S34lteam6 Phase 2 Final Project Report

Friday, August 29, 2025

1. Introduction

The purpose of this project was to simulate a **cloud-native cyber range on AWS** that could replicate real-world attack and defense scenarios within a secure, isolated environment. The cyber range was designed to emphasize four key principles: isolation, realism, automation, and integrated defense.

The objectives of the project were to build secure and automated infrastructure using **Terraform**, conduct offensive and defensive operations in a controlled AWS environment, and map all activities to the **MITRE ATT&CK® framework** to demonstrate visibility and detection capabilities.

The environment integrated a wide range of tools and platforms, including **AWS services** (VPC, Security Groups, NACLs, CloudTrail, VPC Flow Logs) and **Kali Linux** for offensive operations. Infrastructure as Code was implemented through **Terraform**, ensuring scalability, repeatability, and cost-effectiveness.

Collaboration was managed using **Jira** for task tracking, **Slack** and **Zoom** for communication, **When2meet** for scheduling, and **Google Docs** and **Google Slides** for shared documentation and evidence management. See Appendix A.

The team was structured into distinct roles to balance responsibilities and ensure accountability:

- **Team Leads – Janina Vallon and Maria Fraguas Jover** coordinated the project timeline, ensured quality control, and managed reporting.

- **Terraform Engineer – Tadius Frank** automated the deployment of the cyber range using Infrastructure as Code.
- **Network Architect – Bazchaun Rogers** drafted the secure AWS VPC and networking controls.
- **Offensive Security Engineers – Sheniese Aracena-Baez and Darian Lawrence** conducted reconnaissance, scanning, and exploitation with Kali Linux.
- **Defensive Security Analysts – Janina Vallon and Maria Fraguas Jover** configured AWS-native defenses and mapped detections to the MITRE ATT&CK® framework.

Each team member contributed both to the **Final Project Report** by documenting their technical work and to the **Demo Day presentation**, preparing and presenting slides aligned with their role.

2. Project Architecture & Setup

Infrastructure as Code Implementation

Overview

The Red vs Blue Team Cybersecurity Lab leverages Infrastructure as Code (IaC) principles through HashiCorp Terraform to create a fully automated, reproducible, and scalable cybersecurity training environment. This approach transforms traditional manual infrastructure provisioning into a declarative, version-controlled process that ensures consistency across deployments while reducing human error and operational overhead. The implementation demonstrates advanced cloud architecture patterns, security best practices, and educational technology integration within a single, cohesive platform.

Terraform Architecture and Design Patterns

The infrastructure implementation follows a modular, single-file approach that balances simplicity with comprehensive functionality. The main Terraform configuration (`main.tf`) employs a logical sectioning strategy that separates concerns while maintaining readability and maintainability. Provider configuration establishes AWS as the target cloud platform with version constraints ensuring compatibility and stability across different Terraform versions. The architecture leverages data sources for dynamic resource discovery, including availability zone enumeration and AMI selection, creating an adaptive infrastructure that responds to regional differences and AWS updates automatically.

Resource dependencies are carefully orchestrated through implicit and explicit dependency chains, ensuring proper provisioning order without manual intervention. The implementation utilizes local values and random resources to generate dynamic configurations, including randomized CIDR blocks for network isolation and secure password generation for team member accounts. This approach creates unique environments for each deployment while maintaining predictable structure and functionality.

Network Infrastructure Automation

The network architecture implements a sophisticated three-tier design through Terraform automation, creating distinct security zones for different operational requirements. The Virtual Private Cloud (VPC) foundation

utilizes dynamically generated CIDR blocks, ensuring network isolation between concurrent lab deployments while maintaining realistic enterprise networking patterns. Subnet creation employs calculated CIDR allocation through Terraform's built-in functions, automatically distributing IP address space across public, private application, and private database tiers.

Internet connectivity architecture demonstrates advanced cloud networking concepts through automated provisioning of Internet Gateways for public access and NAT Gateways for secure private subnet internet connectivity. Route table associations and routing rules are programmatically defined, creating proper traffic flow patterns that mirror production environments. The implementation includes Elastic IP allocation for NAT Gateway stability, ensuring consistent outbound connectivity for private resources throughout the lab lifecycle.

Detailed Network Configuration

The finalized VPC design follows a **/16 CIDR block (10.0.0.0/16)** deployed in **us-east-1** across **two Availability Zones (us-east-1a and us-east-1b)** to support high availability. Subnets were segmented into distinct tiers with dedicated purposes:

- **Public Subnets**: 10.0.1.0/24 (AZ1a) and 10.0.2.0/24 (AZ1b), hosting the Internet Gateway and Application Load Balancer.
- **Application Subnets**: 10.0.11.0/24 (AZ1a) and 10.0.12.0/24 (AZ1b), running the application servers behind the load balancer.
- **Database Subnets**: 10.0.21.0/24 (AZ1a) and 10.0.22.0/24 (AZ1b), restricted to internal-only communication.
- **Blue Team Services Subnets**: 10.0.31.0/24 (AZ1a) and 10.0.32.0/24 (AZ1b), dedicated to logging, monitoring, and defensive controls.

Routing and Connectivity were configured according to security best practices:

- Public Route Table routes `0.0.0.0/0` to the Internet Gateway.
- Private Route Tables (per AZ) route `0.0.0.0/0` through the local NAT Gateway, enabling outbound internet while preventing inbound access.
- No unintended cross-AZ routing is permitted.

Security Group Design enforced the principle of least privilege:

- **Application SG**: Allowed HTTP/HTTPS (80/443) from the ALB and controlled SSH access.
- **Database SG**: Restricted MySQL (3306) access to the Application SG only; no internet ingress.
- **Kali SG**: Outbound only, simulating external attacker traffic.
- **Blue Team SG**: Inbound restricted to administrative and log collection functions.

This layered security model ensures that each subnet and service has **only the minimal access required**, mirroring real-world enterprise segmentation. The original design included a bastion host in the public subnet, but this was removed in the final iteration to reduce the attack surface.

Security Controls and Access Management

Identity and Access Management (IAM) automation represents a critical component of the infrastructure implementation, creating role-based access controls that separate Red Team and Blue Team capabilities while maintaining security boundaries. The system generates distinct IAM groups, policies, and user accounts with programmatically assigned permissions that align with operational security principles. Red Team members receive restricted access limited to attack platform resources, while Blue Team members obtain comprehensive monitoring and incident response capabilities.

Security group automation implements network-level access controls through code, defining ingress and egress rules that create appropriate attack surfaces while maintaining realistic security postures. The implementation includes Session Manager integration through automated IAM role creation and instance profile assignment, eliminating traditional SSH access requirements while providing secure, auditable remote access capabilities. This approach demonstrates modern zero-trust architecture principles within educational contexts.

Monitoring and Alerting Automation

The infrastructure includes comprehensive security monitoring capabilities through automated CloudWatch integration, creating real-time threat detection and alerting systems. VPC Flow Logs are programmatically configured with appropriate IAM roles and CloudWatch Log Groups, enabling network traffic analysis and anomaly detection. Metric filters are automatically deployed to identify specific attack patterns, including port scanning activities, SSH brute force attempts, and suspicious database connections.

CloudWatch alarm automation creates threshold-based alerting for security events, with SNS topic integration providing immediate notification capabilities to security personnel. The monitoring infrastructure includes dashboard creation through Terraform, providing visual interfaces for threat analysis and incident response activities. This automated approach ensures consistent monitoring coverage across all lab deployments while reducing manual configuration overhead.

Configuration Management and Initialization

Server configuration automation leverages Terraform's user data capabilities to execute comprehensive initialization scripts during instance provisioning. The application server receives automated vulnerable application deployment, creating realistic attack targets with intentional security flaws including SQL injection, command injection, and directory traversal vulnerabilities. Database server initialization includes monitoring tool deployment and logging configuration for Blue Team analysis activities.

The Kali Linux attack platform receives automated penetration testing tool installation and configuration, creating ready-to-use Red Team capabilities without manual setup requirements. Configuration templates utilize Terraform's templatefile function to inject dynamic values into initialization scripts, enabling environment-specific customization while maintaining code reusability.

Scalability and Reproducibility Benefits

The Infrastructure as Code implementation provides significant advantages over traditional manual provisioning approaches, particularly in educational and training contexts. Deployment consistency ensures identical lab environments across multiple teams, eliminating configuration drift and environmental discrepancies that could impact learning outcomes. The single-command deployment process (`terraform apply`) reduces setup time from hours to minutes, enabling rapid lab provisioning for classes, workshops, and training events.

Version control integration allows infrastructure evolution tracking, enabling rollback capabilities and collaborative development of lab enhancements. The declarative configuration approach provides self-documenting infrastructure, where the Terraform code serves as both implementation and documentation of the environment architecture. Resource tagging automation ensures proper cost allocation and resource management across multiple concurrent deployments.

Cost Optimization and Resource Management

The implementation incorporates AWS Free Tier optimization through careful instance type selection and resource sizing, ensuring educational accessibility while maintaining realistic performance characteristics. Automated resource cleanup through `terraform destroy` prevents cost accumulation from forgotten resources, providing complete environment lifecycle management. The infrastructure includes cost-conscious defaults while allowing customization for different training requirements and budget constraints.

CloudWatch log retention policies are programmatically configured to balance monitoring capabilities with storage costs, demonstrating real-world operational considerations within educational contexts. The modular design enables selective resource deployment for different training scenarios, allowing instructors to customize lab complexity and cost profiles based on specific educational objectives.

3. Offensive Operations:

As part of a controlled AWS security lab exercise, we operated as the red team and simulated attack techniques to trigger Amazon GuardDuty, Security Hub, and CloudWatch findings for the blue team to investigate.

Using an Amazon Linux 2 AMI (amzn2-ami) as our attacker machine, we simulated two types of malicious activity:

- Reconnaissance → Using Nmap to identify open ports and running services (webapps) on a target EC2 instances.
- Credential Access Attempts → Using Hydra to simulate SSH brute-force attacks against the same target.

These actions were designed to generate GuardDuty findings, allowing the blue team to practice detection and analysis.

Reconnaissance with Nmap

Nmap (Network Mapper) is an open-source tool used for network discovery and security auditing. It enables attackers and penetration testers to scan target systems, identify open ports, detect running services, and determine software versions. In this lab, the Red Team used Nmap from the attacker instance to perform

reconnaissance against the vulnerable EC2 target. The purpose was not to exploit services directly but to simulate how attackers map out potential entry points before attempting exploitation. See Appendix D.

Nmap scan commands and purpose

- `nmap <target-ip>`
  Performs a basic scan to identify whether the host is reachable and which ports are open.

- `nmap -sS <target-ip>`
  Executes a **TCP SYN scan**, a common stealth scanning technique that identifies open ports without completing the full TCP handshake.

- `nmap -A <target-ip>`
  Runs an **aggressive scan**, which combines OS detection, version detection, script scanning, and traceroute to gather extensive system information.

- `nmap -sV -p- <target-ip>`
  Performs **version detection** across **all 65,535 ports**, identifying the services and software versions running on the target.

- `nmap -p 3306 --script mysql-enum <target-ip>`
  Uses the **mysql-enum script** against port 3306 to enumerate MySQL database information if exposed, simulating a database reconnaissance attempt.

Exploitation with Hydra

🔹 **Hydra** (THC-Hydra) is an open-source password-cracking tool used to perform **brute-force** attacks against authentication services, such as SSH, HTTP, FTP, and others. In security testing, Hydra automates rapid attempts to identify weak credentials and misconfigured services. In this lab, Hydra was used from the attacker machine to simulate SSH brute-force attempts against the target EC2 instance. The purpose was not to gain unauthorized access, but to generate GuardDuty findings**.** See Appendix D

- **Hydra attack commands**
  - hydra -l ec2-user -p password123 ssh://<target-ip>
    - Tests one username (ec2-user) with one password (password123) to simulate a basic failed SSH login.
  - hydra -l ec2-user -p welcome123 -t 4 ssh://<target-ip>
    - This command tells Hydra to try logging in to the EC2 instance using the username ec2-user and the password welcome123. By adding -t 4, Hydra makes four login attempts in quick succession, this speeds up testing and creates more noticeable activity. This additional activity helps Amazon GuardDuty detect simulated brute-force attacks more quickly and consistently within a short timeframe. This kind of attack can be easily noticed. This was tested in 4 and 10 quick successions.

Exploitation with curl (SQL Injection)

🔹 **curl** is a command-line tool used to interact with web applications and APIs(Application Programming Interface) by sending requests and receiving responses directly from the terminal. In this lab, the attacker machine was used to simulate a SQL injection attack against a vulnerable login form hosted on the vulnerable EC2 instance.

The goal of this simulation was not to gain unauthorized access or compromise sensitive data but to demonstrate how a SQL injection attempt would appear in a vulnerable web application. By sending a payload, we generated malicious HTTP request patterns in the web server's logs.

The payload admin' OR '1'='1' -- was designed to manipulate the SQL query used by the login form. By adding the condition OR '1'='1', the database interprets the statement as always true, while the sequence comments out the rest of the query. In a vulnerable application, this could allow an attacker to bypass authentication entirely.

