SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V11I10P107

Volume 11 Issue 10, 54-61, October 2024

Original Article

The Impact of Cloud Architecture and Design on the Effectiveness of Rate-Based Distributed Denial of Service Attacks

Manish Sinha

Facebook, California, USA.

[1]*Corresponding author : manishsinha27@gmail.com*

Received: 25 August 2024 Revised: 30 September 2024 Accepted: 19 October 2024 Published: 31 October 2024

Abstract - *A highly available service is now a cornerstone requirement of any cloud service. Amongst the threats faced by any cloud service, Distributed Denial of Service (DDoS) Attacks are a significant concern when designing services with high availability. Good design decisions help detect and mitigate attacks quickly, whereas poor decisions can introduce tech debt and complicate the detection and mitigation of attacks. In this paper, we will focus primarily on rate-based DDoS attacks, a kind of attack in which the malicious actor tries to exhaust the service's resources by sending fraudulent requests from zombie computers worldwide, making it hard to detect and pinpoint the source of the attack. We will explore different architecture and design decisions that can be used to mitigate DDoS attacks with minimum degradation of latency.*

Keywords - *Cloud architecture, Cloud computing, Cybersecurity, Performance analysis, Distributed denial of attacks.*

1. Introduction

In the recent decade, cloud services have become an integral part of our digital and web infrastructure, and they have now relieved businesses from having to implement all the low-level functionality and capability that power their web service. With higher-order features available on cloud providers, companies now have a reliable, scalable, cost-effective solution where they can outsource much work that is not part of their core competency [1]. With the growing reliance on cloud services, the security risk grows significantly and distributed denial-of-service attacks can compromise the availability of these services. Primarily, DDoS attacks work by flooding the target service with a vast amount of traffic, which is higher than what the service was initially designed for [2].

The distributed nature of such an attack makes it difficult to distinguish it from legitimate customer traffic. Such attacks can cause degradation of the quality of service, which customers can experience as slow loading times, or it can lead to complete unavailability of the service, which is usually shown as “This site cannot be reached,” which is displayed by the browser when the server returns 503. If the load balancing cannot accept further requests, we get the “The request has timed out” error on the browser. DDoS attacks are highly effective and can achieve that level because they use a fleet of compromised devices, known as a botnet [3]. A botnet is a network of infected devices like laptops, desktops, and Internet-of-Things devices like smart refrigerators and intelligent weight scales. A malicious actor controls these without the knowledge of the owner of the infected device. Mirai Botnet [4] is one of the most infamous botnets. It arose in 2016 and comprises thousands of infected IoT devices like cameras and routers. One of the most notorious impacts of Mirai has been on Dyn, a DNS provider. Mirai has evolved since becoming the basis of many other botnet systems. The two primary ways service availability can be affected are bad design decisions that cause degradation under heavy loads from legitimate users and Distributed Denial of Service attacks that seek to overwhelm the compute, network, and data capacity associated with the service.

Unlike regular Denial of Service attacks, which are much more centralized, Distributed Denial of Service (DDoS) attacks are distributed in design, making it difficult to find a mitigation strategy [5]. Not all DDoS attacks are created equal and can come in different flavors. A DDoS attack might send low traffic per IP but from enough machines that the total traffic exceeds the service's capacity. This is primarily an issue with websites that must be designed or configured to handle large amounts of traffic. Mitigating the effects of DDoS is a complex problem that requires a combination of well-designed cloud architecture, effective defense mechanisms built into the system [6], and proper incident response planning. Poor design behaves like a weak link in the chain, becoming a single critical point of failure. A good design will mitigate the

issue ahead of time rather than let the effects propagate throughout the architecture. This paper primarily focuses on the effects of cloud architecture designs in the event of a DDoS, including design considerations that can help us maintain the high availability and security of the service.

2. Related Work

Research has been conducted on the impact of DDoS on cloud services. Such research has proposed different mitigation techniques with varying levels of effectiveness. Somani et al. (2017) discuss four attack prevention approaches: Challenge Response, Hidden Servers/ports, Restrictive Access, and Resource Limit [7]. This research describes how DDoS attacks happen, their types and flavors, and their eventual impact on service availability.

Idhammad et al. (2018) explore a semi-supervised machine learning approach for detecting DDoS attacks. It distinguishes between Direct DDoS attacks and reflection-based DDoS attacks. The approach uses a sliding time window algorithm [8] to estimate the entropy of network header features of the network traffic.

