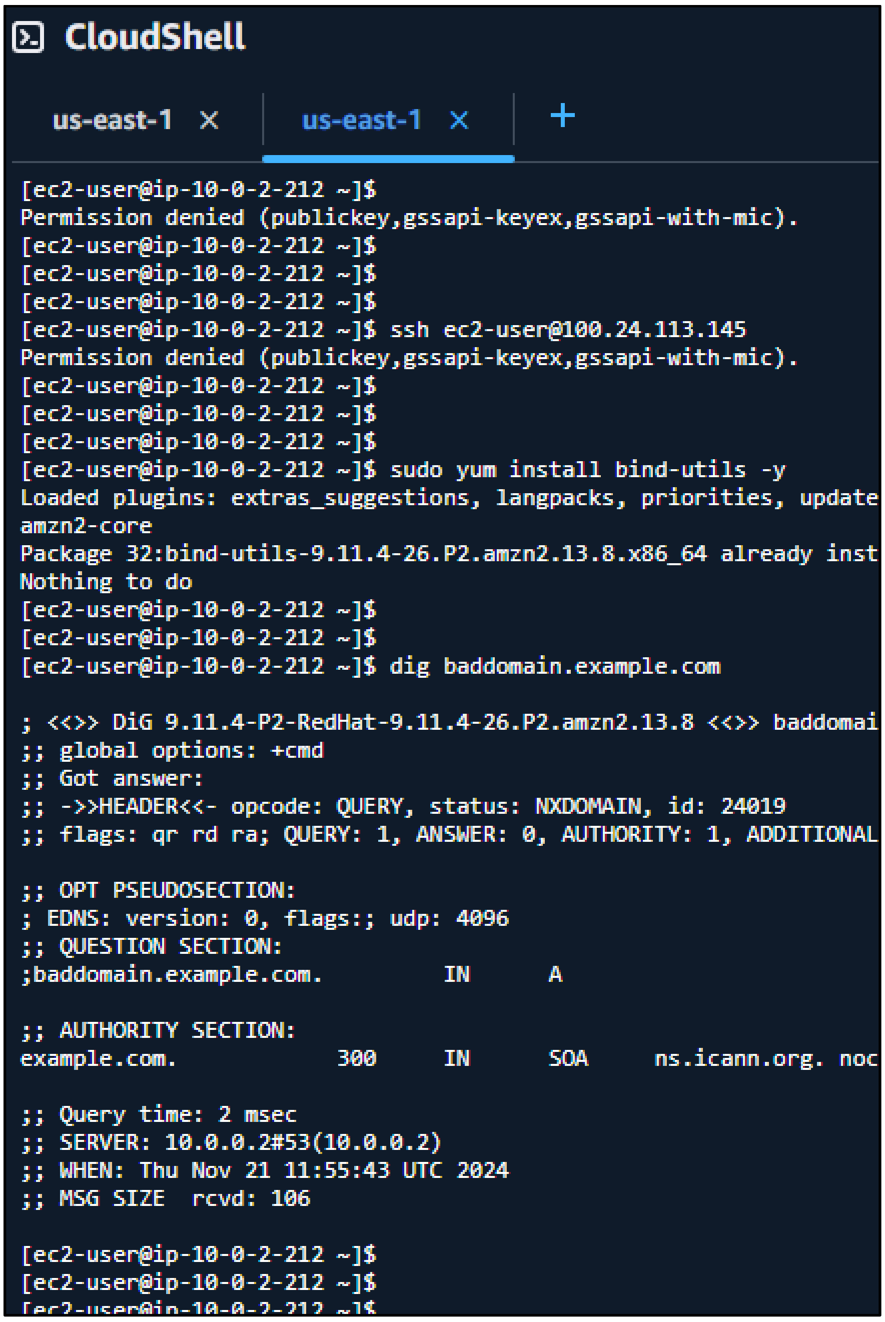
Figure 6: Status of the AWS Trails service which will be used later for the AWS Guard security.

