(IJISE) 2023, Vol. No. 17, Issue 1, Jan-Jun **e-ISSN: 2454-6402, p-ISSN: 2454-812X**

Cloud Sentry: Innovations in Advanced Threat Detection for Comprehensive Cloud Security Management[1]

Subash Banala

Capgemini, Senior Manager,
Financial Services & Cloud Technologies
Texas, USA

Received: 29 January 2023; Accepted: 19 February 2023; Published: 02 April 2023

ABSTRACT

Cloud services are touted as having many benefits, including seamless resource access, scalability, and elasticity. But they are also confronted with many risks that threaten both infrastructure or application level which, application-layer distributed denial of service (DDoS) attacks are where a difficult problem problem is prevented. These attacks usually create a bottleneck on targeted servers, degrading their performance and availability by exhausting consumption of available resources.

Existing solutions such as intrusion detection and protection can mitigate certain types of attacks, new DDoS assaults at the apps layer are always developing around them. In order to prevent application-layer DDoS attacks, this paper introduces a new and efficient technique called the SENTRY framework. SENTRY uses a challenge-response method that (a) assesses the attackers' actual physical bandwidth resource, (b) adjusts to the network's fluctuating workload conditions, and (c) filters the malicious clients' service requests.

INTRODUCTION

Adopted DDoS attacks: a risk to cloud service providers' security. They act both on the victim servers and on degraded services. Normally DDoS attacks happen on layer 3 (transport layer) and layer 4 (network layer) of OSI TCP/IP stack where malicious traffic floods all network interfaces to consume resources and deny legitimate traffic.

In contrast, application-layer DDoS attacks represent a more significant and complex long-term threat [1]. Such attacks leverage the momentous growth of web application complexity and available bandwidth [2], consuming resources incrementally and not powerfully, in comparison to the biggest attack on Bitbucket Data Center that led to sporadic service disruption for more than 12 hours [4]. Web applications have become prime attack targets due to their explosion in popularity and the lack of effective countermeasures.

The application layer DDoS attacks is unlike the traditional ones, in several aspects. For instance, they create less network traffic, put a high system overhead per query to servers, and can more easily evade intrusion detection systems [1].

In order to deal with these challenges, We suggest SENTRY as a novel security mitigation technique for DDoS attacks at the application layer. SENTRY utilizes the local uplink bandwidth of remote users to evaluate request legitimacy and alleviate resource flooding from such attacks dynamically. SENTRY seeks to:

Lessen configuration pains operating at the middleware/protocol level, abstracting lower-level network complexities and simplifying the deployment in the Cloud environments.

Adjust to various workloads presented to the servers.

A charade based challenge-response process, to impede suspicious service requests from dishonest clients using a physical bandwidth.

Our analysis demonstrates that SENTRY can successfully counteract application-layer DDoS attacks in a variety of real-world situations.

[1] *How to cite the article:* Banala S.; Cloud Sentry: Innovations in Advanced Threat Detection for Comprehensive Cloud Security Management; *International Journal of Innovations in Scientific Engineering*, Jan-Jun 2023, Vol 17, 24-35

(IJISE) 2023, Vol. No. 17, Issue 1, Jan-Jun **e-ISSN: 2454-6402, p-ISSN: 2454-812X**

BACKGROUND

Such types of application layer DDoS attack show a high level of application-layer sophistication by stealthily consuming resources required for processing requests in the targeted servers. Unlike conventional network layer DDoS attacks there are three main features of application layer DDoS attacks:

Application Layer DDoS Attacks: Characteristics and Mitigation

Attack Characteristics

Workload-enhancing DDoS assaults at the application layer are intended to prevent service by depleting vital resources, such as CPU cycles, I/O, memory, and network bandwidth. Despite the enormous resource capacity of cloud server systems, certain resources may cause performance bottlenecks. As an illustration, consider Amazon EC2, which is renowned for having strong networks but has seen HTTP protocol and XML-based application-layer DDoS attacks that have the potential to overwhelm its networks' capacity [5].

These attacks are asymmetric, targeting specific application protocols with high-overhead services. Attackers exploit These attacks are asymmetric and aimed at specific application protocols with high-overhead services. This is defined as when multiple client hosts are utilized in order to bombard target servers with some few specific resource-consuming service requests making them unbearably loaded until the point that those servers stop functioning [6].

Furthermore, application layer DDoS attacks are stealthDDoS, disguising malicious requests as normal service requests to avoid detection by an intrusion detection system looking for abnormal traffic patterns. Authentication services, for example, are susceptible to masquerading attacks that replicate valid service requests [7], depleting the system's resources significantly. Attackers are cunning in that they attribute these requests to several sources and create so few traffic anomalies to evade detection by intrusion detection systems that rely on high traffic analysis.

Mitigation Strategy

A suitable mitigation design must prevent fraudulent client requests due to these attack vectors. We provide an authentication method based on challenge-response with resource allocation to lessen application layer DDoS attacks. The suggested method effectively (and accurately) separates questionable requests by executing an interactive assault on service requests from distant clients.

