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 renowned for their touted benefits, such as seamless resource access, scalability, and elasticity. However, they also face significant challenges from various threats at both infrastructure and application levels, with application-layer distributed denial of service (DDoS) attacks posing challenging problems to counter. These attacks typically overwhelm targeted servers, causing performance degradation and service unavailability by exhausting available resources.

While some existing solutions like intrusion detection and protection can mitigate specific attacks, evolving application-layer DDoS attacks often find ways to evade these defences. In response, this paper introduces SENTRY, a novel and efficient methodology designed to combat application-layer DDoS attacks. SENTRY employs a challenge-response strategy that (a) assesses attackers' physical bandwidth resources, (b) dynamically adjusts to varying workload conditions, and (c) blocks suspicious service requests from potentially malicious clients.

INTRODUCTION

Distributed denial of service (DDoS) attacks presents a significant security challenge for Cloud service providers. They overwhelm victim servers and lead to degraded services. Typically, DDoS attacks target the transport layer (layer 3) and network layer (layer 4) of the OSI TCP/IP stack, flooding network interfaces with malicious traffic to exhaust resources and disrupt legitimate traffic.

While network-layer attacks remain a formidable challenge due to their scale, application-layer DDoS attacks pose a growing and complex long-term threat [1]. These attacks exploit web applications' increasing complexity and bandwidth availability [2], depleting resources gradually rather than flooding them, as seen in a notable incident affecting Bitbucket Data Center, which experienced intermittent service disruptions lasting over 12 hours [4]. The proliferation of web applications and limited effective mitigation strategies make them prime targets for attackers.

Application-layer DDoS attacks differ from traditional DDoS attacks in several ways. For example, they generate less network traffic, impose higher system overhead per request on servers, and are more adept at evading intrusion detection systems [1].

To address these challenges, we propose SENTRY, a novel security mitigation scheme tailored for application-layer DDoS attacks. SENTRY leverages remote users' local uplink bandwidth to dynamically assess request legitimacy and mitigate resource flooding caused by these attacks. By implementing a challenge-response mechanism based on uplink bandwidth, SENTRY aims to:

- Reduce configuration burdens by operating at the middleware/protocol level, abstracting lower-level network complexities, and facilitating streamlined deployment in Cloud environments.

- Adapt to diverse workload conditions encountered by servers.

- Thwart deceptive tactics by employing a physical bandwidth-based challenge-response process to block suspicious service requests from dishonest clients.

[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

Our evaluation demonstrates SENTRY's efficacy in mitigating application-layer DDoS attacks across various practical scenarios.

BACKGROUND

The application layer DDoS attack represents a sophisticated method that surreptitiously exhausts resources on targeted servers. In contrast to traditional network layer DDoS attacks, application layer DDoS attacks exhibit three primary characteristics:

Application Layer DDoS Attacks: Characteristics and Mitigation

Attack Characteristics

Application layer DDoS attacks manifest as workload-enhancing assaults that aim to deny service by depleting critical resources such as CPU cycles, I/O, memory, and network bandwidth. Despite the robust resource capacities of cloud server systems, specific resources can still bottleneck overall performance. For instance, Amazon EC2, renowned for its strong network capabilities, faced saturation due to XML and HTTP protocol-based application layer DDoS attacks [5].

These attacks are asymmetric, targeting specific application protocols with high-overhead services. Attackers exploit multiple client hosts to inundate target servers with a few selective but resource-intensive service requests, overwhelming them until they cease functioning [6].

Moreover, application layer DDoS attacks are stealthy, camouflaging malicious requests as regular service requests to evade intrusion detection systems focused on abnormal traffic patterns. For example, authentication services are vulnerable to masquerading attacks that mimic legitimate service requests, consuming significant system resources [7]. Attributing such requests across numerous sources, attackers create minimal traffic deviations that evade high-traffic analysis-based intrusion detection systems.

Mitigation Strategy

Given these attack characteristics, an effective mitigation strategy must thwart dishonest client requests. Our proposed solution introduces a resource-based challenge-response scheme to combat application layer DDoS attacks. This scheme interactively challenges and verifies service requests from remote clients to identify and block suspicious requests effectively.

MODELS

Attacker Model

The attacker aims to overwhelm server resources, causing legitimate clients to experience high latency and reduced throughput. Attackers disrupt service quality by targeting high-overhead operations associated with victim services, necessitating a proactive approach to understanding their behaviour.

Victim Model

The victim model evaluates the performance impact of application layer DDoS attacks on server resources. It distinguishes between normal operational states and attack scenarios, highlighting the distinct workload patterns observed during such attacks.