c. Simulating the Network Activities



```
[cloudshell-user@ip-10-132-35-20 ~]$ aws ec2 describe-instances --region us-east-1
{
    "Reservations": [
        {
            "ReservationId": "r-026ed6fd3a6fc900d",
            "OwnerId": "562178670191",
            "Groups": [],
            "Instances": [
                {
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/xvda",
                            "Ebs": {
                                "AttachTime": "2024-11-13T21:20:20+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-06009f832c032131c"
                            }
                        }
                    ],
                    "ClientToken": "65aa08fb-a048-4545-9763-8219183edf6f",
                    "EbsOptimized": false,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "NetworkInterfaces": [
                        {
                            "Association": {
                                "IpOwnerId": "amazon",
                                "PublicDnsName": "ec2-44-201-128-15.compute-1.amazonaws.com",
                                "PublicIp": "44.201.128.15"
                            },
                            "Attachment": {
                                "AttachTime": "2024-11-13T21:20:20+00:00",
                                "AttachmentId": "eni-attach-04c6b570eaafafc2d",
                                "DeleteOnTermination": true,
                                "DeviceIndex": 0,
                                "Status": "attached",
                                "NetworkCardIndex": 0
                            },
                            "Description": "",
                            "Groups": [
                                {
                                    "GroupId": "sg-0eed8bbea3f56c0fc",
                                    "GroupName": "launch-wizard-2"
                                }
                            ],
                            "Ipv6Addresses": [],
                            "MacAddress": "12:80:bb:91:d0:7d",
                            "NetworkInterfaceId": "eni-0f95e710cb070e376",
```

Figure 7: Details of the EC2 instance shown in the AWS Cli

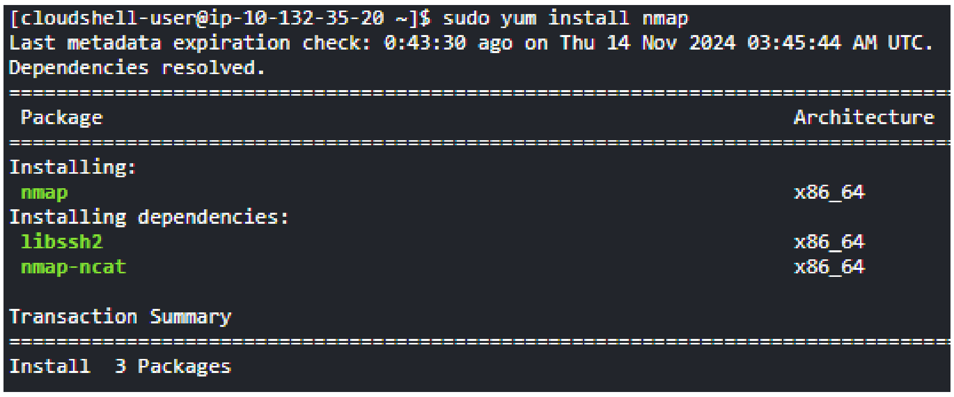