- **curl attack commands**
  - curl -I http://<target-ip>/login.php
    - Asks the server for headers only (metadata) instead of downloading the entire page. This quickly checks if the web application is reachable and responding. Before we run an SQL injection, we want to make sure the web application is reachable and responding.
  - curl -X POST \ -d "username=admin' OR '1'='1' -- &password=anything" \ http://<target-ip>/login.php
    - We used -X POST with curl because login forms expect data to be sent as a POST request. Without -X POST, the server might reject the request or fail to process the submitted username and password.

The SQL injection payload was sent to the vulnerable login form, and while authentication was not bypassed, the web application displayed the executed SQL query in its response. This confirmed that the input was not properly sanitized, and the simulated attack generated identifiable HTTP request patterns. See Appendix D

In this controlled AWS lab, we simulated reconnaissance, brute-force attacks, and SQL injection testing to demonstrate common attack techniques. Nmap was used to identify open ports, communication between instances,and type and version of the web application. Hydra simulated SSH brute-force attempts to trigger GuardDuty findings, and cURL was used to test a SQL injection payload against a vulnerable web app. All activities were performed in a safe, isolated environment to generate realistic data for the blue team to practice findings and log analysis.

Attack	Tools	Kill Chain Stage	Purpose	Outcome in Lab
Reconnaissance	Nmap	Reconnaissance	Scan for open ports and services	Identified active services and endpoints
Brute Force	Hydra	Exploitation Attempt	Test multiple SSH login attempts	Triggered failed login patterns
SQL Injection	cURL	Exploitation Attempt	Send a malicious payload to the login form	Generated a vulnerable SQL query response and failure

4. Defensive Operations

Amazon GuardDuty

Purpose: GuardDuty is a managed threat detection service that uses machine learning and threat intelligence to identify suspicious behavior in your AWS environment.

Setup Steps:

1. Open the **AWS Management Console** and navigate to **GuardDuty**.
2. Click **Enable GuardDuty** for your account (region-specific).
3. (Optional but recommended) Enable **multi-account setup** if you are using AWS Organizations. This lets you centralize findings in a single master account.
4. GuardDuty begins analyzing CloudTrail logs, VPC Flow Logs, and DNS logs automatically. No additional data sources need to be configured.

Usage:

- View findings in the GuardDuty console.
- Findings are categorized (e.g., reconnaissance, unauthorized access, cryptocurrency mining).
- Integrate findings with **CloudWatch Events** or **Security Hub** for automated responses.

AWS Security Hub

Purpose: Security Hub acts as a central dashboard for all security findings, pulling in data from GuardDuty, Inspector, IAM Access Analyzer, and third-party tools. It also runs checks against industry standards like **CIS AWS Foundations**.

Setup Steps:

1. Navigate to **AWS Security Hub** in the console.
2. Click **Enable Security Hub**.
3. Enable **security standards** (e.g., CIS, PCI DSS, NIST) to benchmark your environment.
4. Integrate Security Hub with GuardDuty and other AWS services so all findings appear in one place.
5. (Optional) Connect to third-party security products through **AWS Partner integrations**.

Usage:

- Use the dashboard to see compliance status across accounts.
- Findings are normalized into a standard AWS Security Finding Format (ASFF).
- You can route findings to **CloudWatch Events** or **AWS Lambda** for automated remediation.

AWS CloudTrail

Purpose: CloudTrail records all API calls and account activity, making it the backbone of AWS forensics and auditing.

Setup Steps:

1. Go to the **CloudTrail console**.
2. Create a new trail:
    - Give it a name.
    - Choose **Apply to all regions** (best practice).
    - Select an **S3 bucket** for log storage.
    - Enable **log file validation** for integrity.
3. (Optional) Send CloudTrail logs to **CloudWatch Logs** for real-time monitoring.

Usage:

- Use CloudTrail to see *who did what and when* in your AWS account.
- Essential for investigating suspicious activity, IAM misuse, or insider threats.
- CloudTrail integrates with **GuardDuty**, which uses it as a primary data source.

VPC Flow Logs

Purpose: VPC Flow Logs capture metadata about traffic going in and out of your VPC, subnets, and network interfaces. Useful for spotting brute force attempts, port scanning, or data exfiltration.

Setup Steps:

1. Go to the **VPC console** → choose **Your VPCs**.
2. Select a VPC (or subnet or ENI) → click **Flow Logs** → **Create flow log**.
3. Choose **Filter**:
    - `ALL` → captures all traffic.
    - `ACCEPT` → only successful traffic.
    - `REJECT` → only denied traffic (useful for attack detection).
4. Select a **destination**:
    - **CloudWatch Logs** (real-time monitoring and alerts).
    - **S3 bucket** (archival and analysis with Athena).

5. Assign the IAM role automatically created for publishing logs.

Usage:

- Detect port scans (one source IP hitting many ports).
- Spot brute force attempts (many rejected connections).
- Investigate suspicious outbound traffic from compromised instances.

MITRE ATT&CK® Correlation Table

Observed Attack	MITRE ATT&CK® Technique ID	Description	Defensive Analysis	Proposed Remediation
Brute Force Login Attempt	**T1110** – Brute Force	Repeated login attempts to gain unauthorized access	GuardDuty detects unusual authentication attempts; CloudTrail logs record all IAM sign-in events; VPC Flow Logs may show repeated failed SSH attempts	- Enable MFA for all accounts- Implement IAM password policies- Restrict SSH access using Security Groups or Session Manager only- Configure CloudWatch alarm for multiple failed login attempts
Port Scan Attempt	**T1046** – Network Service Scanning	External or internal scanning of open ports to identify vulnerable services	VPC Flow Logs detect unusual connection patterns (multiple ports accessed from a single IP); GuardDuty can generate reconnaissance findings	- Block suspicious IPs via Security Group or Network ACLs- Use GuardDuty threat lists for automated detection- Enable CloudWatch metric filters for repeated rejected connection attempts
Suspicious Database Access	**T1078** – Valid Accounts (Abuse of Credentials) **T1213** – Data from Information Repositories	Unexpected queries or connections to database instances, possibly exfiltrating data	CloudTrail logs API calls to RDS/DynamoDB; VPC Flow Logs capture unauthorized DB traffic; Security Hub centralizes findings	- Limit DB access with IAM roles and fine-grained policies- Enable database logging and auditing- Monitor and alert on anomalous query patterns using CloudWatch- Rotate credentials regularly

MITRE ATT&CK® Correlation Table

Attack Technique	Tool Used	ATT&CK Tactic	ATT&CK Technique ID	Observed Purpose	Detection Source	Outcome in Lab
Port Scanning (TCP SYN, Version, Aggressive)	Nmap (`-sS` , `-A` , `-sV` , `-p-` , `--script mysql-enum`)	Reconnaissance	T1046 – Network Service Scanning	Enumerate open ports, services, and versions (e.g., MySQL on 3306)	VPC Flow Logs, GuardDuty Reconnaissance findings, CloudTrail (API enumeration)	Identified active services and endpoints for further exploitation
Brute-Force (SSH)	Hydra (`hydra -l ec2-user -p password123 ssh://<target-ip>`)	Credential Access	T1110 – Brute Force	Test multiple SSH login attempts with common credentials	GuardDuty UnauthorizedAccess:EC2/SSHBruteForce, VPC Flow Logs (multiple rejected attempts)	Triggered failed SSH login attempts; generated GuardDuty brute-force findings
SQL Injection	curl (`curl -X POST -d "username=admin' OR '1'='1' --&password=anything" http://<target-ip>/login.php`)	Initial Access / Execution	T1190 – Exploit Public-Facing Application	Exploit vulnerable login form with SQL payload to bypass authentication	Web server logs, VPC Flow Logs (HTTP POST anomalies), Security Hub (aggregated finding)	Generated identifiable SQL injection patterns; confirmed input not sanitized

5. Automation & Cost Management

Terraform Scripts for Repeatable Deployment/Teardown

The lab infrastructure uses Terraform Infrastructure as Code to automate complete environment provisioning and destruction. The main deployment workflow involves `terraform init`, `terraform plan`, and `terraform apply` to create the entire 3-tier network architecture, EC2 instances, IAM roles, and monitoring infrastructure in under 10 minutes. For teardown, `terraform destroy` removes all resources cleanly, ensuring no orphaned components remain. The infrastructure state is managed automatically, allowing identical recreation of the environment for consistent training scenarios.

Cost Control Measures (Ephemerality & Avoiding Idle Resources)

The lab is optimized for minimal cost through free tier eligible t3.micro instances, ephemeral daily operation (deploy morning, destroy evening), and elimination of idle resources. Monthly costs are kept under $5 within AWS free tier limits, with the primary expense being the NAT Gateway (~$3-5/month). The design avoids persistent storage, uses minimal 7-day CloudWatch log retention, operates in a single availability zone, and leverages Session Manager to eliminate bastion host costs. This ephemeral approach reduces weekly costs from $21-35 to just $1-2.

Issues Encountered and Troubleshooting Steps

Common deployment issues include AWS credential configuration errors (solved by verifying `aws sts get-caller-identity --profile terraform`), missing SSH key files (resolved by generating keys with `ssh-keygen`), and IP address access problems requiring updates to the `my_ip` variable in terraform.tfvars. Session Manager connection failures typically need the Session Manager plugin installation via platform-specific package managers. Resource limit errors may require using different AWS regions or requesting limit increases, while unexpected costs can be immediately addressed with `terraform destroy` and billing alert configuration.

6. Team Member Roles and Contributions

The team agreed that **all members would contribute to both the final report and the slide deck**, with each person responsible for the sections aligned with their role. This approach ensured clear accountability, while also producing a comprehensive and cohesive final deliverable.

Team Lead: Maria Fraguas Jover

- Oversaw overall project success, coordination, and timeline management.
- Coordinated team roles and tasks through Jira, ensuring transparency and accountability.
- Provided verification and quality assurance across deployments, evidence, and rubric requirements.
- Created the outline and structure for the Final Project Report and Team Contributions Document.

- Reviewed, edited, and filled gaps to ensure the final report was cohesive and complete.

- Contributed to the Final Project Report, slide deck and Demo Day presentation with a section on project workflow, coordination, and lessons learned.

- During the first week, when delays arose with the Terraform infrastructure, Maria and Sheniese worked together to develop a plan B functioning Terraform deployment, ensuring the project could move forward.

- Assumed additional responsibilities, completing the slide deck, final presentation, and key screenshots to document team progress, while filling in missed items to maintain quality and completeness.

Team Lead: Janina Vallon

- Oversaw overall project success, coordination, and timeline management

- Facilitated team communication by sending meeting announcements and reminders, supporting collaboration and timeline adherence.

- Monitored project success via Jira and coordinated with fellow Team Lead. Assisted with troubleshooting in real-time during testing.

- Contributed to the Final Project Report, slide dec,k and Demo Day presentation with a section on project workflow, coordination, and lessons learned.

Terraform Engineer: Tadius Frank

- Automated the entire cyber range infrastructure using Terraform, and Bash.

- Authored, managed, and executed Infrastructure as Code (IaC) with secure credential handling.

- Ensured repeatability, scalability, and cost-effectiveness through automation.

- Contributed to the Final Project Report, slide deck and Demo Day presentation, with a section on the Terraform automation process.

- Managed the git repo with Terraform project files and authored README.md for CI/CD and code shareability.

Network Architect: Bazchaun Rogers

- Designed the initial network topology map.

Offensive Security Engineer Lead: Sheniese Aracena-Baez

- Set up the Kali Linux environment for simulated attacks as teaching practice.

- Conducted reconnaissance, scanning, exploitation, and attack chains.

- Documented offensive actions for later correlation with defensive detections.

- Contributed to the Final Project Report, slide deck and Demo Day presentation, with a section on offensive operations and attack methodology.
- During the first week, when delays arose with the Terraform infrastructure, Maria and Sheniese worked together to develop a plan B functioning Terraform deployment, ensuring the project could move forward.

Offensive Security Apprentice: Darian Lawrence

- Conducted reconnaissance and was present during team meetings.

Defensive Security Analysts: Janina Vallon and Maria Fraguas Jover

- Analyzed AWS-native defenses (CloudTrail, VPC Flow Logs).

- Analyzed alerts and correlated findings with offensive actions using the MITRE ATT&CK® Correlation Table.

- Proposed remediation steps and strengthened the environment's defensive posture.

- Contributed to the Final Project Report, slide deck and Demo Day presentation with a section on defensive operations, detections, and ATT&CK mapping.

7. Lessons Learned & Reflection

Technical Challenges

- Encountered **password provisioning issues** after deployment, which required manual intervention through the AWS console.

- Faced **service availability limitations**: beginning July 15, 2025, new AWS accounts no longer provided free-tier access to GuardDuty or Security Hub (see Appendix F for AWS support correspondence).

- **Infrastructure delays**: although the initial due date was August 24, Terraform infrastructure was not fully functional until August 28, which impacted the timeline for simulating attacks and defensive responses.

