

Report: Lambda-Based SDN Controller for Dynamic Traffic Policy Enforcement in AWS

1. Introduction

As modern cloud environments grow in scale and complexity, traditional static security controls often fall short in responding to dynamic threats. To address this challenge, this project explores the design and implementation of a Software-Defined Networking (SDN)-inspired controller using AWS Lambda. The goal is to create a lightweight, automated system capable of analyzing VPC Flow Logs and enforcing real-time network security policies by programmatically modifying security groups and network ACLs.

By leveraging serverless architecture and custom traffic policies written in Python, the solution aims to provide dynamic response capabilities—such as blocking unauthorized SSH attempts—without manual intervention. The system is tested against simulated attack scenarios to evaluate its responsiveness, reliability, and compliance readiness. This report outlines the design decisions, implementation steps, and key findings from the simulation, with a focus on scalability, automation, and security best practices in a cloud-native context.

2. Objective

This project aims to build a serverless SDN controller in AWS that dynamically enforces network security policies by analyzing VPC Flow Logs in real-time with AWS Lambda. Using a Python-based DSL, it allows customizable traffic rules to automatically detect and respond to suspicious activities—such as blocking unauthorized access—by programmatically updating security groups and network ACLs. The system also generates detailed logs for auditing and compliance, and it is tested through simulated attacks to ensure reliable policy enforcement and rapid incident response. By automating network traffic control, the solution reduces the need for manual intervention and enhances security agility in cloud environments. Ultimately, this approach demonstrates how event-driven cloud services can improve threat detection and mitigation while supporting scalable and flexible network management.

Key goals include:

- **Real-Time Traffic Analysis Using VPC Flow Logs**

The system should continuously ingest and analyze VPC Flow Logs to monitor network activity across the AWS environment. This real-time analysis enables the identification of suspicious traffic patterns, such as repeated failed connection attempts, traffic from unknown IPs, or unexpected access to sensitive ports. The goal is to ensure timely detection of potential threats without relying on manual log review.

- **Automated Policy Enforcement with AWS Lambda**

A core objective is to automate the enforcement of network policies using AWS Lambda. Based on predefined rules, the Lambda function should trigger actions—such as blocking a source IP or adjusting port access—with human intervention. This allows for faster incident response and supports scalability by removing the need for constant manual oversight.

- **Customizable Policy Logic via Python DSL**

To provide flexibility and control, the project includes a Python-based domain-specific language (DSL) that allows administrators to define custom traffic rules. These rules dictate how the system reacts to specific network conditions, making it easier to tailor security behavior to different use cases or compliance requirements.

- **Dynamic Modification of Security Groups and NACLs**

The solution must be capable of modifying AWS security groups and network ACLs programmatically based on analysis results. This ensures that unauthorized or malicious traffic can be swiftly blocked, and legitimate traffic can be allowed without delay. Dynamic reconfiguration helps maintain strong security postures even in rapidly changing environments.

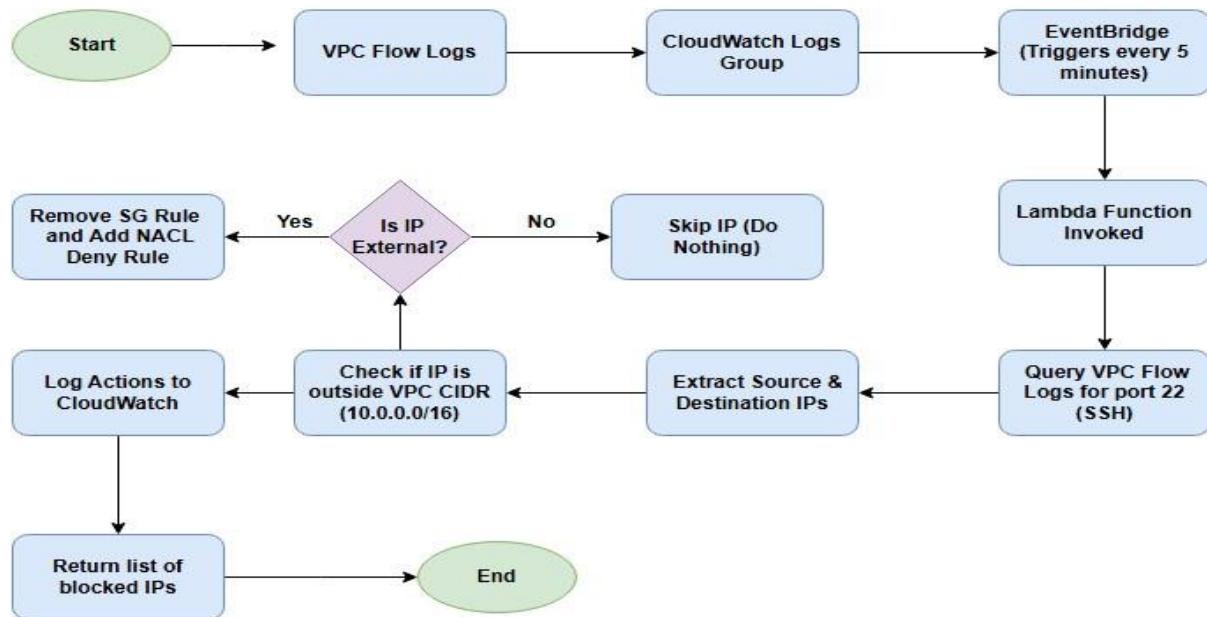
- **Comprehensive Logging and Audit Reporting**

To support compliance and post-incident analysis, the system should generate detailed logs of all actions taken, including traffic evaluations, rule matches, and security group changes. These logs will be used to produce audit-ready reports that provide insight into system behavior, policy enforcement outcomes, and overall security effectiveness.

3. Tools And Techniques

- **AWS Lambda** – Serverless compute service used to process VPC Flow Logs and execute traffic policy logic in real-time.
- **VPC Flow Logs** – Network traffic logs from AWS VPCs used for monitoring and analysis.
- **Python DSL** – Custom domain-specific language implemented in Python for defining dynamic traffic control policies.
- **AWS Security Groups & NACLs** – Network access control mechanisms dynamically modified via code to enforce security policies.
- **AWS SDK (boto3)** – Python library used to programmatically interact with AWS services for policy enforcement.
- **SSH Login Attempts** – Used to simulate unauthorized access attempts for validating policy enforcement.
- **AWS CloudWatch** – Service used for logging, auditing, and monitoring system activities.

4. Workflow Diagram



5. Methodology

This project focuses on building a dynamic Software-Defined Network (SDN) controller using AWS Lambda to monitor and automatically adjust network security policies in real-time. The objective is to create a secure cloud environment that detects unauthorized SSH login attempts via VPC Flow Logs, processes these logs with Lambda functions, and dynamically updates security groups and Network ACLs (NACLs) to block malicious IP addresses. The methodology is divided into distinct phases encompassing network setup, monitoring, automation, attack simulation, and validation.