In conclusion, application layer DDoS attacks pose significant challenges to cloud service systems, exploiting specific vulnerabilities to disrupt operations. Effective mitigation strategies like the resource-based challenge-response scheme can bolster defences against such sophisticated attacks, ensuring continued service availability and reliability.

Cloud server systems are designed to handle large numbers of concurrent clients requesting diverse services. Our victim model focuses on services susceptible to application layer DDoS attacks. Specifically, we adopt the model proposed in [6], which examines the varying system overheads incurred by processing different service requests within an online server environment. This model categorizes service requests into distinct classes based on their processing demands.

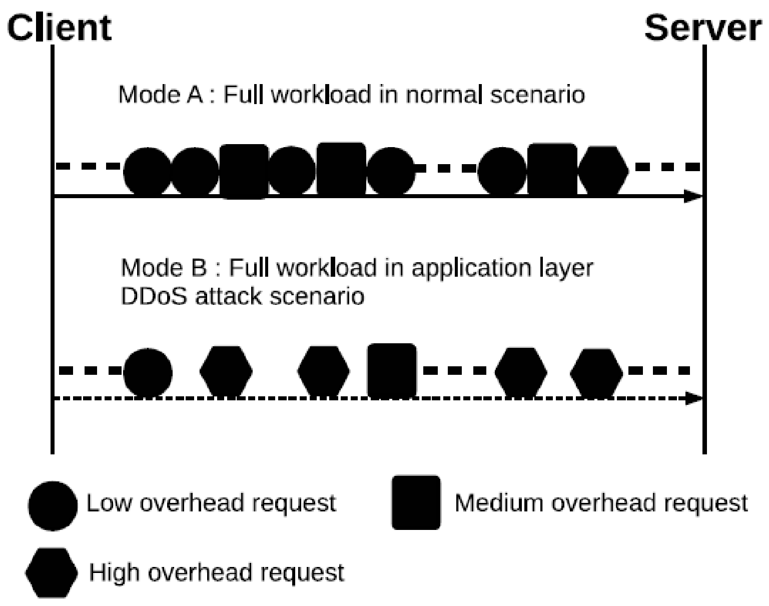

Client
Server
Mode A : Full workload in normal scenario
Mode B : Full workload in application layer DDoS attack scenario
Low overhead request
Medium overhead request
High overhead request

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

To illustrate, we consider an online bookstore operating on a multi-tiered architecture representative of typical e-commerce applications. Figure 2, adapted from [6], demonstrates how processing times fluctuate across different service requests within this bookstore application.

This model provides insights into the differential impact of various service requests on system resources, thereby aiding in identifying and mitigating vulnerabilities targeted by application layer DDoS attacks.

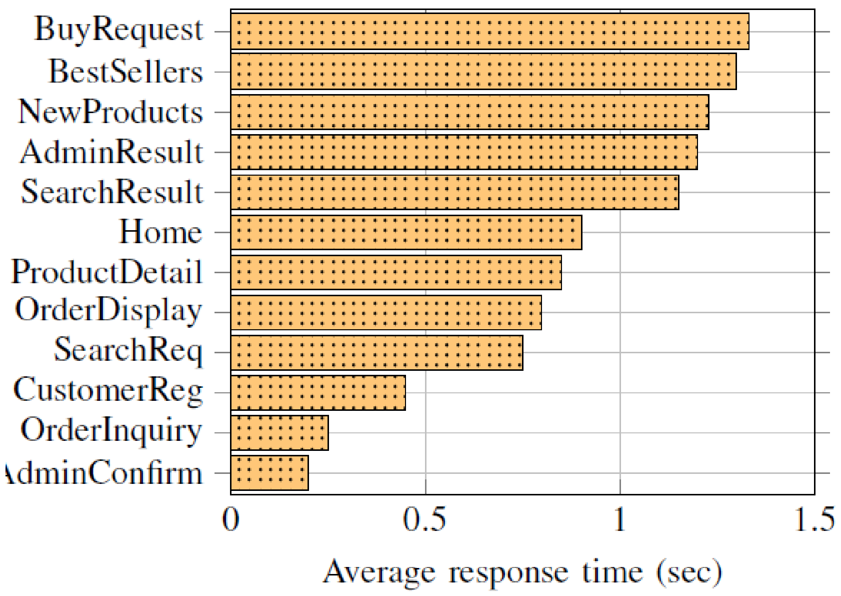


Fig. 2: Processing times for different dynamic contents requests in online bookstore application [6]

In Figure 2, the "Bestsellers" service request incurs significantly higher processing overhead than the "Admin Confirm" service request. This discrepancy arises due to the varying system resources required for these operations. For instance, the "Bestsellers" request involves resource-intensive tasks such as querying the database, sorting results, and delivering the outcome to the user.