Teamwork Lessons

- **Coordination and timing**: project execution highlighted the importance of early initiation of critical-path components (such as Terraform), since delays in foundational infrastructure affected all other dependent tasks.

- **Workload balance**: when infrastructure setup was delayed, other team members temporarily supported outside their primary roles to help accelerate progress, emphasizing the value of adaptability

but also the risks of misalignment.

- **Communication**: consistent use of collaboration tools (Jira, Slack, Zoom) proved essential, but clearer escalation practices could have reduced uncertainty and time lost.

Insights into Cloud Security and Incident Response

- **Service dependency awareness**: hands-on work reinforced how heavily cloud security operations depend on underlying service availability, IAM permissions, and correct configurations.

- **Incident response parallels**: unexpected delays and manual troubleshooting mirrored real-world incident response workflows, where teams must adapt rapidly to unanticipated service limitations or system failures.

- **Purple team synergy**: the integration of offensive and defensive activities demonstrated the importance of feedback loops in security operations, as successful detection required close alignment between simulated attacks and configured defenses.

8. Conclusion

Summary of Accomplishments

This cybersecurity lab successfully delivers a comprehensive AWS-based training environment that simulates real-world attack and defense scenarios through Infrastructure as Code. Key accomplishments include deploying a fully automated 3-tier network architecture with intentionally vulnerable applications, implementing comprehensive monitoring and alerting systems, establishing role-based access controls for Red and Blue teams, and maintaining cost-effective operations within AWS free tier limits. The lab provides hands-on experience with network reconnaissance, web application exploitation, security monitoring, incident detection, and response procedures while ensuring secure access through AWS Session Manager and automated user management.

Value of Purple-Team Approach (Attack + Defense Integration)

The integrated Red/Blue team architecture demonstrates the powerful value of the purple-team methodology by enabling simultaneous offensive and defensive operations within the same environment. Red Team members gain practical experience with penetration testing tools (nmap, hydra, curl) and vulnerability exploitation techniques, while Blue Team members develop critical skills in real-time threat detection, log analysis, and incident response. This collaborative approach breaks down traditional silos between security teams, fostering better communication and understanding of both attack vectors and defensive measures. The shared environment allows defenders to observe actual attack patterns, improving their detection capabilities, while attackers gain insight into monitoring systems, leading to more realistic security assessments.

Next Steps for Improvement Future enhancements should focus on expanding the attack surface with additional vulnerable services (DNS, FTP, SMB), implementing advanced threat detection capabilities through AWS GuardDuty integration, and adding automated incident response workflows using AWS

Lambda functions. The lab would benefit from incorporating container-based vulnerabilities using ECS or EKS, expanding to multi-region scenarios for advanced persistent threat simulations, and integrating threat intelligence feeds for more realistic attack scenarios. Additional improvements include implementing network segmentation bypass challenges, adding Windows-based targets for diverse operating system coverage, developing automated scoring systems for competitive training exercises, and creating integration with popular security tools like Splunk, Elastic Stack, or SIEM platforms for enterprise-grade monitoring experience.

9. References

AWS Documentation

- AWS GuardDuty Documentation: https://docs.aws.amazon.com/guardduty/
- AWS Security Hub Documentation: https://docs.aws.amazon.com/securityhub/
- AWS CloudTrail Documentation: https://docs.aws.amazon.com/awscloudtrail/
- AWS VPC Flow Logs: https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html
- AWS Identity and Access Management (IAM): https://docs.aws.amazon.com/iam/
- AWS Terraform Provider Docs: https://registry.terraform.io/providers/hashicorp/aws/latest

Terraform & IaC

- Terraform Documentation (HashiCorp): https://developer.hashicorp.com/terraform/docs
- Terraform AWS Provider: https://registry.terraform.io/providers/hashicorp/aws/latest

Offensive Security Tools

- Nmap Official Documentation: https://nmap.org/book/man.html
- Hydra (THC-Hydra) Official Repo: https://github.com/vanhauser-thc/thc-hydra
- curl Manual: https://curl.se/docs/manual.html

Security Frameworks & Models

- MITRE ATT&CK Framework: https://attack.mitre.org/
- Cyber Kill Chain (Lockheed Martin): https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html
- https://owasp.org/www-community/attacks/SQL_Injection
- https://www.cve.org/

Best Practices

- CIS AWS Foundations Benchmark: https://www.cisecurity.org/benchmark/amazon_web_services
- NIST Cybersecurity Framework (CSF): https://www.nist.gov/cyberframework

10. Appendices

Appendix A – Project Setup & Collaboration

Figure A1: When2meet for team scheduling.

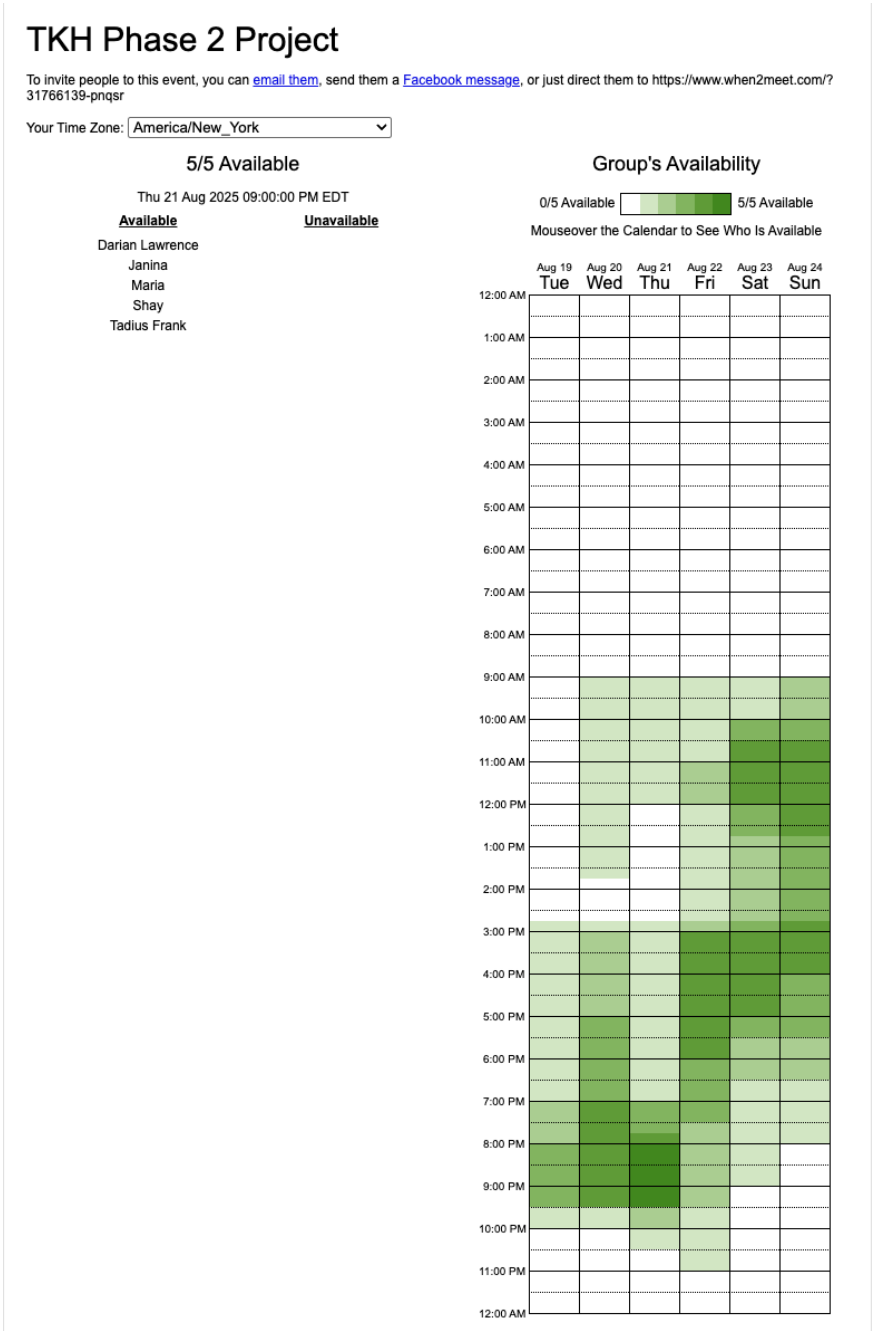


Figure A2: Jira Timeline Overview.