5.1 Network Setup and Resource Configuration

This phase establishes the cloud network infrastructure necessary for hosting resources and capturing traffic data. It ensures a well-architected environment where monitoring and policy enforcement can operate reliably.

Purpose: Establish a foundational and fully functional AWS networking environment to simulate real-world scenarios and enable monitoring.

- Created a Virtual Private Cloud (VPC), subnet, Internet Gateway (IGW), route tables, and attached them appropriately to establish a functional networking infrastructure.

(A VPC named "MyVPC" was created.)

The screenshot shows the AWS VPC Details page for a VPC named "MyVPC". The VPC ID is "vpc-04dd6a1baf8e0fea7". The VPC is in an "Available" state. It has a Main network ACL named "acl-0f1471843d20dfa98". The VPC has no IPv6 CIDR assigned. The Default VPC setting is "No". The Network Address Usage metrics are disabled. The Block Public Access setting is off. The DHCP option set is "dopt-077976cb8ef894f8e". The IPv4 CIDR is "10.0.0.0/16". The Route 53 Resolver DNS Firewall rule groups are empty. The DNS hostnames are disabled. The Main route table is "rtb-022ac3c0dc8327f3f". The IPv6 pool is empty. The Owner ID is "976965259312". The page includes standard AWS navigation and search bars at the top.

(Created a subnet and added the VPC to it.)

The screenshot shows the AWS VPC Subnets page. The subnet 'subnet-0124bcedd67ea9487' is listed with the following details:

Details	
Subnet ID	subnet-0124bcedd67ea9487
IPv4 CIDR	10.0.1.0/24
Availability Zone	eu-north-1a
Route table	rtb-0c9e7328bb83e8bbf MyRouteTable
Auto-assign IPv6 address	No
IPv4 CIDR reservations	-
Resource name DNS A record	Disabled
Subnet ARN	arn:aws:ec2:eu-north-1:976965259312:subnet/subnet-0124bcedd67ea9487
Available IPv4 addresses	250
Availability Zone ID	eun1-az1
Network ACL	acl-0f1471843d20dfa98
Auto-assign customer-owned IPv4 address	No
IPv6 CIDR reservations	-
Resource name DNS AAAA record	Disabled
State	Available
IPv6 CIDR	-
Network border group	eu-north-1
Default subnet	No
Customer-owned IPv4 pool	-
IPv6-only	No
DNS64	Disabled
Block Public Access	Off
IPv6 CIDR association ID	-
VPC	vpc-04dd6a1baf8e0fea7 MyVPC
Auto-assign public IPv4 address	Yes
Outpost ID	-
Hostname type	IP name
Owner	976965259312

(Created an Internet Gateway and attached it to the VPC.)

The screenshot shows the AWS VPC Internet Gateways page. The internet gateway 'igw-0e8615dee6ae9aa20' is listed with the following details:

Details	
Internet gateway ID	igw-0e8615dee6ae9aa20
State	Attached
VPC ID	vpc-04dd6a1baf8e0fea7 MyVPC
Owner	976965259312

Tags

Key	Value
Name	MyIGW

(Created a route table, associated the subnet to it, and selected the Internet Gateway by editing the routes.)

The screenshot shows the AWS VPC Route Tables page. The route table 'rtb-0c9e7328bb83e8bbf' is listed with the following details:

Details	
Route table ID	rtb-0c9e7328bb83e8bbf
VPC	vpc-04dd6a1baf8e0fea7 MyVPC
Main	No
Owner ID	976965259312
Explicit subnet associations	subnet-0124bcedd67ea9487 / MySubnet
Edge associations	-

Routes

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0e8615dee6ae9aa20	Active	No
10.0.0.0/16	local	Active	No

- Launched an EC2 instance named **AppServer** within the VPC **MyVPC** to serve as the target for simulated SSH attacks.

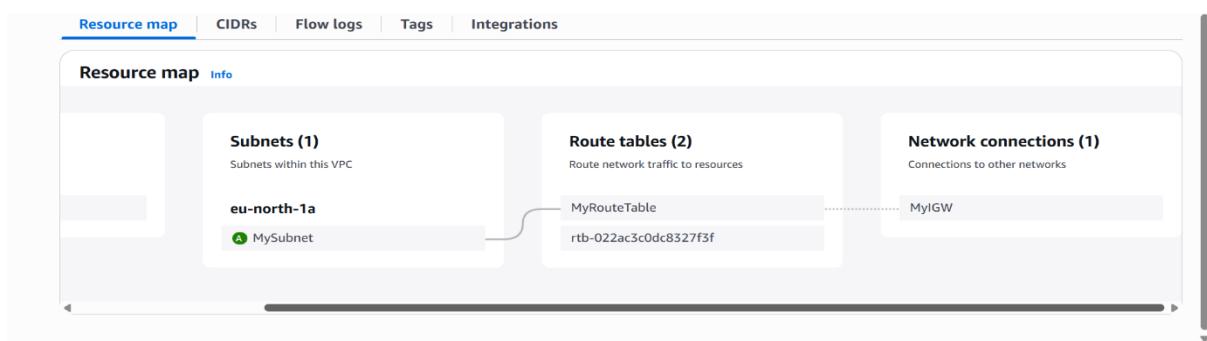
(An instance named AppServer was created, using the VPC and subnet that we created.

Instance summary for i-08cf32a6ffb3e4a7a (AppServer1) [Info](#)

Updated less than a minute ago

Instance ID i-08cf32a6ffb3e4a7a	Public IPv4 address 13.51.47.160 open address	Private IPv4 addresses 10.0.1.180
IPv6 address -	Instance state Running	Public DNS -
Hostname type IP name: ip-10-0-1-180.eu-north-1.compute.internal	Private IP DNS name (IPv4 only) ip-10-0-1-180.eu-north-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t3.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address 13.51.47.160 [Public IP]	VPC ID vpc-04dd6a1baf8e0fea7 (MyVPC)	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-0124bcedd67ea9487 (MySubnet)	Managed false
IMDSv2 Optional	Instance ARN arn:aws:ec2:eu-north-1:1976965259312:instance/i-08cf32a6ffb3e4a7a	

- Generated a resource map of the VPC to visualize the architecture and resource relationships.



5.2 Logging and IAM Role Setup

Effective monitoring requires detailed traffic logs and appropriate permissions. This phase focuses on enabling VPC Flow Logs to capture network traffic metadata and creating the required IAM roles that allow AWS services to interact securely.

Purpose: Enable detailed network traffic visibility and assign appropriate permissions for automation and monitoring.

