Prevention of DDOS Attack for Cloud Using User Session Rate Analysis

1Dhakshin raja.R, [2]Jagadeesan M
[1]Final Year PG Scholar, [2] Assistant Professor (SR. Grade)
[1]Master of Computer Applications,
[1]Kongu Engineering College, Perundurai, Tamil Nadu,India.

Abstract--- Cloud computing is an emerging paradigm that allows customers to obtain cloud resources and services according to their demand. Service level agreements (SLA) regulate the costs that the cloud customers have to pay for the provided quality of service (QoS). The success of the cloud computing paradigm is mainly due to its on-demand, pay-by-use and self-service nature. According to this standard, the effects of Denial of Service (DoS) attacks involve not only the quality of the delivered service, but also the service maintenance costs in terms of resource usage. Specifically, if the detection delay is longer, the cost will be more. Therefore, a particular attention has to be paid for stealthy Denial of Service attacks. They aim at minimizing their visibility, and at the same time, they can be as harmful as the brute-force attacks. They are sophisticated attacks tailored to leverage the worst-case performance of the target system through specific periodic, pulsing, and low-rate traffic patterns. In this study, a strategy is proposed to orchestrate stealthy attack patterns, which exhibit a slowly-increasing-intensity trend designed to inflict the maximum financial cost to the cloud customer, while respecting the job size and the service arrival rate imposed by the detection mechanisms. Here both how to apply the proposed strategy, and its effects on the target system deployed in the cloud is described.

Keyword: Cloud Computing, DDOS Defense, WAP protocol, WRAPS Model,, Real-time Misbehaviors.

I.INTRODUCTION

A denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack)[1] is an attempt to make a computer resource unavailable to its intended users. Although the targets of a DoS attack may vary, it generally consists of the concerted efforts of a person or people to prevent an Internet site or service from functioning efficiently, that may be temporarily or indefinitely.

Denial-of-service attacks are designed to shut down or render inoperable a system or network. The goal of the denial-of-service attack is not to gain access or information but to make a network or system unavailable for use by other users. It is called a denial-of-service attack, because the end result is to deny legitimate users access to network services. Such attacks are often used to exact revenge or to punish some individual or entity for some perceived slight or injustice. Unlike real hacking, denial-of-service attacks do not require a great deal of experience, skill, or intelligence to succeed.

Committers of DoS attacks typically target sites or services hosted on high-profile web servers such as banks, credit card payment gateways, and even root name servers. The term is generally used with regards to computer networks, but is not limited to this field, for example, it is also used in reference to CPU resource management.

One common method of attack involves saturating the target(victim) machine with external communications requests, such that it cannot respond to legitimate traffic, or responds so slowly as to be rendered effectively unavailable. In general terms, DoS attacks are implemented by either forcing the targeted computer(s) to reset, or consuming its resources so that it can no longer provide its intended service or obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

The paper aims to protect DDOS attack day to day issues in the server. The administrator had all privileges to access this website. The administrator logins to the web site protect from the hackers and also DDOS attack. All the denied attacks are blocked the corresponding IP address in the server. It is easy to be made through online by clerks of the concern.

The web is a complicated referral graph, in which a node (website) refers its visitors to others through hyperlinks. They propose to use this graph as a resilient infrastructure to defend against distributed denial-of-service (DDoS) attacks that plague websites today. Suppose eBay allows its trusted neighbors (websites linking to it) such as PayPal to refer legitimate clients to its privileged service through a privileged referral channel.

A trusted client needs to only click on a privileged referral hyperlink on PayPal to obtain a privilege URL fore Bay, which certifies the client's service privilege. When eBay is undergoing a DDoS attack and not accessible directly, routers in its local network will drop unprivileged packets to protect privileged clients' flows.

As such, a client being referred can still access eBay even during the attack. Referral relations can be extended over the site graph: e.g., PayPal may refer its neighbors' clients to eBay. In this way, a website could form a large-scale referral network to fend off attack traffic negligible. Indeed, a website that links to others provides a better experience to its own customers if the links it offers are effective, and so websites have an incentive to serve privileged URLs for the sites to which they link. The overheads experienced by this website's users will be either nonexistent if the website offers privileged referrals to only customers that have already authenticated for other reasons, or minimal if the website will refer any client after it demonstrates it is driven by a human user (in the limit, asking the user to pass a reverse Turing test or "CAPTCHA"). As user will show, the referrer incurs only negligible costs in order to make referrals via user technique.