The proposed approach achieves a high accuracy rate of over 98%. Bhardwaj et al. (2021) focus on the architectural layer of DDoS mitigation efforts in cloud services. It introduces a multi-layer defense that uses techniques like traffic filtering, load balancing, and auto-scaling [9]. This is with the singular focus on achieving high service availability even when the service is experiencing DDoS attacks.

The architecture is evaluated using synthetic traffic and shows its effectiveness against DDoS attacks compared to traditional approaches. None of the above research covers the actual architecture design, and none of them focuses on rate-limiting capability as a cornerstone ability to mitigate DDoS attacks.

3. Methodology

This section will explore multiple methods and approaches to mitigate rate-based DDoS attacks. We will use AWS WAF for reference, and all our materials, including sample rate-limiting rules, will be based on AWS WAF syntax.

These rules have pros and cons, and the effectiveness of each approach is highly reliant on the architecture of the cloud service. We will review multiple approaches and explain when each would be a practical rule.

3.1. Rate Limiting per IP address

3.1.1. Overview

This rule allows specific traffic from an IP address in a specified time window. In the case of AWS WAF, if not specified, the default time window is 5 minutes, but it can be overridden to be 1 minute, 2 minutes, or 10 minutes [10].

3.1.2. Pros and Cons

Pros	**Cons**
Very effective in blocking individual attackers or single compromised devices	If multiple users use the same IP, they can be negatively affected. For example, users behind a NAT
Simple and easy to write and maintain in the long run. Requires less time to understand	If the attacker wants to DDoS a service, they can get around IP-based rate limits by using multiple IPs

3.1.3. Example rule

```
{
  "Name": "RateLimitPerIP",
  "Priority": 1,
  "Action": {
    "Block": {}
  },
  "RateBasedStatement": {
    "Limit": 1000,
    "AggregateKeyType": "IP"
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
    "CloudWatchMetricsEnabled": true,
    "MetricName": "RateLimitPerIP"
  }
}
```

3.1.4. Architecture Considerations

This approach is more useful if the service's traffic is served via a Content Delivery Network (CDN). It should be implemented at the edge. If no CDN is used, this rule should be used at the Load Balancer or API Gateway level. Implementing IP-based rate limits at the edge rather than the origin server is more computationally efficient. If the CDN uses Anycast to route traffic to the closest PoP [11], then each PoP serves traffic for a mostly exclusive set of IP addresses. Rate limiting can be much more effective as the key space for the rate-limiting dictionary would be smaller, using less memory of the PoP infrastructure. If this rate limiting is implemented at the Load Balancer inside one or more specific regions of the cloud infrastructure, it would have to rate limit a much larger subset of IP addresses. Considering this, rate limiting per IP is most useful when deployed on edge servers.

3.2. Rate Limiting by Country or Region

3.2.1. Overview

This rule allows specific traffic from a country or a geographical region in a specified time window [12]. This can be useful for services whose primary business is in a particular country and receives limited traffic from outside that region. For example, a state's unemployment portal can have a higher rate limit for traffic originating in that state and a much tighter limit for traffic from the country but out of that state. Lastly,

traffic outside that country can be significantly restricted but not blocked.

3.2.2. Pros and Cons

Pros	Cons
Possible to rate limit or block traffic from unwanted regions or sanctioned countries.	Possible for legitimate users to be blocked or rate-limited if the threshold is not set adequately.
Allows for much finer-grained control over traffic from specific regions for effective allocation of resources	Attackers can use VPNs or proxies to bypass such geographical restrictions.

3.2.3. Example rule

```
{
  "Name": "RateLimitByUSCA",
  "Priority": 2,
  "Action": {
    "Block": {}
  },
  "RateBasedStatement": {
    "Limit": 5000,
    "AggregateKeyType": "IP",
    "ScopeDownStatement": {
      "GeoMatchStatement": {
        "CountryCodes": ["US", "CA"]
      }
    }
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
    "CloudWatchMetricsEnabled": true,
    "MetricName": "RateLimitByCountry"
  }
}
```

3.2.4. Architecture Considerations

This kind of rate-limiting rule can be implemented either at the edge or at the load balancer level. The number of "keys" would be limited to the number of countries or geographic areas. It is still considerably fewer than IP addresses. The total [13] IPv4 addresses is 10^{32}, and the total IPv6 addresses is 10^{128}. Rate-limiting by country or region is much more computationally efficient than by IP addresses. We can see that this rate-limit rule is not dependent on the architecture of the service since this rule can be enforced at any level of the architecture stack.