To facilitate subsequent discussions in this paper, we establish the following assumptions:

1. Cloud service providers can monitor incoming service requests on the server side. Services like Amazon's "Amazon CloudWatch" [8] exemplify this capability, which provides real-time monitoring of AWS resources.

2. Attackers possess complete control over-exploited hosts, enabling manipulation of local system resources.

PROPOSED MITIGATION SCHEME

This section proposes a resource-based challenge-response scheme to mitigate application layer DDoS attacks. Our approach involves actively verifying the legitimacy of request senders and filtering out suspicious requests based on the responses received from them.

To execute an application layer DDoS attack, attackers inundate a target server with a high volume of requests, aiming to overwhelm it with sufficient attack strength—defined as the aggregate rate of attack requests per second. Attack participants typically maximize their local bandwidth resources, sending high-overhead service requests more frequently than regular users, whose requests commonly exhibit a uniform arrival rate [9]. Consequently, these high-overhead requests lead to excessive consumption of system resources. Therefore, an effective security mitigation solution must reduce the attack strength to alleviate system overhead. This is achieved by identifying and discarding high-overhead attack requests from the service flow, which is accomplished by examining request responses to specially generated challenge messages.

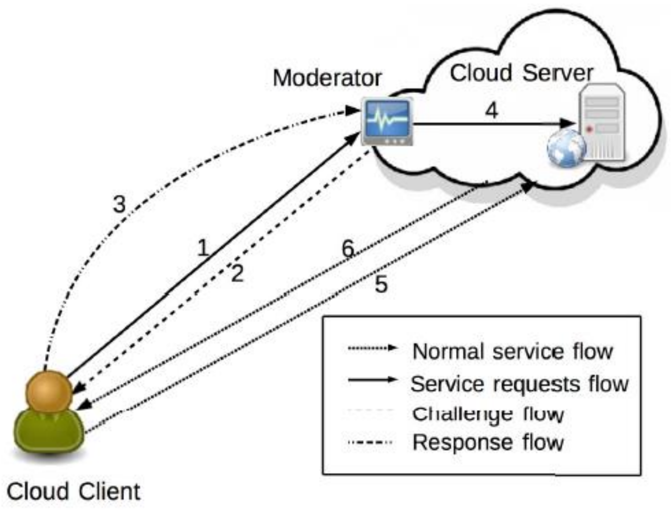


Fig. 3: System overview

We illustrate our mitigation scheme by outlining the system architecture and detailing the functionality of the "Moderator" component.

System Overview

Our system comprises:

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

- Cloud Server: Receives and processes incoming service requests.

Moderator: This is a novel mitigation component situated on the server side. It conducts challenge-response processes against incoming service requests to mitigate application-layer DDoS attacks.

This setup ensures that our mitigation strategy addresses the challenges posed by sophisticated DDoS attacks at the application layer, leveraging proactive challenge-response mechanisms to safeguard system resources and maintain service availability.

Our research has strategically leveraged the client-side physical uplink bandwidth as a foundational metric for designing challenges within our SENTRY mitigation scheme. This deliberate choice is backed by several key reasons that highlight its effectiveness in countering application layer DDoS attacks, keeping you, our esteemed audience, well-informed and engaged in our approach.

Firstly, most network applications predominantly utilize downlink bandwidth resources when delivering services to remote clients, with only a few exceptions, such as peer-to-peer transmissions [10]. Consequently, focusing on uplink bandwidth for our mitigation strategy ensures that our efforts minimally impact the performance of these network applications. By targeting uplink bandwidth, we mitigate the risk of interference with the smooth delivery of services to end-users, maintaining overall system efficiency and responsiveness.

Secondly, the client-side bandwidth is rigorously managed by the user's Internet Service Provider (ISP) and remains beyond the control of potential DDoS attackers. This inherent attribute, often referred to as 'speculation-proof quality ', means that attackers cannot manipulate or inflate client bandwidth remotely. This property enhances the reliability of our challenge-based mitigation approach, fortifying the integrity of our challenge-response mechanism against malicious exploitation.

Moderator Description

The moderator component plays a pivotal role in managing the challenge and response processes within the SENTRY framework. It acts as a crucial intermediary layer between incoming service requests and the server infrastructure, implementing proactive measures to mitigate the impact of high overhead requests and potential DDoS attacks. Its significance in our framework is not to be underestimated, underscoring the importance of its function to our esteemed audience.