CloudShell

us-east-1 ✕ | us-east-1 ✕ | +

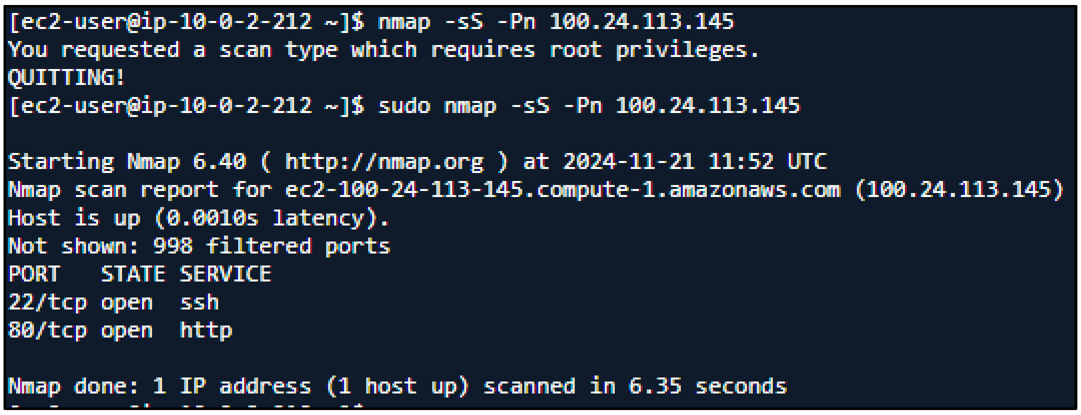
```
[ec2-user@ip-10-0-2-212 ~]$
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$ ssh ec2-user@100.24.113.145
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$ sudo yum install bind-utils -y
Loaded plugins: extras_suggestions, langpacks, priorities, update
amzn2-core
Package 32:bind-utils-9.11.4-26.P2.amzn2.13.8.x86_64 already inst
Nothing to do
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$ dig baddomain.example.com

; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.amzn2.13.8 <<>> baddomai
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 24019
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;baddomain.example.com.         IN      A

;; AUTHORITY SECTION:
example.com.            300     IN      SOA     ns.icann.org. noc

;; Query time: 2 msec
;; SERVER: 10.0.0.2#53(10.0.0.2)
;; WHEN: Thu Nov 21 11:55:43 UTC 2024
;; MSG SIZE  rcvd: 106

[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
```

Figure 8: CLI screenshot showing the various necessary installations for the attack type vectors and their dependencies.



```
[cloudshell-user@ip-10-132-35-20 ~]$ sudo yum install nmap
Last metadata expiration check: 0:43:30 ago on Thu 14 Nov 2024 03:45:44 AM UTC.
Dependencies resolved.
================================================================================
 Package                                                         Architecture
================================================================================
Installing:
 nmap                                                            x86_64
Installing dependencies:
 libssh2                                                         x86_64
 nmap-ncat                                                       x86_64

Transaction Summary
================================================================================
Install  3 Packages
```

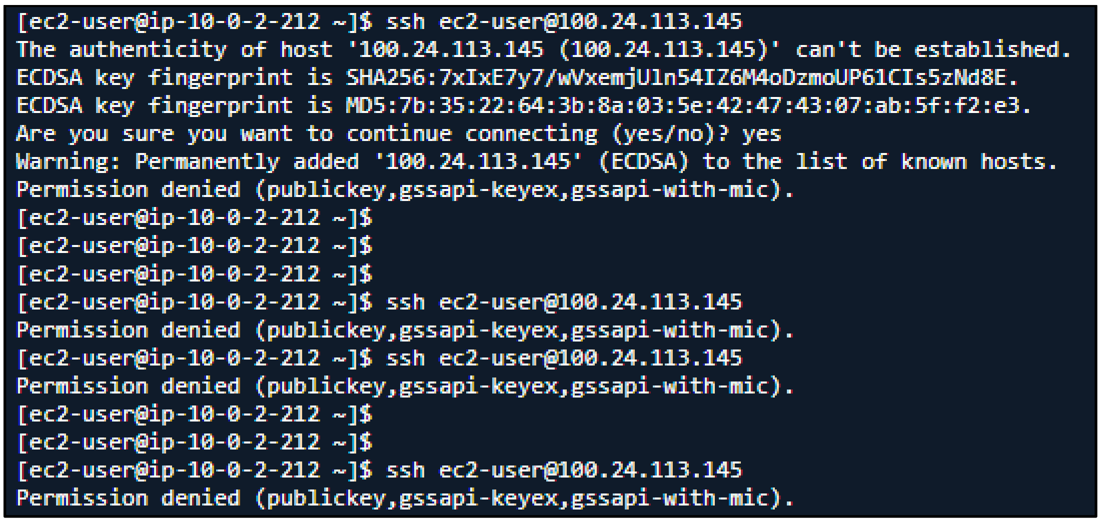
Figure 9: The nmap package installation on the RAN attacker EC2 instance



```
[ec2-user@ip-10-0-2-212 ~]$ nmap -sS -Pn 100.24.113.145
You requested a scan type which requires root privileges.
