Project Methodology Report

on

"DDOS Prevention System using AWS Serverless-Architecture"

Submitted to
Vishwakarma University, Pune
In Partial Fulfillment of the Requirements
For The Award of Degree

Bachelor of Technology (Computer Engineering)

By

Anamay Brahme, Aryan Deshmukh,
Asad Vathare,Chanakya Patil

UNDER THE GUIDANCE OF

Prof. Noshir Z. Tarapore

Table of Contents

Introduction

Denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks are cyberattacks that aim to disrupt the availability of a service by overwhelming the target server or network. DoS attacks originate from a single source, while DDoS attacks involve multiple sources, often using a botnet of compromised hosts controlled by the attacker.

These attacks can take various forms, including UDP flooding and HTTP-based application layer attacks. Implementing effective anti-DDoS measures poses challenges,particularly in handling large volumes of data without incurring unnecessary costs during normal operation. Serverless architecture, which emerged with the introduction of AWS Lambda in 2014, provides a promising solution. It enables rapid scalability by several orders of magnitude while minimizing operational expenses during non-attack periods.

Serverless computing allows users to define and invoke small pieces of custom code known as serverless functions or serverless functions-as-a-service (FaaS). These functions are executed in a dedicated server environment without the need for launching virtual machines. Resource allocation, scaling, and load balancing are managed by the cloud provider, and users are billed based on the function's execution duration. While individual serverless functions may not handle heavy workloads, their horizontal scalability enables them to collectively perform intensive distributed tasks.

Cloud providers also offer integrations around serverless functions, such as logging, monitoring, and invocation by other services or functions. These integrations make serverless functions suitable as building blocks within a microservice architecture, a popular approach for developing cloud applications.

It is important to note that while the above information provides an overview of serverless architecture for DDoS prevention, it is essential to avoid plagiarism by properly citing and paraphrasing the original sources from where the information is derived.

Project Objectives

Aim:

The aim of this project is to create a serverless architecture that can protect web servers from frequency-based distributed denial-of-service (DDoS) attacks.
Scope and Objectives:

1. Develop a serverless architecture using Amazon Web Services (AWS) resources, including Lambda serverless functions, API Gateway, CloudWatch, DynamoDB, and Simple Notification Service (SNS).
2. Monitor the frequency of incoming HTTP requests to identify potential DDoS attacks.
3. Trigger an alarm when the invocation frequency surpasses the threshold, indicating a potential DDoS attack.

The ultimate goal is to provide web server administrators with an efficient and reliable defense mechanism against frequency-based DDoS attacks, reducing the risk of service disruptions, minimizing downtime, and ensuring a smooth user experience.

Methodology

Summary of the Implementation should be differentiated between a normal connection and an Attackers Connection to evaluate the system for the behavioral changes for possible outcomes to test the architecture.

Normal Connection:

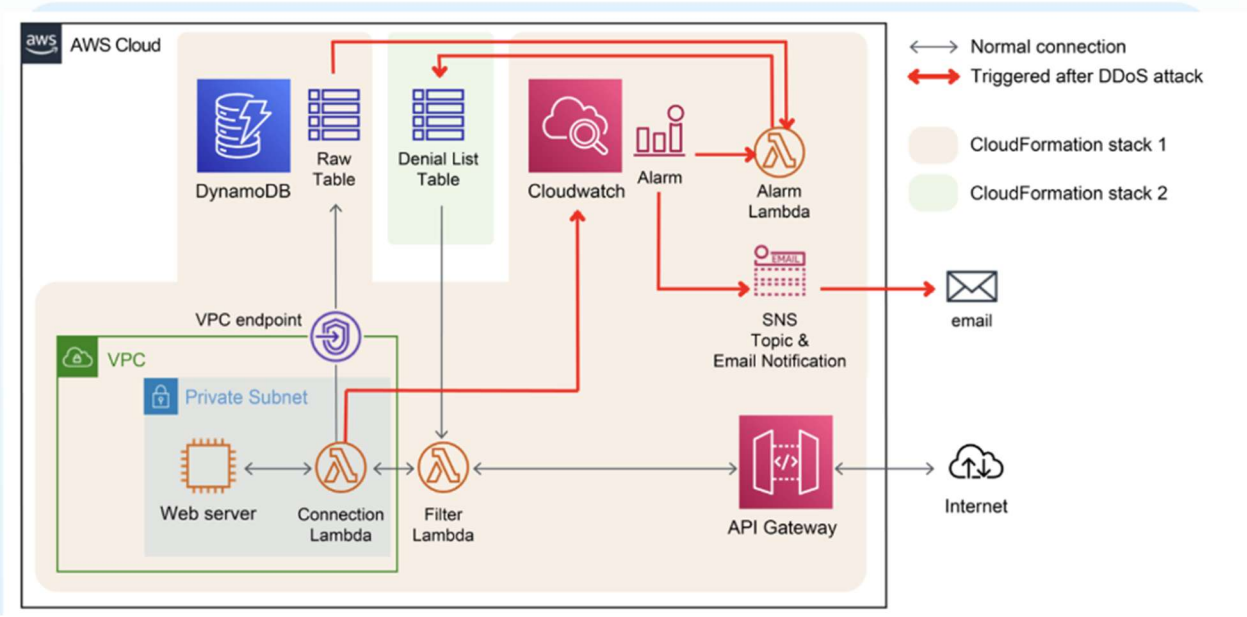
1. Normal User Request: A legitimate user initiates an HTTP request from their device to access a web service.
2. API Gateway: The request is directed to an API endpoint created by the AWS API Gateway service. The API follows the REST style and supports HTTPS for secure communication.
3. Filter Lambda: For every request passing through the endpoint, the Filter Lambda function is invoked. It checks the request against a Denial List DynamoDB table to determine if the request should proceed.
4. Connection Lambda: After passing the filter check, the request is passed to the Connection Lambda function. Connection Lambda sends the request to the Web server instance located in a private subnet within a Virtual Private Cloud (VPC). Additionally, the request information is recorded in the Raw table of DynamoDB.
5. Response Flow: The response from the Web server travels back in the reverse direction. It passes through the Connection Lambda, Filter Lambda, API endpoint, and is finally returned to the user, ensuring a secure response flow.

Attacker’s Connection:

1. Attacker Request: An attacker sends an HTTP request, similar to a normal user, to the API endpoint.
2. Filter Lambda Check: The request is intercepted by the Filter Lambda function, which checks the request against the Denial List DynamoDB table.
3. Denial List Check: If the attacker's IP address and user agent are not listed in the Denial List, the request proceeds to the Connection Lambda.
4. Connection Lambda and Web Server: The Connection Lambda forwards the request to the Web server instance within the private subnet of the VPC, just like in the normal connection scenario.

5. Alarm Trigger: If the invocation frequency of the Connection Lambda exceeds a threshold, an AWS CloudWatch alarm is triggered.
6. Alarm Lambda Response: The triggered alarm invokes the Alarm Lambda function, which parses the Raw table to identify high-frequency requesters. The identified requesters are added to the Denial List table.
7. Denial of Attackers: As a result of the alarm response, requests associated with high-frequency attackers are blocked from reaching the Web server instance. However, normal user requests remain unaffected.
8. Continuous Monitoring: If the attacker continues the DDoS attack by bombarding the endpoint with different source IP addresses and/or user agents not yet listed in the Denial List, those requests can be captured in the next alarm cycle. The CloudWatch alarm typically takes a few minutes to trigger after high-frequency invocations start.
9. Iterative Blocking: The process of identifying and blocking high-frequency attackers through the Alarm Lambda and Denial List table may require several rounds, depending on the number of IP addresses and user agents used, their frequency of change, and how they are distributed over time.

