

Automating Incident Response using Amazon Alexa and AWS

Submitted in partial fulfillment of the requirements of

Cloud Computing Laboratory (CSL803)

for

Third Year of Computer Engineering

By

Bhaves Dhake 19102A0027

Chaitanya Shetye 19102A0072

Devish Gawas 19102A0024

Pratik Borhade 19102A0049

Under the Guidance of

Prof. Divya Nimbalkar

Department of Computer Engineering



Vidyalankar Institute of Technology
Wadala(E), Mumbai-400437

University of Mumbai

2021-22

CERTIFICATE OF APPROVAL

This is to certify that the project entitled

**“Automating Incident Response using
Amazon Alexa and AWS”**

is a bonafide work of

Bhavesh Dhake 19102A0027

Chaitanya Shetye 19102A0072

Devish Gawas 19102A0024

Pratik Borhade 19102A0049

submitted to the University of Mumbai in partial fulfillment of

Cloud Computing Laboratory (CSL803)

for

Final Year of Computer Engineering

Guide
(Name)

Head of Department

Principal

Mini Project Report Approval

This project report entitled *Automating Incident Response using Amazon Alexa and AWS* by

- 1. Bhavesh Dhake 19102A0027**
- 2. Chaitanya Shetye 19102A0072**
- 3. Devish Gawas 19102A0024**
- 4. Pratik Borhade 19102A0049**

is approved for Mini Project (CSM501) for Third Year of Computer Engineering.

Internal Examiner





External Examiner

Date:

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

	Name of student	Roll No.	Signature
1)	Bhavesh Dhake	19102A0027	
2)	Chaitanya Shetye	19102A0072	
3)	Devish Gawas	19102A0024	
4)	Pratik Borhade	19102A0049	

Date:

Place: Mumbai

Acknowledgements

This Project wouldn't have been possible without the support, assistance, and guidance of a number of people whom we would like to express our gratitude to. First, we would like to convey our gratitude and regards to our mentor **Prof. Divya Nimbalkar** for guiding us with his constructive and valuable feedback and for his time and efforts. It was a great privilege to work and study under his guidance.

We would like to extend our heartfelt thanks to our Head of Department, **Dr. Sachin Bojewar** for overseeing this initiative which will in turn provide every Vidyalankar student a distinctive competitive edge over others.

We appreciate everyone who spared time from their busy schedules and participated in the survey. Lastly, we are extremely grateful to all those who have contributed and shared their useful insights throughout the entire process and helped us acquire the right direction during this research project.

Abstract

Cyber assaults and data breaches are appearing on security radars at an alarming rate. If your organization does not have a strong and timely response process in place to such security incidents, it is vulnerable to a wide range of threats, including ransomware, phishing attempts, zero-day exploits, Man-in-the-Middle (MitM) attacks, Distributed Denial-of-Service (DDoS) attacks, and SQL injections, to name a few. Automation speeds up routine reactions and repetitive processes, requiring little to no human participation to detect and respond to security risks and occurrences. Automation in incident response attempts to assist enterprises in achieving a 24-hour defensive system. The benefits of automating incident response are limitless for any business looking to better its defenses in how it manages and responds to attacks in this continuously changing environment. The business as a whole, and the security operations team, in particular, benefit from having an efficient – and automated – incident response strategy in place. This implemented project tries to solve these problems by implementing an Automated Incident Response mechanism using AWS and the Alexa device.

Table of Contents

Sr No	Description	Page No
1	Introduction	8-9
2	Problem Definition	10
3	Literature Survey	11-12
4	Proposed System	13-19
5	Results and Discussion	20-22
6	Conclusion & Future Scope	23
7	References	24
8	Recorded video link	24

1. Introduction

When you see the term Incident Response, what that refers to is an organization's ability to identify and investigate attacks and breaches and reduce their impact. We call this process, Assess and Mitigate. This has often been done in the past with human elements monitoring traffic, investigating suspicious activity, drafting procedures when new threats arrive, etc.

