IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.133138

Introduction to Application Layer DDoS Attacks And Protection Against It

Bhargavi Nalluri[1], Viswanadha Phani Koushik[2], Zunaid Yasir Syed[3], Kantam Pujitha[4], Gandi Lakshmi Vara Prasad[5]

Asst Prof, Department of CSE, KL University, Andhra Pradesh, India[1].

CSE, KL University, Andhra Pradesh, India[2-5]

project104cloud@gmail.com

Abstract: The increasing prevalence of distributed denial of service (DDoS) attacks has highlighted the urgent need for robust protection strategies, particularly at the application layer in the computers. DDoS attacks [2] targeting the application layer exploit vulnerabilities in web applications and services, making traditional network-layer defenses ineffective. This abstract explores effective measures for protecting against application layer DDoS attacks [2], emphasizing proactive strategies for safeguarding web applications, and ensuring uninterrupted service availability. Prominent protection measures include the deployment of web application firewalls (WAFs), which analyze incoming traffic for suspicious patterns and block malicious requests. Rate-limiting mechanisms can be used to reduce the influx of requests from a single source, mitigating the impact of an attack. Application-layer DDoS detection systems [5], supported by machine learning algorithms, enable rapid identification of abnormal behavior and malicious traffic.

Keywords: DDoS attacks, Application Layer DDoS attacks, HTTPS Floods, Slowloris Attack, DNS Amplification.

I. INTRODUCTION

The arrival of the digital age has led to unprecedented advancements in technology. However, with this progress comes a darker underbelly—cybersecurity threats that can disrupt the very foundations of our online world. Among these threats, distributed denial of service (DDoS) [2] attacks have emerged as a formidable adversary, capable of crippling online services, businesses, and even critical infrastructure. DDoS attacks, in their various forms, pose a grave danger to the availability and integrity of web applications and services. While network-layer defenses can mitigate some of these threats, the evolution of attack techniques has rendered them increasingly inadequate. Of particular concern are DDoS attacks [2] that target the application layer, where the vulnerabilities of web applications are exposed, leading to service disruption. This study investigates the application layer DDoS attacks, shedding light on their complexities, consequences, and the urgent need for effective defense measures. The objective is to provide businesses and cybersecurity experts with a comprehensive understanding of the risks posed by these attacks and the proactive steps they can take to safeguard their digital assets. The upcoming sections will discuss application layer DDoS attacks, their impact, and ways to enhance defense tactics for businesses. We can protect ourselves by gaining knowledge and putting effective measures in place, protect web services from application layer DDoS attacks in the digital era.

II. UNDERSTANDING APPLICATION LAYER DDOS ATTACKS : A DEEPER LOOK

Application Layer DDoS attacks [3], otherwise labeled as Layer 7 DDoS onslaughts, epitomize an exquisitely intricate and pinpointed manifestation of cyber assault. In contrast to conventional DDoS attacks [2] that concentrate on inundating network resources, Application Layer attacks singularly zero in on the application layer of the OSI (Open Systems Interconnection) model. This layer bears the onus of processing and reciprocating to user entreaties.

DDoS assaults on the application layer emerge for the following reasons:

1. Protocol exploitation Malefactors exploit flaws in application-layer protocols, including SIP (Session Initiation Protocol), HTTP, and DNS. They inundate the system with an onslaught of deceitful entreaties, adroitly camouflaging them as legitimate traffic to elude detection.

2. The strategy of The Low and the Slow: Attacks on the application layer often employ cunning lowand-slow master plan. Assailants dispatch a scant number of appeals over an extended duration in a bid to gradually deplete the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.133138

server's resources. An illustrious exemplification is Slowloris, which bombards the target server with several connections and dispatches fragmented HTTP entreaties while perpetually keeping each association open and essentially attack the server's resources.

III. TYPES OF DDOOS ATTACKS

A. HTTP/HTTPS Floods :

Attackers bombard web servers with a significant quantity of. HTTP or HTTPS requests to deplete server resources. These attacks frequently target pages, images, and scripts on a website., among other components. They rely on a high request volume, which exceeds the server's capacity for processing. Legitimate users are consequently subjected to lengthy loading times or total service interruptions. The figure below represents how this flood is working.