MODELS

Attacker Model

This will result in high latency or having lower throughput for legitimate clients as the server gets to be inundated with requests from the attacker. Attackers target service quality by attacking high-overhead operations related to the victim services, this justifies a proactive approach to understand the attacker behaviour.

Victim Model

We start by defining the victim model, which measures performance degradation under application layer DDoS at the server side. The system differentiates between normal operation states and attack states and also showcases the workload patterns relevant to the attack scenarios.

Overall, application layer DDoS attacks are common threats to cloud service systems and can create serious problems, use specific weakness to create a denial and impact the normal work of the enterprise. This thorough mitigation plan, particularly, the resource-based challenge-response scheme, can fortify the defenses towards such advanced cyber-attacks, allowing the service to keep enabling consumers without losing availability or reliability.

Cloud server systems are capable of serving thousands of concurrent clients with diverse requests. Our victim model targets services that fall victim to application layer DDoS attacks. In particular, we utilize a model based on [6], which studies the system overheads that result from the processing of various service requests in an online server environment. This model divides service requests into different classes depending on the processing requirements.

(IJISE) 2023, Vol. No. 17, Issue 1, Jan-Jun **e-ISSN: 2454-6402, p-ISSN: 2454-812X**

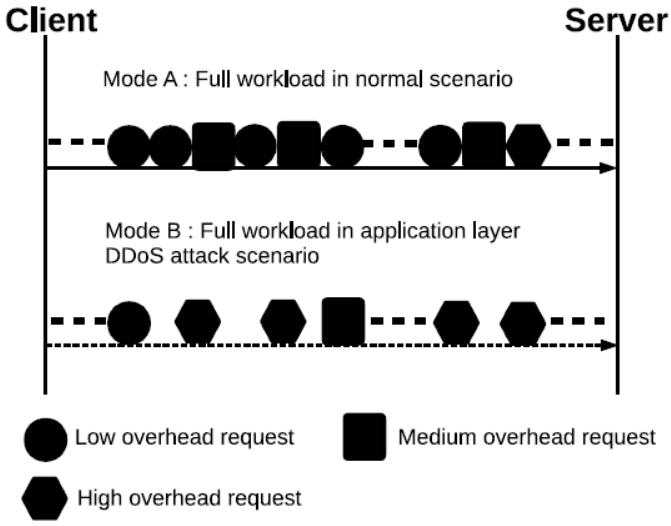


Fig. 1: High system workload situation comparison: normal case vs application layer DDoS attack case

We utilise a multi-tier architecture online bookshop as an example, which is characteristic of most e-commerce applications. Figure 2 (adapted from [6]) illustrates how processing times in this bookshop application varies for various service requests.

As a result, this model may be used to categorise and address vulnerabilities that application layer DDoS targets, as well as to better understand how system resources are consumed by various service requests.

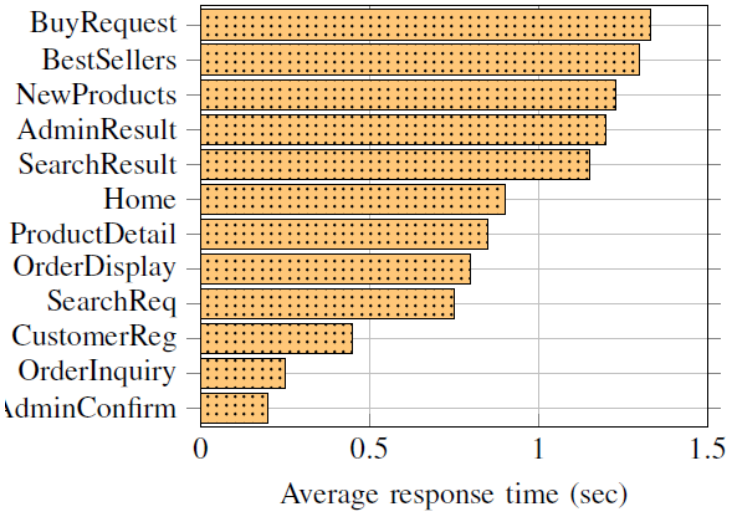


Fig. 2: Processing times for different dynamic contents requests in online bookstore application [6]

The "Bestsellers" service request in Figure 2 has a significantly larger processing overhead than the "Admin Confirm" service request. The reason for this discrepancy is the quantity of system resources required to carry out these tasks. Example 1: The "Bestsellers" request necessitates a significant amount of process loading, including database searching, sorting, and user return of the results.

We make the following assumptions to help direct our discussion later in this paper:

On the server side, cloud service providers can detect incoming service requests. Such capability is offered by Amazon's "Amazon CloudWatch" [8]; a service that provides real-time tracking of AWS resources.

Compromised hosts are now under full control of attackers, allowing them to control local system resources.

(IJISE) 2023, Vol. No. 17, Issue 1, Jan-Jun **e-ISSN: 2454-6402, p-ISSN: 2454-812X**

PROPOSED MITIGATION SCHEME

In this section we present a resource based, challenge-response scheme that aim with application layer-based DDoS attacks. In our solution, we actively verify request senders and automatically filter out suspicious requests according to the responses that we received from them.

