International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

DDOS Prevention System using AWS Serverless-Architecture

Anamay Brahme[1], Aryan Deshmukh[2], Asad Vathare[3], Chanakya Patil[4], Prof. Noshir Z. Tarapore[5]

[5]Assistant professor and Guide, Vishwakarma University, Pune, B. TECH in Computer Engineering
[1,2,3,4]Student, Vishwakarma University, Pune, B. TECH in Computer Engineering

Abstract

The cloud is referred to as the servers or endpoints that can be accessed and controlled remotely. By using cloud computing, users and companies do not have to manage and take care of the physical servers themselves or even run the software applications on their own machines. The advantages of using cloud enables the users to access the same files and applications, because the computing and storage takes place on servers in large data centers, instead of storing locally on the user device itself, saving a lots of costs and manpower. Amazon Web Services is a leading cloud computing platform provided by Amazon. It's incredibly useful for individuals and businesses for several reasons. AWS offers scalability, which allows users to easily adjust resources to match their demands, which reduces costs and improves performance. Since AWS has a cost-effective pay-as-you-go model, you only pay for what you use. With data centers that are globally distributed, AWS ensures high reliability and availability of resources, ideal for building resilient applications. Tools like IAM (Identity Access Management) are used for resource access control and encryption for data protection, which increases security.

A Distributed Denial of Service Attack widely known as DDoS attack is a type of cyberattack that uses up the resources of the web server in order to disrupt the regular functioning of the website. DDoS is the malicious attempt to make resources unavailable for the intended users temporarily. In DDos attacks, multiple compromised computers known as “bots” are used to increase the attack intensity. These computers are part of a botnet which is a network of computers that is controlled by hackers.

To mitigate DDoS attacks, organizations employ multiple strategies. Firewalls and Access Control Lists (ACLs) are used to control the traffic reaching applications, while rate limiting techniques set restrictions on incoming traffic. These measures collectively enhance network security and protect against malicious traffic.

Keyword: DoS, DDoS, AWS, HTTP, HTTPS, Serverless-Architecture, Lambda

Introduction

The digital world relies heavily on web applications and online services, which makes them highly vulnerable to Distributed Denial of Service (DDOS) attacks, if not configured properly. DDoS attacks can disrupt these services, leading to economic losses and damages to reputation. Traditional DDoS

prevention methods are often costly and complex. To address this challenge, this research explores the potential of AWS Serverless Architecture as a cost-effective and scalable solution for DDoS attack prevention. By adopting this approach, organizations or companies can boost their ability to detect and mitigate DDoS attacks, in real-time.

This paper examines the practical implementation of AWS Serverless Architecture for DDoS prevention, offering insights and guidelines to bolster cybersecurity defenses. The goal is to make online services more secure and resilient in the face of persistent threats.

Problem Statement

Modern web applications and online services face a critical and persistent threat from Distributed Denial of Service attacks. This can disrupt the availability and functionality of these services. Conventional DDoS prevention methods often involve costly infrastructure investments and complex configurations. This presents a significant challenge for organizations, especially when it comes to the small to medium-sized businesses, to effectively mitigating DDoS attacks.

In this context, the problem we aim to address is how to enhance DDoS attack prevention and mitigation measures by leveraging the capabilities of AWS Serverless Architecture, with a focus on improving cost-effectiveness, scalability, and real-time responsiveness, while mitigating one of the most ferocious threats.

Motivation

Working on this project solely focused on Preventing DDOS attack. Using AWS serverless architecture is worthwhile undertaking for multiple reasons.

In this digital world, where everyone can basically gain knowledge about anything in just one click, any company/industry/business can be vulnerable to these types of attacks, for instance competing business or rivalry industry owning publicly accessible websites trying to bring down them. Learning/Working on this project can give us valuable insight about this real-world problem. The field of cybersecurity never-ending, developing such projects can help in various ways like resume building, learning opportunities etc. Choosing this project with such complexity can help personal growth like problem solving skills and working at the forefront of the technologies.