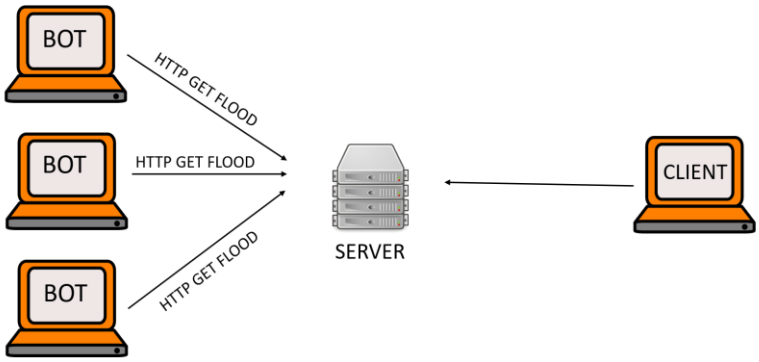

Fig. 1 HTTP/HTTPS Floods

B. HTTP Post Flood:

In this attack, attackers flood a web server with A considerable quantity of HTTP POST requests, many of which have large payloads. These payloads may include uploaded files., form submissions, or other data. The main objective is to use up as much CPU and memory as possible on the server while processing requests. As a result, performance degradation and slow response times may occur on the server. The figure below represents how this flood is working.

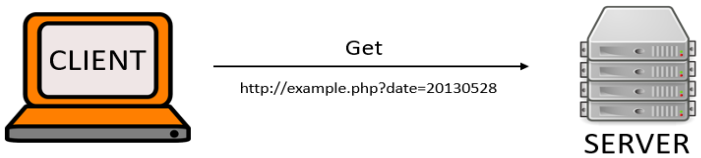

Fig. 2 HTTP POST Flood

C. HTTP Get Flood:

Like POST floods, GET floods involve flooding the server with numerous HTTP GET requests. Attackers may target URLs or application resources. The impact of the attack is in the consumption of resources, particularly the strain on server resources, which causes slow loading times for legitimate users and may result in service interruption. The figure below represents how this flood is working.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.133138

Fig. 3 HTTP GET Flood

D. Slowloris Attack:

The Slowloris attack involves repeatedly connecting to a web server and slowly sending HTTP headers. This trickery keeps connections open for a long time without processing the requests. Slowloris aims to reach the server's maximum number of concurrent connections, rendering it unresponsive to brand-new, legitimate connections. The figure below represents how this flood is working.

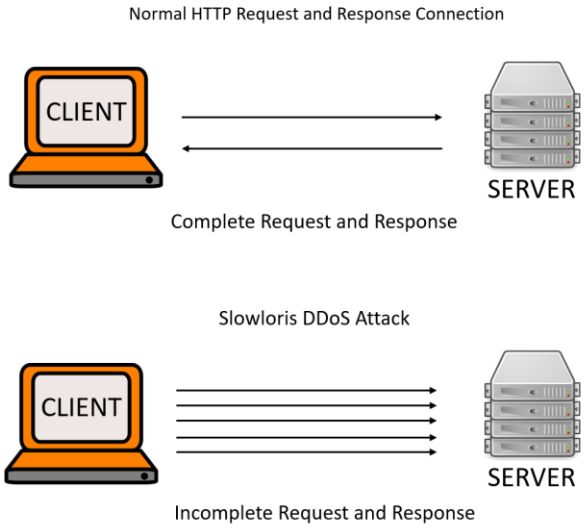

Fig. 4 Slowloris attack

E. DNS Amplification:

Attackers use incorrectly configured DNS servers to magnify their attacks. They send open DNS resolvers DNS requests with forged source IP addresses. The victim's server receives DNS responses from the resolvers which is much larger than the initial queries, which generates a large amount of traffic. This attack uses a lot of bandwidth and may interfere with services. The figure below represents how this flood is working.