- Created a VPC Flow Log named **MyFlow** to collect network traffic metadata and an IAM role with permissions for publishing these logs.

(I created the IAM role for the Flow Log.)

The screenshot shows the AWS IAM Roles page. The role name is "VPCFlowLogs-Cloudwatch-1747805015714". The "Summary" section includes the creation date (May 21, 2025, 10:56 (UTC+05:30)), last activity (12 minutes ago), ARN (arn:aws:iam::976965259312:role/service-role/VPCFlowLogs-Cloudwatch-1747805015714), and maximum session duration (1 hour). There are "Delete" and "Edit" buttons on the right.

This IAM policy configuration enables the ability to create log streams and publish log events within any log stream associated with any log group under the AWS account 976965259312 in the eu-north-1 region. Additionally, it grants permissions to describe existing log streams and create new log groups within the same account and region, supporting log management and monitoring tasks. Furthermore, it allows listing and describing all log groups globally, across all regions, which is useful for gaining visibility into log group configurations throughout the account.

(Permission Policy)

The screenshot shows the AWS IAM Policy editor. The policy document is as follows:

```

1  {
2    "version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": [
7          "logs:CreateLogStream",
8          "logs:PutLogEvents"
9        ],
10       "Resource": "arn:aws:logs:eu-north-1:976965259312:log-group:*:log-stream"
11     },
12     {
13       "Effect": "Allow",
14       "Action": [
15         "logs:DescribeLogStreams",
16         "logs>CreateLogGroup"
17       ],
18       "Resource": "arn:aws:logs:eu-north-1:976965259312:log-group:)"
19     },
20     {
21       "Effect": "Allow",
22       "Action": "logs:DescribeLogGroups",
23       "Resource": "*"
24     }
25   ]
26 }
  
```

The "Actions" tab is selected in the top right. A sidebar on the right lists various AWS services with checkboxes for selecting actions.

This IAM trust policy allows the AWS VPC Flow Logs service (vpc-flow-logs.amazonaws.com) to assume the role, but only if the request comes from your AWS account (976965259312) and specifically from VPC Flow Logs resources within the eu-north-1 region of that account. This ensures that only legitimate VPC Flow Logs within your account can use this role, adding a layer of security by restricting access based on account and resource ARN.

(Trust Relationship.)

The screenshot shows the AWS IAM Trust Relationships page for a role named "VPCFlowLogs-Cloudwatch-1747805015714". The "Trust relationships" tab is selected. The "Trusted entities" section displays a JSON policy document:

```
[{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": "vpc-flow-logs.amazonaws.com"}, "Action": "sts:AssumeRole", "Condition": {"StringEquals": {"aws:SourceAccount": "976965259312"}, "ArnLike": {"aws:SourceArn": "arn:aws:ec2:eu-north-1:976965259312:vpc-flow-log/*"}}}]}
```

(Created a flow log and set the destination to send data to CloudWatch Logs.)

The screenshot shows the "Create flow log" page under the "Your VPCs" section. The "Flow log settings" section includes:

- Name - optional:** MyFlow
- Filter:** All (radio button selected)
- Maximum aggregation interval:** 1 minute (radio button selected)
- Destination:** Send to CloudWatch Logs (radio button selected)
- Destination log group:** /aws/vpc/flow-logs

(Added the previously created service role.)

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.

/aws/vpc/flow-logs

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 [Info](#)
The IAM role that has permission to publish to the Amazon CloudWatch log group.

VPCFlowLogs-Cloudwatch-1747805015714

[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](#)

(The flow log was created successfully.)

Resource map | CIDRs | **Flow logs** | Tags | Integrations

Flow logs (1) [Info](#)

Name	Flow log ID	Filter	Destination type	Destination name
MyFlow	fl-01709aca222c58d90	ALL	cloud-watch-logs	/aws/vpc/flow-logs

- Verified successful delivery of flow logs to the designated CloudWatch Logs group by inspecting the log stream for incoming entries.

(The log group is created.)

aws | Search [Alt+S]

CloudWatch > Log groups > /aws/vpc/flow-logs

/aws/vpc/flow-logs

Log group details

Log class Info Standard	Metric filters 0	Data protection -
ARN arn:aws:logs:eu-north-1:976965259312:log-group:/aws/vpc/flow-logs:*	Subscription filters 0	Sensitive data count -
Creation time 4 hours ago	Contributor Insights rules -	Field indexes Configure
Retention Never expire	KMS key ID -	Transformer Configure
Stored bytes -	Anomaly detection Configure	

Log streams | Tags | Anomaly detection | Metric filters | Subscription filters | Contributor Insights | Data protection | Field in

(Creation of the log stream was successful.)

The screenshot shows the AWS CloudWatch Log Streams interface. At the top, there are tabs for Log streams, Tags, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, Data protection, and Field info. The Log streams tab is selected. Below the tabs, a search bar contains the placeholder 'Filter log streams or try prefix search'. To the right of the search bar are buttons for Delete, Create log stream, and Search all log streams. Further right are checkboxes for 'Exact match' and 'Show expired' with an 'Info' link. A pagination indicator shows '1' of '1' with a refresh icon. Below the search bar, there's a table with a single row. The first column is 'Log stream' with the value 'eni-0f054ad70df6fec76-all'. The second column is 'Last event time' with the value '2025-05-21 08:30:38 (UTC)'. On the far right of the table is a dropdown arrow.

(Verified logs within the log group successfully.)

The screenshot shows the AWS CloudWatch Log Events interface. The URL in the address bar is 'aws CloudWatch > Log groups > /aws/vpc/flow-logs > eni-0f054ad70df6fec76-all'. The main heading is 'Log events'. A note says 'You can use the filter bar below to search for and match terms, phrases, or values in your log events.' with a link 'Learn more about filter patterns'. Below is a search bar with placeholder 'Filter events - press enter to search', a time range selector from 'Clear' to 'Custom', and a 'UTC timezone' dropdown. There are also 'Actions', 'Start tailing', and 'Create metric filter' buttons. A 'Display' button is highlighted in blue. The main area lists log events with columns for 'Timestamp' and 'Message'. Each event entry includes a timestamp like '2025-05-21T08:59:39.000Z' and a detailed message log. A note at the bottom says 'No newer events at this moment. Auto retry paused. Resume' with a 'Resume' button. A 'Back to top' button is in the bottom right corner.

- Created a separate IAM role with policies granting Lambda function necessary permissions to access flow logs and modify network resources.

(The IAM role for the Lambda function was created.)