Architecture Overview



Software/AWS Services Used

Tools and Software Used -

1. Python 3 -
    a. BOTO3 -
    Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python, providing an easy-to-use interface to interact with AWS services programmatically.
    b. JSON -
    JSON (JavaScript Object Notation) is a lightweight data interchange format commonly used in Python and other programming languages for storing and transmitting structured data.
    c. OS -
    OS stands for "operating system" and it is a module in Python that provides functions for interacting with the operating system, such as file operations, directory management, and process handling.
2. Amazon Web Services -
    a. AWS Lambda -
    AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows you to run code without provisioning or managing servers.
    b. AWS DynamoDB -
    AWS DynamoDB is a fully managed NoSQL database service that provides fast and flexible storage for applications requiring low-latency access to large amounts of data.
    c. CloudWatch -
    AWS CloudWatch is a monitoring and observability service provided by AWS that collects and tracks operational data and logs from various AWS resources and applications, allowing users to gain insights and take actions based on the collected data.
    d. API Gateway -
    API Gateway is a service that acts as a single entry point for multiple APIs, enabling organizations to manage, secure, and monitor their API endpoints in a centralized manner.
    e. Cloud Formation -

AWS CloudFormation is a service that allows you to create and manage AWS resources using declarative templates, enabling infrastructure as code and automated provisioning of cloud environments.

Monitoring and Logging

Through the continuous monitoring and iterative blocking process, the system can effectively mitigate DDoS attacks by blocking requests from known high-frequency attackers. The architecture distinguishes between normal user connections and attacker connections based on the combination of source IP address and user agent, enabling accurate identification and targeted blocking.

Overall, this project demonstrates a comprehensive approach to protecting web services from DDoS attacks by leveraging AWS services and implementing an intelligent workflow. By promptly detecting and mitigating attacks, the system ensures the availability and reliability of the web service for legitimate users.

Cost Optimization

Existing DDoS solutions should definitely be considered for a fast and reliable strategy. For example, an AWS Web Application Firewall (WAF) could be attached to the API Gateway endpoint to block known attack sources and apply other filtering rules. AWS also provides a powerful DDoS prevention service called AWS Shield Advanced that protects against HTTP as well as UDP reflection flood, SYN flood andDNS query flood attacks (Priyam, 2018).

AWS Shield Advanced is expensive (e.g. $3,000 monthly fee plus data transfer), but its response time is less than seconds. Note out that a full DDoS load test was not performed on the Lambda-based architecture described here.The purpose of this project was to come up with a design that works on a small scale as a proof of concept, and not intended to replace existing or recommended DDoS response services.

Future Scope

Future Scope 1: Approval List

An additional enhancement for the project could be the implementation of an approval list, which would override the denial list. This feature would be beneficial as it allows known users to bypass the denial list, preventing them from triggering alarms unnecessarily. In case a legitimate user is mistakenly added to the denial list, an administrator would have the capability to manually add them to the approval list, ensuring uninterrupted access for trusted users.

Future Scope 2: Expanded Request Information

To enhance the system's capabilities, the architecture can be extended to support additional request information beyond just the source IP address and user agents. For instance, specific paths or parameters of the requests could be included in the combination used for identification and filtering. This expansion would provide a more comprehensive and nuanced approach to differentiate between normal user connections and potential attackers, enhancing the system's accuracy and effectiveness in detecting and mitigating DDoS attacks.

Conclusion

In conclusion, the described project implements a robust system architecture for protecting against Distributed Denial of Service (DDoS) attacks. By utilizing AWS services such as API Gateway, Lambda functions, DynamoDB, and CloudWatch, the system effectively filters and monitors incoming HTTP requests, allowing normal user requests to reach the web server while identifying and mitigating high-frequency attackers.

To sum it up, our project has built a strong defense system against DDoS attacks using AWS services like API Gateway, Lambda functions, DynamoDB, and CloudWatch. This system can tell apart real users from harmful traffic, ensuring that our website stays available and responsive even during attacks. We've used smart strategies like rate limiting and anomaly detection to keep the bad traffic away while letting genuine users through.

With all these parts working together – from stopping bad traffic to adapting to changes – our project shows how AWS Serverless Architecture can help websites stay strong against cyber threats.

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 countermeasures” 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