By making this project open source, people interested in this field will be able to find innovation.

Literature Review

Existing systems for DDoS prevention typically involve the use of specialized hardware appliances or dedicated software solutions. These systems often rely on traditional network-level defenses, for example, firewalls and intrusion detection systems, in order to detect and prevent incoming DDoS attacks. They employ various techniques, such as rate limiting, traffic analysis, and IP blocking, to identify and block malicious traffic.

In contrast, DDoS prevention using AWS serverless architecture takes advantage of the cloud computing paradigm and AWS services, specifically AWS Lambda. With serverless architecture, the focus shifts from hardware-based solutions to a more scalable and cost-effective approach. AWS Lambda allows the users to write and invoke blocks of codes, known as serverless lambda functions, that can be executed in

a dedicated server environment, without the need of managing the infrastructure. This eliminates the overhead of launching virtual machines and provides the flexibility to scale up rapidly when an attack occurs.

The key difference lies in the scalability and resource management. Existing systems often require dedicated hardware or software resources that may need to be provisioned in advance or scaled manually during an attack. In contrast, AWS serverless architecture automatically handles resource allocation, scaling, and load balancing, allowing for rapid scaling by several orders of magnitude when needed. This dynamic scalability ensures that the system can handle the increased traffic during an attack without incurring excessive costs during normal operation.

Additionally, serverless architecture offers integrations with other AWS services, such as logging, monitoring, and event-driven triggers, which further enhance the DDoS prevention capabilities. These integrations thus enables the serverless functions to work like building blocks just like in a microservice architecture, providing greater flexibility and extensibility in implementing anti-DDoS measures.

By leveraging AWS serverless architecture, organizations can benefit from a highly scalable and cost-efficient approach to DDoS prevention. The on-demand nature of serverless computing allows for efficient resource utilization and reduces the operational burden associated with maintaining and managing dedicated hardware or software solutions.

Methodology

The foundation of the architecture is designed using AWS resources and its services, although a similar framework can be implemented using other public cloud platforms that can offer equivalent functionalities similar to AWS Services used. The basic concept of this architecture involves the utilization of two Lambda functions situated between web server and the incoming traffic, along with the implementation of an alert system which is triggered by invocations from one of these functions. This alarm activates a third Lambda function which is responsible for sending an SOS message and thereafter updating the implemented filtering mechanism i.e., the denial list table.

All HTTP requests coming from the internet is channeled through the Application Programming Interface (API) endpoint which is created using the AWS API Gateway service. This API follows the principles of the Representational State Transfer (REST) architectural style, and it is protected using the Hypertext Transfer Protocol Secure (HTTPS). A Lambda function, named "Filter Lambda," is integrated into the API, it processes each request as it passes through this endpoint. This function scrutinizes the request against a Denial List table which is a Dynamo DB table that stores the suspicious IPs and then forwards the request to a second Lambda function named "Connection Lambda." The Connection Lambda is responsible for forwarding the request to the web server instance, which is protected within a private subnet in a Virtual Private Cloud (VPC). Simultaneously, it will log the essential request information such as IP address, Date and Time, User-Agent into a DynamoDB table referred to as the "Raw table." The response from the web server will follow the same path in reverse order, passing through the Connection Lambda, Filter Lambda, API endpoint, finally reaching the end user.