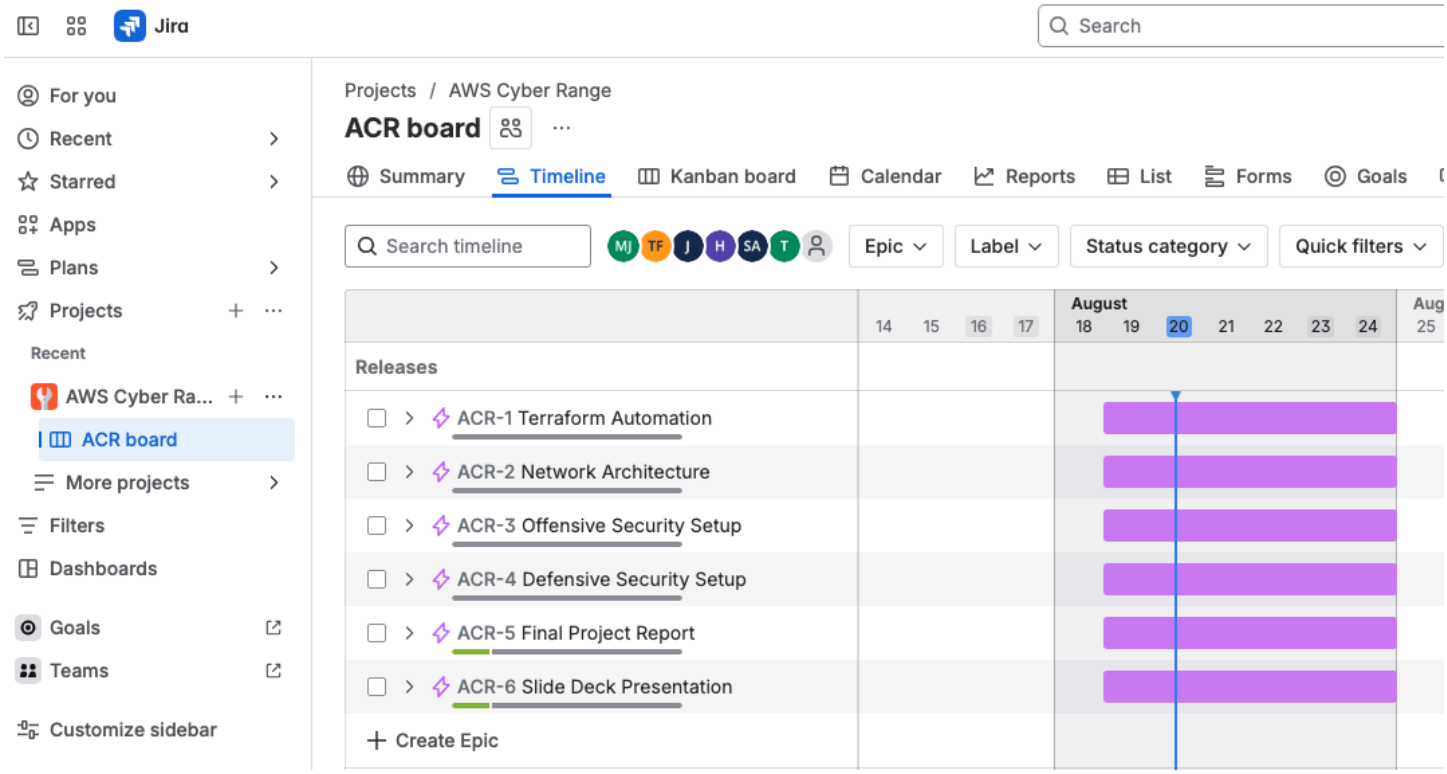


Figure A3: Task Distribution amongst team members on Jira, using Kanban board

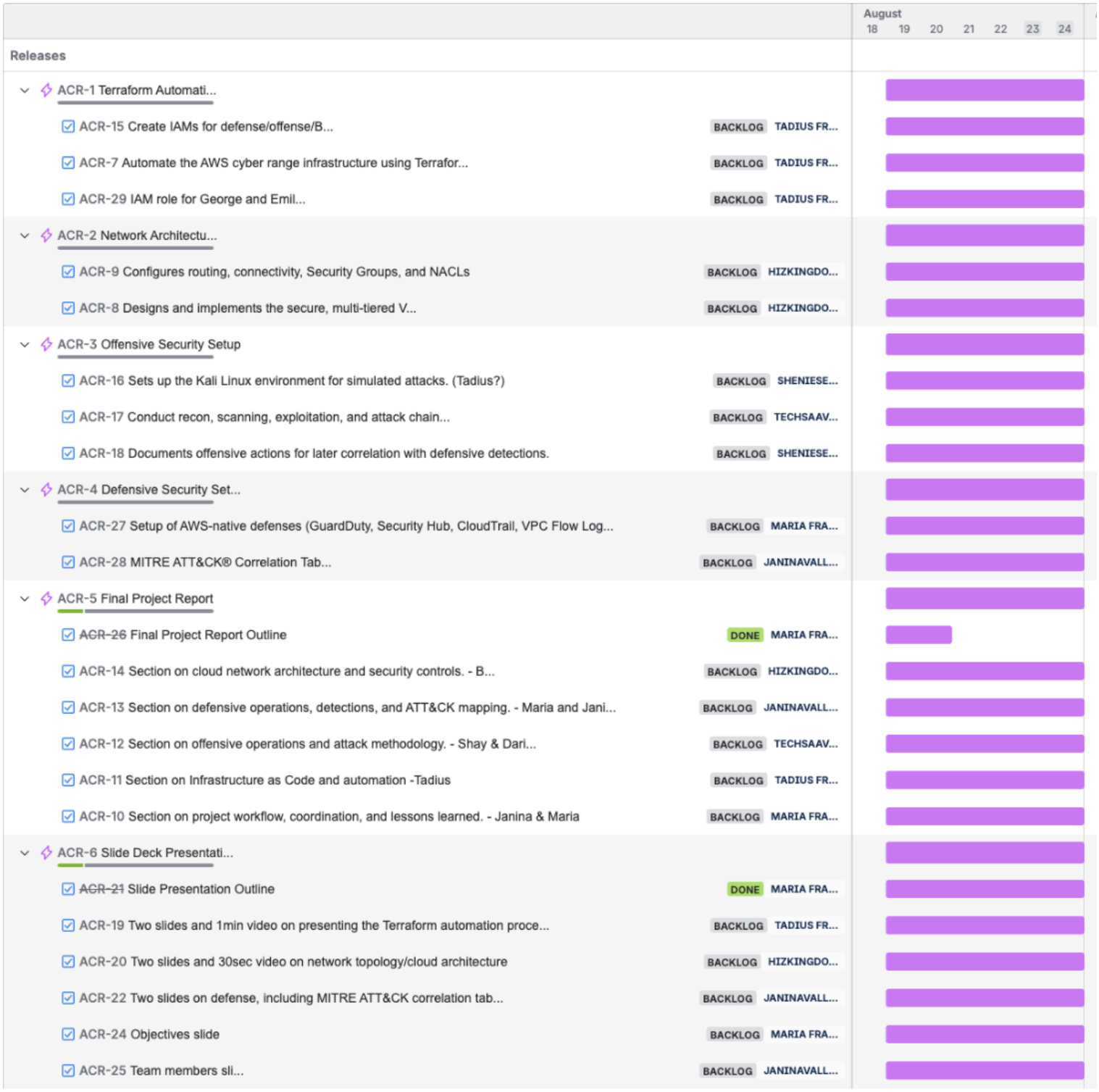


Figure A4: Team collaboration tools in action (Google Docs, Slides, Zoom)

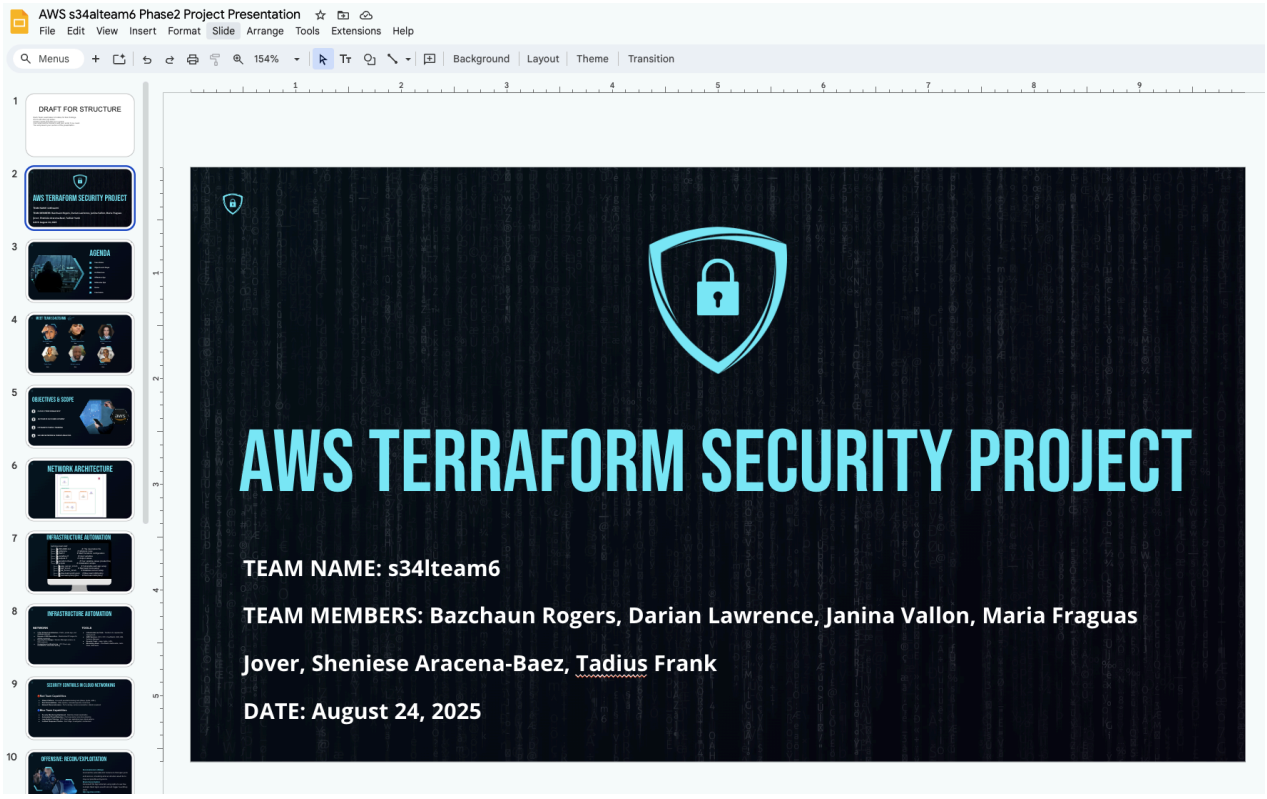


Figure A5: Jira Cumulative Workflow Diagram, August 19-28th, 2025

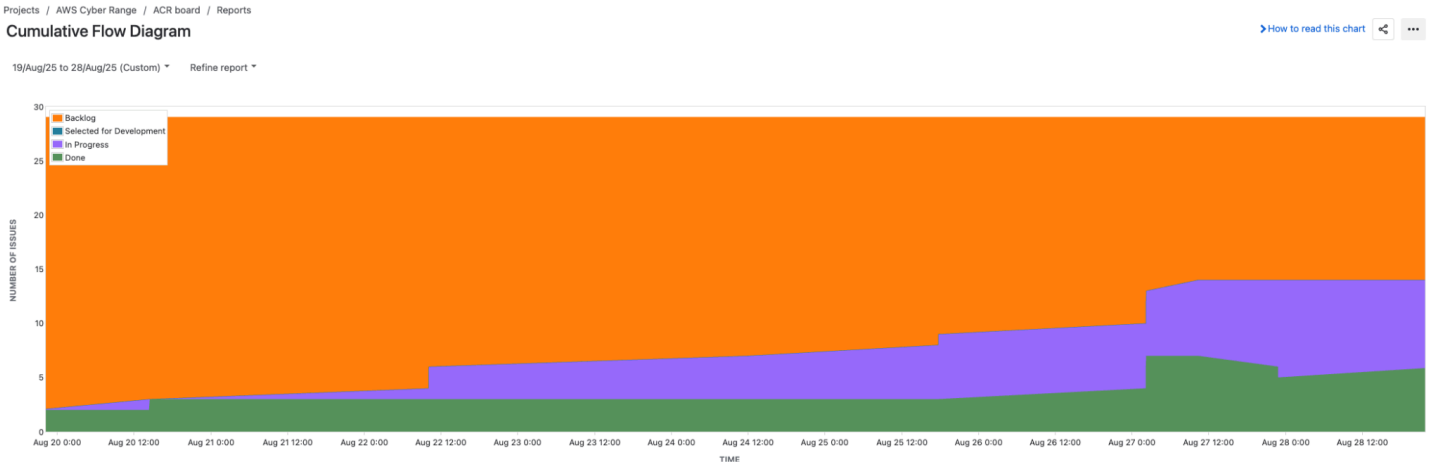


Figure A6: Jira Done work items Report:

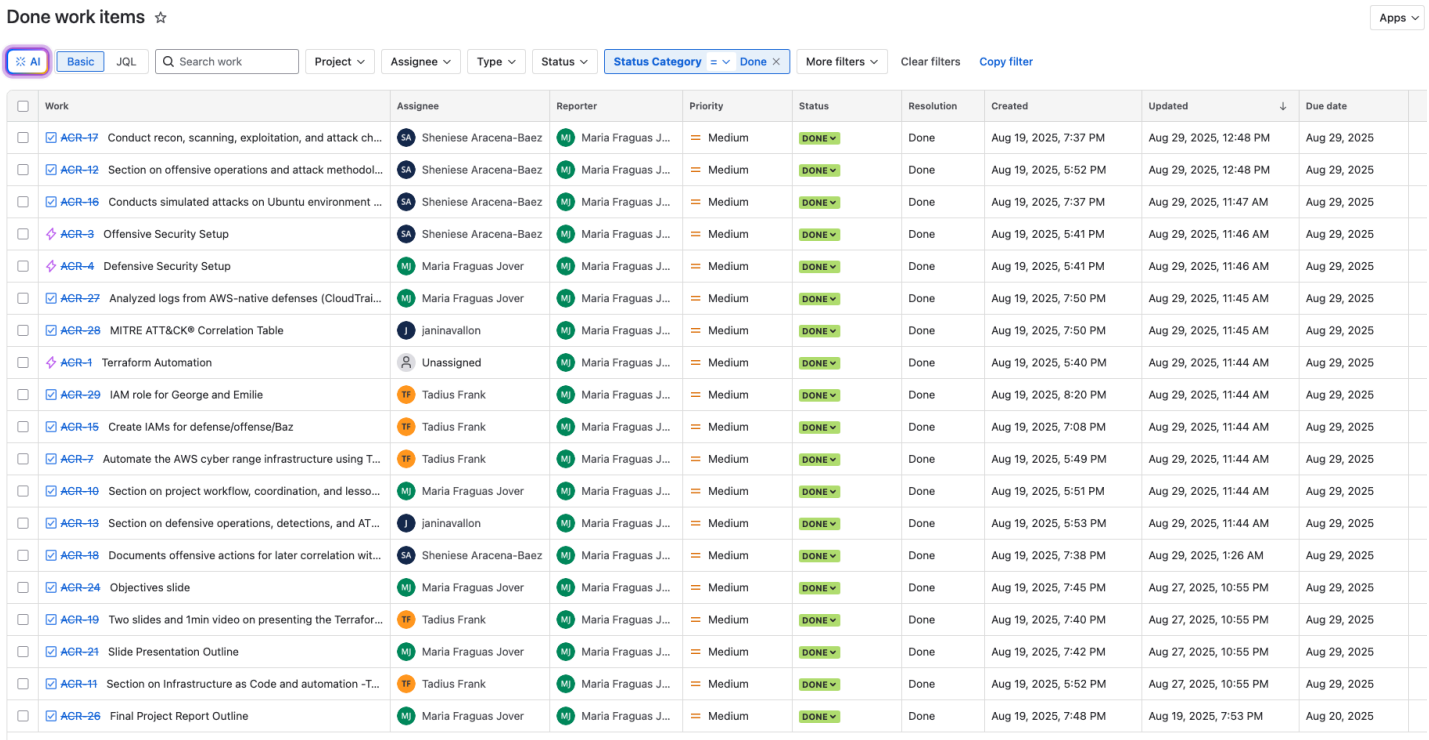
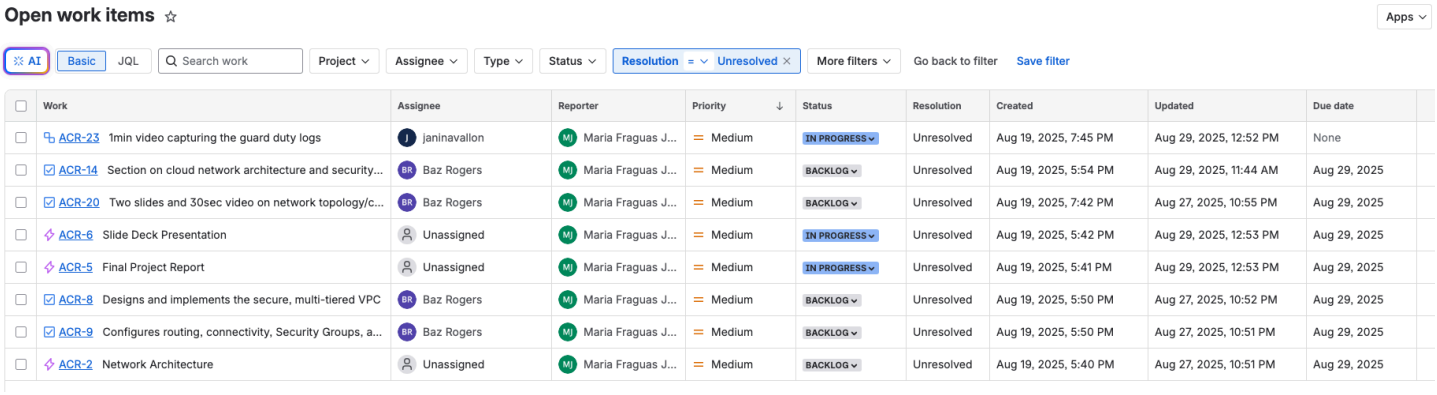


Figure A7 Open work items:



Appendix B – Infrastructure as Code (Terraform)

Figure B1: Terraform directory structure (main.tf, variables.tf, etc.)