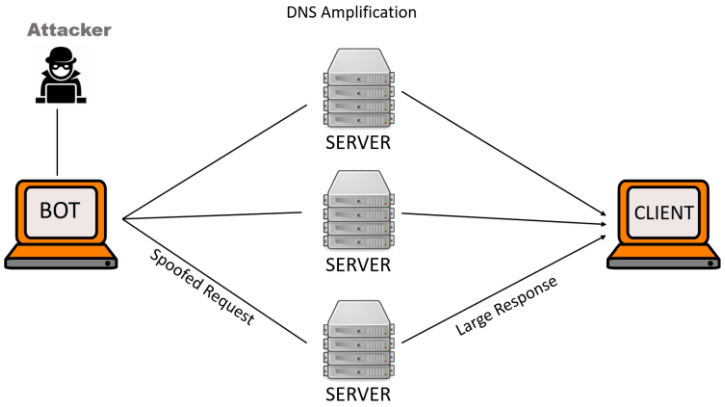

Fig. 5 DNS Amplification

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.133138

F. SYN/ACK Flood(Layer7):

During a Layer 7 SYN/ACK flood, attackers produce a flood of SYN/ACK packets at the application layer with the intention of attacking a particular server service or application. As it tries to make connections, this attack aims to use up server resources like CPU and memory. Consequently, it differs from conventional network layer SYN/ACK floods in that it may result in service degradation or unavailability. The figure below represents how this flood is working.

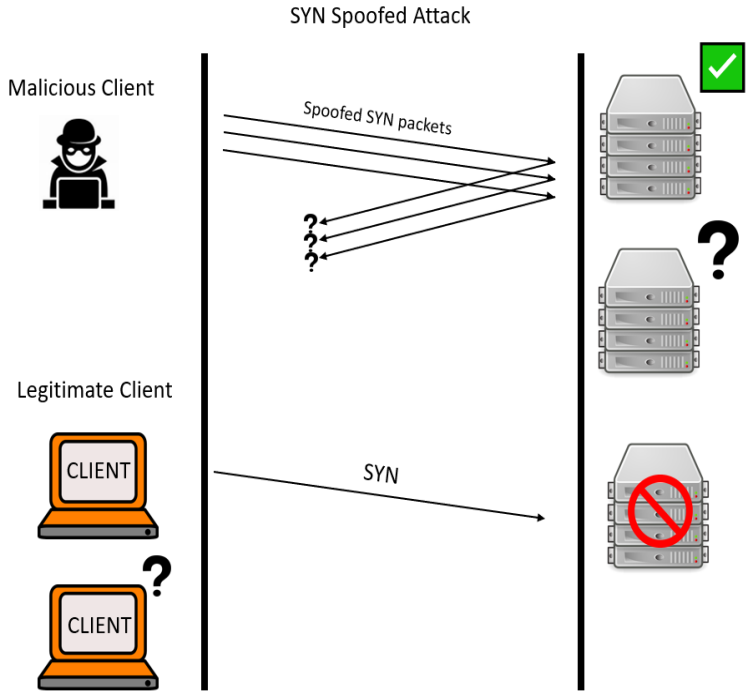

Fig. 6 SYN/ACK Flood

IV. EFFECTS OF THESE ATTACKS IN REAL TIME

Let us dive into the real-time effects of these attacks

A. HTTP/HTTPS Floods real example:

In the 2016 Dyn attack, the Mirai botnet orchestrated a massive HTTP flood against Dyn's DNS infrastructure. This attack disrupted access to numerous major websites, including Twitter, Netflix, Reddit, and CNN. The consequences included not only user inconvenience but also financial losses for businesses relying on these services.

B. Slowloris Attack real example:

During the 2009 Iranian post-election protests, the Slowloris attack was used against various Iranian government websites. By tying up connections slowly and stealthily, these attacks contributed to the unresponsiveness of these sites amid political turmoil, demonstrating the attacker's ability to disrupt critical online services during sensitive periods.

C. HTTP POST Flood real example:

GitHub, a prominent code hosting platform, faced a massive HTTP POST flood attack in 2018. Peaking at 1.35 Tbsp. This attack briefly impacted service availability, highlighting the potential vulnerability of even well-resourced online platforms to Application Layer DDoS attacks [5].

D. HTTP GET Flood real example:

In 2016, KrebsOnSecurity, a well-known security blog, was targeted by a GET flood attack reaching 620 Gbps. The attack disrupted access to the site for some time, underscoring the need for robust mitigation strategies to protect against Application Layer DDoS attacks [5].

E. DNS Amplification real example:

In 2013, Spamhaus, an anti-spam organization, became the one of the targets to the largest DNS amplification [12] attacks on record. Attackers exploited vulnerable DNS resolvers to amplify their traffic. The attack peaked at around

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024
DOI: 10.17148/IJARCCE.2024.133138

300 Gbps, causing widespread internet routing issues in Europe. This incident demonstrated the potential for DNS amplification [12] attacks to disrupt not only specific services but also broader internet infrastructure.