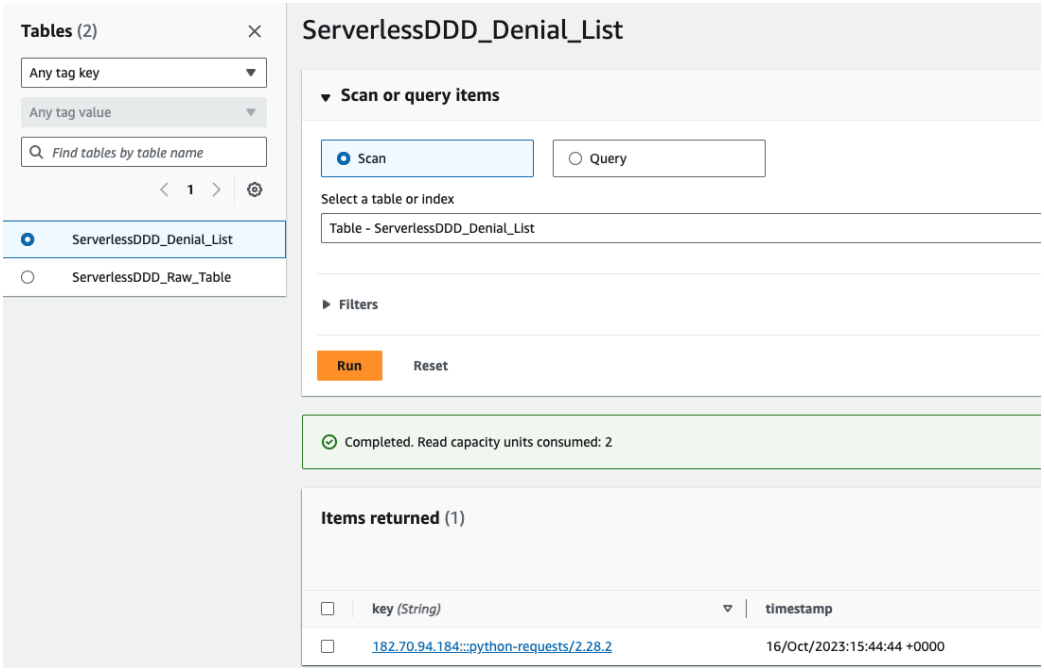
In case that the frequency of incoming traffic captured by Connection Lambda surpasses a predefined threshold, AWS Service known as CloudWatch alarm is set to trigger. This alarm will, in turn, activate a third Lambda function known as "Alarm Lambda". The primary role of this function is to analyze requests in the Raw table in order to identify the entities which are responsible for the high-frequency incoming requests and subsequently add the suspicious requesters to the Denial List table. Now, the alarm

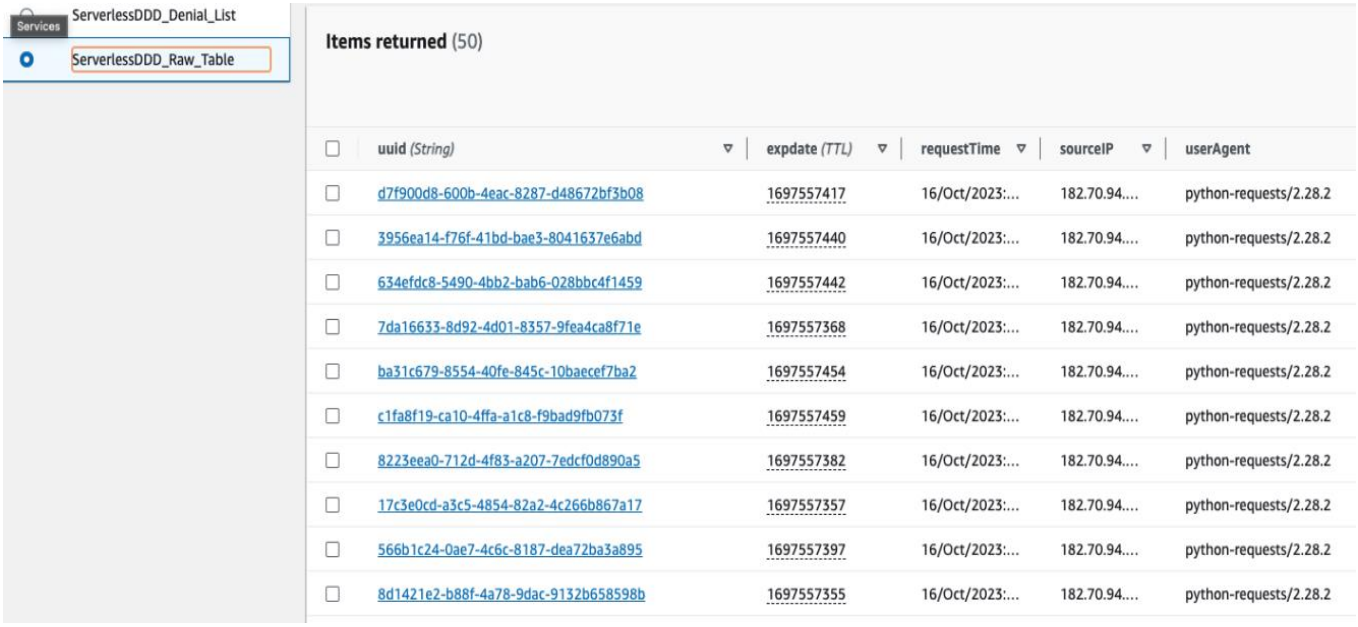
mechanism will send an alert mail to the specified administrator's email using AWS Simple Notification Service (SNS).