However, as the name suggests, automated incident response eliminates the human element from the process. It automates repetitive tasks, expedites threat detection and response, and provides 'round-the-clock defense, allowing your SOC team the time and space to further develop and strengthen your security posture in other ways.

Analysts may allocate their time to more critical and less monotonous duties by automating incident response. Without automation, security analysts waste significant time manually searching through alarms from diverse security solutions to determine which require action. Because it takes longer to extract the true risks from the noise, the amount of time spent on regular data collection increases their mean time to respond (MTTR) to important threats. Automation allows analysts to focus on the key issues that demand their attention and speeds up data gathering, placing pertinent facts at the analyst's fingertips for actual analysis.

It's critical to expediting the incident response process in order to minimize the potential damage of a cyber incident. Manual analyses of events are rarely feasible, and neither are manual reviews of every alert raised by security tools. Automated incident response addresses these limitations.

The benefits of automating the processes involved in responding to rapidly evolving cyber threats and incidents cannot be overemphasized. The following summarizes the primary ways organizations benefit from automating their incident response.

1. **Better decision making** - Having an automated IR plan in place will not only speed up the decision-making process in the event of an attack but also ensures that the right decision-makers for every action or threat level are clearly established, defined, and automatically engaged when required.
2. **Damage limitation** - An automated incident response plan puts your organization in a better position to take strong and swift actions in the event of a cyber attack or security breach to limit its effect on the overall business.

3. ***Internal and external coordination*** - An effective automated IR plan helps coordinate the interactions not only between an organization's internal departments and units but also externally with suppliers and partners in the event of a security incident. Automatically bringing the relevant parties together as soon as a security event occurs is vital to managing risk and reducing brand reputation damage.
4. ***Improved MTTD and MTTR*** - Businesses with automated incident response detect and respond a lot faster to threats and attacks than those still running entirely manual processes in their incident response. Automation rapidly improves an organization's mean time to respond and mean time to detect security threats or breaches by speeding the identification of real threats from false positives.
5. ***Reduced SOC operational costs*** - Since SOC analysts no longer have to focus on repetitive and unproductive tasks, valuable man-hours are redirected to more productive tasks more alerts can be handled by fewer people. Overall, automation reduces the running costs of the SOC.

The right technology platform is essential to automate incident detection and response. Such tools provide integrated workflows, automated scripts, and pre-built tasks, so the organization's security infrastructure can automatically take actions for threat detection, response, containment, and closure.

2. Problem Definition

For many organizations, security tools generate an overwhelming number of alerts. To determine whether these alerts refer to genuine threats or false positives, analysts have to manually review each alert. This is fine as long as alerts are low, but for most businesses and organizations, SOC teams can spend days tracking down one day's worth of alerts. This leads to what we call alert fatigue. Alert fatigue often results in genuine issues being ignored, which makes the organization far more vulnerable. Automated incident response takes care of this problem by completely eliminating the human element from alert analysis and response. This benefit also enables security teams to analyze and remediate more threats, and thus strengthen enterprise security.

Assume you have a hosted EC2 instance or a Beanstalk instance running in AWS that hosts your website/servers, and you discover that the hosted website/server is down and you are receiving several email messages about it. This might be the result of an assault on your website/server. To avoid this, just ask your Alexa device (any Alexa device will suffice) to check the status of the instances and promptly shut down the afflicted one. The Alexa gadget will do the task for you.

Added benefits of using an automated system for IR include:

- faster response times
- optimized threat intelligence
- reduced manual operations and standardization of processes
- streamlined operations
- reduced cyber security impact
- easy technology and tools integration
- lowered costs, and
- automated reporting and metrics capabilities.

3. Literature Survey

Kroll, Red Canary, and VMware conducted a survey of over 400 information security and 100 legal and compliance leaders from companies with over \$500M in annual revenue to capture the current state of incident response from a technical perspective.