The screenshot shows the AWS IAM Roles interface. The URL in the address bar is 'aws IAM > Roles > LambdaVPCFlowLogsRole'. The role name is 'LambdaVPCFlowLogsRole' with an 'Info' link. A note says 'Allows Lambda functions to call AWS services on your behalf.' To the right is a 'Delete' button. Below the role name is a 'Summary' section. It shows 'Creation date' as 'May 20, 2025, 23:16 (UTC+05:30)', 'Last activity' as '6 minutes ago', and ARN as 'arn:aws:iam::976965259312:role/LambdaVPCFlowLogsRole'. It also shows 'Maximum session duration' as '1 hour'. There is an 'Edit' button in the top right of the summary section.

(Permissions granted to the role.)

The screenshot shows the AWS IAM Permissions page. On the left, there's a sidebar with navigation links like Dashboard, Access management, Roles, Policies, and Access reports. The main area is titled "Permissions policies (4) Info" and says "You can attach up to 10 managed policies." It features a search bar and a filter section labeled "Filter by Type" with "All types" selected. A table lists four policies: "AmazonEC2FullAccess" (AWS managed), "AWSLambdaBasicExecutionRole" (AWS managed), "CloudWatchLogsFullAccess" (AWS managed), and "ControllerLambdaPolicy" (Customer inline). Each row includes a checkbox, a preview icon, the policy name, its type, and the count of attached entities.

This policy grants broad permissions to manage and view EC2 security components, including describing security groups and network ACLs, as well as authorizing or revoking security group ingress rules and managing network ACL entries and network interfaces. Additionally, it allows starting and retrieving CloudWatch Logs queries, describing log groups and streams, and creating log streams and putting log events, including full access to all logs and specifically to the Lambda function log group /aws/lambda/ControllerLambda.

(Policy.)

The screenshot shows the AWS IAM Policy editor. At the top, there's a breadcrumb trail: IAM > Roles > LambdaVPCFlowLogsRole > Edit policy. Below that, a progress bar indicates "Step 2 Review and save". The main area is titled "Policy editor" and has tabs for "Visual", "JSON" (which is selected), and "Actions". The JSON code in the editor is as follows:

```
1 {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Action": [  
7                 "ec2:DescribeSecurityGroups",  
8                 "ec2:AuthorizeSecurityGroupIngress",  
9                 "ec2:RevokeSecurityGroupIngress",  
10                "ec2:DescribeNetworkAcls",  
11                "ec2:CreateNetworkAclEntry",  
12                "ec2:DeleteNetworkAclEntry",  
13                "ec2:DescribeNetworkInterfaceAttribute",  
14                "ec2:DescribeNetworkInterfaces"  
15            ],  
16            "Resource": "*"  
17        },  
18        {  
19            "Effect": "Allow",  
20            "Action": [  
21                "logs:StartQuery",  
22                "logs:GetQueryResults",  
23                "logs:DescribeLogGroups",  
24                "logs:DescribeLogstreams",  
25            ]  
26        }  
27    ]  
28}
```

To the right of the editor, there are sections for "Edit statement", "Add actions", "Choose a service" (with a dropdown menu showing "Filter services"), and "Included" and "Available" services lists. The included services list contains "EC2". The available services list includes AI Operations, AMP, API Gateway, API Gateway V2, ARC Zonal Shift, ASC, and Access Analyzer.

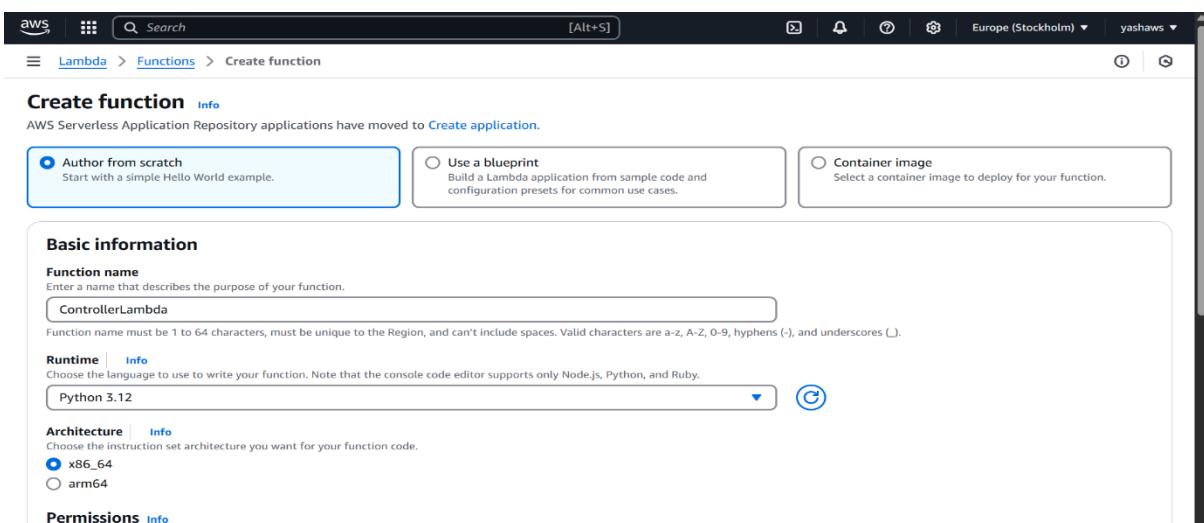
5.3 Lambda Function and Scheduling

At the heart of automation lies the Lambda function, which processes logs and applies security rules. This phase develops the function and sets up scheduled execution, ensuring continuous vigilance over network traffic.

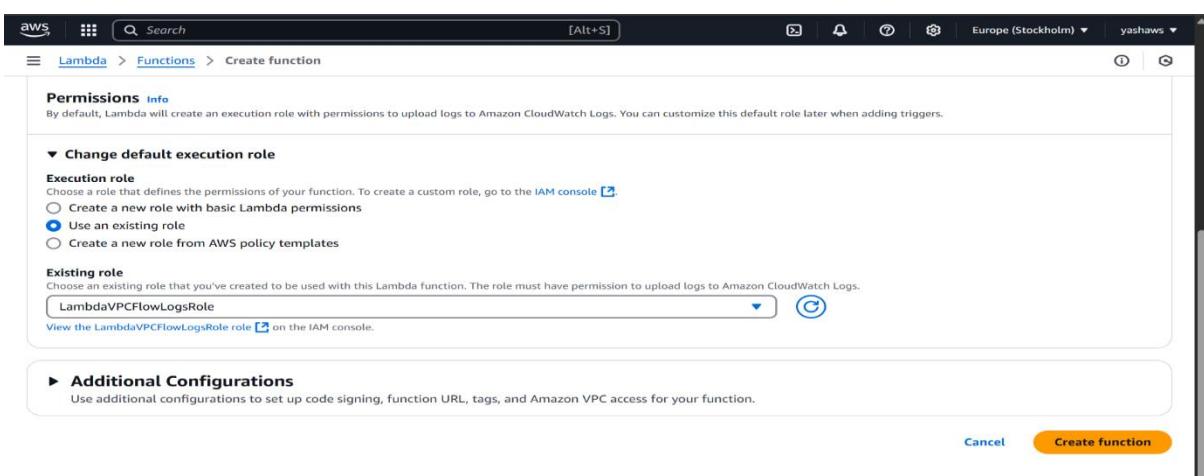
Purpose: Automate the analysis of network logs and the enforcement of dynamic security policies in near real-time.