F. SYN/ACK Flood (Layer 7) real example:
In 2016, the well-known cybersecurity blog KrebsOnSecurity suffered a Layer 7 SYN/ACK flood attack, reaching an astonishing 665 Gbps. This attack overwhelmed the site's defenses, causing a temporary outage. The incident highlighted the evolving tactics of DDoS attackers, as traditional SYN/ACK floods usually operate at the network layer but were adapted for application layer attacks [5] in this case.

V. LITERATURE STUDY

A literature review of DDoS attacks at the application layer [5] can provide an overview of the various aspects, techniques, and countermeasures related to this type of cyber threat. Below, I have provided an overview of some of the key research papers and resources related to application-layer DDoS attacks up to the point of my understanding in September 2021. Please note that since then, there may have been important developments in this area.

A. A. A. Fernandes [10] investigated the possible security risks [1] connected to IoT devices, which are frequently included in botnets used for DDoS attacks. The paper delves into the topic of application-layer DDoS attacks and examines the security threats [1] that result from them, even though it does not exclusively concentrate on them.

In a study titled DDoS Attacks Against Content Delivery Networks, M. Antonakakis [11] examined DDoS attacks that target Content Delivery Networks (CDNs) and talked about how they affect the application layer. It draws attention to how difficult it is to repel such attacks.

Understanding the Mirai Botnet, a study by J. Antonakakis [12], examines the Mirai botnet, which was implicated in multiple DDoS attacks, including application layer attacks. It offers information about how the botnet functions and how application layer attacks employ it.

R. Stone [13] looked into the defense against and characterization of application-layer DDoS attacks in his study. The application layer is the main topic of this study. This paper highlights the danger that denial-of-service (DDoS) attacks pose to web services by describing their anatomical structure. In order to strengthen applications against such attacks, the paper examines a number of defense mechanisms, such as rate limitation and request classification, offering useful information to network administrators and security experts.

An extensive examination of HTTP Flood attacks [2] is given in "HTTP Flood Attack [2] and Defense: A Review," which also highlights the attacks' evolution and variants. It provides a thorough analysis of defense strategies and investigates how these attacks affect web servers, making it an invaluable tool for comprehending and fending off HTTP Floods [2].

Information about server security against HTTP-Flood attacks can be found in "Evaluating and Enhancing the Security of Servers Against HTTP-Flood Attacks [2]". The importance of rate limitation and anomaly detection in preventing these attacks is emphasized in the discussion of detection and prevention strategies.

"Slow Loris HTTP Denial of Service" goes into detail about the Slow Loris attack, describing how it works and how it can effectively use up server resources. The effectiveness of Slow Loris as a tool for DDoS attacks is also covered in the paper, along with mitigation and detection techniques.

The comprehensive study "Understanding Slow Loris Attacks: Analysis, Detection, and Mitigation" highlights the stealthy character of Slow Loris attacks. It looks at ways to spot Slow Loris-based attacks and provides advice on how to avoid them.

The article "HTTP POST Attacks: An Overview" provides a thorough rundown of HTTP POST Flood attacks and emphasizes their importance in denial-of-service attacks. It looks at how attacks are carried out, how they affect web services, and what mitigation techniques are available.

"Mitigating HTTP POST Flood Attacks Using Request Analysis" suggests a method for preventing HTTP POST Flood attacks that is based on request analysis. The analysis and implementation of an effective defensive strategy are the main topics of this essay.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024
DOI: 10.17148/IJARCCE.2024.133138

In order to defend against HTTP GET Floods, "Defending Against HTTP GET Flood Attacks: A Behavioral Analysis Approach" employs a behavioral analysis strategy. To effectively detect and mitigate threats, the research highlights how crucial it is to comprehend the behavior of both malicious and legitimate clients.

The investigation of HTTP GET Flood attacks is covered in "Analysis and Detection of HTTP GET Flooding DoS Attacks." It presents a detection algorithm based on request patterns and offers insights into the features of these attacks.

The article "DNS Amplification Attacks [12]: A Comprehensive Overview" offers a thorough explanation of DNS Amplification attacks, explaining their workings and the effects they have on DNS infrastructure. Response Rate Limiting (RRL) is one of the mitigation strategies covered in the paper.