As a result, the requests that are associated with the high-frequency requesters are prevented from reaching the web server instance. To differentiate the high-frequency requesters from the regular ones, a combination of source IP address and user agent information is utilized. This comprehensive architectural framework in turn, provides a robust and flexible solution for securing and managing incoming web requests.

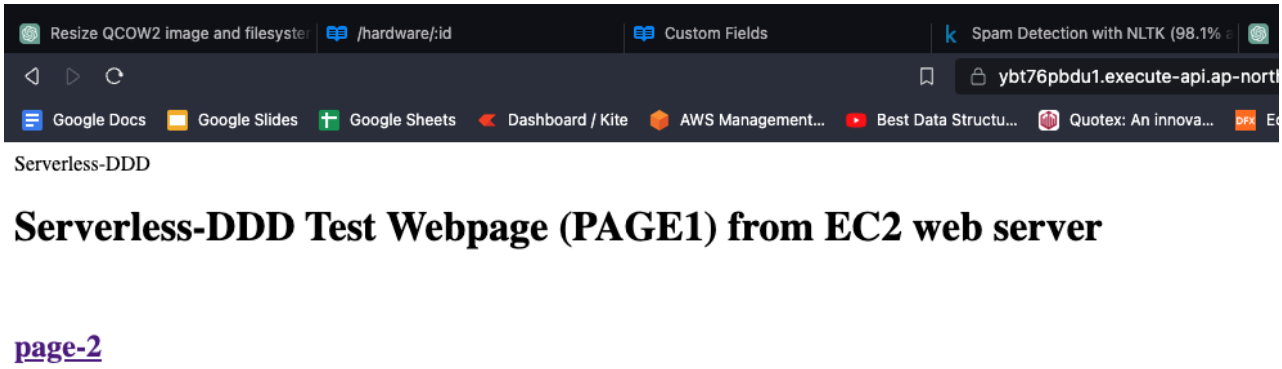
In the event of DDoS attack that manages to partially evade the blocking mechanisms and is persistently bombarding the endpoint with requests from multiple different source IP addresses and user agents that the first alarm cycle couldn't include in the Denial List, these incoming requests are detected in multiple alarm cycles. Usually, it requires a few minutes for the AWS CloudWatch Alarm to trigger after a surge in invocations at a high frequency begins. This alarm condition often remains active for a few minutes, providing time for visual inspection and assessment of the situation. However, this deliberation for visual inspection results in a slight delay in the activation of the next alarm cycle. This approach ensures a proactive response to emerging threats while allowing for human intervention in the security monitoring procedure.