3.3. Rate Limiting by Known Bad IP addresses

3.3.1. Overview

The approach ties in with rate-limiting IP addresses, with a twist that the limits are significantly lower for a list of known bad IP addresses [14]. Ideally, we should block these IP addresses from accessing the service, but that is not feasible. IP addresses are randomly assigned to end users, indiscriminately blocking access can affect legitimate users. Since these IP addresses have a known history of abuse, we should allow them access to the service at a much lower rate.

3.3.2. Pros and Cons

Pros	Cons
Manage traffic from known bad IPs without completely blocking them	It requires us to maintain a list of known bad IP addresses that can be fraught with mistakes.
Allows for flexibility of IP addresses to be added and removed from the list at regular intervals without updating the rule	It is not always effective since bad actors can learn the known bad IP addresses when they get throttled and switch to a new IP address.

3.3.3. Example rule

```
{
  "Name": "RateLimitBadIPs",
  "Priority": 3,
  "Action": {
    "Block": {}
  },
  "RateBasedStatement": {
    "Limit": 100,
    "AggregateKeyType": "IP",
    "ScopeDownStatement": {
      "IPSetReferenceStatement": {
        "ARN": "arn:aws:wafv2:us-east-
1:123456789012:regional/ipset/BadIpSet/
1234a1b2-5678-90ab-1234-c56789defgh"
      }
    }
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
    "CloudWatchMetricsEnabled": true,
    "MetricName": "RateLimitBadIPs"
  }
}
```

3.3.4. Architecture Considerations

When it comes to known bad IP addresses, it should be implemented at multiple levels of the cloud architecture. Depending on the CDN, it might require its origin server to be publicly accessible [15]. This opens up a path for the malicious actors to bypass CDN and attack the origin. Applying rate limits to all the layers ensures that no matter the future changes to the architecture, known bad IP cannot misuse the server resources, provided the list of IP addresses is regularly updated.

3.4. Rate Limiting by User Agents

3.4.1. Overview

This rule allows the service owner to limit the traffic from specific types of user agents [16], like curl or Python-

requests/x.y.z, which is used by bots or web scrapers. Additionally, we can use it to allow certain User-Agent patterns with a much higher traffic rate.

3.4.2. Pros and Cons

Pros	Cons
Low effort and high reward to block specific tools or scripts written by low-effort	User agents are sent by the clients themselves and can easily be spoofed.
Useful for rate-limiting blocking traffic from outdated or suspicious user agents	Legitimate users using non-popular browsers can be significantly affected.

3.4.3. Example rule

```
{
  "Name": "RateLimitUserAgents",
  "Priority": 4,
  "Action": {
    "Block": {}
  },
  "RateBasedStatement": {
    "Limit": 500,
    "AggregateKeyType": "IP",
    "ScopeDownStatement": {
      "ByteMatchStatement": {
        "SearchString": "curl",
        "FieldToMatch": {
          "SingleHeader": {
            "Name": "user-agent"
          }
        },
        "TextTransformations": [
          {
            "Priority": 0,
            "Type": "LOWERCASE"
          }
        ],
        "PositionalConstraint":
"CONTAINS"
      }
    }
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
    "CloudWatchMetricsEnabled": true,
    "MetricName":
"RateLimitUserAgents"
  }
}
```

3.4.4. Architecture Considerations

This kind of rate limiting should not be implemented at the edge. This is especially true if most of the service's traffic is from valid browsers with valid user agents. Implementing this rule on the edge at the PoP would result in a performance penalty for most users just for restricting a smaller subset of bots. The best place to implement this rate-limiting rule would be the Load Balancer running on the service infrastructure.

3.5. Different Rate Limits for URL Paths

3.5.1. Overview

This approach considers different resource availability and operational complexity of different operations and features of a service. Some operations [17], like login and data aggregation, are expensive and can become a bottleneck, bringing down the service if the attacker figures out the expensive operations. On the other hand, we can list down the cached paths [18] and allow them much higher rate limits.

3.5.2. Pros and Cons

Pros	Cons
Optimize resource consumption based on the requirements of each path	It requires careful categorisation analysis and understanding the resources available for each path.
Allows for fine-tuned control – higher rates for static cached content and lower for expensive operations	It's not really a solution since expensive paths/routes can still be DDoS-ed.