QUITTING!
[ec2-user@ip-10-0-2-212 ~]$ sudo nmap -sS -Pn 100.24.113.145

Starting Nmap 6.40 ( http://nmap.org ) at 2024-11-21 11:52 UTC
Nmap scan report for ec2-100-24-113-145.compute-1.amazonaws.com (100.24.113.145)
Host is up (0.0010s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 6.35 seconds
```

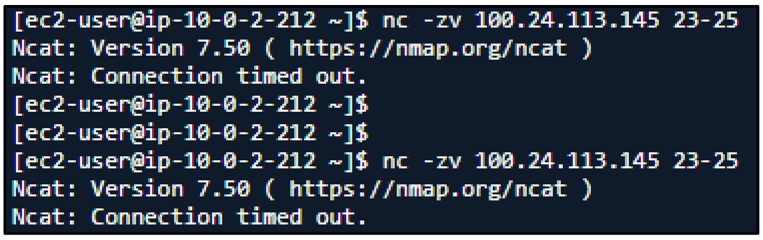
Figure 10: The nmap scanning of the Core instance launched from the RAN EC2 instance

1. Core Subnet (MVP-5G-Core-Subnet): Assigned the IP range 10.0.1.0/24, this subnet hosted the Open5GS core network functions.
2. 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).



```
[ec2-user@ip-10-0-2-212 ~]$ ssh ec2-user@100.24.113.145
The authenticity of host '100.24.113.145 (100.24.113.145)' can't be established.
ECDSA key fingerprint is SHA256:7xIxE7y7/wVxemjU1n54IZ6M4oDzmoUP61CIs5zNd8E.
ECDSA key fingerprint is MD5:7b:35:22:64:3b:8a:03:5e:42:47:43:07:ab:5f:f2:e3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '100.24.113.145' (ECDSA) to the list of known hosts.
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$ ssh ec2-user@100.24.113.145
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-10-0-2-212 ~]$ ssh ec2-user@100.24.113.145
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$ ssh ec2-user@100.24.113.145
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

Figure 11: RAN attacking the Core using the brute SSH attack to be flagged by the AWS GuardDuty.

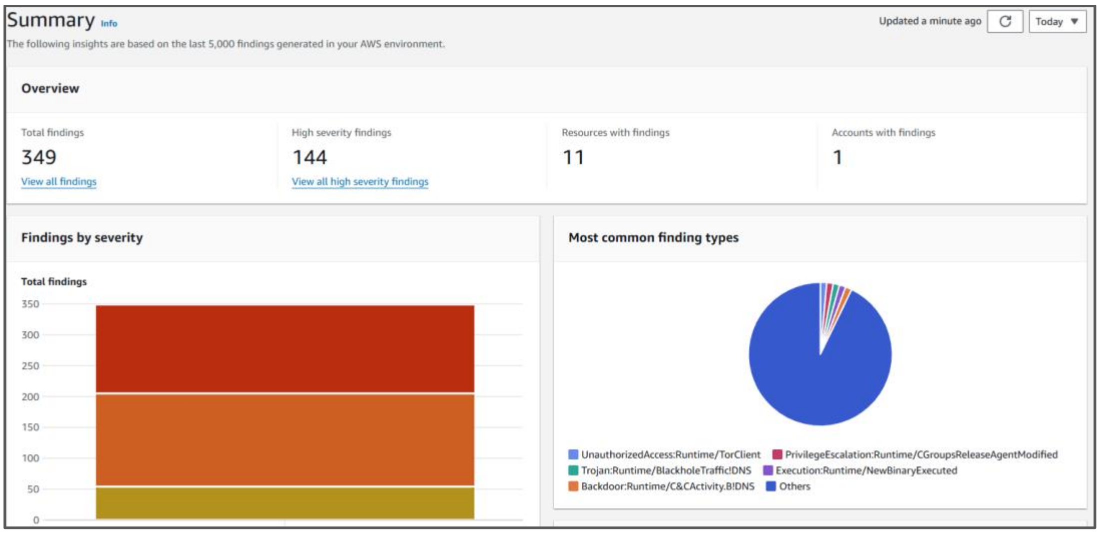


```
[ec2-user@ip-10-0-2-212 ~]$ nc -zv 100.24.113.145 23-25
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connection timed out.