The WRAPS enables clients to circumvent a very intensive flooding attack against a website, and imposes reasonable costs on both edge routers and referral websites. A limitation of WRAPS is that it requires modifications to edge routers, as many capability-based approaches. WRAPS does not require installing anything on a Web client. User explores the importance of web site graph topology to the efficacy of WRAPS. User also describe a simple mechanism that helps a website to acquire referral sites at a negligible cost and helps legitimate clients to retrieve referral relationships from the Internet.

- A client may obtain a privilege URL either directly from the target website
- The border of this mechanism is the site's ISP's edge routers
- Translate fictitious addresses in privilege URLs into the website's real address.
- A neighbor website refers a trusted client to the target website's privileged service.
- The referral is done through a simple proxy script running on the referrer site
- Client acquires a redirection instruction leading to the privilege URL
- Edge routers drop packets addressed to the privilege port of that website.

A DDoS attack can be perpetrated in a number of ways.

Consumption of computational resources such as bandwidth, disk space an processor time.

1. Disruption of configuration information, such as routing information.
2. Disruption of state information, such as unsolicited resetting of TCP sessions.
3. Disruption of physical network components.
4. Obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

A DDoS attack may include execution of malware intended to,

- Max out the processor's usage, preventing any work from occurring.
- Trigger errors in the microcode of the machine.
- Trigger errors in the sequencing of instructions, so as to force the computer into an unstable state or lock-up.
- Exploit errors in the operating system, causing resource starvation and/or thrashing, i.e. to use up all available facilities so no real work can be accomplished.

It proposes to protect websites against DDoS attacks, which user refers to as the "web referral architecture for privileged service" or "WRAPS", is built upon existing referral relationships among websites. Incentives for deployment, therefore, are not a significant barrier, provided that the overhead of the referral mechanism is negligible. Indeed, a website that links to others provides a better experience to its own customers if the links it offers are effective, and so websites have an incentive to serve.

II. Related Works

In the paper "WRAPS: Denial-of-Service Defense through Web Referrals" by XiaoFeng Wang and Michael K. Reiter. The web is a complicated graph, with millions of web-sites interlinked together. In this paper, they proposed to use this web site graph structure to mitigate flooding attacks on a website, using new web referral architecture for privileged service ("WRAPS").

In the paper "CAPTCHA: Using Hard AI (Artificial Intelligence) Problems For Security". They introduce captcha, an automated test that humans can pass, but current computer programs can't pass: any program that has high success over a captcha can be used to solve an unsolved Artificial Intelligence problem. They provide several novel constructions of captchas.

In this paper [11] "Preventing Internet Denial-of-Service with Capabilities", by Tom Anderson Timothy Roscoe and David Wetherall. In this paper, they proposed a new approach to preventing and constraining denial-of-service attacks. Instead of being able to send anything to anyone at any time, in user architecture, nodes must first obtain "permission to send" from the destination; a receiver provides tokens, or capabilities, to those senders whose traffic it agrees to accept.

In this Paper [21] "Implementing Pushback: Router-Based Defense Against DDoS Attacks" by John Ioannidis and Steven M. Bellovin, Pushback is a mechanism for defending against distributed denial-of-service attacks. DDoS attacks are treated as a congestion-control problem, but because most such congestion is caused by malicious hosts not obeying traditional end-to-end congestion control, the problem must be handled by the routers.

In this Paper[23] "Controlling High-Bandwidth Flows at the Congested Router" by Ratul Mahajan, Sally Floyd and David Wetherall, FIFO (First In First Out) queueing is simple but does not protect traffic from high-bandwidth flows, which include not only flows that fail to use end-to-end congestion control, but also short round-trip time TCP flows. At the other extreme, per-flow scheduling mechanisms provide max-min fairness but are more complex, keeping state for all flows going through the router. This paper presents RED-PD, a mechanism that combines simplicity and protection by keeping state for just the high-bandwidth flows. RED-PD uses the packet drop history at the router to detect high-bandwidth flows in times of congestion and preferentially drops packets from these flows.