3.5.3. Example rule

```
{
  "Name": "RateLimitByPath",
  "Priority": 5,
  "Action": {
    "Block": {}
  },
  "RateBasedStatement": {
    "Limit": 1000,
    "AggregateKeyType": "IP",
    "ScopeDownStatement": {
      "ByteMatchStatement": {
        "SearchString": "/api/",
        "FieldToMatch": {
          "UriPath": {}
        },
        "TextTransformations": [
          {
            "Priority": 0,
            "Type": "URL_DECODE"
          }
        ],
        "PositionalConstraint":
"STARTS_WITH"
      }
    }
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
    "CloudWatchMetricsEnabled": true,
    "MetricName": "RateLimitByPath"
  }
}
```

3.5.4. Architecture Considerations

An excellent place to implement such a rate-limiting rule would be on the API itself. AWS allows us to associate API Gateway with AWS WAF and enforce rate-limiting on paths and routes of a RESTful service.

4. Discussion and Considerations

4.1. Design Diagram

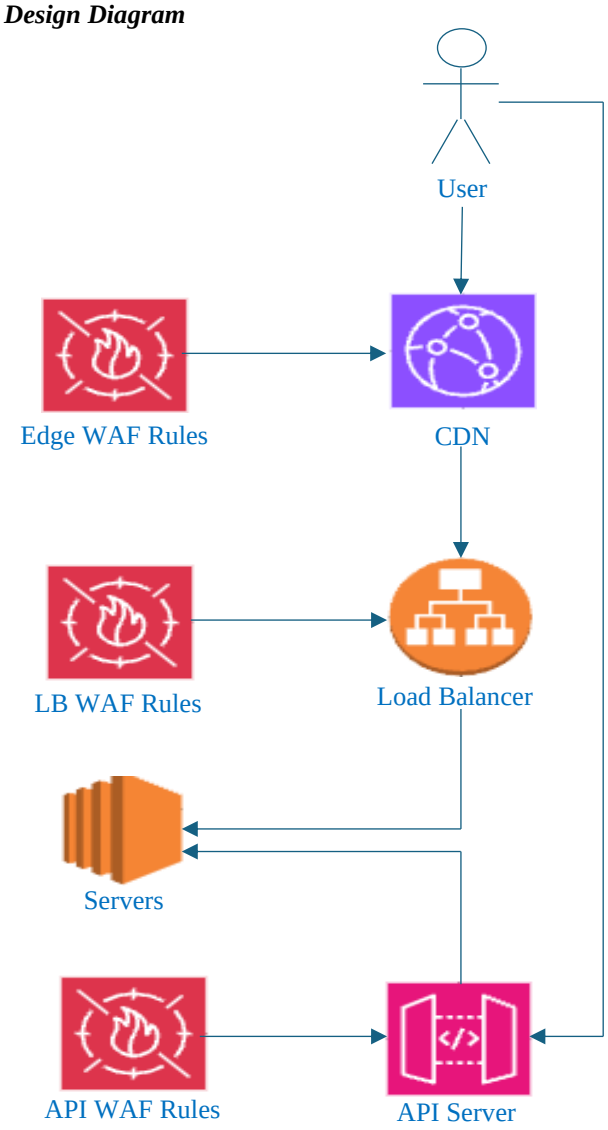

Fig. 1 An outline of how the traffic flows from one component to another

4.2. API Server behind CDN

Depending on the architecture and specific technology used for the API Server, it can be behind the CDN or directly accessible from the end-user. Based on the design considerations at that point in time, this determination must be made on a case-by-case basis. The criteria and decision should be documented so that the decision can be revisited in the future if the requirements have changed. In AWS, the API Server is implemented by API Gateway, which can have a WAF associated with it. API Gateway does not need to be put behind a CDN. The CDN implementation for AWS – CloudFront cannot selectively strip headers from the request to sanitize the input [19]; the stripping of headers makes it challenging to use with AWS API Gateway, which requires proper usage of request headers. The other reason why API Gateway should not reside behind CDN is that such API endpoints are regional, and we do not need to use the global capabilities of the CDN. If using the AWS Edge Optimized API Gateway instance, we get the CDN's benefits without setting them up explicitly [20]. Sometimes, it can be a good idea to put AWS Cloudfront in front of API Gateway to inject authorization headers that should not be exposed to users [21].