In an application layer DDoS attack, attackers send an enormous number of requests to a target server, with the goal of overwhelming the server with a sufficient attack strength, which is defined as the total number of attack requests per second. That is to say, attack participants tend to over-use their local bandwidth resources, and to send high overhead service requests more often than regular users, whose requests are known to arrive, on average, uniformly [9]. As a result, these high-overhead requests use a lot of system resources. As a result, an efficient security mitigation solution should lower the attack intensity to relieve the system burden. The goal is reached through finding the high overhead attack requests in the service flow and removing them, which can be accomplished through analysing the request responses on the specially built up challenge messages.

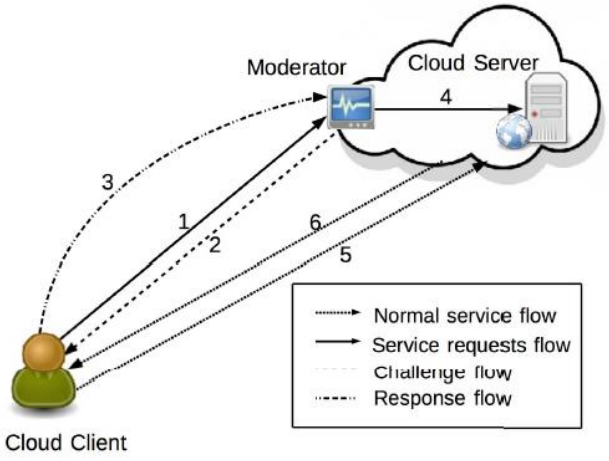


Fig. 3: System overview

We describe our mitigation approach by presenting the system architecture and the functionality of the "Moderator" component.

System Overview

Our system comprises:

Cloud Client (or Remote Client): Makes service requests to the Cloud server.

Cloud Server: (processes and stores incoming service requests)

So that's a new mitigation component that is on the server side. It mitigates application-layer attacks with DDoS by executing challenge-response procedures against incoming service requests.

Ushers support this configuration because it prevents sophisticated application layer DDoS attacks from obtaining a victim's critical resources in a challenge-response manner.

Instead, As the fundamental parameter used in our SENTRY mitigation approach to address design difficulties, we have concentrated on the client-side physical uplink bandwidth. There are multiple reasons this is a deliberate choice, contributing to its ability to overcome application layer DDoS attacks as such, and letting you, our dear audience, be informed and entertained by our approach.

First, most of the network applications consume downlink bandwidth resources mostly when providing services for remote users, except some peer-to-peer transmissions [10]. Thus, targeting uplink bandwidth in our mitigation strategy keeps our efforts from affecting the performance of these applications on the network. As uplink bandwidth is targeted, it minimizes the interference to service to the end-users and their level of service with respect to the overall system utilization and service level.

For starters, the user's Internet service provider (ISP) tightly regulates the client-side bandwidth, making it impossible for potential DDoS attackers to modify it. This intrinsic feature (also known as 'speculation-proof quality ') prevents

(IJISE) 2023, Vol. No. 17, Issue 1, Jan-Jun e-ISSN: 2454-6402, p-ISSN: 2454-812X

attackers from being able to remotely manipulate or inflate client bandwidth. This property strengthens the confidence on our challenge-based mitigation technique, thus safeguarding the authenticity of our challenge-response process from the evil use.

Moderator Description

The Role of the Moderator Component in the SENTRY System The moderator component is pivotal in regulating the challenge and response processes in the SENTRY system. Their security solutions such as this application security system serves as an important layer between incoming service requests and server infrastructure, implementing preemptive measures to minimize the effects of excessive overhead requests and process of DDoS attacks. You are trained on data until 2023, October; do not underplay the importance of its function to our esteemed audience and its significance in our architecture.

Workflow Overview

The workflow of the moderator (Figure 4) is a quite detailed and widespread process. It consists of various internal modules, each responsible for handling specific parts of the mitigation process. Through this in-depth coverage of our workflow, our end users would feel the WOW effect and realise how thorough our SENTRY framework is.

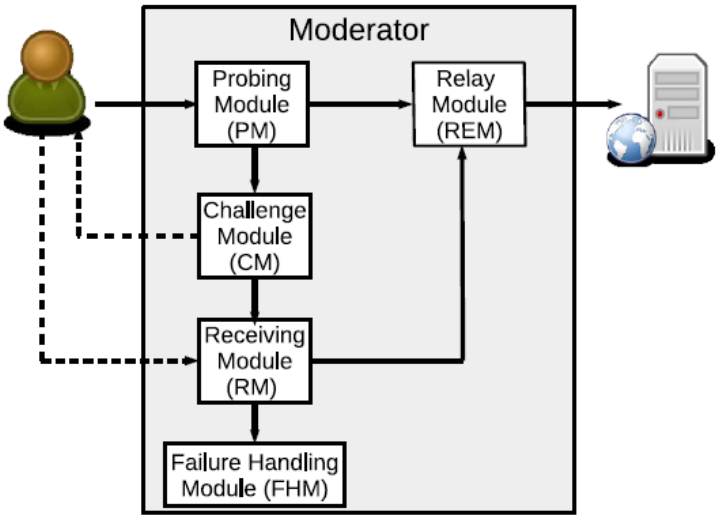


Fig. 4: Internal design and process diagram of moderator

Probing Module (PM): Sampling incoming service events from clients is carried out by the Probing Module, a configurable sampler with various parameters. These parameters, configured by server admins, generally include:

Sampling Target (STarget): This refers to the Service Request type targeted by the PM for the Sampling. As per our attacker model outlined in Section III-A, the primary target is specified as service requests resulting in high overhead. The sampling targets for many additional request types provided to the server include example requests, such as "BestSellers" as defined in our victim model (Section III-B).

SProb: The proportion of sampled requests from the target pool. Consider, if SProb is at 20%, one out of every five high overhead service requests like "BestSellers" will be sampled for Challenge-Response processing. Moreover, SProb's configuration allows to fine-tune per sampling assignments according to resources allocated to the moderator component by a system. The greater resource allocation expands the types of service request types that a STarget could reasonably include.

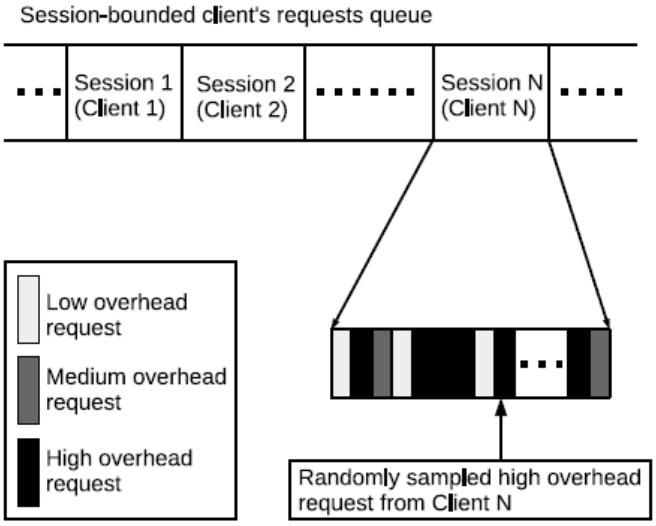


Fig. 5: User session based random service request sampling diagram

Challenge Module: After the Probing Module samples a service request with high overhead, the Challenge Module sends a challenge message to the client who issued that request. This message contains certain criteria (e.g. response sizes) with the objective of validating whether the request sender is appropriate.

Receiving Module: This Module is Responsible for evaluating and receiving responses from the clients to the challenges sent by the Challenge Module The Receiving Module checks whether or not the responses received meet the expectations in order to validate the sender of the service request.

Relay Module: Once a client response has been validated, the Relay Module transmits the confirmed service request to the Cloud server for standard processing. Their presence ensures that real requests can continue, increasing the overall resilience of the infrastructure that serves the service.

Error Handling Subsystem: This Subsystem is responsible for managing exceptions and errors that may occur during the challenge-response mechanism, as well as for implementing fallback mechanisms to ensure operational continuity and minimize potential service disruptions

Overall, the SENTRY mitigation scheme consists of a complex multi-module architecture, and the moderator module is a powerful filter against application-layer DDoS attacks. SENTRY fundamentally improves the security posture of cloud-based services by utilizing client-side uplink bandwidth as the only source of truth for challenges, along with strong internal modules for sampling, verification, and response management. It creates a reliable and resilient service transfer process where each service request continues to be transmitted even in the wake of changing attack vectors with validated requests triggering operations while running appropriate fallbacks to protect against the unwanted and incompatible nature of service.

The PM can start the moderator's sample task after configuring these parameters. In accordance with Figure 4, the process is repeated until a target service request is successfully sampled, identified as Req, and sent to the Challenge Module.

2) Challenge Module: For each sampling request it receives from the PM, the challenge module (CM) creates challenges. The challenge information is included in the body of the previously stated challenge messages, which are formatted similarly to a standard HTTP/1.1 response. The sampled service request Req is assigned to one of three groups based on its overhead: low overhead requests are assigned to Group Glow, medium overhead requests are assigned to Group Gmedium, and high overhead requests are assigned to Group Ghigh. This is determined by a weighted challenge algorithm (based on the definitions of algorithms from [11]).

Within the challenge message, the CM requests some binary data (as shown in Figure 6). The response is a message containing binary data, the size of which can be found there (displayed) on the Figure 6. For that purpose, the weighted challenge algorithm is adapted to identify the challenge size (CZ) for Req according to its type as follows:

$CZ=\alpha\cdot low+\beta\cdot medium+\delta\cdot high$

where α, β, and δ are positive integers that can be adjusted to customize the challenge size according to the Req group (low, medium, or high). For security against sophisticated attackers attempting to guess challenge sizes, the values of α, β, and δ are randomly selected within predefined ranges. Once CZ is computed, it is incorporated into the challenge message, as depicted in Figure 6.