Workflow Overview

The workflow of the moderator, as depicted in Figure 4, is a comprehensive and thorough process. It encompasses several internal modules, each designed to handle different facets of the mitigation process. This detailed overview of our workflow is aimed at making our esteemed audience feel the depth and breadth of our SENTRY framework, ensuring you are fully aware of its capabilities.

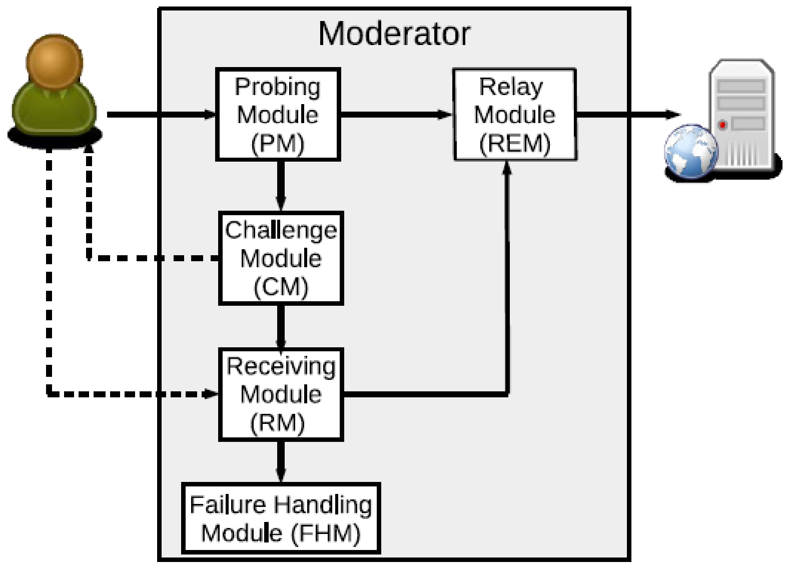


Fig. 4: Internal design and process diagram of moderator

1. Probing Module (PM): Responsible for sampling incoming service requests from clients, the Probing Module operates as a versatile sampler with configurable parameters. These parameters, adjusted by server administrators, include:

- Sampling Target (STarget): This specifies the type of service requests targeted for sampling by the PM. In our approach, high overhead service requests are designated as the primary STarget, aligning with our attacker model detailed in Section III-A. For instance, requests akin to "BestSellers", as outlined in our victim model (Section III-B), serve as the sampling targets amidst various other request types received by the server.

- Sampling Probability (SProb): This parameter determines the percentage of sampled requests from the targeted pool. For instance, if SProb is set at 20%, one of every five high overhead service requests (such as "BestSellers") is sampled for subsequent challenge-response processing. The configuration of SProb enables fine-tuning sampling rates based on system resource allocations for the moderator component. A higher allocation of resources broadens the scope of service request types that can be included within STarget.

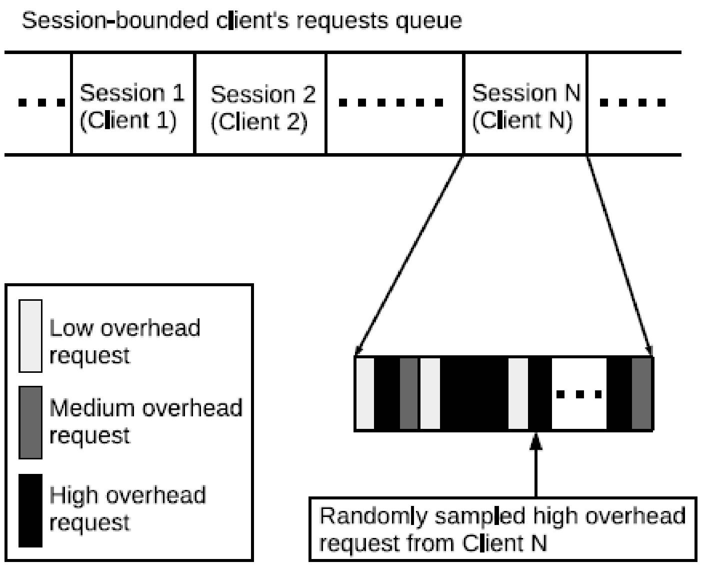


Fig. 5: User session based random service request sampling diagram

2. Challenge Module: Once the Probing Module samples a high overhead service request, the Challenge Module initiates a challenge message directed at the client associated with that request. This message includes specific criteria, such as expected response sizes, designed to validate the legitimacy of the request sender.

3. Receiving Module: This Module is Responsible for receiving and evaluating responses from clients to the challenges issued by the Challenge Module. The Receiving Module verifies whether the received responses align with the expected criteria, thereby determining the authenticity of the service request sender.