"Analyzing and Mitigating DNS Amplification Attacks" is a book that focuses on these two processes. It offers case studies from real-world situations as well as useful strategies for spotting and thwarting these attacks.

Rate limitation, CAPTCHA challenges, and machine learning-based strategies are just a few of the methods for preventing HTTP Flood attacks that are compared in a paper titled "A Comparative Analysis of HTTP Flood Attack Mitigation Techniques."

The use of machine learning algorithms for HTTP Flood attack detection and mitigation is the main topic of a study titled "Detecting [16] and Mitigating HTTP Flood Attacks Using Machine Learning." It investigates the efficacy of various machine learning models and how well they work in real time.

An extensive analysis of the Slowloris attack, including methods for recognizing and successfully mitigating it, can be found in the study "Slowloris Attack [16]: Identifying, Mitigating, and Evaluating Effectiveness." It also assesses how well various mitigating techniques work.

The use of machine learning techniques to improve the detection of Slowloris assaults [16] is investigated in the article "Enhancing Slowloris [16] Detection Using Machine Learning Algorithms." It discusses how machine learning can be used to detect subtle assault patterns.

The focus of a study titled "Defending Web Applications Against HTTP POST Flood Attacks" is on protecting web applications from POST flood attacks. In order to safeguard web servers, it goes over methods for tracking and limiting POST requests.

A paper titled "Advanced Defense Mechanisms Against Large-Scale HTTP POST Flood Attacks" investigates these kinds of attacks and their defenses. The application of behavior-based analysis and proactive response techniques are covered.

The focus of a study named "Detecting and Mitigating HTTP GET Flood Attacks in Real-Time" is on HTTP GET Flood attack detection and mitigation in real-time. It offers a proactive method for spotting and thwarting nefarious GET requests.

The paper "Anomaly-Based Detection of HTTP GET Flood Attacks Using Machine Learning" explores the use of machine learning in anomaly-based detection methods to recognize and prevent HTTP GET Flood attacks.

Effective methods for identifying and thwarting DNS amplification attacks are covered in a study titled "Efficient Detection [16] and Mitigation of DNS Amplification Attacks [3]." It highlights how crucial it is to react to these kinds of attacks quickly.

Response Rate Limiting (RRL) and traffic filtering are two of the approaches for mitigating DNS amplification attacks [12] that are compared in a study titled "DNS Amplification Attack Mitigation [3]: A Comparative Analysis of Approaches." The use of stateful packet inspection (SPI) to lessen SYN/ACK Flood attacks is investigated by the authors of a study titled "Mitigating SYN/ACK Flood Attacks Using Stateful Packet Inspection." The benefits of SPI in recognizing and obstructing malicious traffic are covered.

Researchers explore SYN/ACK Flood attacks, covering attack patterns, detection strategies, and various countermeasures, in a study titled "A Comprehensive Study on SYN/ACK Flood Attacks and Countermeasures." It offers a comprehensive strategy for countering these assaults.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024

DOI: 10.17148/IJARCCE.2024.133138

VI. PROTECTION AGAINST THESE ATTACKS WITH ALGORITHMS

A. Token Bucket Algorithm for Http/Https Flood:

A network can control requests based on the amount of traffic through the token bucket algorithm. Each bucket holds a fixed number of tokens that represents network requests, like logging into an account or sending a message. Whenever a user makes a request, one token is added to the bucket.

If users send too many requests in a short amount of time, the algorithm will stop working. This is because the bucket can only hold a limited number of tokens. Whenever the network performs a "bucket refill," the number of allowed tokens is reset. Until then, all new requests will be denied.

1. Token Generation: In the Token Bucket algorithm, tokens are generated at a constant rate known as the token replenishment rate. For example, tokens may be produced at a rate of X tokens per second.
2. Token Consumption: When an HTTP or HTTPS request arrives, it consumes one token from the Token Bucket. If there are not enough tokens, the request is either dropped or delayed.
3. Bucket Capacity: The Token Bucket has a maximum capacity, representing the maximum number of tokens it can hold at any given time. If the bucket is full, no additional tokens are generated.
4. Rate Limiting: The speed at which requests are allowed depends on the token generation rate and bucket capacity. Requests can only be processed at the token replenishment rate when the bucket is full.