- Developed the Lambda function named **ControllerLambda** that analyzes incoming VPC Flow Logs to detect unauthorized SSH attempts and programmatically modifies security groups and NACLs.

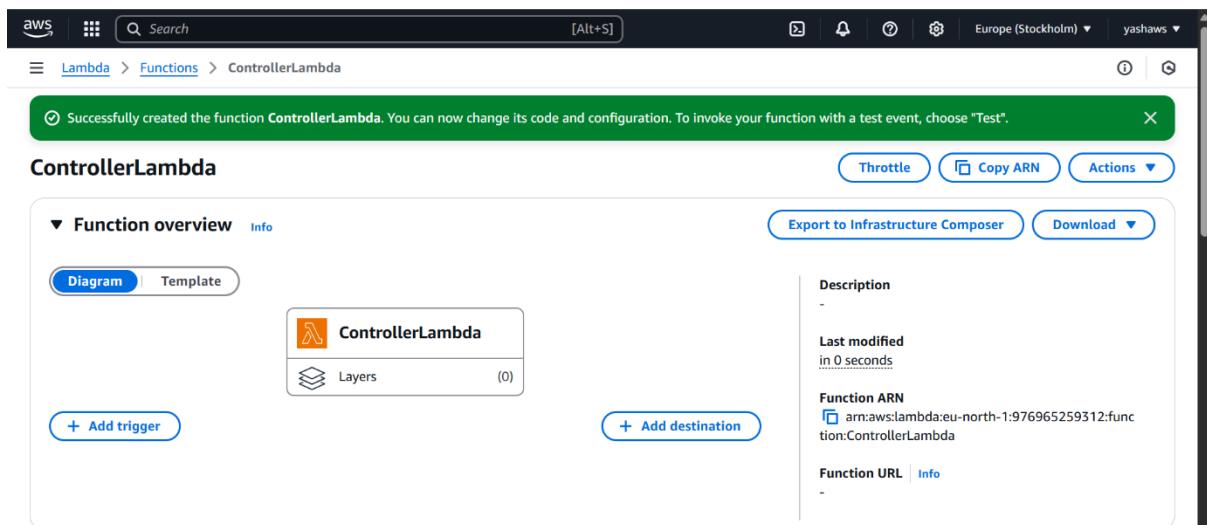
(A Lambda function was created.)



(Utilized the existing role created earlier.)



(Lambda function created successfully.)



The Lambda function code is provided below.

```
import boto3
import time
import ipaddress
import logging

ec2 = boto3.client('ec2')
logs = boto3.client('logs')

LOG_GROUP = '/aws/vpc/flow-logs'
SECURITY_GROUP_ID = 'sg-025b0819a4b5799ab'
NACL_ID = 'acl-0f1471843d20dfa98'
VPC_CIDR = '10.0.0.0/16'

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    logger.info("Lambda execution started")
```

```
end_time = int(time.time())
start_time = end_time - 5 * 60

query_string = """
fields @timestamp, srcAddr, dstAddr, dstPort, action
| filter dstPort = 22 and action = "ACCEPT"
| sort @timestamp desc
| limit 50
"""

logger.info(f"Starting Logs Insights query: {query_string.strip()}")
start_query_response = logs.start_query(
    logGroupName=LOG_GROUP,
    startTime=start_time,
    endTime=end_time,
    queryString=query_string,
    limit=50
)
query_id = start_query_response['queryId']
logger.info(f"Query started with ID: {query_id}")

timeout = 30
elapsed = 0
while True:
    response = logs.get_query_results(queryId=query_id)
    status = response['status']
    logger.info(f"Query status: {status}")
    if status in ['Complete', 'Failed', 'Cancelled']:
        break
    time.sleep(1)
    elapsed += 1
    if elapsed >= timeout:
        logger.error("Logs query timed out")
        return {'statusCode': 500, 'body': 'Query timeout'}
```

```
if status != 'Complete':
    logger.error(f"Logs query failed or cancelled: {status}")
    return {'statusCode': 500, 'body': 'Query not completed'}

blocked_ips = []
logger.info(f'Query returned {len(response["results"])} results')

for result in response['results']:
    src_ip = None
    dst_ip = None
    for field in result:
        if field['field'] == 'srcAddr':
            src_ip = field['value']
        elif field['field'] == 'dstAddr':
            dst_ip = field['value']

    logger.info(f'Found src IP: {src_ip}, dst IP: {dst_ip}')

for ip in [src_ip, dst_ip]:
    if ip:
        if is_unknown_ip(ip):
            logger.info(f'IP {ip} is outside VPC CIDR, attempting to block')
            removed = remove_sg_rule(ip)
            added = add_nacl_deny(ip)
            if removed or added:
                blocked_ips.append(ip)
        else:
            logger.info(f'IP {ip} is inside VPC CIDR, skipping')
    else:
        logger.warning("IP field missing")

logger.info(f'Blocked IPs in this run: {blocked_ips}')
```

```

return {
    'statusCode': 200,
    'body': f'Blocked IPs: {blocked_ips}'
}

def is_unknown_ip(ip):
    try:
        vpc_network = ipaddress.ip_network(VPC_CIDR)
        ip_obj = ipaddress.ip_address(ip)
        result = ip_obj not in vpc_network
        logger.info(f"is_unknown_ip({ip}) -> {result}")
        return result
    except ValueError:
        logger.error(f"Invalid IP address: {ip}")
        return False

def remove_sg_rule(ip):
    try:
        sg =
        ec2.describe_security_groups(GroupId=[SECURITY_GROUP_ID])['SecurityGroups'][0]
        perms = sg.get('IpPermissions', [])
        to_revoke = []

        for perm in perms:
            if perm.get('FromPort') == 22 and perm.get('ToPort') == 22 and
            perm.get('IpProtocol') in ['tcp', '-1']:
                for ip_range in perm.get('IpRanges', []):
                    if ip_range.get('CidrIp') == f'{ip}/32':
                        to_revoke.append({
                            'IpProtocol': perm['IpProtocol'],
                            'FromPort': perm['FromPort'],
                            'ToPort': perm['ToPort'],
                        })
    
```

```
'IpRanges': [ {'CidrIp': f'{ip}/32'}]
        })
    if to_revoke:
        ec2.revoke_security_group_ingress(GroupId=SECURITY_GROUP_ID,
IpPermissions=to_revoke)
        logger.info(f'Removed SG rule for IP {ip}')
        return True
    else:
        logger.info(f'No SG rule found to remove for IP {ip}')
        return False
except Exception as e:
    logger.error(f'Failed to remove SG rule for {ip}: {e}')
    return False
```

```
def find_free_rule_number(nacl, starting_point=100):
    used = {entry['RuleNumber'] for entry in nacl['Entries']}
    for num in range(starting_point, 0, -1):
        if num not in used:
            return num
    for num in range(starting_point + 1, 32766):
        if num not in used:
            return num
    return None
```

```
def add_nacl_deny(ip):
    try:
        nacl = ec2.describe_network_acls(NetworkAclIds=[NACL_ID])['NetworkAcls'][0]

        for entry in nacl['Entries']:
            if (entry['RuleAction'] == 'deny' and
                entry.get('CidrBlock') == f'{ip}/32' and
                entry.get('Protocol') == '6' and
```

```
entry.get('PortRange', {}).get('From') == 22 and
not entry.get('Egress', True)):
    logger.info(f"NACL deny rule already exists for IP {ip}")
    return False

allow_rule_numbers = [entry['RuleNumber'] for entry in nacl['Entries']]
if entry['RuleAction'] == 'allow' and not entry.get('Egress', True)]
```