Findings reveal that:

- Less than half (48%) of organizations conduct regular security readiness exercises with corporate leadership.
- Over 90% of respondents are not fully confident in their organization's ability to identify the root cause of a cyber attack
- Nearly half (46%) of organizations are unable to contain a threat less than an hour after the initial compromise, and most respondents identified 24/7 detection, response, containment and remediation as a primary goal for 2021.

When asked about the current threat landscape, the most pressing concerns were:

- Exposure to ransomware attacks (57%)
- Reduction in endpoint visibility due to employees working from home (54%)
- Hasty migration to cloud services (50%)

Close to 70% of organizations are receiving 100 or more threat alerts every day, but only 20% are investigating more than 20 events per day. Closing the gap between alerts and investigations is key to a more efficient incident response process.

TOP INCIDENT RESPONSE CHALLENGES

N=500



Nearly half (46%) of organizations are unable to contain a threat in less than an hour after initial compromise, and for 23% of organizations that reported more than three data compromises in the past 12 months, containment takes at least 12 hours.

The influence of automated incident response is most noticeable when it comes to recognizing and responding to threats in real-time. For example, 91 percent of cyberattacks begin with a phishing email and have an automated incident response system in place. These alarms and hazards, however, may be addressed successfully without the need for human interaction. Automation minimizes the need for analysts to go through hundreds of warnings daily, from obtaining malware intelligence to following established protocols and remediating risks.

However, most firms are still in the early stages of incorporating automation into their IR procedures. According to the SANS Institute, the most automated operations are for remotely distributing bespoke content or signatures from security vendors and restricting command and control to malicious IP addresses, followed by eliminating rogue files. At the moment, the processes that are least likely to be automated include isolating infected workstations from a network during cleanup and shutting down, and taking systems offline.

4. Proposed system:

The proposed system is quite simple. The steps of our proposed system are as follows:

- a. Suppose there is an EC2 instance or a BeanStalk instance running on AWS where a user/admin is hosting their server/website.
- b. Then there is a malicious body who tries to access the server/website using his/her machine with their own set of tools.
- c. The moment where the malicious body starts the assault on the server/website, an email is sent to the owner of the instance with a body that says that there has been an illegal attempt to access the server and there are many packets flowing which is creating massive traffic.
- d. Then it is the responsibility of the owner who receives the email to mitigate this issue.
- e. The owner then fires up the Alexa device or the Alexa developer console and specifies what he has to do with the instance.
- f. As of now, there are only two commands that the owner can execute:
 - i. Status Report - This command shows the owner how many instances are running and their description.
 - ii. Shut Down Instance - This command shuts down the instance which is under assault and saves the server from being compromised.
- g. After execution of these commands, a notification is sent to the owner regarding the changes that happened in the state of the instance.

Now, discussing the AWS Services we have used in our Project. We have used a total of 6 AWS Services which are as follows:

1. ***AWS SNS (Simple Notification Service)*** - Amazon Web Services Simple Notification Service (AWS SNS) is a web service that automates the process of sending notifications to the subscribers attached to it. It uses the publishing/subscribe paradigm for the push and delivery of messages. The data loss is prevented by storing the data across multiple availability zones. It provides developers with the highly scalable, cost-effective, and flexible capability to publish messages from an application and sends them to other applications. We have used this service for the purpose of sending notifications if there is any change in the state of the running instance.
2. ***AWS EC2 (Elastic Compute Cloud)*** - Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for the capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios. We are using EC2 for hosting a server, and alternatively, we can also use AWS BeanStalk where we can host a website

and implement this project.