B. Connection Timeout Enforcement algorithm for Slow Loris attack:

The Connection Timeout Enforcement algorithm works to protect against Slow Loris attacks by actively managing and enforcing strict connection timeout settings. Here is a stepby-step explanation of how this algorithm functions to safeguard web servers.

Configuration: Administrators configure the web server with a relatively short and strict connection timeout value. This value represents the maximum allowable duration for a client to complete the connection establishment process with the server.

Incoming Connection Tracking: The server continuously tracks all incoming connection attempts from clients. It records the initiation time for each connection.

Timeout Timer: As soon as a connection attempt is initiated, the server starts a timer for that connection based on the configured connection timeout value. This timer is set to expire after the specified duration.

Timeout Expiry Detection: The server continually monitors the connection timers for all ongoing connection attempts. When the timer for a particular connection expires, it means that the client has not completed the connection setup within the defined time frame.

Automatic Connection Termination: Upon detecting the expiration of a connection's timeout, the server takes proactive action and automatically terminates that specific connection. This termination is essential as it releases server resources that were tied up by incomplete connections.

Resource Preservation: By enforcing strict connection timeouts and promptly terminating incomplete connections, the Connection Timeout Enforcement algorithm effectively preserves server resources. Slow Loris attacks rely on keeping numerous connections open for an extended period to exhaust server resources.

However, the algorithm prevents this by reclaiming resources from incomplete connections. The server continues to enforce these connection timeouts for all incoming connections. This means that Slow Loris attackers, who may try to keep multiple connections open, are consistently thwarted as their connections are terminated when they fail to complete the connection setup within the allowed time.

Dynamic Adjustment: Administrators have the flexibility to dynamically adjust the connection timeout value based on evolving traffic patterns and potential threats. This adaptability ensures that the defense remains effective against changing attack tactics.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024
DOI: 10.17148/IJARCCE.2024.133138

C. Request Size Limiting Algorithm for HTTP POST Flood:

HTTP POST Flood attacks involve sending a high volume of POST requests, often with large payloads, to overwhelm a web server. The Request Size Limiting algorithm helps mitigate this attack by setting a maximum allowable request size. Here is how it works:

Configuration: Administrators configure the web server with a maximum allowable request size. This value represents the largest payload size that the server will accept for incoming POST requests.

Incoming Request Inspection: When the server receives incoming POST requests, the algorithm inspects their payload size. This inspection occurs before processing the request.

Request Size Check: The algorithm checks if the incoming POST request payload is within the configured size limit.

Decision Making: If the request payload size is within the allowed limit, the server processes the request as usual. If the request payload size exceeds the configured limit, the server takes one of the following actions:

1. Rejects the request outright, sending an error response to the client.
2. Delays the request, effectively slowing down the rate at which it is processed.
3. Drops the request, preventing it from consuming server resources.

Resource Conservation: By limiting the size of incoming POST requests, the algorithm ensures that excessively large payloads, often associated with HTTP POST Flood attacks, are either rejected or delayed. This prevents attackers from overwhelming the server with large amounts of data.

Attack Mitigation: HTTP POST Flood attacks typically various points in the network infrastructure, such as within a web application firewall or load balancer, providing scalability and protection across the entire server environment.

Dynamic Configuration: Administrators have the flexibility to dynamically adjust the maximum allowable request size based on evolving traffic patterns and potential threats. This adaptability ensures that the defense remains effective against changing attack tactics.

HTTP POST Flood attacks involve sending a high volume of POST requests, often with large payloads, to overwhelm a web server. The Request Size Limiting algorithm helps mitigate this attack by setting a maximum allowable request size. Here's how it works:

Configuration: Administrators configure the web server with a maximum allowable request size. This value represents the largest payload size that the server will accept for incoming POST requests.

Incoming Request Inspection: As incoming POST requests are received by the server; the algorithm inspects the size of the request payload. This inspection occurs before processing the request.

Request Size Check: The algorithm checks whether the size of the incoming POST request payload exceeds the configured maximum allowable request size.

Decision Making: If the request payload size is within the allowed limit, the server processes the request as usual. If the request payload size exceeds the configured limit, the server takes one of the following actions:

1. Rejects the request outright, sending an error response to the client.
2. Delays the request, effectively slowing down the rate at which it is processed.
3. Drops the request, preventing it from consuming server resources.