4. Relay Module: Upon successfully verifying a client's response, the Relay Module facilitates the transmission of the validated service request to the Cloud server for normal processing. This ensures that legitimate requests proceed without disruption, contributing to the overall resilience of the service infrastructure.

5. Failure Handling Module: This Module manages exceptions and errors encountered during the challenge-response process, implementing appropriate fallback mechanisms to maintain operational continuity and mitigate potential service disruptions.

In summary, the moderator component within the SENTRY mitigation scheme embodies a sophisticated, multi-module architecture designed to effectively safeguard against application-layer DDoS attacks. By leveraging client-side uplink bandwidth as a basis for challenges, coupled with robust internal modules for proactive sampling, verification, and response handling, SENTRY enhances the security posture of cloud-based services. It ensures reliable and uninterrupted service delivery amidst evolving cybersecurity threats by managing the challenge and response processes, facilitating the transmission of validated service requests, and implementing appropriate fallback mechanisms to maintain operational continuity and mitigate potential service disruptions.

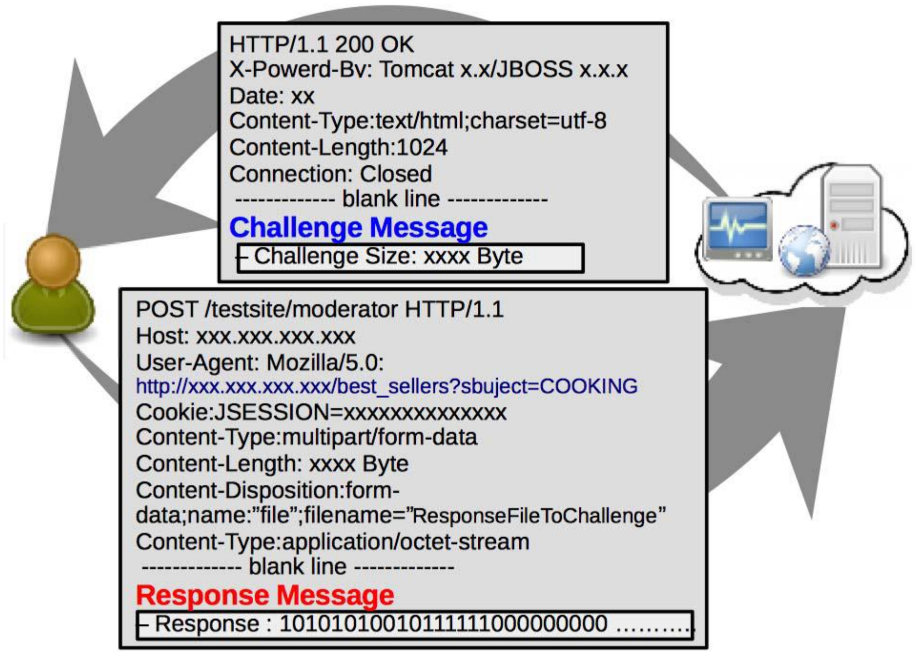
Once these parameters are set up, the PM can initiate the sampling task for the moderator. Upon completion of this process, the successfully sampled target service request (denoted as Req) is forwarded to the Challenge Module, as illustrated in Figure 4.

2) Challenge Module: The Challenge Module (CM) issues challenge messages for each sampled request received from the PM. These challenge messages conform to the standard HTTP/1.1 response format, embedding challenge information within the message body. To handle various types of service requests, a weighted challenge algorithm (based on algorithms specified in [11]) classifies the sampled service request Req into three main groups based on their overhead: Group Glow for requests with low overhead, Group Gmedium for medium overhead, and Group Ghigh for high overhead.

Within the challenge message, the CM requests a specific amount of binary data, specified in Figure 6. The client responds with a message containing binary data of the specified size (also detailed in Figure 6). The weighted challenge algorithm determines the challenge size (CZ) for Req based on its group, calculated 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.



```
HTTP/1.1 200 OK
X-Powerd-Bv: Tomcat x.x/JBOSS x.x.x
Date: xx
Content-Type:text/html;charset=utf-8
Content-Length:1024
Connection: Closed
 ------------- blank line -------------
Challenge Message
 Challenge Size: xxxx Byte
```

```
POST /testsite/moderator HTTP/1.1
Host: xxx.xxx.xxx.xxx
User-Agent: Mozilla/5.0:
http://xxx.xxx.xxx.xxx/best_sellers?sbuject=COOKING
Cookie:JSESSION=xxxxxxxxxxxxxx
Content-Type:multipart/form-data
Content-Length: xxxx Byte
Content-Disposition:form-
data;name:”file”;filename=”ResponseFileToChallenge”
Content-Type:application/octet-stream
 ------------- blank line -------------
Response Message
 Response : 1010101001011111000000000 .........
```

Fig. 6: Challenge and response messages

3) Receiving Module (RM):

The Receiving Module (RM) is responsible for receiving and validating the challenge response from the remote client. Its primary functions include verifying the response size to ensure it matches the expected challenge response size specified by the Challenge Module (CM). The RM identifies the sender using the client's Session ID (SIDReq) from the request and either forwards or discards the client's service request (Req).