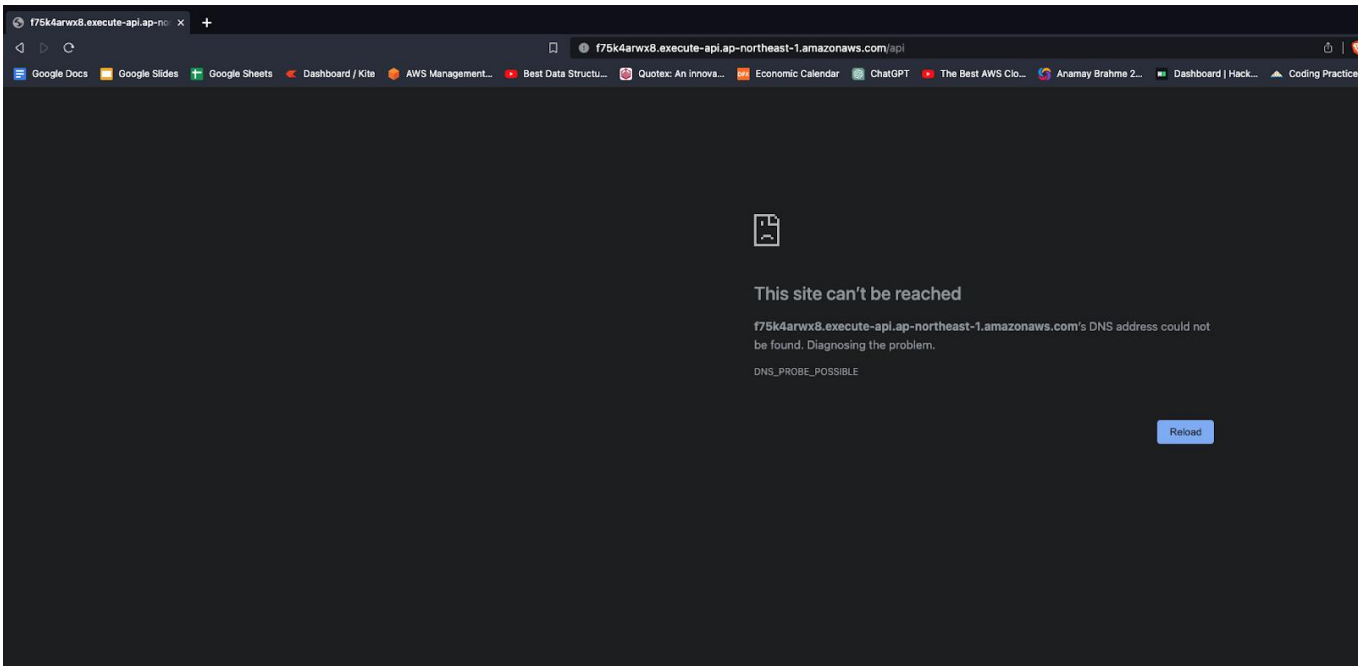
DynamoDB Tables:


Tables (2)
Any tag key
Any tag value
Find tables by table name
ServerlessDDD_Denial_List
ServerlessDDD_Raw_Table
ServerlessDDD_Denial_List
Scan or query items
Scan
Query
Select a table or index
Table - ServerlessDDD_Denial_List
Filters
Run
Reset
Completed. Read capacity units consumed: 2
Items returned (1)
key (String)
timestamp
182.70.94.184:::python-requests/2.28.2
16/Oct/2023:15:44:44 +0000