Table 2.1 Review Survey

S.No	Methodology / Algorithm	Advantages	Disadvantages
1	Collaborative Access Control for OSN with Policy Aggregation (CACO-PA)	Effective trust-based access control technique	Improve Policy aggregation methods

2	Both self reported and from observed Mining Model	Provides information about how their data set may be similar or different from the whole of public	Either similar or different from the whole of public and Private content.
3	Conflict Detection Algorithm and Conflict Resolution	In order to find a solution to the conflict that can be acceptable by all negotiating users, it is key important Point data	Historical behavior is a problem worth further studying.
4	PPDP and PPDM techniques	how to pay the data owners, and more importantly, how he should protect the owners' privacy	Historical behavior is a problem worth further studying.
5	Model of Forest Fire Attacks	Construct a probabilistic model to formalize the threats of forest fire attacks	Less security and usability. Optimizing forest fire attacks given a time constraint

III. SYSTEM METHODOLOGY

A. DOS Attacks against Cloud Applications

In this section are presented several attack examples, which can be leveraged to implement the proposed SIPDAS attack pattern against a cloud application. In particular, we consider DDoS attacks that exploit application vulnerabilities [10], [12], [30], including: the Oversize Payload attack that exploits the high memory consumption of XML processing; the Oversized Cryptography that exploits the flexible usability of the security elements defined by the WS-Security specification , the Resource Exhaustion attacks use flows of messages that are correct regarding their message structure, but that are not properly correlated to any existing process instance on the target server based document, which must be read and processed completely, before they may safely be discarded); and attacks that exploit the worst-case performance of the system, for example by achieving the worst case complexity of Hash table data structure, or by using complex queries that force to spend much CPU time or disk access time. In this paper, they use a Coercive Parsing attack as a case study, which represents one of the most serious threats for the cloud applications.

It exploits the XML verbosity and the complex parsing process (by using a large number of namespace declarations, oversized prefix names or namespace URIs). In particular, the Deeply-Nested XML is a resource exhaustion attack, which exploits the XML message format by inserting a large number of nested XML tags in the message body. The goal is to force the XML parser within the application server, to exhaust the computational resources by processing a large number of deeply-nested XML tags.

B. Stealthy DOS Characterization and modeling

This section defines the characteristics that a DDoS attack against an application server running in the cloud should have to be stealth. Regarding the quality of service provided to the user, we assume that the system performance under a DDoS attack is more degraded, as higher the average time to process the user service requests compared to the normal operation. Moreover, the attack is more expensive for the cloud customer and/or cloud provider, as higher the cloud resource consumption to process the malicious
requests on the target system. From the point of view of the attacker, the main objective is to maximize the ratio between the amount of ‘damage’ caused by the attack (in terms of service degradation and cloud resources consumed), and the the cost of mounting such an attack (called ‘budget’).

Therefore, the first requirement to design an efficient DDoS attack pattern is the ability of the attacker to assess the damage that the attack is inflicting to the system, by spending a specific budget to produce the malicious additional load. The attack damage is a function of the ‘attack potency’, which depends on the number of concurrent attack sources, the request-rate of the attack flows,

and the job-content associated to the service requests to be processed. Moreover, in order to make the attack stealthy, the attacker has to be able to estimate the maximum attack potency to be performed, without that the attack pattern exhibits a behavior that may be considered anomalous by the mechanisms used as a protection for the target system.

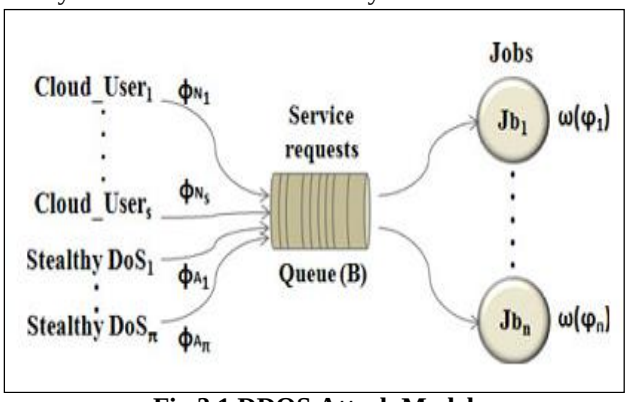

Fig 3.1 DDOS Attack Model

In the above Fig 3.1ns, starting from a synthetic representation of the target system, we describe the conditions the attack pattern has to satisfy to minimize its visibility as long as possible, and effectively affect the target system performance in the cloud environment.

C. Server Under Attack Model

In order to assess the service degradation attributed to the attack, we define a synthetic representation of the system under attack. They suppose that the system consists of a pool of distributed VMs provided by the cloud provider, on which the application instances run. Moreover, we assume that a load balancing mechanism dispatches the user service requests among the instances. The instances can be automatically scaled up or down, by monitoring some parameter suitable to assess the provided QoS (e.g., the computational load, the used memory, and the number of active users). Specifically, we model the system under attack with a comprehensive capability zM, which represents a global amount of work the system is able to perform in order to process the service requests. Such capability is affected by several parameters, such as the number of VMs assigned to the application, the CPU performance, the memory capability, etc. Each service request consumes a certain amount wi of the capability zM on the base of the payload of the service request.