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$
[ec2-user@ip-10-0-2-212 ~]$ nc -zv 100.24.113.145 23-25
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connection timed out.
```

Figure 12: DNS Exfiltration attack highlighting the sender's bad reputation which would be flagged by the AWS GuardDuty as malicious activity.

- sudo yum install bind-utils -y
- dig baddomain.example.com
- nc -zv 100.24.113.145 23-25

d. AWS GuardDuty



Summary

The following insights are based on the last 5,000 findings generated in your AWS environment.

Overview

Total findings	High severity findings	Resources with findings	Accounts with findings
349	144	11	1

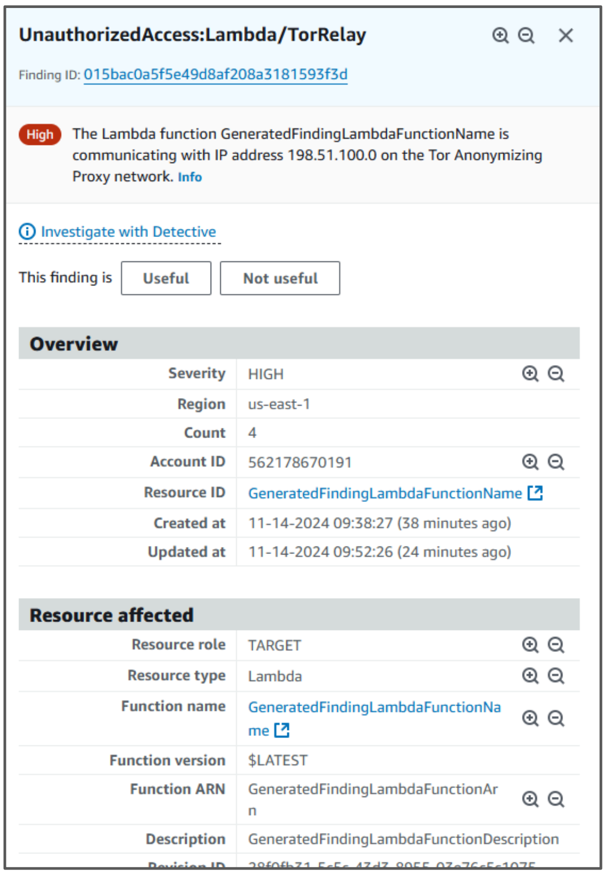
Findings by severity

Total findings

Most common finding types

- 🟦 UnauthorizedAccess:Runtime/TorClient
- 🟥 PrivilegeEscalation:Runtime/CGroupsReleaseAgentModified
- 🟩 Trojan:Runtime/BlackholeTraffic!DNS
- 🟪 Execution:Runtime/NewBinaryExecuted
- 🟧 Backdoor:Runtime/C&CActivity.B!DNS
- 🟦 Others

Figure 13: The above illustration shows the Summary of the AWS GuardDuty.



UnauthorizedAccess:Lambda/TorRelay

Finding ID: 015bac0a5f5e49d8af208a3181593f3d

High The Lambda function GeneratedFindingLambdaFunctionName is communicating with IP address 198.51.100.0 on the Tor Anonymizing Proxy network.