if allow_rule_numbers:

```
    min_allow = min(allow_rule_numbers)
    rule_number = find_free_rule_number(nacl, starting_point=min_allow - 1)
else:
    rule_number = find_free_rule_number(nacl, starting_point=100)

if rule_number is None:
    logger.error("No available rule numbers to create deny rule")
    return False

ec2.create_network_acl_entry(
    NetworkAclId=NACL_ID,
    RuleNumber=rule_number,
    Protocol='6',
    RuleAction='deny',
    Egress=False,
    CidrBlock=f'{ip}/32',
    PortRange={'From': 22, 'To': 22}
)
logger.info(f"Added NACL deny rule for IP {ip} with rule number {rule_number}")
return True

except Exception as e:



```
 logger.error(f'Failed to add NACL deny rule for {ip}: {e}')
 return False
```


```

The Lambda function is the core component of the dynamic SDN controller. It automates the process of detecting unauthorized SSH login attempts and enforces network policies by modifying AWS Security Groups and Network ACLs based on real-time log analysis.

Function Overview

The Lambda function performs the following tasks:

1. Queries recent VPC Flow Logs for accepted SSH (port 22) connections using CloudWatch Logs Insights.
2. Extracts source and destination IPs from these logs.
3. Evaluates whether the IPs are external (i.e., not part of the internal VPC CIDR).
4. If an IP is suspicious:
 - It removes any existing Security Group rules allowing SSH from that IP.
 - Adds a new deny rule for the IP in the Network ACL.
5. Logs every step and decision made for auditing and debugging.

Core Code Logic – IP Evaluation Snippet

```
def is_unknown_ip(ip):  
    vpc_network = ipaddress.ip_network(VPC_CIDR)  
    ip_obj = ipaddress.ip_address(ip)  
    return ip_obj not in vpc_network
```

This helper function checks whether a given IP address is **outside the private VPC range**, treating it as suspicious if so. Only these IPs are subject to blocking.

Triggering Mechanism

The Lambda function is triggered every **5 minutes** using **Amazon EventBridge (formerly CloudWatch Events)**. This ensures continuous monitoring and rapid response to threats.

Security Operations Performed

Security Group Cleanup: The function scans for rules allowing SSH from suspicious IPs and revokes them using:

```
ec2.revoke_security_group_ingress()
```

NACL Rule Insertion:

A deny rule is added for port 22 using:

```
ec2.create_network_acl_entry()
```

Rule numbers are dynamically chosen to avoid conflict with existing entries.

Logging and Observability

The function logs all events via AWS CloudWatch, including:

- Queries executed
- IPs processed
- Rules added or removed
- Any errors or timeouts

- Configured Amazon EventBridge to trigger the Lambda function every 5 minutes, ensuring near real-time processing and response.

(Scheduled the Lambda function to run at 5-minute intervals.)

The screenshot shows the AWS EventBridge Schedules console. The top navigation bar includes the AWS logo, a search bar, and links for 'Schedules' and 'InvokeLambdaEvery5Minutes'. The main page displays a single schedule entry with the following details:

Schedule detail			
Schedule name	InvokeLambdaEvery5Minutes	Status	Enabled
Description	-		
Schedule group name	default	Schedule ARN	arn:aws:scheduler:eu-north-1:976965259312:schedule/default/InvokeLambdaEvery5Minutes
Action after completion	NONE		
Schedule start time		Flexible time window	
-		5 minutes	
Schedule end time		Created date	
-		May 21, 2025, 11:40:03 (UTC+05:30)	
Execution time zone		Last modified date	
Europe/Stockholm		May 21, 2025, 11:40:03 (UTC+05:30)	

Below the main table, there are tabs for 'Schedule', 'Target', 'Retry policy', 'Dead-letter queue', and 'Encryption'. The 'Schedule' tab is currently selected. A sub-section titled 'Schedule' shows a fixed rate of 5 minutes.

The screenshot shows the Amazon EventBridge console with the path: Amazon EventBridge > Schedules > InvokeLambdaEvery5Minutes. A green success message at the top states: "Schedule InvokeLambdaEvery5Minutes has been saved successfully." Below it, the "Target" tab is selected, showing the target configuration. The target is set to "ControllerLambda" (AWS Lambda), API is "Invoke", and Payload is "-". The "Target ARN" is listed as "arn:aws:lambda:eu-north-1:976965259312:function:ControllerLambda". The "Execution role" is "Amazon_EventBridge_Scheduler_LAMBDA_6dc0ab72b1". Other tabs include "Schedule", "Retry policy", "Dead-letter queue", and "Encryption".

5.4 Security Policy Verification and Attack Simulation

To validate the system's responsiveness, this phase involves testing the baseline security posture and simulating attack scenarios. It confirms the environment's readiness and the controller's detection capability.

Purpose: Test the baseline security posture and validate the detection and response capabilities of the system.

- Inspected existing NACL and security group rules, confirming they initially allowed broad access (0.0.0.0/0) to port 22 (SSH).

(All inbound traffic is allowed.)

The screenshot shows the AWS VPC Network ACLs console with the path: VPC > Network ACLs > acl-0f1471843d20dfa98. The details for the Network ACL "acl-0f1471843d20dfa98" are shown, including its association with subnet-0124bcdd67ea9487 / MySubnet, owner 976965259312, and VPC ID vpc-04dd6a1baf8e0fea7 / MyVPC. The "Inbound rules" tab is selected, displaying two rules: one allowing all traffic on port 22 from 0.0.0.0/0 with an "Allow" status, and another allowing all traffic on port 22 from 0.0.0.0/0 with a "Deny" status. Other tabs include "Outbound rules", "Subnet associations", and "Tags".