Services
ServerlessDDD_Denial_List
ServerlessDDD_Raw_Table
Items returned (50)
uuid (String)
expdate (TTL)
requestTime
sourceIP
userAgent
d7f900d8-600b-4eac-8287-d48672bf3b08 1697557417 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
3956ea14-f76f-41bd-bae3-8041637e6abd 1697557440 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
634efdc8-5490-4bb2-bab6-028bbc4f1459 1697557442 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
7da16633-8d92-4d01-8357-9fea4ca8f71e 1697557368 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
ba31c679-8554-40fe-845c-10baecef7ba2 1697557454 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
c1fa8f19-ca10-4ffa-a1c8-f9bad9fb073f 1697557459 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
8223eea0-712d-4f83-a207-7edcf0d890a5 1697557382 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
17c3e0cd-a3c5-4854-82a2-4c266b867a17 1697557357 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
566b1c24-0ae7-4c6c-8187-dea72ba3a895 1697557397 16/Oct/2023:... 182.70.94.... python-requests/2.28.2
8d1421e2-b88f-4a78-9dac-9132b658598b 1697557355 16/Oct/2023:... 182.70.94.... python-requests/2.28.2

Webpage:


Resize QCOW2 image and filesyste
/hardware/:id
Custom Fields
Spam Detection with NLTK (98.1%
ybt76pbdu1.execute-api.ap-nort
Google Docs
Google Slides
Google Sheets
Dashboard / Kite
AWS Management...
Best Data Structu...
Quotex: An innova...
Serverless-DDD
Serverless-DDD Test Webpage (PAGE1) from EC2 web server
page-2



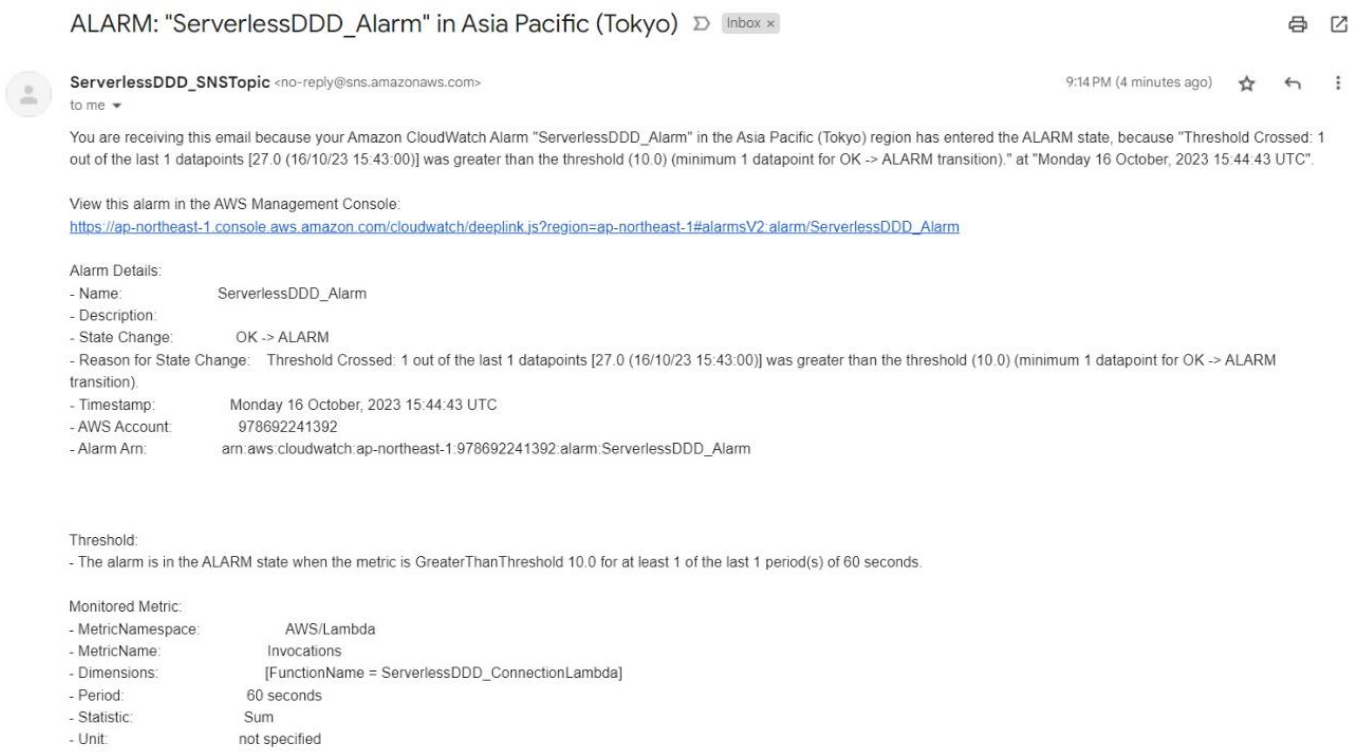
CloudWatch:

Python Attack Script:

```
PS C:\Users\Omen> python -u "c:\Users\Omen\OneDrive\Desktop\LY Proj\fake_attack.py\fake_attack.py"
Request 1
status_code=502
content={"message": "Internal server error"}
Request 2
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 3
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 4
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 5
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 6
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 7
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 8
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 9
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>
```

```
Request 59
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 60
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 61
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 62
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 63
status_code=200
content=<html><head>Serverless-DDD</head><body><h1>Serverless-DDD Test Webpage (PAGE1) from EC2 web server</h1><br><a href='page2.html'><h2>page-2</h2></a></body></html>

Request 64
status_code=403
content=Error: Forbidden to access requested page
Request 65
status_code=403
content=Error: Forbidden to access requested page
Request 66
status_code=403
content=Error: Forbidden to access requested page
Request 67
status_code=403
content=Error: Forbidden to access requested page
Request 68
status_code=403
content=Error: Forbidden to access requested page
Request 69
status_code=403
content=Error: Forbidden to access requested page
```

Simple Notification Service [SNS]:



ALARM: "ServerlessDDD_Alarm" in Asia Pacific (Tokyo) Inbox ×

ServerlessDDD_SNSTopic <no-reply@sns.amazonaws.com> 9:14 PM (4 minutes ago)
to me

You are receiving this email because your Amazon CloudWatch Alarm "ServerlessDDD_Alarm" in the Asia Pacific (Tokyo) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [27.0 (16/10/23 15:43:00)] was greater than the threshold (10.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Monday 16 October, 2023 15:44:43 UTC".

View this alarm in the AWS Management Console:
https://ap-northeast-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=ap-northeast-1#alarmsV2:alarm/ServerlessDDD_Alarm

Alarm Details:
- Name: ServerlessDDD_Alarm
- Description:
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [27.0 (16/10/23 15:43:00)] was greater than the threshold (10.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Monday 16 October, 2023 15:44:43 UTC
- AWS Account: 978692241392
- Alarm Arn: arn:aws:cloudwatch:ap-northeast-1:978692241392:alarm:ServerlessDDD_Alarm

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 10.0 for at least 1 of the last 1 period(s) of 60 seconds.

Monitored Metric:
- MetricNamespace: AWS/Lambda
- MetricName: Invocations
- Dimensions: [FunctionName = ServerlessDDD_ConnectionLambda]
- Period: 60 seconds
- Statistic: Sum
- Unit: not specified

Impact

1. Enhanced Service Availability: By effectively preventing DDoS attacks, the organization can maintain high service availability, ensuring that its online services remain accessible to users without disruption.
2. Reduced Financial Losses: DDoS attacks can be costly due to potential service downtime, loss of revenue, and increased infrastructure expenses. A successful prevention system helps mitigate these financial losses.
3. Improved User Experience: Users experience uninterrupted access to the organization's services, leading to higher satisfaction and retention rates.