3. **AWS EventBridge** - Amazon EventBridge is an AWS service that enables you to respond to state changes in an AWS resource. You can use AWS Step Functions Standard Workflows with EventBridge, while Express Workflows do not emit events to EventBridge. There are two ways to use Step Functions Standard Workflows with EventBridge. You can configure Step Functions to emit EventBridge events when an execution status changes. This enables you to monitor your workflows without having to constantly poll using the DescribeExecution API. Based on changes in state machine executions you can use an EventBridge target to start new state machine executions, call AWS Lambda functions, publish messages to Amazon Simple Notification Service (Amazon SNS) topics, and more. You can also configure a Step Functions state machine as a target in EventBridge. This enables you to trigger an execution of a Step Functions workflow in response to an event from another AWS service. As the definition of EventBridge suggests, we are using this service for capturing events happening to the hosted instance and then integrating it with the AWS SNS to send the notifications accordingly.
4. **AWS Lambda** - AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. Therefore you don't need to worry about which AWS resources to launch, or how well you manage them. Instead, you need to put the code on Lambda, and it runs.
In AWS Lambda the code is executed based on the response of events in AWS services such as add/delete files in the S3 bucket, HTTP requests from Amazon API gateway, etc. However, Amazon Lambda can only be used to execute background tasks. AWS Lambda function helps you to focus on your core product and business logic instead of managing operating system (OS) access control, OS patching, right-sizing, provisioning, scaling, etc. In here Python code is being run, and this function is the base of this project. Without this, nothing will work. After writing the code, do not just upload the code as it is in the Lambda Function. The code is needed to be packaged and then uploaded. The steps to package the code are as follows:
 1. mkdir directory
 2. cd directory
 3. virtualenv --python=/usr/local/bin/python3.9 .
 4. source bin/activate
 5. pip install boto3
 6. pip install ask-sdk
 7. deactivate
 8. cd lib/python3.7/site-packages
 9. vi lambda_function.py

Now you write the code the lambda_function.py file and compress the whole code package into a zip file and then upload.

The Python code is as follows:

```

import logging

from ask_sdk_core.skill_builder import SkillBuilder
from ask_sdk_core.dispatch_components import AbstractRequestHandler
from ask_sdk_core.dispatch_components import AbstractExceptionHandler
from ask_sdk_core.utils import is_request_type, is_intent_name
from ask_sdk_core.handler_input import HandlerInput

from ask_sdk_model.ui import SimpleCard
from ask_sdk_model import Response

sb = SkillBuilder()

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

class LaunchRequestHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_request_type("LaunchRequest")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speech_text = "Welcome Admin. You can say Hello or You can ask me- Status Report, Shutting down an instance. Which one would you like to try?"

        handler_input.response_builder.speak(speech_text).set_card(
            SimpleCard("Hello World", speech_text)).set_should_end_session(
                False)
        return handler_input.response_builder.response

class HelloWorldIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_intent_name("HelloWorldIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speech_text = "Hello, People? "

        handler_input.response_builder.speak(speech_text).set_card(
            SimpleCard("Hello World", speech_text)).set_should_end_session(
                True)
        return handler_input.response_builder.response

```

```

class HelpIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_intent_name("AMAZON.HelpIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speech_text = "You can say hello to me!"

        handler_input.response_builder.speak(speech_text).ask(
            speech_text).set_card(SimpleCard(
                "Hello World", speech_text))
        return handler_input.response_builder.response

class CancelOrStopIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return (is_intent_name("AMAZON.CancelIntent")(handler_input) or
            is_intent_name("AMAZON.StopIntent")(handler_input))

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speech_text = "Goodbye!"

        handler_input.response_builder.speak(speech_text).set_card(
            SimpleCard("Hello World", speech_text))
        return handler_input.response_builder.response

class FallbackIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_intent_name("AMAZON.FallbackIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speech_text = (
            "The Hello World skill can't help you with that. "
            "You can say hello!!")
        reprompt = "You can say hello!!"
        handler_input.response_builder.speak(speech_text).ask(reprompt)
        return handler_input.response_builder.response

class SessionEndedRequestHandler(AbstractRequestHandler):