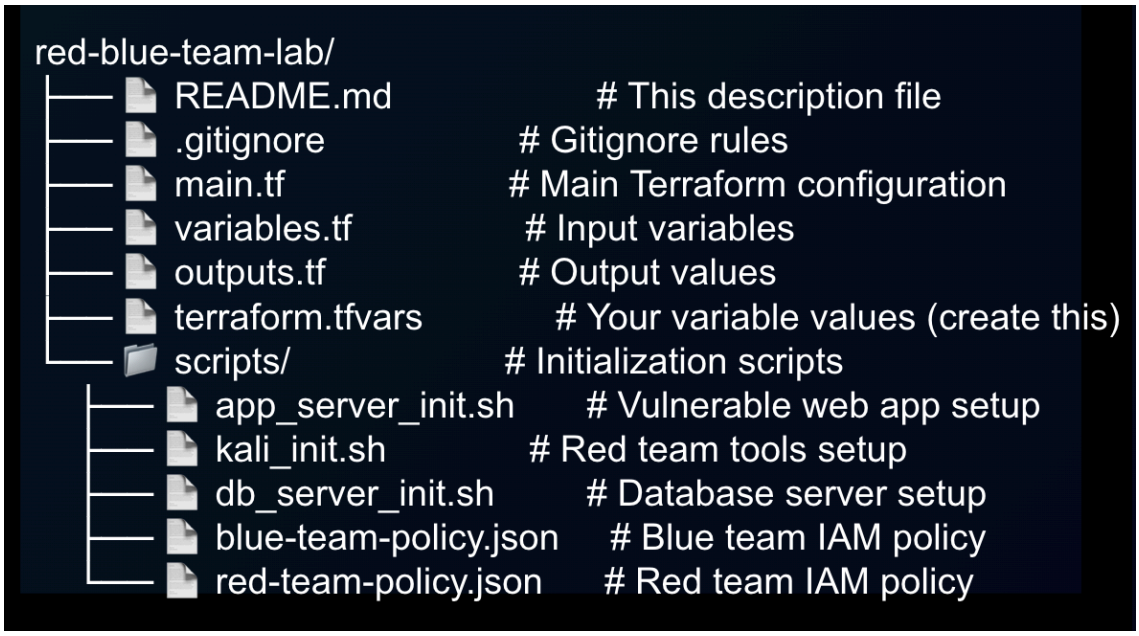


Figure B2: Terraform init/plan/apply output (state management in S3/DynamoDB)

```
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

Outputs:

connection_info = {
  "app_server_id" = "i-04868fbb78ead6067"
  "app_server_ip" = "10.237.2.189"
  "app_ssm_command" = "aws ssm start-session --target i-04868fbb78ead6067"
  "db_server_id" = "i-01732019e5ead8f0f"
  "db_server_ip" = "10.237.3.130"
  "db_ssm_command" = "aws ssm start-session --target i-01732019e5ead8f0f"
  "kali_id" = "i-0e38e4015b25e3ca0"
  "kali_private_ip" = "10.237.1.188"
  "kali_public_ip" = "13.221.26.186"
  "kali_ssm_command" = "aws ssm start-session --target i-0e38e4015b25e3ca0"
  "private_app_subnet_cidr" = "10.237.2.0/24"
  "private_db_subnet_cidr" = "10.237.3.0/24"
  "public_subnet_cidr" = "10.237.1.0/24"
  "vpc_cidr" = "10.237.0.0/16"
}
```