Specifically, the RM receives an HTTP POST request message, which is the client's response to the challenge issued by CM. This response contains the necessary data submitted by the remote client. The RM extracts the client's session information (SIDReq) from the message header and checks the size of the binary data in the message body. It then compares this response with the challenge information issued by CM. If the response matches the challenge, Req is deemed to come from an honest client who correctly responds to the specified challenge size. Consequently, RM forwards Req and its session information (SIDReq) to the Relay Module (REM) for further processing.

Req is considered a suspicious or attacking request if the response does not match the issued challenge. In such cases, RM sends Req and its session information (SIDReq) to the Failure Handling Module (FHM), as depicted in Figure 4.

It's important to note that not all failures in this challenge-response process indicate malicious activity. Legitimate clients may occasionally experience transient connection issues or hardware failures. In such cases, clients are expected to resubmit their requests and respond correctly when challenged by the moderator. However, attackers must answer correctly to challenge messages or manage only a limited number of service requests if they attempt to mimic normal client behaviour. In the former case, all attacking requests are rigorously filtered out. In the latter case, the impact of attacking requests is significantly reduced since only a limited number are processed successfully, while most are blocked due to unsuccessful responses.

4) Relay Module (REM):

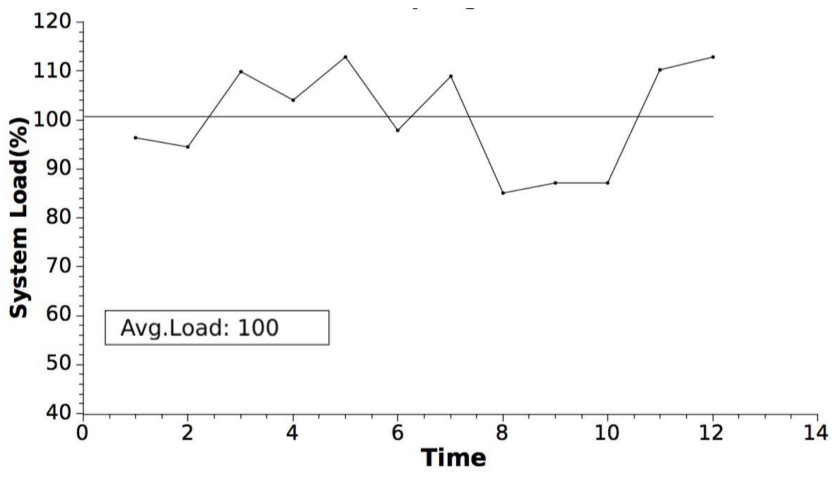
The Relay Module (REM) is the interface through which the moderator forwards sampled service requests (Req) to the server system. All non-sampled service requests are directly relayed to the server systems, as illustrated in Figure 4.

5) Failure Handling Module (FHM):

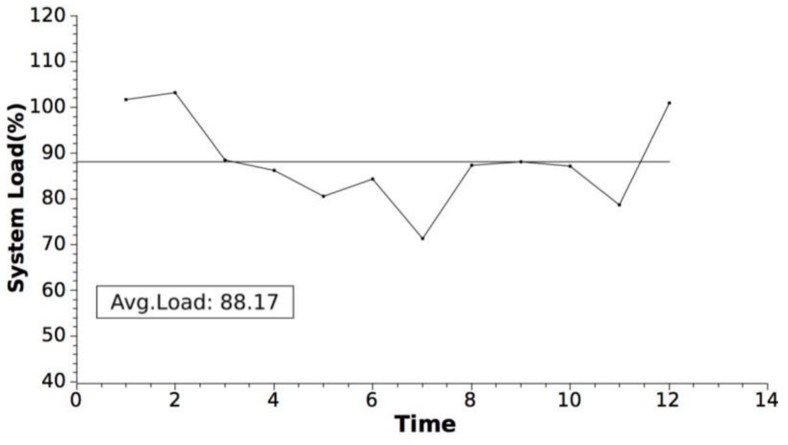
The Failure Handling Module (FHM) is an optional component in our design. When a sampled service request (Req) fails to respond correctly to the challenge message, it is routed to FHM. FHM is responsible for executing post-challenge processes, which may include intrusive IP banning, request redirection, user information logging, and other specified actions.