4. Protection of Reputation: Effective DDoS prevention safeguards the organization's reputation, demonstrating a commitment to security and reliability, which can enhance trust among customers and partners.
5. Lower Infrastructure Costs: Leveraging AWS serverless architecture allows for efficient resource utilization, reducing infrastructure costs compared to traditional dedicated mitigation solutions.
6. Minimized Downtime Costs: Even brief service interruptions due to DDoS attacks can result in substantial financial losses. Prevention reduces or eliminates these downtime costs.
7. Faster Incident Resolution: With automated detection and mitigation, the time required to resolve DDoS incidents is significantly reduced, minimizing service disruption.
8. Scalability: The architecture's ability to scale dynamically ensures the organization can handle traffic spikes and increased user demand without performance degradation.

Technologies Used

1. Amazon Web Services (Serverless):

a. AWS Lambda Service: AWS Lambda is used to run custom code in response to DDoS detection events. This code can analyze traffic patterns, detect anomalies, and trigger protective measures.

b. AWS SNS (Simple Notification Service): AWS SNS is used for real-time alerting. When a DDoS attack is detected, it can send notifications to administrators or other systems to take action.

c. AWS DynamoDB: DynamoDB can be used to store configuration data, logs, and DDoS attack detection records. It is a highly scalable NoSQL database.

d. API Gateway: AWS API Gateway can be used to create an API to manage and monitor the system. It can also help in throttling and controlling incoming traffic.

e. VPC (Virtual Private Cloud): VPC is used to isolate the DDoS prevention system from the public internet. It provides a secure network environment to deploy resources like Lambda functions and DynamoDB.

2. Amazon Web Services:

a. AWS EC2: EC2 instances can be used to host the DDoS protection front-end, such as a web application that provides an interface for administrators to configure protection rules and view alerts.

b. AWS IAM (Identity and Access Management): IAM is used for managing access to AWS resources securely. You can define roles and permissions to ensure that only authorized entities have access to your system components.

c. AWS CloudWatch: CloudWatch can be used for monitoring and logging. You can set up CloudWatch alarms to trigger actions when specific events or metrics occur.

3. Python 3.6:

Python is a programming language used to develop custom code for AWS Lambda functions. It can be used for implementing algorithms and logic for detecting DDoS attacks.

4. Linux Virtual Machine:

Used to host the webpage to be tested the DDOS Attack on API URL. In this architecture, when a DDoS attack is detected, Lambda functions can automatically update security rules in your VPC, and CloudWatch can trigger notifications. The EC2 instance can host a web-based dashboard to manage DDoS

protection settings and view alerts. DynamoDB can store data related to DDoS attack detection for further analysis or auditing.

Conclusion

In conclusion, the integration of AWS Serverless Architecture into DDoS prevention has yielded a cost-effective, scalable, and responsive solution. The project's use of AWS services, Python 3.6, and a Linux Virtual Machine has demonstrated the viability of this approach, offering small to medium-sized businesses an affordable means to defend against DDoS attacks. This innovative combination has the potential to reshape the cybersecurity landscape, making online services more resilient and secure in the face of persistent threats. Further research and practical implementations hold the key to refining and expanding this groundbreaking methodology.

References

1. Soohyun Lee (May 10 2021), Conference details, date, year, volume. A serverless architecture for frequency-based HTTPrequest filtering against distributed denial-of-service(DDoS) attacks
2. Ajay Singh Chauhan (2018) “DoS and DDoS attacks.” Practical network scanning. Packt publishing.
3. Cade Metz (2009) “DDoS attack rains down on Amazon cloud” The Register. Retrieved from https://www.theregister.com/2009/10/05/amazon_bitbucket_outage/
4. Nick Galov (2021, Jan 16) “39 Jaw-Dropping DDoS”
5. Specht and Lee (2004) “Distributed Denial of Service: taxonomies of attacks, tools and countermeas-ures” ISCA PDCS. 543–550
6. Ahmed Bakr ,Abd El-Aziz Ahmed ,Hesham A. Hefny (2019) ,”A Survey on Mitigation Techniques against DDoS Attacks on Cloud Computing Architecture “ Vol. 28, No. 12, (2019), pp. 187-200