4.3. Monitoring and Alerting

The rate rules generate metrics and logs, which should be analyzed appropriately. They play a critical role in detecting and responding to DDoS attacks promptly. The key metrics to monitor would be request rates, error rates, and resource utilization at any time. These can indicate abnormal traffic patterns or resource exhaustion, which can significantly deteriorate the quality of the service.

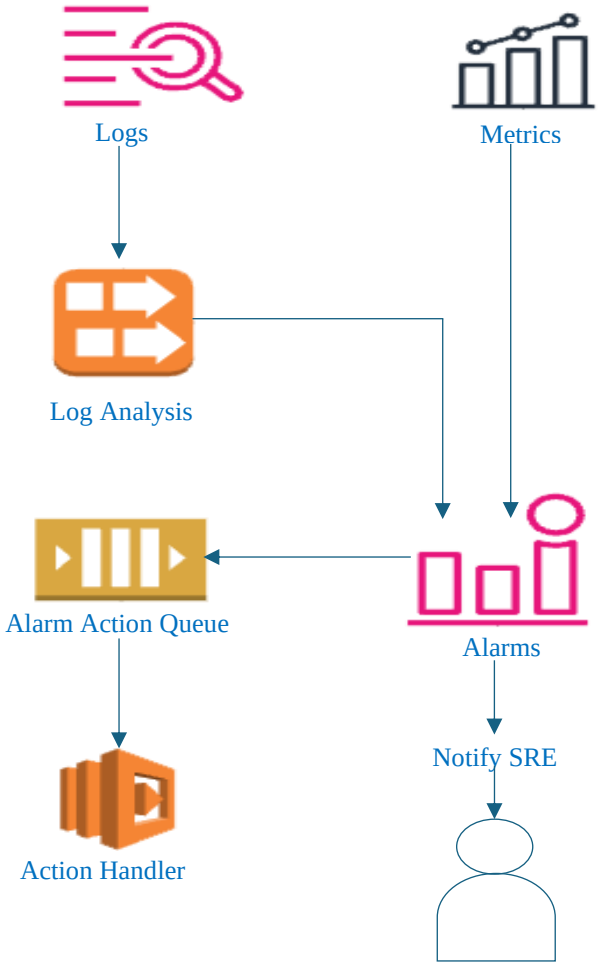

Fig. 2 An example of how to design the monitoring and alerting subsystem when using rate-limiting rules

Setting up alerts is crucial so appropriate individuals can be notified when manual intervention is needed to mitigate an incident or situation. Such incidents can be misconfigured bots sending too much traffic or an attacker trying to bring down the service. Alerts do not always have to involve individuals. Automated failovers and healing actions can be taken based on pre-determined conditions [22]. This reduces the cognitive load on the individuals responsible for the operation of the service and, at the same time, provides an audit log of the actions taken by the system.

4.4. Rules Ordering

In all the WAF implementations, the rules are kept together in a single logical container, and when a request arrives, the request is matched one by one from the first rule to the last. AWS WAF container for rules is WebACL, whereas, for Akamai WAF, it is called Security Configuration [23]. These implementations have actions like Allow, Block, Captcha, or Monitor. Allow and block are short-circuit evaluations, which means no further rules will be processed. The existence of such short-circuit rule action can be beneficial if used wisely. If we have Block rules, we must sort them by most likely to least likely. We can explicitly Allow certain kinds of traffic, for example, if we find a specific header with a particular string, which acts as a pre-authorized token. Such Allow rules should be cheap to evaluate and kept on top of the rules list if they can effectively allow a bulk of legitimate traffic.

4.5. Deployment with limited side effects

Adding, removing, or updating a rate-limiting rule on production traffic can be risky. Engineers are much more likely to test the new rules on a testing copy first and use synthetic traffic to validate their effects. This is a valid approach if we are trying to reduce the risk of unintended outages. Unfortunately, it is not always possible to reproduce the actual traffic using a synthetic traffic generator. In such cases, we would have to deploy the updated collection of rules to production WAF. Adding a new rule with the Block action by testing it on staging is not enough. The block action denies traffic to requests and can significantly affect legitimate users if such a rule is not crafted properly. Thankfully, all WAF solutions allow a Monitor action, which does not block or negatively affect the request. It generates metrics and collects logs specifically, which can help us identify how well such a new rule is performing. If the metrics generated show much higher traffic matched with this rule, we need to look at the corresponding traffic log and understand if such traffic is malicious or legitimate. Then, we can use a deploy-validate-repeat loop to identify the adequate rate-limiting rule we wish to use. Once we are confident of the quality of the new rate-limiting rule, we can change the action from Monitor to Block.