```



```

def can_handle(self, handler_input):
    # type: (HandlerInput) -> bool
    return is_request_type("SessionEndedRequest")(handler_input)

def handle(self, handler_input):
    # type: (HandlerInput) -> Response
    return handler_input.response_builder.response

class CatchAllExceptionHandler(AbstractExceptionHandler):
    def can_handle(self, handler_input, exception):
        # type: (HandlerInput, Exception) -> bool
        return True

    def handle(self, handler_input, exception):
        # type: (HandlerInput, Exception) -> Response
        logger.error(exception, exc_info=True)

        speech = "Sorry, there was some problem. Please try again!!"
        handler_input.response_builder.speak(speech).ask(speech)

        return handler_input.response_builder.response

class GatherServersIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_intent_name("GatherServersIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        import boto3
        total_instance_count = 0
        client = boto3.client('ec2')
        aws_regions = [region['RegionName'] for region in
            client.describe_regions()['Regions']]
        for region in aws_regions:
            ec2 = boto3.resource('ec2', region_name=region)
            instances = ec2.instances.filter(Filters=[{'Name': 'instance-state-name',
                'Values': ['running']}])
            instance_count = int(len([instance.id for instance in instances]))
            total_instance_count = total_instance_count + instance_count
        if total_instance_count > 0:
            if total_instance_count == 1:
                speech_text = f"There is {total_instance_count} instance running in AWS"
            elif total_instance_count > 1:
                speech_text = f"There are {total_instance_count} instances running in

```

```

AWS"
    else:
        speech_text = "I could not find any servers running right now."
        handler_input.response_builder.speak(speech_text).set_card(
            SimpleCard("Hello World", speech_text)).set_should_end_session(
                False)
        return handler_input.response_builder.response

class TurnOffInstancesIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_intent_name("TurnOffInstancesIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        import boto3
        client = boto3.client('ec2')
        aws_regions = [region['RegionName'] for region in
            client.describe_regions()['Regions']]
        for region in aws_regions:
            ec2 = boto3.resource('ec2', region_name=region)
            instances = ec2.instances.filter(Filters=[{'Name': 'instance-state-name',
                'Values': ['running']}])
            for instance in instances:
                server_client = boto3.client('ec2', region_name=region)
                response = server_client.stop_instances(InstanceIds=[instance.id])
            speech_text = "Acknowledged. Turning off instances."
            handler_input.response_builder.speak(speech_text).set_card(
                SimpleCard("Hello World", speech_text)).set_should_end_session(
                    True)
            return handler_input.response_builder.response

sb.add_request_handler(LaunchRequestHandler())
sb.add_request_handler>HelloWorldIntentHandler())
sb.add_request_handler(HelpIntentHandler())
sb.add_request_handler(CancelOrStopIntentHandler())
sb.add_request_handler(FallbackIntentHandler())
sb.add_request_handler(SessionEndedRequestHandler())
sb.add_request_handler(GatherServersIntentHandler())
sb.add_request_handler(TurnOffInstancesIntentHandler())

sb.add_exception_handler(CatchAllExceptionHandler())

lambda_handler = sb.lambda_handler()

```

5. ***AWS S3 (Simple Storage Service)*** - Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements. This service is being used to store the Lambda Function code. The reason S3 is being implemented is due to the huge size of the packaged code.
6. ***AWS IAM (Identity and Access Management)*** - AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources. This service is being implemented for ensuring that the Lambda Function has the correct and a proper authorization to describe the instances and stop the instances under assault.

5. Results & Discussion:

1. Active EC2 instance created.

The screenshot displays the AWS Management Console interface. The top navigation bar includes links to Mail - Bhavesh Dhake, EC2 Management Console, AWS Management Console, IAM Management Console, and Developer Console. The left sidebar shows the 'Instances' section under 'EC2 Dashboard', with a list of instance types, launch templates, spot requests, savings plans, reserved instances, dedicated hosts, scheduled instances, and capacity reservations. The main content area shows a table of instances with one instance, 'test8', in the 'Running' state. Below the table, the 'Instance: i-09ac1a35704806c83 (test8)' details are displayed, including the instance ID, IP addresses, instance state, and various DNS names.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
test8	i-09ac1a35704806c83	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-196-218-225.co...	54.196.218.225	-