Figure B3: Git Hub Repo Link

https://github.com/tadiusfrank2001/AWS_SECURE_VPC

Appendix C – Network Architecture

Figure C1: Infrastructure Diagram

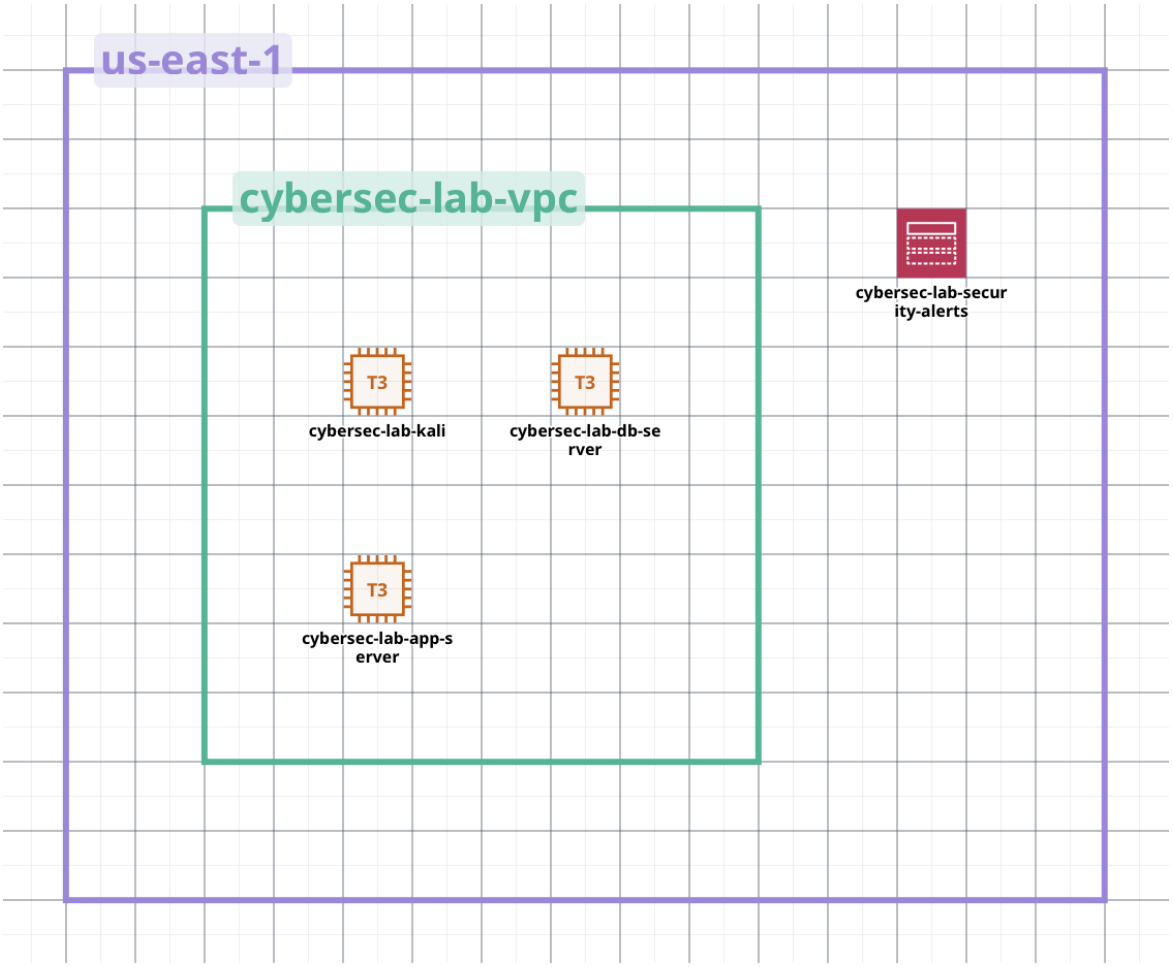


Figure C2: Security Diagram

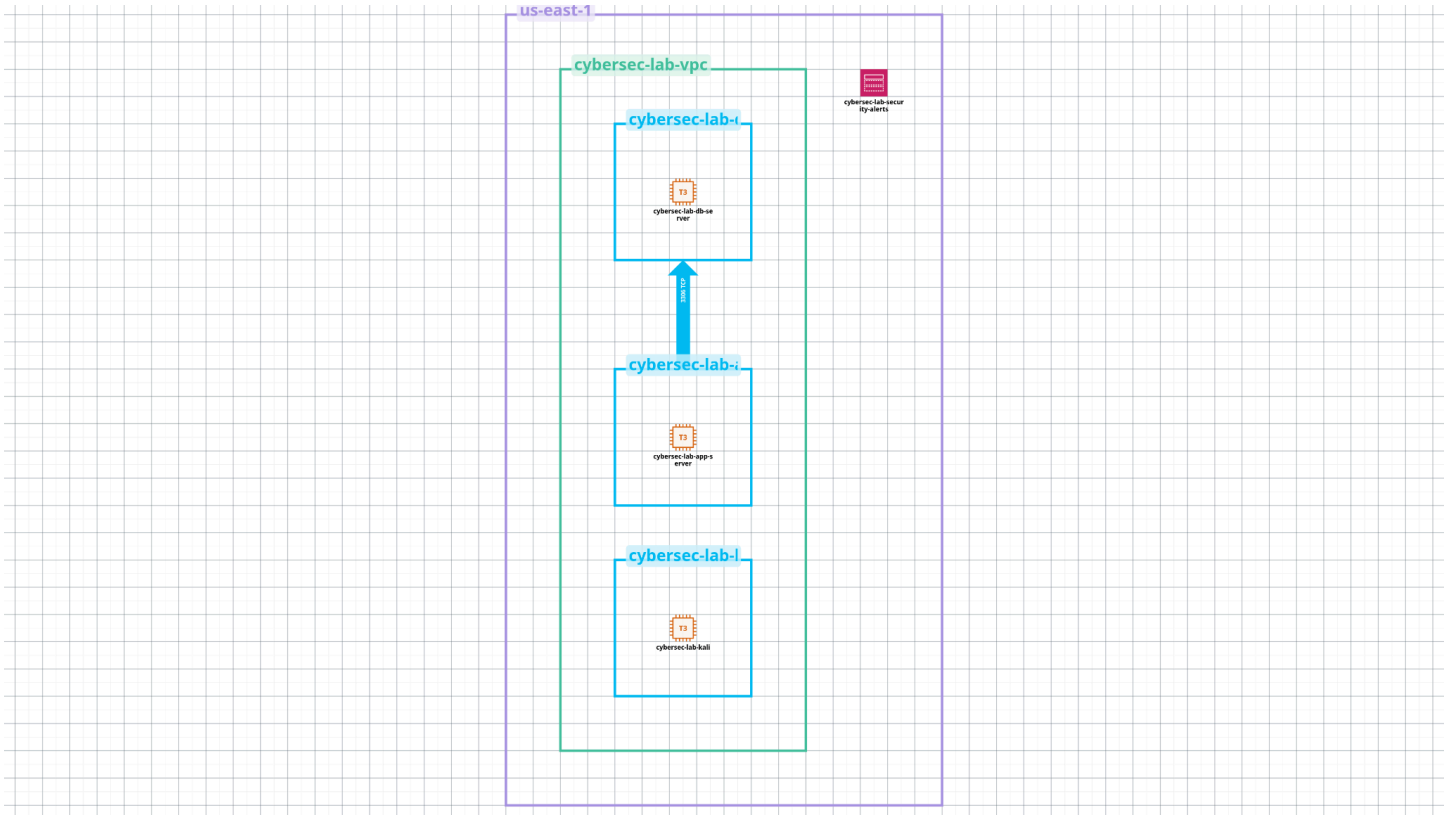


Figure C3: Network Diagram

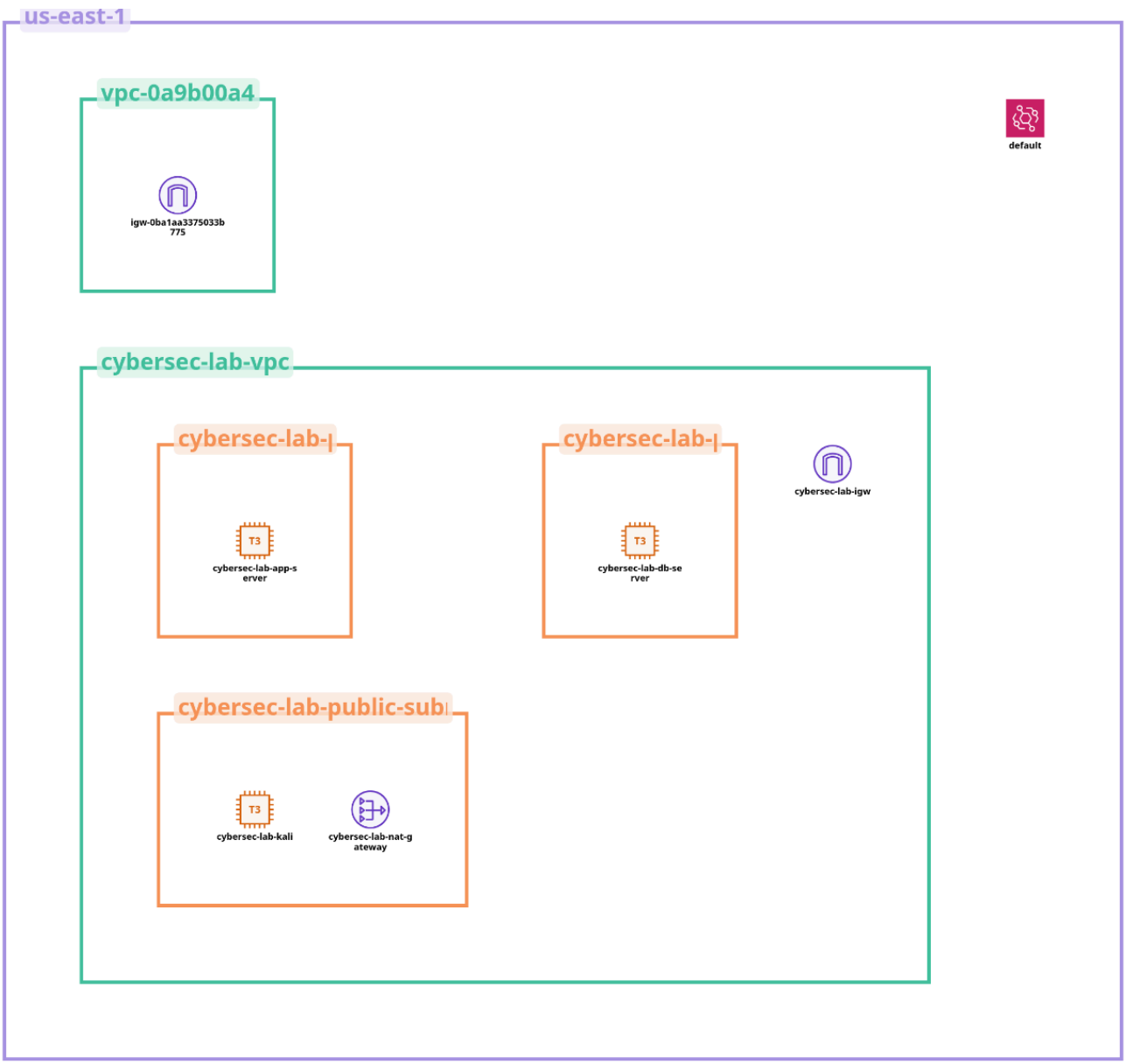


Figure C4: VPC Resource Map

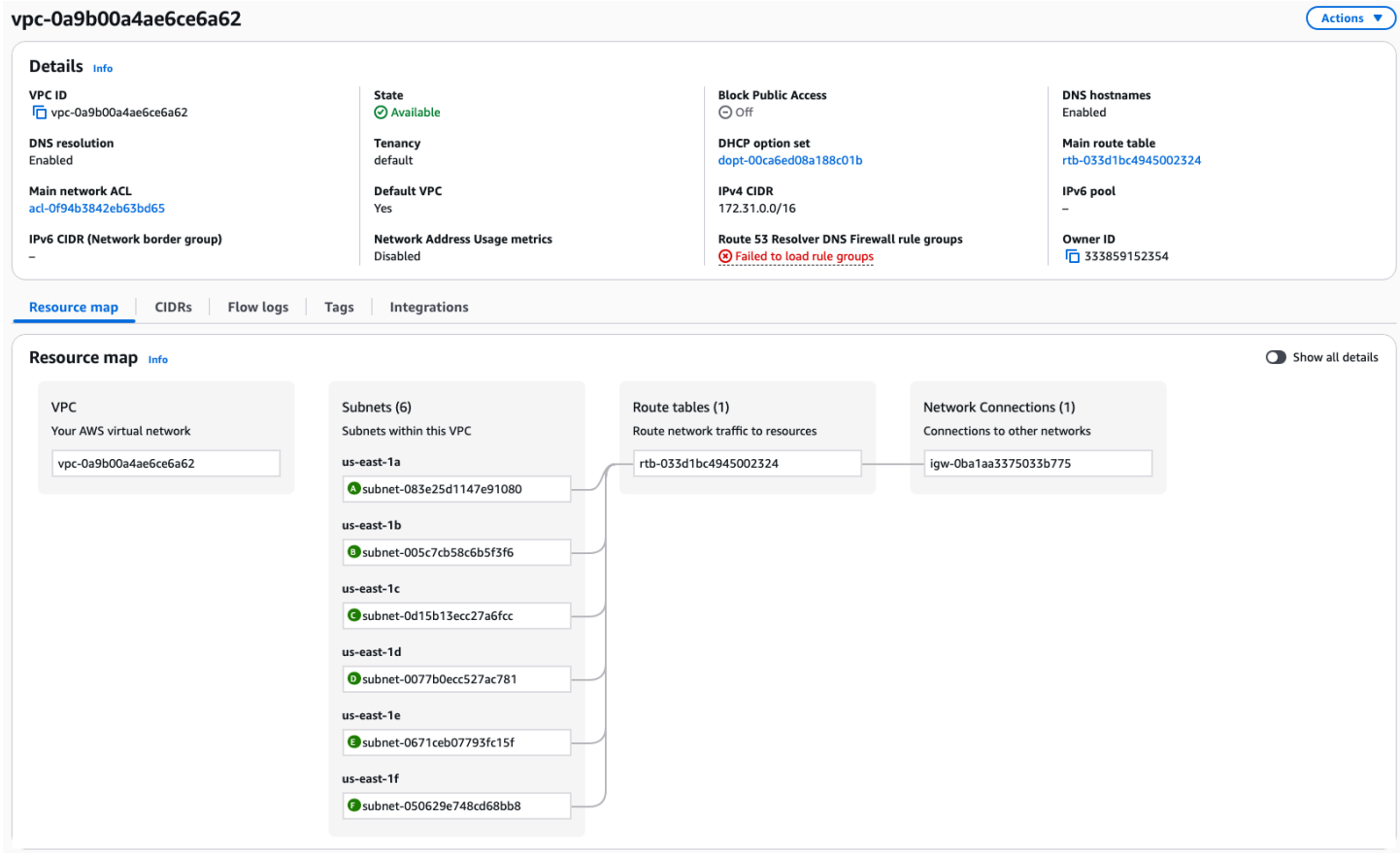
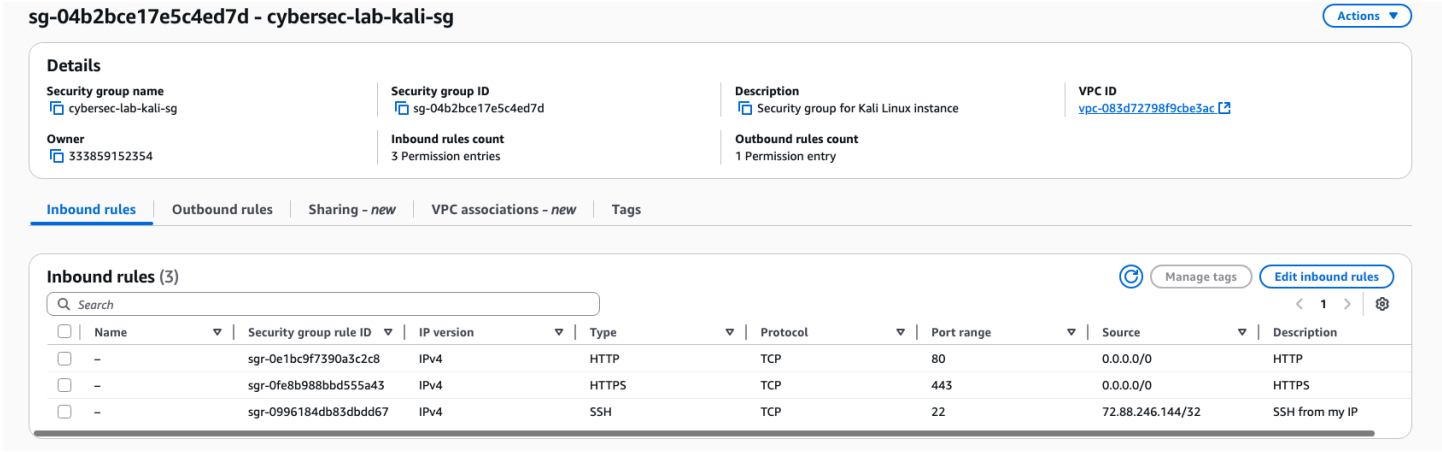


Figure C2: Security Group rules and NACL screenshots



sg-0f94228230a982fab - cybersec-lab-app-sg

Actions

Details

Security group name	Security group ID	Description	VPC ID
cybersec-lab-app-sg	sg-0f94228230a982fab	Security group for application server	vpc-083d72798f9cbe3ac

Owner	Inbound rules count	Outbound rules count
333859152354	2 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags

Inbound rules (2)

Search

Manage tags | Edit inbound rules

1

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0b797a075b080aa4e	IPv4	HTTP	TCP	80	10.124.0.0/16	HTTP from VPC
–	sgr-027e55065313927ea	IPv4	HTTPS	TCP	443	10.124.0.0/16	HTTPS from VPC

sg-081f075ade3958901 - cybersec-lab-db-sg

Actions

Details

Security group name	Security group ID	Description	VPC ID
cybersec-lab-db-sg	sg-081f075ade3958901	Security group for database server	vpc-083d72798f9cbe3ac

Owner	Inbound rules count	Outbound rules count
333859152354	2 Permission entries	3 Permission entries

Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags

Inbound rules (2)

Search

Manage tags | Edit inbound rules

1

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0f4b34adf9e9205af	IPv4	SSH	TCP	22	10.124.2.0/24	SSH for monitoring
–	sgr-089a53121c0691eaf	–	MYSQL/Aurora	TCP	3306	sg-0f94228230a982fab...	MySQL from app

NACLS:

acl-0c7584287eba072ae

Actions

Details Info

Network ACL ID	Associated with	Default	VPC ID
acl-0c7584287eba072ae	3 Subnets	Yes	vpc-083d72798f9cbe3ac / cybersec-lab-vpc

Owner
333859152354

Inbound rules | Outbound rules | Subnet associations | Tags

Inbound rules (2)