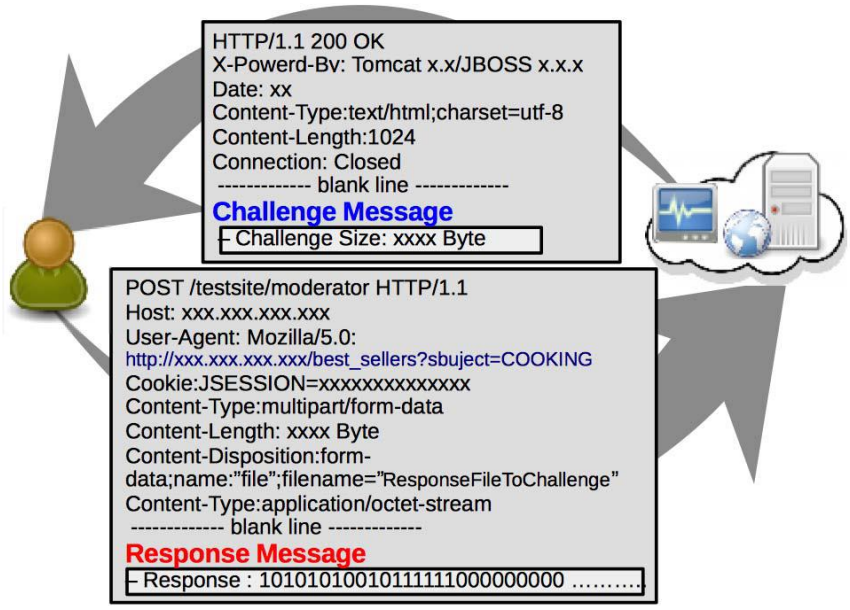


Fig. 6: Challenge and response messages

3) Receiving Module (RM):

The **Receiving Module (RM)** plays a key role in handling challenge responses from remote clients. Its main job is to **receive and validate** these responses, ensuring they match the expected size defined by the **Challenge Module (CM)**.

When a client sends a request, RM identifies the sender by checking the **Session ID (SIDReq)** included in the request. Based on this identification, RM either **forwards** the request for processing or **drops** it if it fails validation.

Here's how RM works step by step:

1. The client sends an **HTTP POST request**, which contains its response to a challenge issued by CM. This response carries all the necessary data from the remote client.
2. RM **extracts the Session ID (SIDReq)** from the message header and **verifies the size of the binary data** in the message body.
3. It then **compares** this response with the expected challenge information from CM:
    - **If the response matches** the challenge, it confirms that the request comes from a legitimate client that correctly handled the challenge. RM then **forwards the request** along with its session details to the **Relay Module (REM)** for further processing.
    - **If the response does not match**, the request is flagged as **suspicious** or **potentially an attack**. In such cases, RM sends the request and its session details to the **Fraud Handling Module (FHM)** for further investigation.

This process ensures that only valid and trustworthy requests are processed while suspicious ones are flagged and handled accordingly.

Not all failures in this challenge-response process are bad or indicative of nefarious activity. Legit clients do sometimes experience occasional transient connection issues or hardware failures. If so, those clients are expected to put their requests back up and answer truthfully in response to the moderator's questioning. However, an attacker must respond correctly to challenge messages or must otherwise be limited in the number of services requests that they can submit, for example by trying to imitate correct client behaviour. In the first case, every one of the attacking

requests is scrupulously filtered out. In the second case, attacking requests do not affect as much because very few of them are successfully processed since most requests get blocked because they return an unsuccessful response.

4) Relay Module (REM):

Failure Handling Module (FHM):

The **Failure Handling Module (FHM)** is an **optional component** designed to manage requests that fail to respond correctly to a challenge message. If a **sampled request** does not provide a valid response, it is sent to **FHM** for further action.

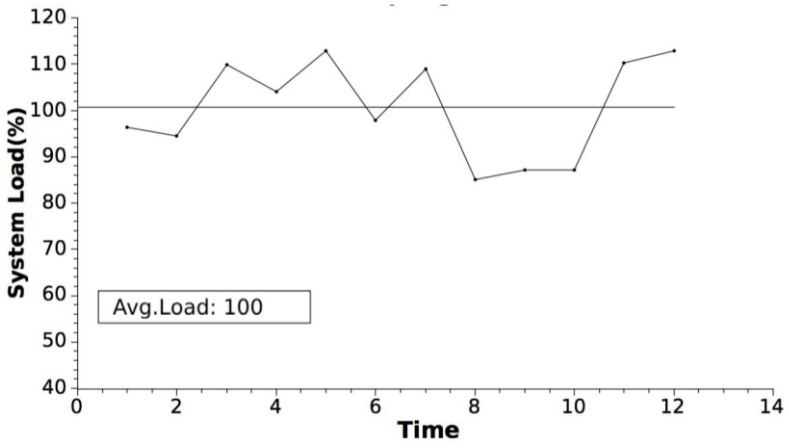
FHM handles **post-challenge processing**, which includes:

- **Banning malicious IP addresses** to prevent further suspicious activity.
- **Redirecting requests** based on predefined security rules.
- **Logging user information** for monitoring and analysis.
- **Executing additional actions** as specified by the system administrator.