Instance: i-09ac1a35704806c83 (test8)

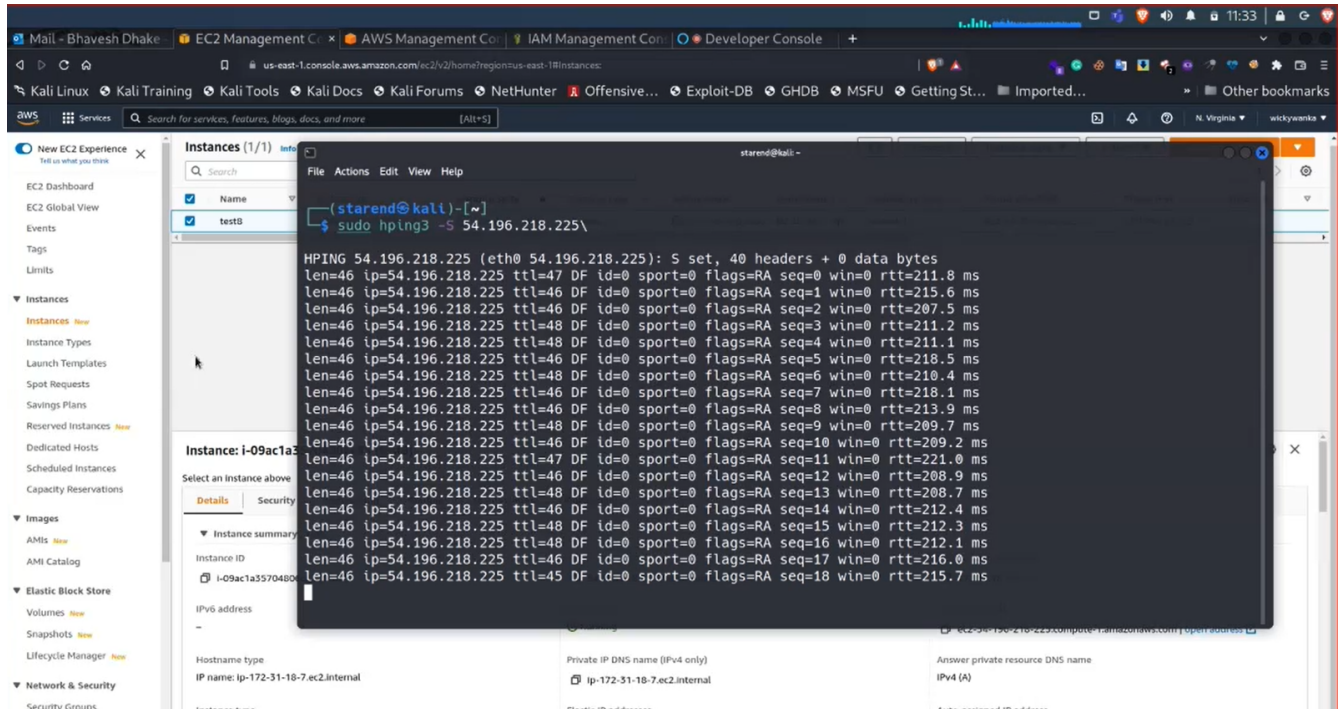
Select an Instance above

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

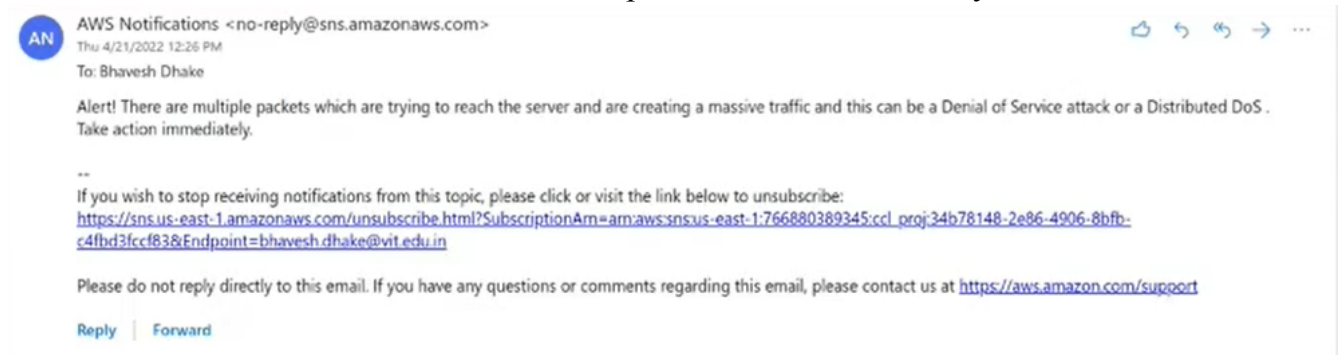
Instance summary | Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-09ac1a35704806c83 (test8)	54.196.218.225 open address	172.31.18.7
IPv6 address	Instance state	Public IPv4 DNS