This finding is Useful Not useful

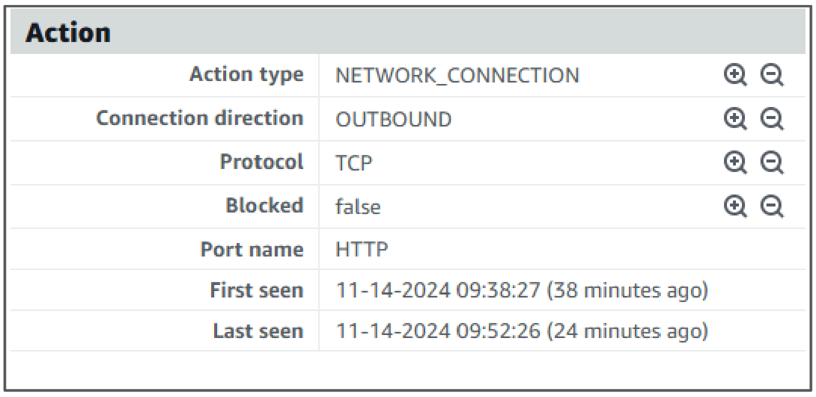
Overview

Severity	HIGH
Region	us-east-1
Count	4
Account ID	562178670191
Resource ID	GeneratedFindingLambdaFunctionName
Created at	11-14-2024 09:38:27 (38 minutes ago)
Updated at	11-14-2024 09:52:26 (24 minutes ago)

Resource affected

Resource role	TARGET
Resource type	Lambda
Function name	GeneratedFindingLambdaFunctionName
Function version	$LATEST
Function ARN	GeneratedFindingLambdaFunctionArn
Description	GeneratedFindingLambdaFunctionDescription

Figure 14: Individual TorRelay high risk attack and its details showcasing the GuardDuty findings.



Action

Action type	NETWORK_CONNECTION
Connection direction	OUTBOUND
Protocol	TCP
Blocked	false
Port name	HTTP
First seen	11-14-2024 09:38:27 (38 minutes ago)
Last seen	11-14-2024 09:52:26 (24 minutes ago)

Figure 15: Network details of the attack type and various protocol information of the attack.

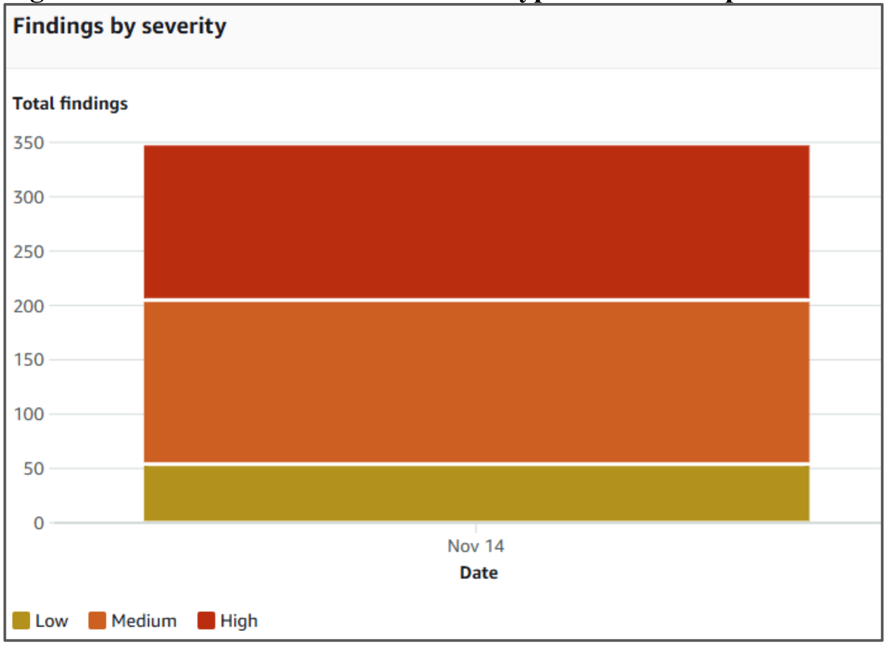


Figure 16: Detail graph of the Findings filtered by the severity of the 350+ findings.