Resource Conservation: By limiting the size of incoming POST requests, the algorithm ensures that excessively large payloads, often associated with HTTP POST Flood attacks, are either rejected or delayed. This prevents attackers from overwhelming the server with large amounts of data.

Attack Mitigation: HTTP POST Flood attacks typically aim to consume server resources by sending numerous requests with large payloads. The Request Size Limiting algorithm thwarts these attempts by controlling the size of incoming requests.

DOI: 10.17148/IJARCCE.2024.133138

Scalability: The algorithm can be implemented at various points in the network infrastructure, such as within a web application firewall or load balancer, providing scalability and protection across the entire server environment.

Dynamic Configuration: Administrators have the flexibility to dynamically adjust the maximum allowable request size based on evolving traffic patterns and potential threats. This adaptability ensures that the defense remains effective against changing attack tactics.

D. Response Rate Limiting (RRL) Algorithm for DNS Amplification Attack Mitigation:

DNS queries with a spoof source IP address are directed to DNS servers whose strength is weak, which consequently transmit amplified responses to the target, causing traffic floods. The RRL algorithm is designed to protect DNS servers by imposing rate limits on DNS responses. Here is how it operates:

Monitoring Incoming DNS Requests: The DNS server continuously monitors incoming DNS query requests, tracking their source IP addresses.

Response Rate Calculation: The RRL algorithm calculates the rate at which responses are generated for each unique source IP address. It may use historical data and current traffic patterns to establish a baseline response rate.

Threshold Detection: If the response rate from a specific source IP address exceeds a predefined threshold, indicating a potentially malicious or amplified query, the RRL algorithm triggers rate-limiting actions.

Rate Limiting: When the threshold is exceeded, the DNS server may respond to subsequent queries from that source IP address at a reduced rate. This means that the server may delay or limit responses to queries from the offending source, making it less effective for attackers seeking amplification.

Logging and Reporting: The algorithm often logs and reports rate-limiting events, allowing administrators to monitor and investigate potential attacks.

Resource Conservation: By limiting responses to potentially malicious requests, the RRL algorithm conserves server resources and reduces the risk of contributing to traffic amplification.

Attack Mitigation: DNS amplification attacks rely on the ability to generate amplified traffic. The RRL algorithm disrupts this by limiting the rate at which responses are generated, making it less effective for attackers.

Customizable Parameters: Administrators can customize the RRL algorithm by setting thresholds and rate-limiting parameters to align with their network's requirements and potential threat scenarios.

Dynamic Adaptation: The algorithm can dynamically adjust rate-limiting parameters based on observed traffic patterns, providing protection against evolving attack tactics.

E. SYN Cookies Algorithm for SYN/ACK Flood Mitigation:

SYN Cookies are a security feature integrated into the TCP/IP stack, primarily designed to protect against SYN Flood attacks. SYN Cookies are highly effective in mitigating SYN/ACK Flood attacks because they prevent the allocation of server resources until a legitimate connection is established. Here is how the SYN Cookies algorithm operates:

Connection Initiation: The server creates a unique SYN Cookie depending on the client's IP address, port, and other connection details when a client establishes a connection with the server by sending a TCP SYN (synchronize) packet.

Sending the SYN Cookie: Instead of allocating server resources immediately, the client receives the SYN Cookie from the server's end in the TCP SYN-ACK (synchronize-acknowledge) packet. The SYN Cookie is a mathematical computation based on the client's connection parameters.

Client Acknowledgment: To complete the connection establishment, the client must return the SYN Cookie in its ACK (acknowledge) packet. The ACK packet contains the SYN Cookie, indicating that the client acknowledges the server's SYN-ACK response.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024
DOI: 10.17148/IJARCCE.2024.133138

Server Validation: When the server receives the ACK packet with the SYN Cookie, it validates the SYN Cookie by recalculating it based on the received parameters. If the recalculated SYN Cookie matches the one sent to the client, the server considers the connection legitimate and allocates the necessary resources to handle it.

Resource Allocation: Only after the SYN Cookie is validated and a legitimate connection is confirmed does the server allocate resources and allow data transfer between the client and the server.

Expiry and Cleanup: SYN Cookies typically have a limited lifetime, after which they expire. Expired SYN Cookies are discarded, freeing up any associated resources. This helps prevent resource exhaustion due to incomplete connections.