Filter inbound rules

Edit inbound rules

1

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

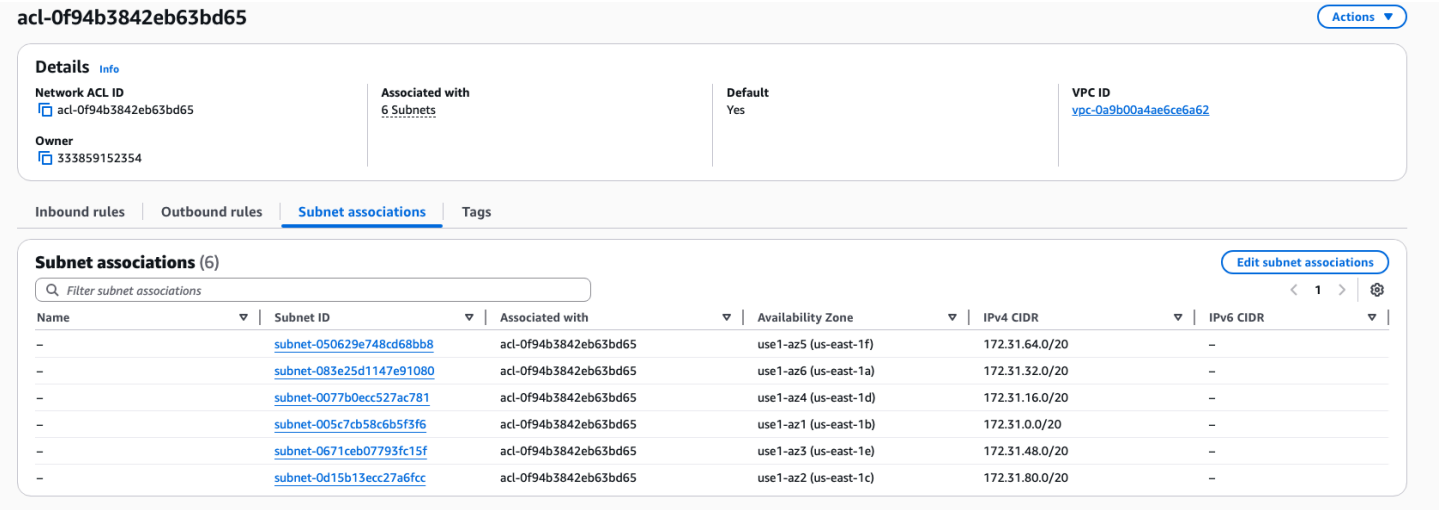
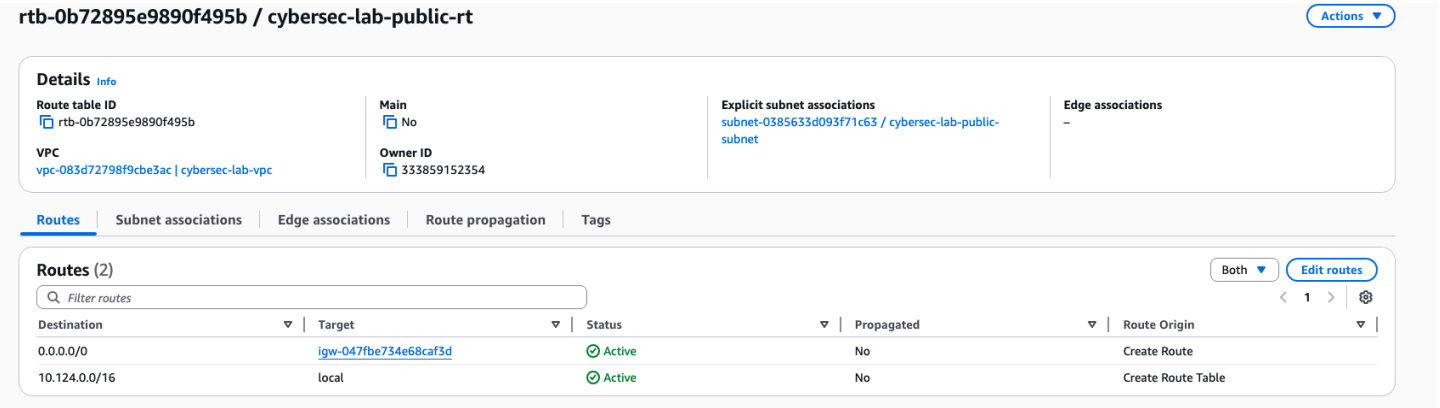
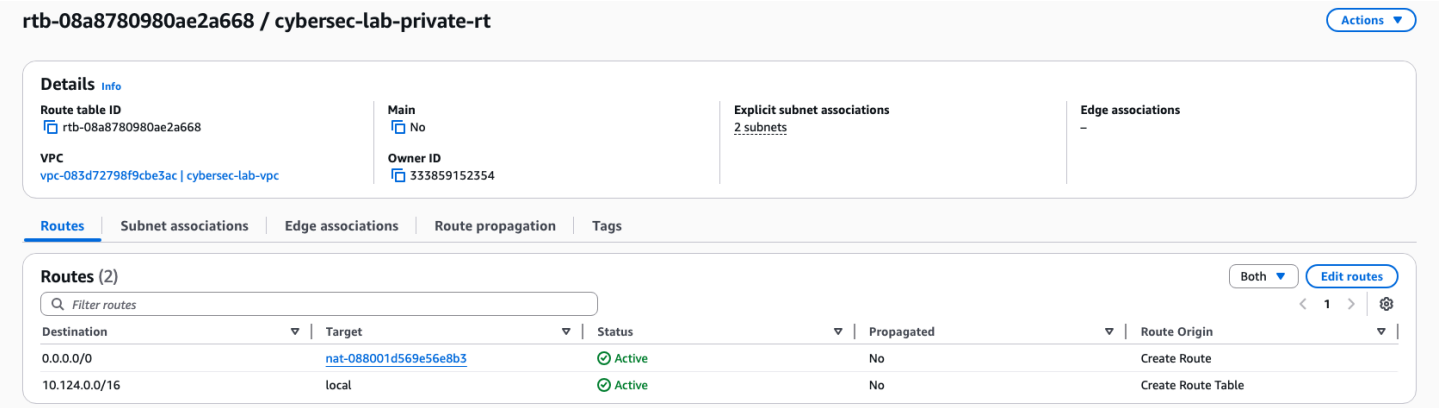


Figure C3: Routing table and subnet associations

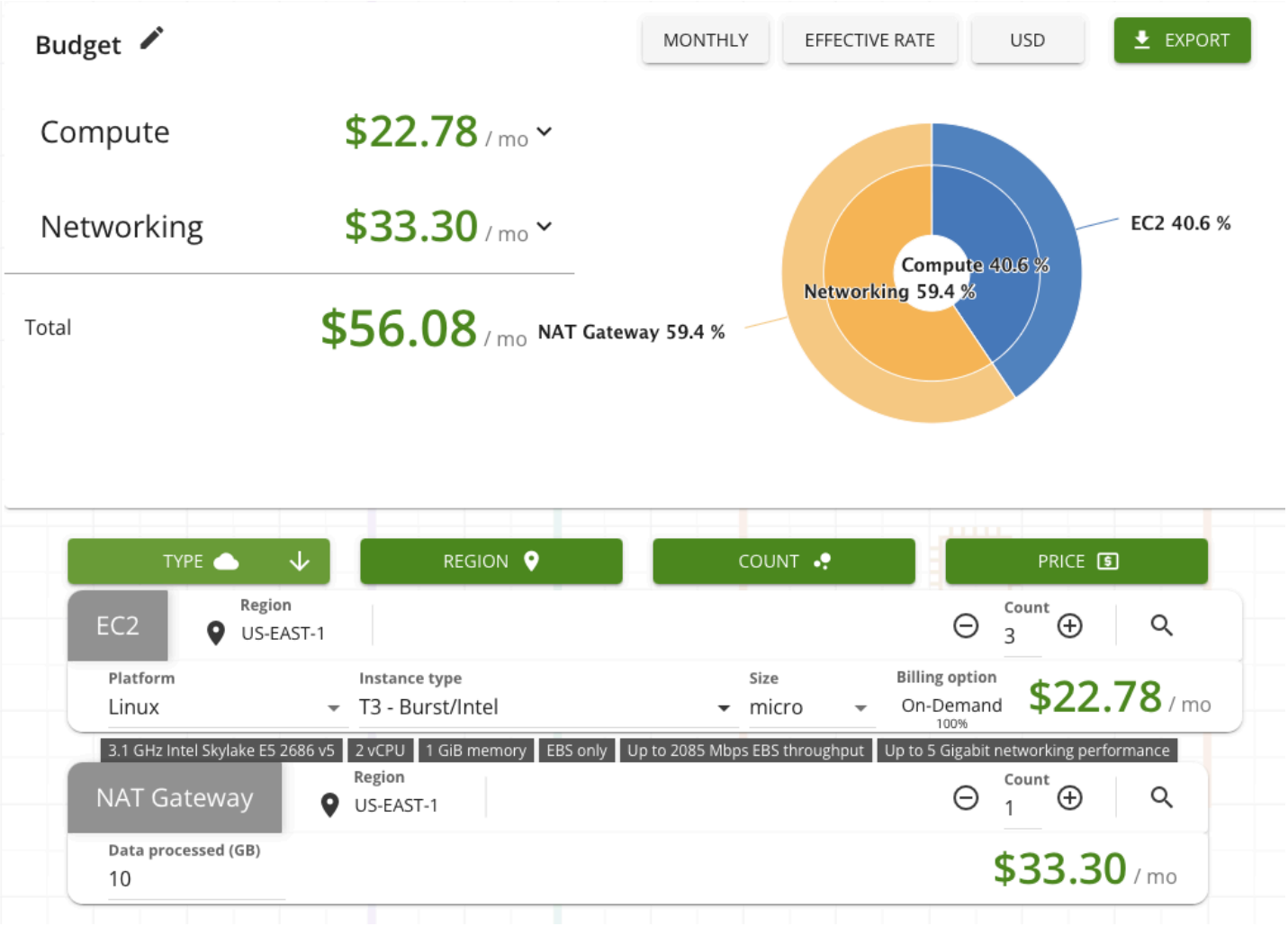
Public Subnet:



Private Subnet:



BUDGET CONSTRAINTS:



Appendix D – Offensive Operations

Figure D1: Kali Linux environment setup

```
AWS_SECURE_VPC % aws ssm start-session --target i-03fc5337dc50f2ade --profile terraform

Starting session with SessionId: root-633falijjqjz99xftz4h7cc4kq
sh-4.2$ nmap --version

Nmap version 6.40 ( http://nmap.org )
Platform: x86_64-koji-linux-gnu
Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
sh-4.2$ hydra --version
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding
, these *** ignore laws and ethics anyway).

hydra: invalid option -- '-'
```

Figure D2: Reconnaissance and scanning (Nmap, etc.)

```
Starting Nmap 6.40 ( http://nmap.org ) at 2025-08-29 03:42 UTC
Nmap scan report for ip-[redacted].ec2.internal ([redacted])
Host is up (0.00029s latency).
Not shown: 998 filtered ports
PORT    STATE  SERVICE
80/tcp  open   http
443/tcp closed https

Nmap done: 1 IP address (1 host up) scanned in 4.53 seconds
sh-4.2$ nmap -sS [redacted]
You requested a scan type which requires root privileges.
QUITTING!
sh-4.2$ nmap -A [redacted]

Starting Nmap 6.40 ( http://nmap.org ) at 2025-08-29 03:43 UTC
Nmap scan report for ip-[redacted].ec2.internal ([redacted])
Host is up (0.00033s latency).
Not shown: 998 filtered ports
PORT    STATE  SERVICE VERSION
80/tcp  open   http    Apache httpd 2.4.64 (() PHP/5.4.16)
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
443/tcp closed https

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.19 seconds
sh-4.2$ nmap -sV -p- [redacted]
```

```
sh-4.2$ nmap -p 3306 --script mysql-enum [redacted]

Starting Nmap 6.40 ( http://nmap.org ) at 2025-08-29 03:48 UTC
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds
sh-4.2$ nmap -Pn [redacted]

Starting Nmap 6.40 ( http://nmap.org ) at 2025-08-29 03:48 UTC
Nmap scan report for ip-[redacted] ec2.internal (10.124.3.83)
Host is up.
All 1000 scanned ports on ip-[redacted] ec2.internal ([redacted]) are filtered

Nmap done: 1 IP address (1 host up) scanned in 201.27 seconds
sh-4.2$
```

Figure D3: Exploitation attempts/screenshots

```
sh-4.2$ hydra -l ec2-user -p welcome123 -t 10 ssh://[redacted]
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-08-29 04:16:47
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://[redacted]/
[ERROR] could not connect to ssh://[redacted]:22 - Timeout connecting to [redacted]
sh-4.2$ hydra -l ec2-user -p welcome123 -t 4 ssh://[redacted]
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-08-29 04:17:43
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://[redacted]:22/
[ERROR] could not connect to ssh://[redacted]:22 - Timeout connecting to [redacted]
```

```
sh-4.2$ hydra -l ec2-user -p welcome123 -t 10 ssh://[redacted]
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-08-29 04:16:47
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://[redacted]/
[ERROR] could not connect to ssh://[redacted] - Timeout connecting to [redacted]
sh-4.2$ hydra -l ec2-user -p welcome123 -t 4 ssh://[redacted]
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-08-29 04:17:43
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://[redacted]:22/
[ERROR] could not connect to ssh://[redacted] - Timeout connecting to [redacted]
```

```
sh-4.2$ curl -I http://[redacted]/login.php
HTTP/1.1 200 OK
Date: Fri, 29 Aug 2025 04:35:21 GMT
Server: Apache/2.4.64 () PHP/5.4.16
Upgrade: h2,h2c
Connection: Upgrade
X-Powered-By: PHP/5.4.16
Content-Type: text/html; charset=UTF-8

sh-4.2$ curl -X POST \
> -d "username=admin' OR '1'='1' -- &password=anything" \
> http://[redacted]/login.php
<!DOCTYPE html>
<html>
<head><title>Employee Login</title></head>
<body>
    <h2>Employee Login Portal</h2>
    <form method="post">
        <p>Username: <input type="text" name="username" required></p>
        <p>Password: <input type="password" name="password" required></p>
        <p><input type="submit" value="Login"></p>
    </form>
    <p style='color:red'>Error: Invalid credentials. Query executed: SELECT * FROM users WHERE username = 'admin' OR '1'='1' -- ' AND password = 'anything'</p>    <p><small>Hint: Try SQL injection techniques lik
e ' OR '1'='1' --</small></p>
    <!-- Developer note: Default credentials admin/admin123 -->
    <p><a href="index.php">Back to Home</a></p>
</body>
</html>
```