-	Running	ec2-54-196-218-225.compute-1.amazonaws.com open address
Hostname type	Private IP DNS name (IPv4 only)	Answer private resource DNS name
IP name: ip-172-31-18-7.ec2.internal	ip-172-31-18-7.ec2.internal	IPv4 (A)
Instance type	Elastic IP addresses	Auto-assigned IP address

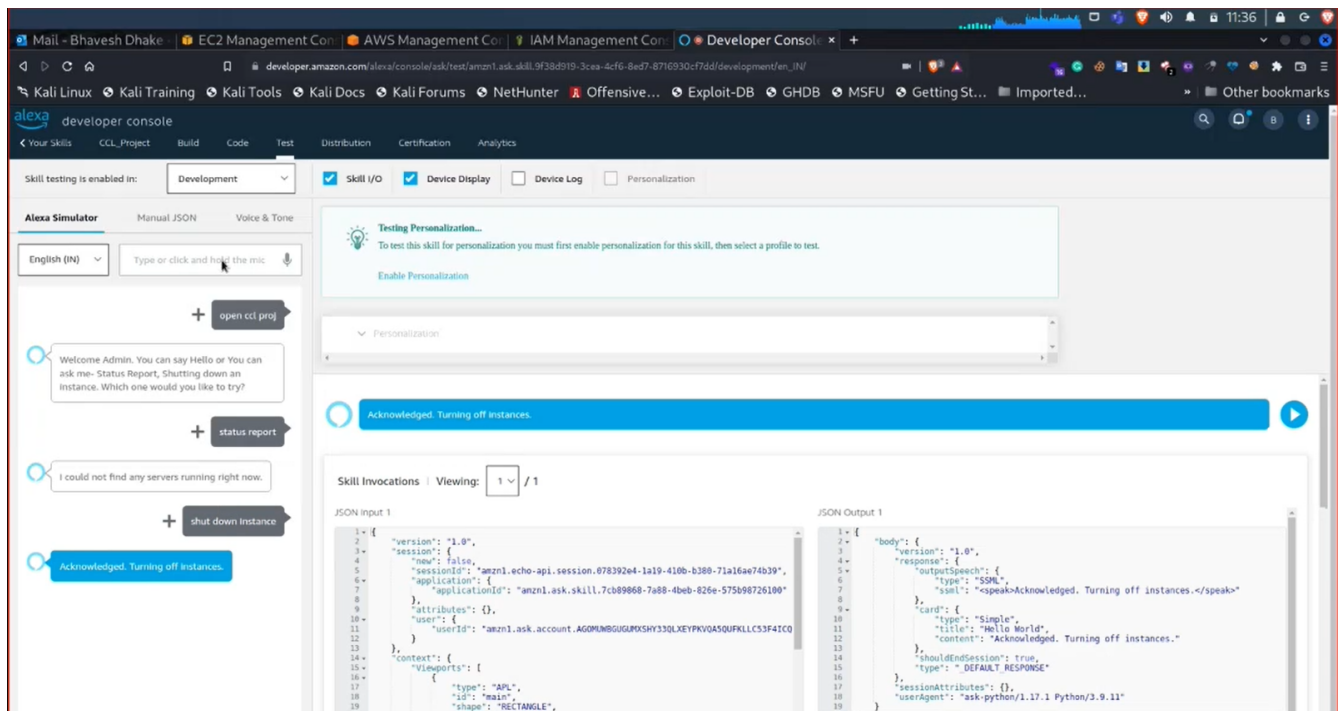
2. Demonstration of a threat to the server(EC2 instance) by using a tool that is used for Denial of Service attacks.