4.6. Incident Response and Mitigation

These rate-limiting rules aim to secure the service and achieve operational excellence. If an incident happens and there is no defined path to restoring the system to its expected state, it provides adequate time and opportunity for the bad actors to cause damage. A well-defined incident response plan is crucial to handle DDoS attacks effectively [24]. The plan should outline the roles and responsibilities of individuals participating in the response. Additionally, a communication method should be established, and criteria for escalation should be established. One single individual leads the incident response and is responsible for making it efficient and smooth. This individual can delegate another individual to take up the role of the journalist, who is responsible for documenting the actions taken step-by-step, including the determination and results of the investigation. This journal is a goldmine of information for the retrospective that should be conducted when the incident is mitigated. When an incident is detected, the first step is to identify the type and source of attacks, followed by placing the affected components and subsystems to prevent further damage. Blackholing is another technique that can be utilized to temporarily mitigate the issue by dumping the traffic on a null interface before it reaches the critical systems. Looking at the logs and metrics, the engineers should update the rate-limiting rules to handle the specific format of the attack for the time being before a more permanent solution can be found. Once the issue has been mitigated, retrospectives should be conducted, and proper action items should be created to ensure such an incident does not repeat.

4.6. Regulations and Laws

The 116th Congress introduced and passed H.R.1668—IoT Cybersecurity Improvement Act of 2020 [25], which focuses on improving the security of Internet devices. It requires the National Institute of Standards and Technology (NIST) and the Office of Management and Budget (OMB) to create standards and guidelines for appropriate use and minimum information security requirements for the agency that controls the device. This law does not compel companies to act specifically but would try to direct the industry to be self-regulated through consensus and guidelines. California's SB-327 Information Privacy: Connected Devices [26] focuses primarily on securing the data on connected devices. The law compels the businesses controlling the connected devices to implement and maintain reasonable security procedures and practices appropriate to the nature of the information. It will take time to determine if this law will reduce the number of devices that can be easily compromised and used as part of large-scale DDoS attacks.

5. Conclusion

This paper presents a collection of methodologies and how they relate to the service's cloud architecture. We explore five critical rate-limiting criteria and discuss whether they should be implemented on the Edge, Load Balancer, or API server. The order in which these rules exist is essential and can significantly affect each web request's round-trip time latency.

Care should be taken to judiciously choose the relevant rate-limiting methodology based on the service's specific architecture, which itself should be based on the business use case. The architect needs to ensure that the design is flexible enough to incorporate evolving business needs. This change in service design might necessitate changes to the rate-limiting methodologies being used. Minor design changes can have an outsized impact on the performance of a specific rate-limiting methodology. Static content that is no longer cached can become expensive in terms of network resources and costs. We have not provided any concrete implementation as no single implementation would do justice to different methodologies. Architects should explore the proper method for their use cases and use trial and error to determine the correct applications. Finally, we provided a sample WAF rule for each methodology, and such a sample was based on AWS WAF technology. AWS WAF's implementation can be ported to any known technology, such as Azure WAF [27], Google's Cloud Armor WAF [28], or Oracle WAF [29]. There is ample scope for further research to be based on this paper. This paper focuses primarily on AWS as the cloud provider and AWS WAF as the firewall technology. The two biggest competitors to AWS are Azure, with Azure WAF, and Oracle Cloud Infrastructure (OCI), with OCI WAF. Further research

References

[1] Abdulaziz Aljabre, "Cloud Computing for Increased Business Value," *International Journal of Business and Social Science*, vol. 3, no. 1, pp. 234-239, 2012. [Google Scholar] [Publisher Link]

[2] What is a DDoS Attack?, Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

[3] What is a Botnet?, Palo Alto Networks. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-botnet

[4] Manos Antonakakis et al., "Understanding the Mirai Botnet," *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, BC, Canada, pp. 1093-1110, 2017. [Google Scholar] [Publisher Link]

[5] DoS Attack vs. DDoS Attack, Fortinet. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/dos-vs-ddos