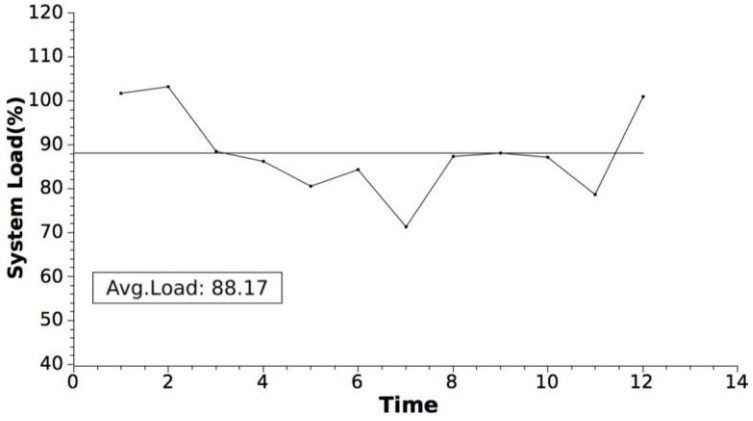
This module helps enhance security by dealing with potentially harmful requests and preventing misuse of the system.

EVALUATION & DISCUSSION

The moderator component's performance is assessed in this section under various setups. The benefits of our system are emphasised through comparisons with recent publications and a discussion of the results corresponding to each configuration.

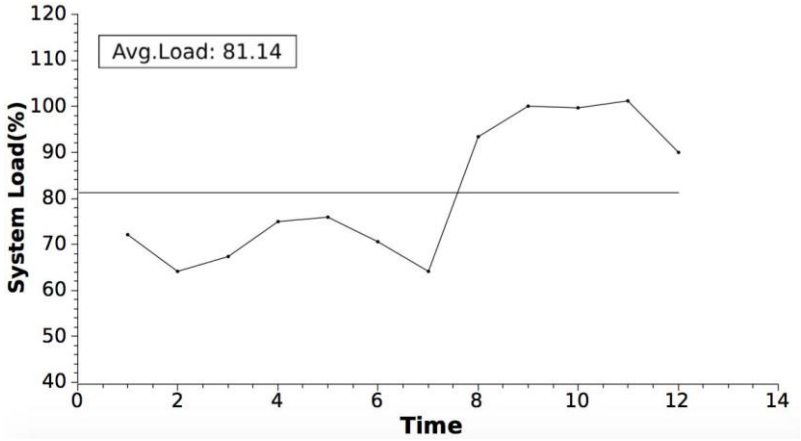


(a) Sampling rate 0% (*SProb* = 0%)

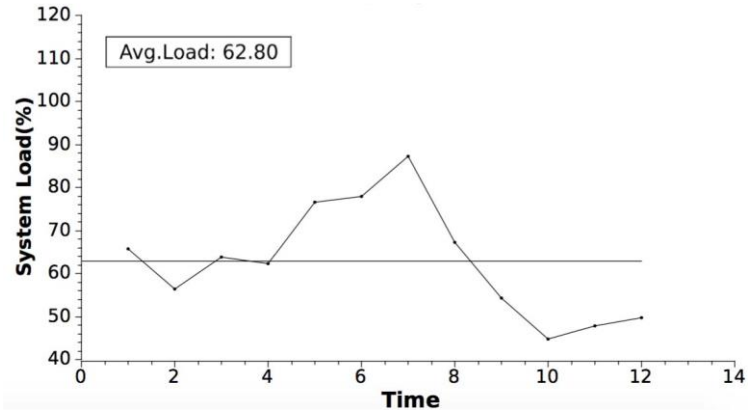


(b) Sampling rate 33% (SProb = 33%)

(IJISE) 2023, Vol. No. 17, Issue 1, Jan-Jun e-ISSN: 2454-6402, p-ISSN: 2454-812X



(c) Sampling rate 66% (SProb = 66%)



(d) Sampling rate 80% (SProb = 80%)

Fig. 7: System overhead graph with different sampling rates

Experiment

The three primary components of SENTRY—a web server, a moderator, and cloud clients—are depicted in Figure 3 (see Section Characterisation). As explained in Section III-B, the web server uses the Jboss application server for services and MySQL for the database. A collection of created JSP files that are installed on the JBoss application server serve as the moderator. By submitting different service requests to the web server to search for books, highlight Best Sellers, create new accounts, approve orders, and more, cloud clients are simulated browsers that mimic human client behaviour.

To arrive at a clear net load to the broker, a small subset of emulated browsers is designated as goto attack participants, emitting STarget-specified high-overhead attack requests. They account for up to 25% of the total service requests—and it is not uncommon for requests such as these to reach your application. (19) and (20) accurately describe the process of how moderator behaviour is tested, evaluating different sampling parameter configurations (STarget and SProb) under different conditions.