(The security group allows all incoming TCP traffic.)

The screenshot shows the AWS VPC Security Groups console. A specific security group, "sg-025b0819a4b5799ab - launch-wizard-4", is selected. The "Details" section shows the security group name, ID, owner, and various counts. The "Inbound rules" tab is active, displaying one rule: "sgr-0f6a5b7901f18811a" allowing SSH (TCP port 22) from 0.0.0.0/0. There is also an "Edit inbound rules" link.

Inbound rules (1)

Name	Security group rule ID	IP version	Type
-	sgr-0f6a5b7901f18811a	IPv4	SSH

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0f6a5b7901f18811a	SSH	TCP	22	Cu... 0.0.0.0/0	0.0.0.0/0

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

- Simulated an attack by attempting SSH logins from unauthorized IP addresses to trigger the Lambda function's detection logic.

(Use SSH on the attacker machine to connect to our instance.)

```
PS E:\> ssh -i E:\SSHkeyPair.pem ubuntu@13.62.19.139
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Wed May 21 09:10:01 UTC 2025

System load: 0.0          Processes:          106
Usage of /: 17.7% of 9.51GB   Users logged in:    1
Memory usage: 26%           IPv4 address for ens5: 10.0.1.180
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed May 21 08:33:27 2025 from 183.83.52.246
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-180:~$
```

5.5 Monitoring, Detection, and Enforcement Validation

This final phase verifies the end-to-end functionality of the SDN controller. It includes analyzing logs for threat detection, confirming the application of blocking rules, and ensuring that unauthorized access attempts are effectively thwarted.

- Monitored Lambda function logs in **/aws/lambda/ControllerLambda** for entries indicating detection of suspicious SSH login attempts.

Purpose: Confirm that the SDN controller correctly identifies threats and enforces security policies dynamically.

(Network traffic was successfully captured.)

The screenshot shows the AWS CloudWatch Log Events interface. On the left, there's a sidebar with navigation links: Favorites and recents, Dashboards, AI Operations (with Preview), Alarms, Logs (with Log groups, New, Log Anomalies, Live Tail, Logs Insights, New, Contributor Insights), Metrics, X-Ray traces, and Events (with Rules). The main area is titled "Log events" and contains a search bar with placeholder "Filter events - press enter to search", time range buttons for "1m", "1h", and "UTC timezone", and a "Display" button. Below these are two columns: "Timestamp" and "Message". The log entries are as follows:

Timestamp	Message
2025-05-21T09:12:55.250Z	[INFO] 2025-05-21T09:12:55.250Z 7e682d98-6b36-4267-bc59-4bccb49493af is_unknown_ip(10.0.1.18..)
2025-05-21T09:12:55.250Z	[INFO] 2025-05-21T09:12:55.250Z 7e682d98-6b36-4267-bc59-4bccb49493af IP 10.0.1.180 is inside..
2025-05-21T09:12:55.250Z	[INFO] 2025-05-21T09:12:55.250Z 7e682d98-6b36-4267-bc59-4bccb49493af is_unknown_ip(183.83.52..)
2025-05-21T09:12:55.250Z	[INFO] 2025-05-21T09:12:55.250Z 7e682d98-6b36-4267-bc59-4bccb49493af IP 183.83.52.246 is out..
2025-05-21T09:12:55.334Z	[INFO] 2025-05-21T09:12:55.334Z 7e682d98-6b36-4267-bc59-4bccb49493af No SG rule found to rem..
2025-05-21T09:12:55.404Z	[INFO] 2025-05-21T09:12:55.404Z 7e682d98-6b36-4267-bc59-4bccb49493af NACL deny rule already ..
2025-05-21T09:12:55.404Z	[INFO] 2025-05-21T09:12:55.404Z 7e682d98-6b36-4267-bc59-4bccb49493af Blocked IPs in this run: 7e682d98-6b36-4267-bc59-4bccb49493af Blocked IPs in this run: ['183.83.52.246']
2025-05-21T09:12:55.414Z	END RequestId: 7e682d98-6b36-4267-bc59-4bccb49493af

- Verified that the Lambda function successfully updated NACLs by blocking the malicious IP addresses.

(The suspicious IP address has been blocked in the NACL rules.)

The screenshot shows the AWS VPC Network ACL details page for 'acl-Of1471843d20dfa98'. The 'Details' tab is selected, showing the Network ACL ID (acl-Of1471843d20dfa98), Associated with subnet-0124bcedd67ea9487 / MySubnet, Default Yes, and VPC ID vpc-04dd6a1baf8e0fea7 / MyVPC. Below this, there are tabs for Inbound rules, Outbound rules, Subnet associations, and Tags. The Inbound rules section displays three entries:

Rule number	Type	Protocol	Port range	Source	Allow/Deny
101	SSH (22)	TCP (6)	22	183.83.52.246/32	Deny
102	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

- Observed connection reset behavior on SSH login attempts from blocked IPs, confirming enforcement effectiveness.

(On the attacking machine, the connection has been reset.)

```
PS E:\> ssh -i E:\SSHKeyPair.pem ubuntu@13.62.19.139
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Wed May 21 09:10:01 UTC 2025

System load: 0.0      Processes:          106
Usage of /: 17.7% of 9.51GB  Users logged in:   1
Memory usage: 26%        IPv4 address for ens5: 10.0.1.180
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed May 21 08:33:27 2025 from 183.83.52.246
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-180:~$ client_loop: send disconnect: Connection reset
PS E:\>
```

(Tried to connect again, but the connection was denied.)

```
PS E:\> ssh -i E:\SSHKeyPair.pem ubuntu@13.62.19.139
ssh: connect to host 13.62.19.139 port 22: Connection timed out
PS E:\>
```

- Reviewed detailed audit logs to ensure comprehensive recording of detection and mitigation actions.

(The logs show that the IP is blocked.)