[6] Opeyemi Osanaiye, Kim-Kwang Raymond Choo, and Mqhele Dlodlo, "Distributed Denial of Service (DDoS) Resilience in Cloud: Review and Conceptual Cloud DDoS Mitigation Framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147-165, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[7] Gaurav Somani et al., "DDoS Attacks in Cloud Computing: Issues, Taxonomy, and Future Directions," *Computer Communications*, vol. 107, pp. 30-48, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[8] Mohamed Idhammad, Karim Afdel, and Mustapha Belouch, "Semi-Supervised Machine Learning Approach for DDoS Detection," *Applied Intelligence*, vol. 48, pp. 3193-3208, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Aanshi Bhardwaj et al., "Distributed Denial of Service Attacks in Cloud: State-of-the-Art of Scientific and Commercial Solutions," *Computer Science Review*, vol. 39, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] What are AWS WAF, AWS Shield Advanced, and AWS Firewall Manager?, AWS. [Online]. Available: https://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html

[11] Matt Calder et al., "Analyzing the Performance of an Anycast CDN," *Proceedings of the 2015 Internet Measurement Conference*, Tokyo, Japan, pp. 531-537, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[12] Geographic Match Rule Statement, AWS. [Online]. Available: https://docs.aws.amazon.com/waf/latest/developerguide/waf-rule-statement-type-geo-match.html

[13] Understanding IP Addressing and CIDR Charts, RIPE NCC. [Online]. Available: https://www.ripe.net/about-us/press-centre/understanding-ip-addressing/

[14] IP Reputation Rule Groups, AWS. [Online]. Available: https://docs.aws.amazon.com/waf/latest/developerguide/aws-managed-rule-groups-ip-rep.html

[15] Origin Server, Akamai Techdocs. [Online]. Available: https://techdocs.akamai.com/property-mgr/docs/origin-server

[16] Yang Zhang et al., "Detecting Malicious Activities With User-Agent-Based Profiles," *International Journal of Network Management*, pp. 306-319, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[17] FieldToMatch, AWS. [Online]. Available: https://docs.aws.amazon.com/waf/latest/APIReference/API_FieldToMatch.html

[18] Matt Auerbach, Zachary Goldberg, and Jay Raval, CDN Caching Improvements for Better App Performance with AWS Amplify Hosting, 2024. [Online]. Available: https://aws.amazon.com/blogs/mobile/cdn-caching-improvements-for-better-app-performance-with-aws-amplify-hosting/

[19] Deliver Custom Content with CloudFront, AWS, 2014. [Online]. Available: https://aws.amazon.com/blogs/aws/enhanced-cloudfront-customization/

[20] Amazon API Gateway Concepts, AWS. [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-basic-concept.html#apigateway-definition-edge-optimized-api-endpoint

[21] Chris Munns, AWS. Protecting your API Using Amazon API Gateway and AWS WAF — Part 2, 2018. [Online]. Available: https://aws.amazon.com/blogs/compute/protecting-your-api-using-amazon-api-gateway-and-aws-waf-part-2/

[22] Configure ALARM Actions for CloudWatch Alarms, Trend Micro. [Online]. Available: https://www.trendmicro.com/cloudoneconformity-staging/knowledge-base/aws/CloudWatch/cloudwatch-alarm-action.html
[23] API Concepts, Akamai Techdocs. [Online]. Available: https://techdocs.akamai.com/application-security/reference/api-concepts
[24] Nivedita Shinde, and Priti Kulkarni, "Cyber Incident Response and Planning: A Flexible Approach," *Computer Fraud & Security*, vol. 2021, no. 1, pp. 14-19, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[25] H.R.1668 - IoT Cybersecurity Improvement Act of 2020, Congress.Gov, 2020. [Online]. Available: https://www.congress.gov/bill/116th-congress/house-bill/1668
[26] Jackson, California Legislative Information, SB-327 Information Privacy: Connected Devices, 2018. [Online]. Available: https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201720180SB327
[27] Custom Rules for Web Application Firewall v2 on Azure Application Gateway, Microsoft Lgnite, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/web-application-firewall/ag/custom-waf-rules-overview
[28] Google Cloud Armor documentation, Google Cloud. [Online]. Available: https://cloud.google.com/armor/docs/security-policy-overview
[29] Overview of Web Application Firewall, Oracle Cloud Infrastructure Documentation. [Online]. Available: https://docs.oracle.com/en-us/iaas/Content/WAF/Concepts/overview.htm