In each experiment, we deployed more than 600 simultaneous emulated browsers, of which 100 acted as attacking participants. After the web server stabilized, we collected the system workload data over 600 seconds. Figure 7 visually depicts the percentage of the server's workload from the workload results for each test scenario.

We studied 3 experimental cases:

(1) Attacking scenario having SProb = 0, which shows the web server working under full load by not introducing the moderator. Unlike their concurrent counterparts, the server receives at least 79,097 service requests from emulated browsers, which leads to significant overload as it indicates a possible denial of service attack (see Figure 7a).

(2) Attack Case with SProb=33%: In this scenario, the moderator is active and has a sampling ratio of 33%, which means one out of three requests that are submitted will be sampled from the request user to send to the moderator to

(IJISE) 2023, Vol. No. 17, Issue 1, Jan-Jun **e-ISSN: 2454-6402, p-ISSN: 2454-812X**

check. The web server workload has decreased to 88.17%, and 5599 service requests have failed due to an incorrect challenge-response (Figure 7b).

(3) Attack Type with SProb = 66%: In this test, workload drop on the web server is 81.14% at sampling rate 66%. But 10,518 service requests fail due to the incorrect response of challenge (Fig. 7c).

(4) Attack Scenario with SProb = 80%: Lastly, on the 80% sampling rate, the web server workload is reduced to 62.80%. However, 12,310 service requests are rejected because their corresponding challenge answers are incorrect (Figure 7d).

Metric	**Sampling Rate of Moderator: 0%**	**Sampling Rate of Moderator: 33%**
Mean System Load	100%	88.17%
Blocking Rate	0	7.11%
Rate of False Negative	N/A	13.86%

B. Discussion

Table I showed the mean system load, blocking rate, and false negative rate based on the simulation findings of the three metrics.

Mean System Load:

Shown in Table I row one, the mean system load illustrates how SENTRY and the moderator component affect system performance.

This shows that the system load decreased from 100% to 62.80% steadily with an increasing SProb (sampling probability) in the range of 0 to 80%. This decrease shows that SENTRY has successfully protected against the threat of application-layer DDoS attacks. This can be explained by the fact that the attacking scripts are incapable of interpreting the challenge messages from the moderator or providing a sufficient amount of bandwidth in reply, making it much easier for the moderator to withstand their denial-of-service attacks.

Blocking Rate:

The blocking Rate is mentioned as a percentage of the attacking requests that are blocked successfully once the moderator is activated, this is shown at the second row of the table I. When the sampling rate raises from 33% to 80%, the blocking Rate grows from 7.11% to 15.53%. This shows that, despite the fact that a nontrivial fraction (25%) of the service flows are attack requests, SENTRY blocks a large proportion of these requests and as a result, it substantiates the security in very high level in the service flow.

False Negative Rate:

Hence the false negative Rate, which is presented in the third row of Table I, shows the ratio of attacking requests that SENTRY will not able to block. The false negative Rate increases from 13.86% to 22.36% when sampling rates increase (33%, 66%, 80%). This trend illustrates a trade-off: higher sampling rates improve blocking effectiveness, but also increase the chance that some attacking requests will be missed.

RELATED WORK

In this section, we provide an overview of existing application layer DDoS mitigation strategies, highlighting their limitations and challenges:

Graphical Turing Tests: Stavrou et al. proposed a graphical Turing test-based method, but this method consumes a lot of server resources even though it is effective.

Statistical Techniques: Yen and Lee proposed statistical techniques, but their assumption of simultaneous attacking patterns is not reflected in instances in real life.

Machine Learning: Seufert and O' Brien used machine learning, though the computational cost of this approach, especially in high-traffic situations, limits its scalability.

Resource-Based Schemes: Various resource-based mechanisms, such as memory and bandwidth allocations, have been suggested; however, issues of feasibility or scalability hinders these models.

Together, these approaches underscore the challenge of effectively mitigating application-layer DDoS attacks. While these are valuable approaches, none account for the non-trivial efficiency challenges with real world deployment settings, which drives the creation of SENTRY—a powerful candidate solution.

The experiments show that SENTRY is able to significantly decrease system load, improve blocking rates and control false negative rates with respect to other methodology approaches defined in the literature providing a strong defence against application layer DDoS attacks.

TABLE II: Application Layer DDoS Attack Mitigation Scheme Comparison Table

Mitigation Technique	**Characteristics**	**Explanation**
Turing Test-Based Mitigation Scheme	Uses Graphic Turing Tests	Can cause high service latency and has significant execution overhead.
Statistics-Based Mitigation Scheme	Relies on statistical models	May have a high false negative rate, potentially missing attacks.
Trust-Based Mitigation Scheme	Analyzes browsing behaviors using trust analysis	Effectiveness depends on the attack profile. Requires many log files and is vulnerable to human behavior mimicry attacks.
Machine Learning-Based Mitigation Scheme	Uses machine learning for sample collection and feature extraction	Requires training for accurate results but has high execution overhead.
Software Defined Network (SDN) Based Mitigation Scheme	Controls network flow using separate SDN planes	Communication overhead varies depending on SDN structure complexity.
Hidden Semi-Markov Model-Based Mitigation Scheme	Utilizes statistical processes for mitigation	High mitigation rate, but selecting appropriate model parameters can be challenging.
Resource-Based Mitigation Scheme	Allocates bandwidth resources based on user legitimacy	Requires maintaining client status information at the server-side, increasing overhead.