EVALUATION & DISCUSSION

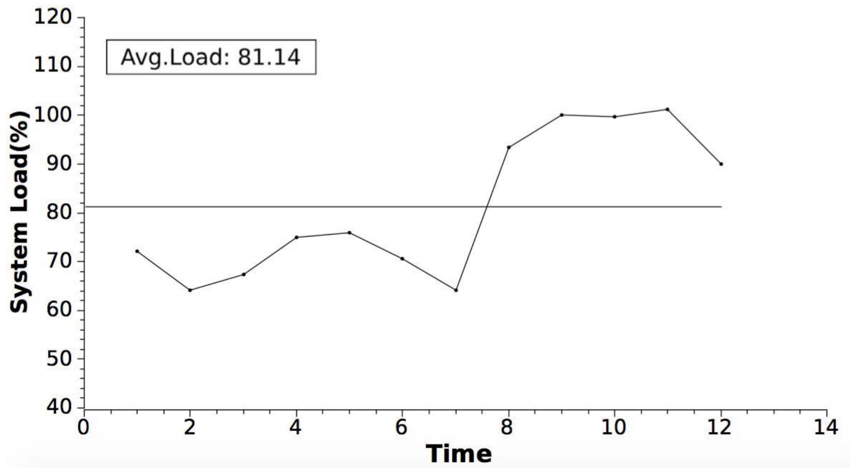
This section evaluates the performance of the moderator component under various configurations. Results corresponding to each setting are discussed, and comparisons with contemporary works highlight the advantages of our design.



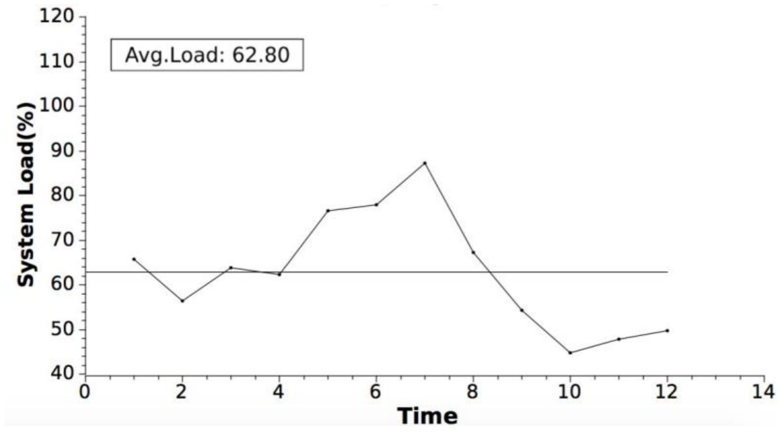
(a) Sampling rate 0% (*SProb* = 0%)



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



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



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

Fig. 7: System overhead graph with different sampling rates

Experiment

SENTRY comprises three key elements (cf., Figure 3): a web server, a moderator, and Cloud clients. The web server utilizes a Jboss application server and MySQL for database services, as Section III-B describes. The moderator consists of developed JSP files deployed on the JBoss application server. Cloud clients are emulated browsers that mimic human client operations by sending different service requests to the web server, such as searching for books, checking Best Sellers, registering new accounts, confirming orders, and more.

Additionally, a subset of emulated browsers is configured as attack participants, generating high-overhead attack requests specified by STarget. These attacking requests constitute up to 25% of the total service requests. Different configurations of sampling parameters (STarget and SProb) are tested to observe moderator behaviour under varying conditions.

We deployed over 600 concurrent emulated browsers in each experiment, with 100 acting as attacking participants. We collected system workload data over 600 seconds after the web server stabilized. The workload results are graphically presented in Figure 7, showing the server's workload percentage over time for each test scenario.

1) Attack Scenario with SProb = 0: This experiment demonstrates the web server operating at full load without moderator intervention. The server receives 79,097 service requests from emulated browsers, resulting in significant overload and indicating a potential denial of service attack (Figure 7a).

2) Attack Scenario with SProb = 33%: Here, the moderator is active with a sampling rate of 33%, meaning one-third of submitted requests are sampled and sent to the moderator for verification. The web server's workload decreases to 88.17%, with 5,599 service requests failing due to incorrect responses to challenge messages (Figure 7b).

3) Attack Scenario with SProb = 66%: In this experiment, the sampling rate is increased to 66%, resulting in a workload reduction of 81.14% on the web server. However, 10,518 service requests fail due to incorrect challenge responses (Figure 7c).