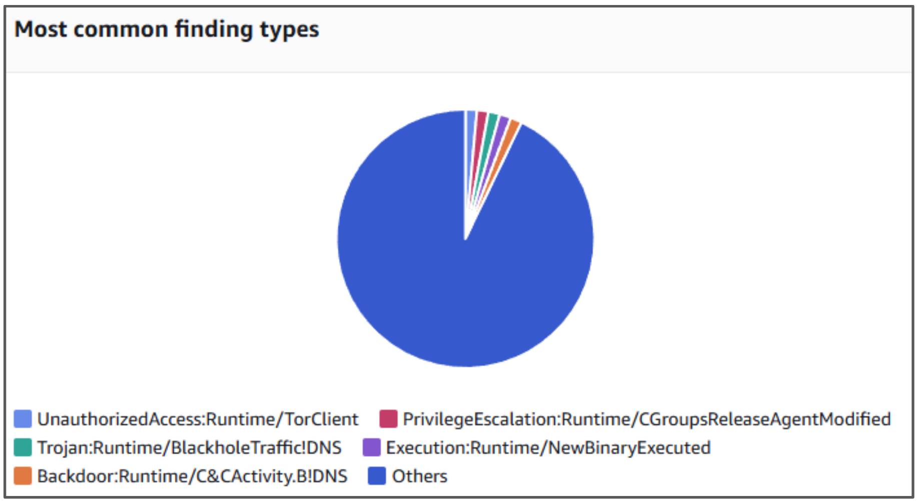


Figure 17: Most common finding types associated with the various cyber attacks recorded by the AWS GuardDuty.

References

1. Open5GS Repository: https://github.com/open5gs/open5gs
2. MongoDB Community Edition: https://www.mongodb.com/docs/manual/installation/
3. srsRAN Repository: https://github.com/srsran/srsRAN
4. Ubuntu Server 20.04 LTS: https://ubuntu.com/download/server

5. Build Tools (build-essential, cmake): https://packages.ubuntu.com/
6. Meson Build System: https://mesonbuild.com/
7. Ninja Build System: https://ninja-build.org/
8. BIND Utilities (dig): https://linux.die.net/man/1/dig
9. Netcat (nc): https://netcat.sourceforge.net/

Appendix: Detail Instructions for Result Reproduction

Instructions	**Code/Output**
Using Root Privileges	
- Access type: AWS Management Console access	
- See that following policies are there or not (atleast)	AmazonEC2FullAccess, AmazonVPCFullAccess, AmazonGuardDutyFullAccess, CloudWatchLogsFullAccess
Create VPC to Simulate 5G Network	
Navigate to VPC Console → Your VPCs → Create VPC	MVP-5G-VPC
	10.0.0.0/16
Core Subnet	MVP-5G-Core-Subnet
	10.0.1.0/24
RAN Subnet	MVP-5G-RAN-Subnet
	10.0.2.0/24
Create and Attach Internet Gateway	
Navigate to Internet Gateways → Create Internet Gateway	MVP-5G-IGW
	MVP-5G-VPC
Configure Route Tables	
Core Route Table	MVP-5G-Core-RT
	Destination 0.0.0.0/0 → Target MVP-5G-IGW
	MVP-5G-Core-Subnet
Launch EC2 Instances	
Core Instance	MVP-5G-Core-Instance
	Amazon Linux 2

	t2.micro
	MVP-5G-Core-Subnet
	SSH (22), HTTP (80)
RAN Instance	MVP-5G-RAN-Instance
	Amazon Linux 2
	t2.micro
	MVP-5G-RAN-Subnet
	SSH (22), Custom TCP (8080)
Install and Configure Software on Core Instance	
Connect to Core Instance via SSH	ssh -i ./AmmadMVPKeyPair.pem ec2-user@100.24.113.145
Update System Packages	sudo yum update -y
Install Apache HTTP Server	sudo yum install httpd -y
Start and Enable HTTP Server	sudo systemctl start httpd sudo systemctl enable httpd
Create a Simple Web Page	echo "<h1>MVP 5G Core Instance</h1>"
Install and Configure Software on RAN Instance	
Connect to RAN Instance via SSH	ssh -i ./AmmadMVPKeyPair.pem ec2-user@3.84.165.84
Update System Packages	sudo yum update -y
Install curl for HTTP Requests	sudo yum install curl -y
Simulate Network Traffic from RAN to Core	
Send HTTP Request to Core Instance	curl http://100.24.113.145
Loop HTTP Requests	while true; do curl http://100.24.113.145; sleep 1; done
Install nmap	sudo yum install nmap -y