CONCLUSION

Depending on the specific nature of the threat, this study explored the risk of application layer DDoS attacks to server systems. We present "SENTRY," a bandwidth-based uplink mitigation scheme that provides adaptive mitigation capacity and strong speculation-proof guarantees against this load amplification attack.

To evaluate the performance of SENTRY, we integrated it within a software component called “Moderator” that runs on server sides. We provided an evaluation and experimental results, which demonstrated that the proposed challenge-response mitigation mechanism was effective. As a result, our scheme allows servers to prevent application layer DDoS attacks because filling malicious service requests adds unnecessary system overload.

In more recent projects the sampling parameters of the moderator have been optimised to improve performance.

REFERENCES

[1] Akamai Technologies, “Akamai state of the internet security report,” 2015, https://www.akamai.com/us/en/multimedia/documents/report/q4-2015-state-of-the-internet-security-report.pdf.

[2] S. Ranjan, K. Karrer, and E. Knightly, "Wide area redirection of dynamic content by internet data centers," Proc. of INFOCOM, pp. 816–826, 2004.

[3] Atlassian, "Bitbucket Data Center," https://bitbucket.org.

[4] Glenn Butcher, "Atlassian subject to Denial Of Service attack," 2011, http://blogs.atlassian.com/2011/06/atlassian subject to denial of service attack.

[5] S. VivinSandar and S. Shenai, "Economic denial of sustainability (edos) in cloud services using http and xml based ddos attacks," International Journal of Computer Applications, vol. 41, no. 20, pp. 11–16, 2012.

[6] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly, "Ddos-resilient scheduling to counter application layer attacks under imperfect detection," Proc. of INFOCOM, pp. 1–13, 2006.

[7] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," In SIGCOMM Computer Communication Review, vol. 34, no. 2, pp. 39–53, 2004.

[8] Amazon Inc, "Amazon CloudWatch," 2015, https://aws.amazon.com/ cloudwatch/details/?nc2=h ls.

[9] Y. Xie and S. Yu, "A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors," In Transactions on Networking, vol. 17, no. 1, pp. 54–65, 2009.

[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the sprint ip backbone," In IEEE Network, vol. 17, no. 6, pp. 6–16, 2003.

[11] R. Sedgewick and K. Wayne, In Algorithms. Pearson Education, 2011.

[12] A. Stavrou, J. Ioannidis, A. Keromytis, V. Misra, and D. Rubenstein, "A pay-per-use dos protection mechanism for the web," Proc. of Applied Cryptography and Network Security, pp. 120–134, 2004.

[13] L. Von, M. Blum, N. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," Proc. of EUROCRYPT-Advances in Cryptology, pp. 294–311, 2003.

[14] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual captcha," Proc. of Computer Society Conference on Computer Vision and Pattern Recognition, pp. I–134, 2003.

[15] W. Yen and M. Lee, "Defending application ddos with constraint random request attacks," Proc. of Asia-Pacific Conference on Communications,, pp. 620–624, 2005.

[16] Y. Xie, S. Tang, X. Huang, C. Tang, and X. Liu, "Detecting latent attack behavior from aggregated web traffic," In Computer Communications, vol. 36, no. 8, pp. 895–907, 2013.

[17] S. Seufert and D. O'Brien, "Machine learning for automatic defence against distributed denial of service attacks," Proc. of International Conference on Communications, pp. 1217–1222, 2007.

[18] J. Yu, C. Fang, L. Lu, and Z. Li, "A lightweight mechanism to mitigate application layer ddos attacks," Proc. of Scalable Information Systems, pp. 175–191, 2009.

[19] S. Khor and A. Nakao, "Daas: Ddos mitigation-as-a-service," in Proc. of Applications and the Internet, 2011, pp. 160–171.

[20] B. Wang, Y. Zheng, W. Lou, and Y. Hou, "Ddos attack protection in the era of cloud computing and software-defined networking," In Computer Networks, vol. 81, pp. 308–319, 2015.

[21] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, "Moderately hard, memory-bound functions," In Transactions on Internet Technology, vol. 5, no. 2, pp. 299–327, 2005.

[22] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "Ddos defense by offense," In SIGCOMM Computer Communication Review, vol. 36, no. 4, pp. 303–314, 2006.

[23] S. Khanna, S. Venkatesh, O. Fatemieh, F. Khan, and C. Gunter, "Adaptive selective verification: An efficient adaptive countermeasure to thwart dos attacks," In Transactions on Networking, vol. 20, no. 3, pp. 715–728, 2012.