4) Attack Scenario with SProb = 80%: Finally, with a sampling rate of 80%, the web server's workload further decreases to 62.80%. Nevertheless, 12,310 service requests fail due to incorrect challenge responses (Figure 7d).

Data	**Moderator Sampling Rate**			
	0	33%	66%	80%
Mean System Load	100%	88.17%	81.14%	62.80%
Blocking Rate	0	7.11%	13.29%	15.53%
False Negative Rate	N. A	13.86%	19.47%	22.36%

Discussion

Three specific data types were collected from the experiments and presented in Table I: mean system load, blocking Rate, and false negative Rate.

Mean System Load:

The mean system load, depicted in the first row of Table I, indicates the moderator component's performance and the effectiveness of SENTRY. As SProb (sampling probability) increases from 0 to 80%, the system load steadily decreases from 100% to 62.80%. This reduction indicates that SENTRY successfully mitigates the threat posed by application layer DDoS attacks. The decrease in system load is attributed to the inability of attacking scripts to decipher the moderator's challenge messages and respond with appropriate bandwidth, thus limiting their effectiveness in generating service-denial attacks.

Blocking Rate:

The blocking Rate, detailed in the second row of Table I, refers to the percentage of attacking requests that are successfully blocked when the moderator is active. As the sampling rate increases from 33% to 80%, the blocking Rate rises from 7.11% to 15.53%. This demonstrates that even with a substantial proportion (25%) of service flows being attack requests, SENTRY effectively blocks a significant portion of these requests, thereby maintaining a high level of security in the service flow.

False Negative Rate:

The false negative Rate, discussed in the third row of Table I, indicates the proportion of attacking requests that SENTRY fails to block. With increasing sampling rates (33%, 66%, 80%), the false negative Rate escalates from 13.86% to 22.36%. This trend underscores a trade-off: while higher sampling rates enhance the blocking effectiveness, they also increase the likelihood of missing some attacking requests.

RELATED WORK

This section provides an overview of existing strategies for mitigating application layer DDoS attacks, emphasizing their limitations and challenges:

- Graphical Turing Tests: Stavrou et al. proposed a method based on graphical Turing tests, which, while effective, consumes substantial server resources.

- Statistical Techniques: Yen and Lee introduced statistical techniques, though their assumption of round-robin attacking patterns may not align with real-world scenarios.

- Machine Learning: Seufert and O'Brien utilized machine learning, but its computational expense limits scalability, especially under heavy traffic conditions.

- Resource-Based Schemes: Various resource-based schemes, such as memory and bandwidth allocations, have been proposed, but they often suffer from feasibility or scalability issues.

These approaches collectively highlight the ongoing challenge of effectively countering application-layer DDoS attacks. Despite their merits, none fully address the complexities of efficiency required in practical deployment scenarios, which motivates the development of SENTRY as a promising solution.

The experiments demonstrate that SENTRY effectively reduces system load, enhances blocking rates, and manages false negatives, offering a robust defence against application layer DDoS attacks compared to existing methodologies outlined in related literature.

TABLE II: Comparison Table of Application Layer DDoS Attack Mitigation Schemes

Mitigation Approach	**Characteristics**	**Comments**
Turing test-based mitigation scheme [12]	Graphic Turing Tests	High service latency High execution overhead
Statistic based mitigation scheme [15]	Statistics based Statistical model dependent	High false negative rate
Trust based mitigation scheme [18]	Trust analysis-based Analysing browsing behaviours	False negative rate depending on attacking profile huge amount of log files required Vulnerable to human behaviour mimic attacks
Machine learning based mitigation scheme [17]	Machine learning based Sample collection Feature extraction algorithm needs training	High execution overhead
Software Defined Network (SDN) based mitigation scheme [20]	SDN based Control network flow with separate planes	Communication overhead depends on SDN structure
Hidden semi-Markov model-based mitigation scheme [16]	Statistic processes based High mitigation rate	Difficulty in model parameter selection
Resource based mitigation scheme [22]	Bandwidth resource based Legal users get higher service possibility	Client status information required to maintain at the server side

CONCLUSION

This study delved into the threat posed by application layer DDoS attacks on server systems. We introduced "SENTRY," a mitigation scheme based on uplink bandwidth that offers flexible mitigation capabilities and robust speculation-proof properties to counteract this threat.

To assess the efficacy of SENTRY, we implemented it in a software component named "Moderator," deployed on server sides. Through rigorous evaluation, our experimental results underscored the effectiveness of the proposed challenge-response mitigation mechanism. Consequently, our scheme enables servers to thwart application layer DDoS attacks by minimizing unnecessary system overhead induced by processing malicious service requests.

Future endeavours will refine the moderator's sampling parameters to enhance performance further.

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.