Attack Mitigation: SYN/ACK Flood attacks aim to overwhelm a server by initiating numerous connections. However, SYN Cookies ensure that server resources are only allocated when a legitimate connection is established, making it significantly more challenging for attackers to exhaust resources.

Scalability: SYN Cookies are implemented at the TCP/IP stack level and are therefore applicable to all incoming connections, making them scalable and suitable for network-wide protection.

Dynamic Adaptation: SYN Cookies adapt to changing traffic patterns and attack tactics by only allocating resources to legitimate connections, regardless of the attack volume.

VII. CONCLUSION

Finally, the increasing sophistication and diversity of distributed denial of service (DDoS) attacks, with an emphasis on the application layer, highlight the crucial significance of effective defense measures. Traditional network-layer defenses are ineffective against the complexities of application layer threats, demanding a move to proactive solutions.

This research has offered a thorough examination of application layer DDoS attacks, shining light on their intricacies, implications, and the critical need for robust defense methods. The study stresses the dynamic nature of the threat landscape by evaluating real-world instances and digging into numerous attack methods, ranging from HTTP/HTTPS floods to Slowloris and SYN/ACK floods.

To combat application layer DDoS attacks, research and development are crucial, as the literature review further emphasizes. The literature review highlights several studies and papers that provide useful insights into understanding, recognizing, and mitigating these threats. Algorithms for machine learning, rate-limiting devices, and behavioral analysis are becoming as critical components of advanced protection systems.

Furthermore, the study introduces algorithms tailored to specific attack types, such as the Token Bucket Algorithm for HTTP/HTTPS floods, Connection Timeout Enforcement for Slowloris attacks, Request Size Limiting for HTTP POST floods, Response Rate Limiting for DNS Amplification attacks, and SYN Cookies for SYN/ACK floods. These algorithms offer dynamic and scalable defense mechanisms, showcasing the adaptability required to combat evolving attack tactics.

REFERENCES

[1]. Mell P, Grance T – https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf
[2]. Shi Dong, Raj Jain, Khushnood Abbas – “https://ieeexplore.ieee.org/document/8735686”
[3]. A. A. A. Fernandes, et al - “Understanding and Mitigating the Security Risks of loT”. IEEE Security & Privacy, 2018.
[4]. M. Antonakakis, et al - "An Inside Look at DDoS Attacks Against Content Delivery Networks".ACM SIGCOMM, 2018.
[5]. R. Stone et al - "Application-Layer DDoS Attacks: Characterization and Defense".
[6]. IqbalAhmed - “https://www.researchgate.net/publication/337953680_A_brief_review_security_issues_in_cloud_computing_and_their_solutions”
[7]. WayneJansenTimothyGrance.Guidelines-https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2024
DOI: 10.17148/IJARCCE.2024.133138

[8]. Ali Nafea Methaq Abdullah Shyaa Hassan Muayad Ibrahim Ayat Saleh Hasan "https://www.researchgate.net/publication/358046663_The_Impacting_on_Network_Security_and_the_Risk_for_Organization_Behaviours"
[9]. R.Sanjeetha,K.N.A.Shastry,H.R.Chetanand A. Kanavalli – https://www.semanticscholar.org/paper/Mitigating-HTTP-GET-FLOOD-DDoSattack-using-an-SDN-Sanjeetha-Shastry/f8575433d605d24f8c7c88c6a22ce270239d18ae
[10]. M. Sikora, T. Gerlich and L. Malina – "https://ieeexplore.ieee.org/document/8970844"
[11]. MAEDA,SHOGO-HTTPS://WWW.RESEARCHGATE.NET/PUBLICATION/331594868_A_BOTNET_DETECTION_METHOD_ON_SDN_USING_DEEP_LEARNING
[12].L.CAI,Y.FENG,J.KAWAMOTOANDK.SAKURAI-HTTPS://IEEEXPLORE.IEEE.ORG/DOCUMENT/7794541
[13].S. H. C. Haris, R. B. Ahmad and M. A. H. A. Ghani – https://ieeexplore.ieee.org/document/5635797
[14].A. Praseed and P. S. Thilagam – https://ieeexplore.ieee.org/document/8466561
[15].S. Bravo and D. Mauricio – https://ieeexplore.ieee.org/document/8356848
[16].S. Black and Y. Kim – "https://ieeexplore.ieee.org/document/9720741"