	nmap -sS -Pn 100.24.113.145
SSH Brute Force Attack	ssh ec2-user@100.24.113.145
DNS Exfiltration	sudo yum install bind-utils -y
	dig baddomain.example.com
	nc -zv 100.24.113.145 23-25
Enable Amazon GuardDuty	
Navigate to GuardDuty Console → Enable GuardDuty	Check the findings too

Instructions	Code/Output
Launch Open5GS Core Instance (Ubuntu 20.04)	Open5GS-Core
	Ubuntu Server 20.04 LTS
	t2.medium
	MVP-5G-Core-Subnet
	SSH (22), SCTP (38412), UDP (2152, 8805) from 10.0.2.0/24
Install Dependencies on Core Instance	
Connect via SSH	ssh -i …
Update System	sudo apt update && sudo apt upgrade -y
Install Required Packages	sudo apt install -y build-essential meson ninja-build pkg-config gcc flex bison git libsctp-dev libgnutls28-dev libgcrypt-dev libssl-dev libidn11-dev libmongoc-dev libbson-dev libyaml-dev libnghttp2-dev libtins-dev
Install Open5GS	
Clone Open5GS Repository	git clone https://github.com/open5gs/open5gs.git
Build and Install Open5GS	cd open5gs meson build --prefix=pwd/install ninja -C build ninja -C build install
Start MongoDB Service	sudo apt install -y mongodb sudo systemctl start mongodb sudo systemctl enable mongodb
Add Subscriber to Open5GS	cd ~/open5gs/misc/db ./open5gs-dbctl add 001010123456789 123456789012345
Start Open5GS Core Services	cd ~/open5gs/install/bin sudo ./open5gs-mmed & sudo ./open5gs-smfd & sudo ./open5gs-amfd & (Start other services similarly)
Verify Services are Running	ps -ef
Reconfigure RAN Instance for srsRAN	
Install Dependencies on RAN Instance	sudo apt update && sudo apt install -y git build-essential cmake libconfig++-dev

Install srsRAN	git clone https://github.com/srsran/srsRAN.git cd srsRAN mkdir build cd build cmake ../ make sudo make install sudo ldconfig
Configure srsRAN	
Edit gNB Configuration	sudo nano /usr/local/etc/srsran/gnb.conf
-Set core network IP to Core Instance's private IP	
Simulate UE Connection	
Start UE Simulator on RAN Instance	sudo srsue
Monitor Core Instance Logs	sudo tail -f /var/log/open5gs/*log
Show 5G Network Operations	
Show running services	`ps -ef
Display UE attachment logs	Outputs showing UE registration and session establishment
Integrate with GuardDuty	
Simulate threats from UE/RAN to Core	Use previous threat commands
	curl http://100.24.113.145 while true; do curl http://100.24.113.145; sleep 1; done nmap -sS -Pn 100.24.113.145 ssh ec2-user@100.24.113.145 nc -zv 100.24.113.145 23-25
Monitor GuardDuty findings	Navigate to GuardDuty Console → Findings