Log events

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 Clear 1m 30m 1h 12h Custom UTC timezone

Display ▾

Timestamp	Message
2025-05-21T09:22:55.536Z	[INFO] 2025-05-21T09:22:55.536Z 7e682d9a-c336-4267-bc59-4bccb49493af Found src IP: 183.83.52.246, dst IP: 10.0.1.180
2025-05-21T09:22:55.536Z	[INFO] 7e682d9a-c336-4267-bc59-4bccb49493af Found src IP: 183.83.52.246, dst IP: 10.0.1.180
2025-05-21T09:22:55.536Z	[INFO] 2025-05-21T09:22:55.536Z 7e682d9a-c336-4267-bc59-4bccb49493af is_unknown_ip(183.83.52.246) -> True
2025-05-21T09:22:55.536Z	[INFO] 2025-05-21T09:22:55.536Z 7e682d9a-c336-4267-bc59-4bccb49493af IP 183.83.52.246 is outside VPC CIDR, attempting to block
2025-05-21T09:22:55.536Z	[INFO] 7e682d9a-c336-4267-bc59-4bccb49493af IP 183.83.52.246 is outside VPC CIDR, attempting to block
2025-05-21T09:22:55.778Z	[INFO] 2025-05-21T09:22:55.778Z 7e682d9a-c336-4267-bc59-4bccb49493af No SG rule found to remove for IP 183.83.52.246
2025-05-21T09:22:55.778Z	[INFO] 7e682d9a-c336-4267-bc59-4bccb49493af No SG rule found to remove for IP 183.83.52.246
2025-05-21T09:22:55.963Z	[INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af Added NACL deny rule for IP 183.83.52.246 with rule

Back to top ^

(Detailed log entries)

The screenshot shows a CloudWatch Logs interface with the following details:

- Log events**: The main heading.
- Actions**: A button with a gear icon.
- Start tailing**: A button.
- Create metric filter**: A button.
- Filter bar**: "Filter events - press enter to search".
- Time range**: Clear, 1m, 30m, 1h, 12h, Custom, UTC timezone.
- Display dropdown**: Set to "Display ▾".
- Table headers**: "Timestamp" and "Message".
- Log entries** (partial list):
 - [INFO] 2025-05-21T09:22:55.778Z 7e682d9a-c336-4267-bc59-4bccb49493af No SG rule found to remove for IP 183.83.52.246
 - [INFO] 2025-05-21T09:22:55.778Z 7e682d9a-c336-4267-bc59-4bccb49493af No SG rule found to remove for IP 183.83.52.246
 - [INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af Added NACL deny rule for IP 183.83.52.246 with rule number 101
 - [INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af Added NACL deny rule for IP 183.83.52.246 with rule number 101
 - [INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af is_unknown_ip(10.0.1.180) -> False
 - [INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af IP 10.0.1.180 is inside VPC CIDR, skipping
 - [INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af Found src IP: 10.0.1.180, dst IP: 183.83.52.246
 - [INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af is_unknown_ip(10.0.1.180) -> False
 - [INFO] 2025-05-21T09:22:55.963Z 7e682d9a-c336-4267-bc59-4bccb49493af IP 10.0.1.180 is inside VPC CIDR, skipping
- Back to top**: A yellow button.

6. Results And Findings

This section presents the observed outcomes of the implemented SDN-style network controller using AWS Lambda. Each finding highlights a critical aspect of system performance, automation accuracy, and threat response capability.

6.1 Successful Flow Log Capture

The VPC Flow Logs were effectively configured to monitor network traffic to and from the EC2 instance. The logs were successfully streamed to CloudWatch, ensuring full visibility into all connection attempts. This foundational step proved critical for detecting unauthorized access patterns and feeding data to the Lambda function for real-time analysis.

6.2 Lambda Detection Logic Worked as Expected

The custom Lambda function, written in Python, accurately parsed flow logs and identified unauthorized SSH login attempts. It correctly filtered logs for port 22, isolated unknown or suspicious IP addresses, and triggered the appropriate response. This validated the effectiveness of the custom detection logic embedded in the function.

6.3 Automated Policy Enforcement

Upon detecting malicious traffic, the Lambda function dynamically updated the Network ACL (NACL) to block the source IP address. This confirmed that real-time policy enforcement was functioning correctly, without requiring manual intervention, thereby achieving the goal of an autonomous SDN controller.

6.4 EventBridge Scheduling Triggered Lambda Reliably

The AWS EventBridge schedule rule was configured to trigger the Lambda function every 5 minutes. During testing, the trigger executed consistently and reliably, allowing for continuous monitoring. This regular execution cycle ensures timely detection and response to any future suspicious traffic.

6.5 Confirmed Blockage and SSH Denial

Following the IP block by the Lambda, SSH access attempts from the attacker IP were met with connection resets, confirming that the mitigation mechanism worked in real-time.

7. Recommendations

Based on the implementation and observations, the following recommendations can enhance the security, scalability, and robustness of the system.

7.1 Expand Policy Coverage Beyond Port 22

Currently, the controller targets port 22 (SSH). However, attackers may exploit other common ports such as HTTP (80), HTTPS (443), RDP (3389), or MySQL (3306). Expanding the Lambda logic to analyze a broader range of ports would enhance threat detection and protect a wider surface of potential attack vectors.

7.2 Integrate with SNS for Alerting

To increase visibility for administrators, integrating the Lambda function with Amazon SNS would provide real-time alerts via email or SMS whenever an attack is detected or a policy is enforced. This ensures rapid awareness and facilitates human oversight for potentially critical actions.

7.3 Implement IP Whitelisting

Introducing a whitelist feature in the Lambda code can prevent trusted IPs (e.g., internal users, management systems) from being mistakenly blocked. This is especially important in environments with shared or fluctuating IP usage, reducing the risk of service disruptions due to false positives.

7.4 Log Archival and Retention Strategy

While CloudWatch provides good short-term log storage, a long-term retention and archival strategy using Amazon S3 is advisable. Archiving flow logs ensures that historical network data is preserved for forensic analysis, regulatory compliance, or machine learning applications in the future.

7.5 Add Rate Limiting and Throttling Controls

Instead of instantly blocking an IP after one connection attempt, implementing rate-limiting logic (e.g., allow 3 failed attempts within 5 minutes) adds nuance to the detection mechanism. This helps avoid false positives from legitimate users making occasional mistakes, while still preventing brute-force attempts.

8. Conclusion

The implementation of a Lambda-based SDN-style network controller within the AWS environment was both technically effective and strategically valuable. Through structured phases—starting from the creation of VPC components and EC2 deployment, to the configuration of flow logs and Lambda automation—the system was designed to monitor and respond to suspicious network activity in real-time. The core mechanism relied on analyzing VPC Flow Logs, detecting unauthorized SSH login attempts, and dynamically modifying Network ACLs to block malicious IP addresses. The use of AWS Lambda, IAM roles, and EventBridge scheduling ensured that the entire process remained automated, scalable, and secure.

Moreover, the test simulation using SSH login attempts confirmed the accuracy of the detection logic and the reliability of the enforcement mechanism. The solution also upheld core cloud security principles such as least privilege access, real-time monitoring, and automated mitigation. The success of this project demonstrates a practical, lightweight alternative to traditional SDN

controllers—well-suited for dynamic cloud environments. Going forward, integrating alerting systems, expanding protocol coverage, and incorporating analytics dashboards could further enhance visibility, control, and response capabilities.