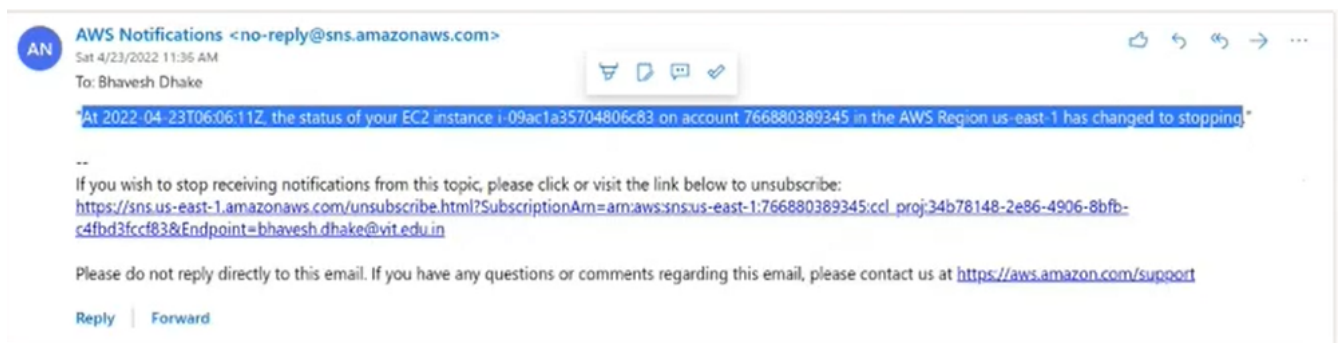
3. Push notification received when multiple threats are received by the server:



4. Alexa developer console where the verbal commands can be executed:



5. Push notification is received when the running instance is closed by Alexa using the voice command:



6. Conclusion & Future scope.

The future iteration of this project includes:

- Writing a more advanced Lambda Function.
- An advanced IR automation for various types of attacks.
- Detailed information about ongoing instances.
- Integration with AWS CloudWatch for better analysis and monitoring.

In conclusion, a robust incident response process is critical to every organization's cybersecurity infrastructure. Because manual processes cannot always provide the proactivity, fast response, or real-time mitigation required to deal with modern threats and threat actors, however, new tools have been developed to help counteract these increasingly complicated threats. Automated incident response provides the solution to these limitations. By investing in automated tools, organizations can strengthen their cybersecurity posture and set themselves up for success.

7. References:

- <https://aws.amazon.com/blogs/security/how-to-automate-incident-response-in-aws-cloud-for-ec2-instances/>
- <https://www.kroll.com/en/insights/publications/cyber/state-of-incident-response>
- <https://www.siemplify.co/blog/how-enterprises-benefit-from-automated-incident-response/>
- <https://www.youtube.com/watch?v=xJtp4XI4to>
- <https://www.threatintelligence.com/blog/automated-incident-response>

8. Video Link:

- https://drive.google.com/drive/folders/1MPNtinWDyDtScx2_s9xszGsQZnSxJqIV?usp=sharing