Appendix E – Defensive Operations

Figure E1: VPC Flow Logs configuration and analysis screenshots

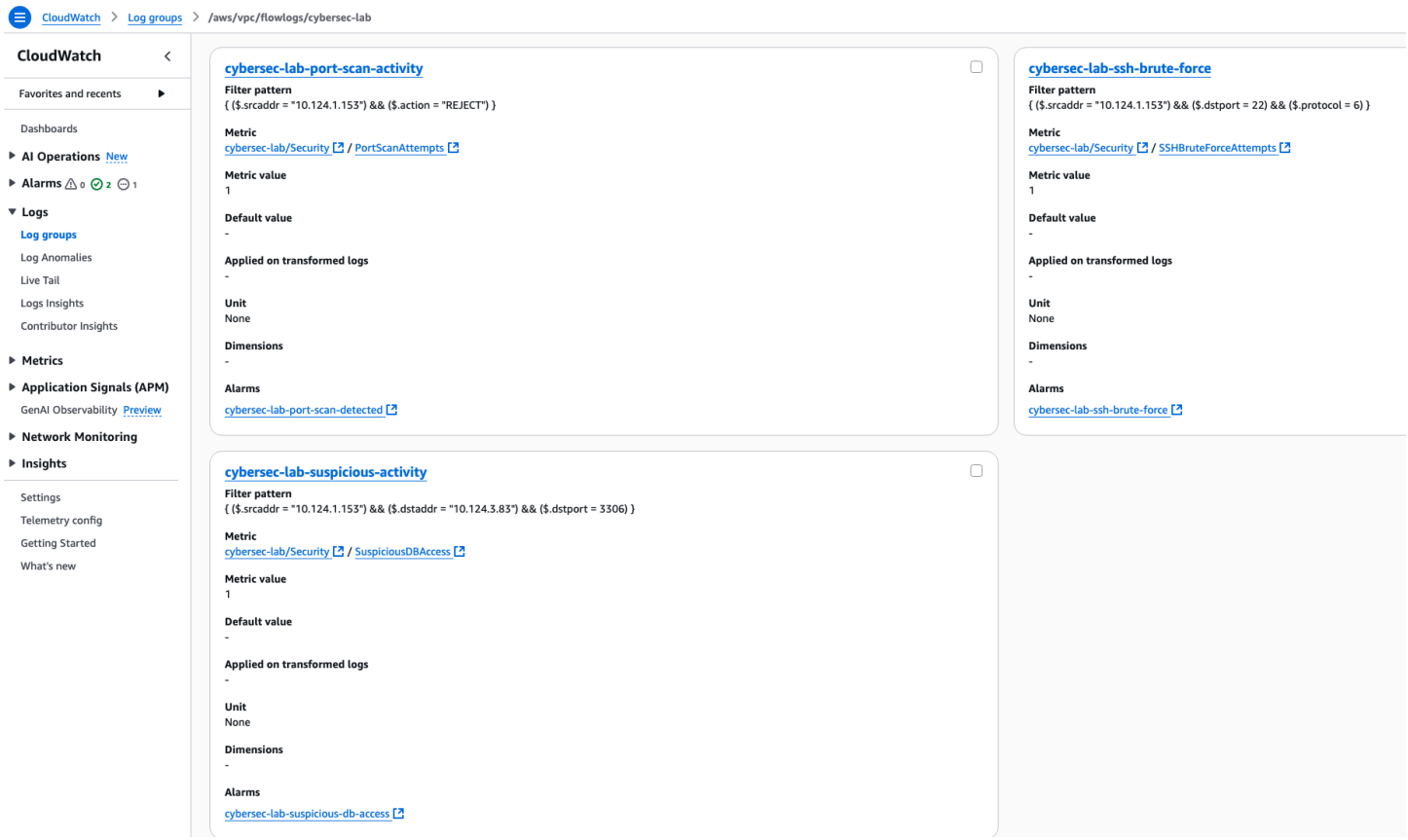
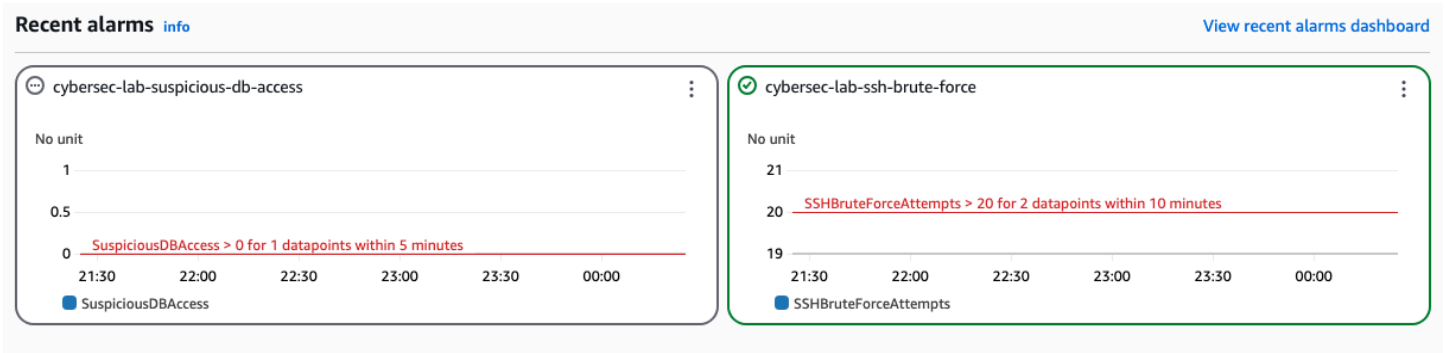


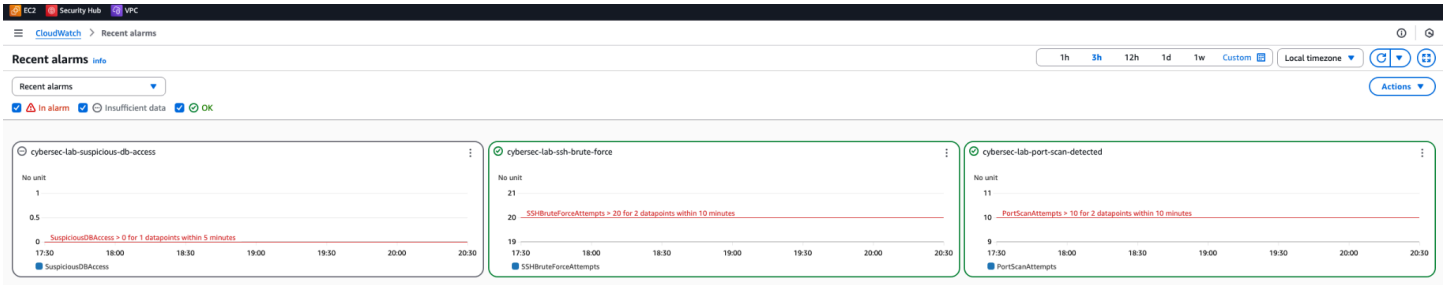
Figure E2: Correlation of defensive alerts to offensive actions

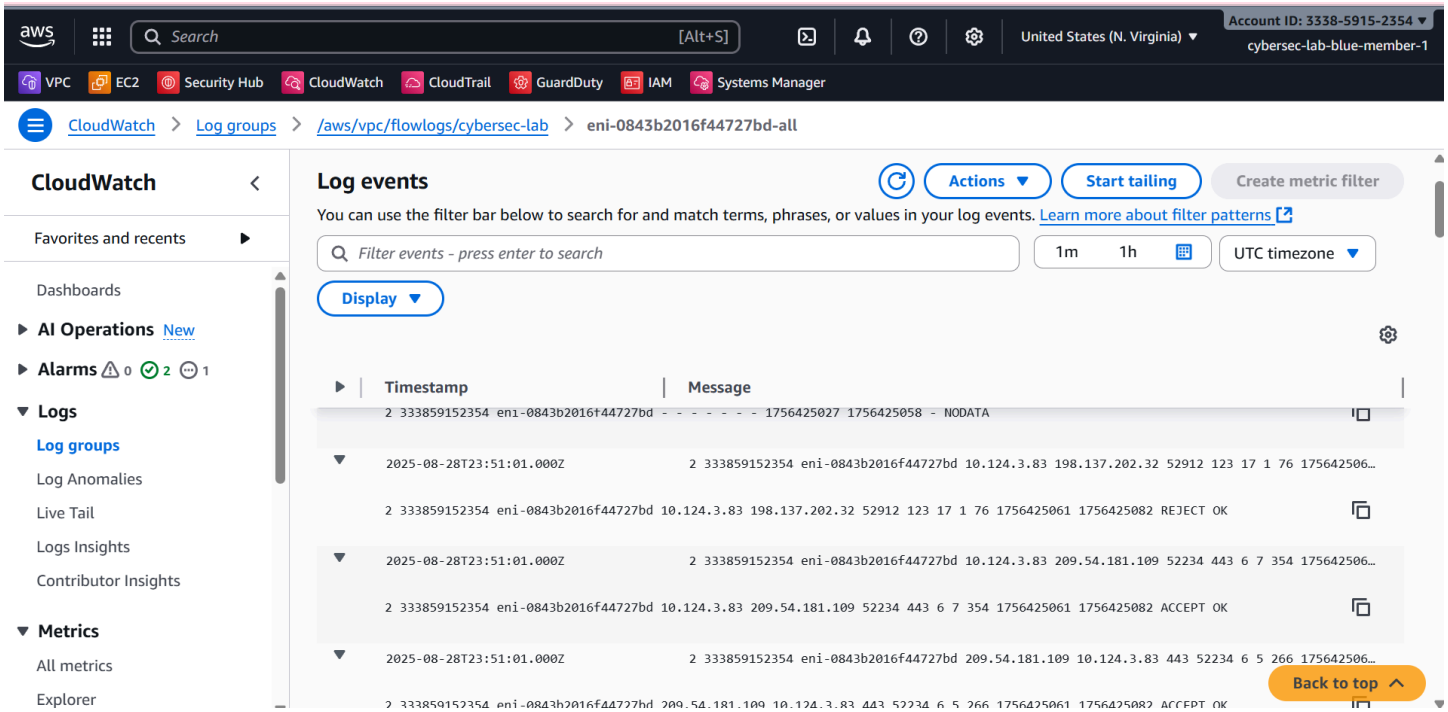
MITRE ATT&CK® Correlation Table

Attack Technique	Tool Used	ATT&CK Tactic	ATT&CK Technique ID	Observed Purpose	Detection Source	Outcome in Lab
Port Scanning (TCP SYN, Version, Aggressive)	Nmap (`-sS` , `-A` , `-sV` , `-p-` , `--script mysql-enum`)	Reconnaissance	T1046 – Network Service Scanning	Enumerate open ports, services, and versions (e.g., MySQL on 3306)	VPC Flow Logs, GuardDuty Reconnaissance findings, CloudTrail (API enumeration)	Identified active services and endpoints for further exploitation
Brute-Force (SSH)	Hydra (`hydra -l ec2-user -p password123 ssh://<target-ip>`)	Credential Access	T1110 – Brute Force	Test multiple SSH login attempts with common credentials	GuardDuty UnauthorizedAccess:EC2/SSHBruteForce, VPC Flow Logs (multiple rejected attempts)	Triggered failed SSH login attempts; generated GuardDuty brute-force findings
SQL Injection	curl (`curl -X POST -d "username=admin' OR '1'='1' --&password=anything" http://<target-ip>/login.php`)	Initial Access / Execution	T1190 – Exploit Public-Facing Application	Exploit vulnerable login form with SQL payload to bypass authentication	Web server logs, VPC Flow Logs (HTTP POST anomalies), Security Hub (aggregated finding)	Generated identifiable SQL injection patterns; confirmed input not sanitized

Figure E3: Cloudwatch Recent Alarms:






aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 3338-5915-2354
cybersec-lab-blue-member-1
VPC
EC2
Security Hub
CloudWatch
CloudTrail
GuardDuty
IAM
Systems Manager
CloudWatch
Log groups
/aws/vpc/flowlogs/cybersec-lab
eni-0843b2016f44727bd-all
CloudWatch
Favorites and recents
Dashboards
AI Operations New
Alarms 0 2 1
Logs
Log groups
Log Anomalies
Live Tail
Logs Insights
Contributor Insights
Metrics
All metrics
Explorer
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
1m
1h
UTC timezone
Display
Timestamp
Message
2 333859152354 eni-0843b2016f44727bd - - - - - - - 1756425027 1756425058 - NODATA
2025-08-28T23:51:01.000Z 2 333859152354 eni-0843b2016f44727bd 10.124.3.83 198.137.202.32 52912 123 17 1 76 175642506…
2 333859152354 eni-0843b2016f44727bd 10.124.3.83 198.137.202.32 52912 123 17 1 76 1756425061 1756425082 REJECT OK
2025-08-28T23:51:01.000Z 2 333859152354 eni-0843b2016f44727bd 10.124.3.83 209.54.181.109 52234 443 6 7 354 175642506…
2 333859152354 eni-0843b2016f44727bd 10.124.3.83 209.54.181.109 52234 443 6 7 354 1756425061 1756425082 ACCEPT OK
2025-08-28T23:51:01.000Z 2 333859152354 eni-0843b2016f44727bd 209.54.181.109 10.124.3.83 443 52234 6 5 266 175642506…
Back to top

Appendix F – AWS Support Correspondence

Figure F1: Attempts to activate Guard Duty and Security Hub