Thus, the load CN of the system at time t can be modeled by a queuing system M=M=n=n with Poisson arrivals, exponentially distributed service times, multiple servers, and n incoming requests in process (system capability). Moreover, the auto scaling feature of the cloud is modeled in a simple way: when new resources (e.g., VMs) are added to the system, the effect is an increase of the system capability zM.

Therefore, given h legitimate type of service requests u = (#1; . . . ; #h), and denoted w as the cost in terms of cloud resources necessary to process the service request' 2 u, an attack against a cloud system can be represented as in Fig. 3.1. Specifically, Fig. 3.1 shows a simple illustrative attack scenario, where the system is modeled as: ðiÞ a queue (that conceptually represents the load balancing mechanism), in which are queued both the legitimate user request flows fN_j and the DDoS flows fA_j (attack sources), and ðiiÞ a job for each service request that is currently processed on the system.

D. Stealthy Attack Objectives

In this section, we aim at defining the objectives that a sophisticated attacker would like to achieve, and the requirements the attack pattern has to satisfy to be stealth. Recall that, the purpose of the attack against cloud applications is not to necessarily deny the service, but rather to inflict significant degradation in some aspect of the service (e.g., service response time), namely attack profit PA, in order to maximize the cloud resource consumption CA to process malicious requests. In order to elude the attack detection, different attacks that use low-rate traffic (but well orchestrated and timed) have been presented in the literature. Therefore, several works have proposed techniques to detect low-rate DDoS attacks, which monitor anomalies in the fluctuation of the incoming traffic through either a time or frequency-domain analysis.

They assume that, the main anomaly can be incurred during a low-rate attack is that, the incoming service requests fluctuate in a more extreme manner during an attack. The abnormal fluctuation is a combined result of two different kinds of behaviors: ðiÞ a periodic and impulse trend in the attack pattern, and ðiiÞ the fast decline in the incoming traffic volume (the legitimate requests are continually discarded). Therefore, in order to perform the attack in stealthy fashion with respect to the proposed detection techniques, an attacker has to inject low-rate message flows fA j¼ ½'j;1; . . . ; 'j;m].

Stealthy DDoS attack pattern in the cloud - Denote p the number of attack flows, and consider a time window T , the DDoS attack is successful in the cloud, if it maximizes the following functions of profit and resource consumption:

$$\begin{cases} maximize \quad P_A = \sum_{j=1}^{\pi} \sum_{i} g(\varphi_{j,i}), \\ maximize \quad C_A = \sum_{j=1}^{\pi} \sum_{i} w(\vartheta_{j,i}), \end{cases}$$

and it is performed in stealthy fashion, if each flow fAj satisfies the following conditions:

$$\begin{cases} (c_1) & minimize & \delta_j, \forall j \in [1..\pi], \\ (c_2) & s.t.to & \varphi_{j,i} \in \theta, \\ (c_3) & s.t.to & exhibits\ a\ pattern\ neither\ periodic \\ & & \qquad nor\ impulsive, \\ (c_4) & s.t.to & exhibits\ a\ slowly\ increasing\ intensity, \end{cases}$$

Where:

- g is the profit of the malicious request ’j;i, which expresses the service degradation
- d j is the average message rate of the flow fAj,
- w is the cost in terms of cloud resources necessary to process ’j;i 2 u.

E. Creating Service Degradation

Considering a cloud system with a comprehensive capability zM to process service requests ’i, and a queue with size B that represents the bottleneck shared by the customer’s flows fN j and the DoS flows fAj (Fig. 1). Denote C0 as the load at time the onset of an attack period T (assumed to occur at time t0), and CN as the load to process the user requests on the target system during the time window T. To exhaust the target resources, a number n of flows fA j have to be orchestrated, such that:

$$C_0(t_0) + C_N(T) + C_A(T) \geq \zeta_M * T,$$

Where, CA(T) represents the load to process the malicious requests i during the period T. If we assume that ð1Þ the attack flows are not limited to a peak rate due to a network bottleneck or an attacker’s access link rate, and ð2Þ the term CN can be neglected during the attack (CA CN), the malicious resource consumption CA can be maximized if the following condition is verified:

$$C_A(T) \geq \zeta_M * T - C_0(t_0) \quad with \quad C_A \gg C_N.$$

Moreover, assume that during the period T, the requests ’i 2 fA burst at an average rate dA, whereas the flow fN bursts at an average rate dN. Denote B0 as the queue size at time t0, and d as the time that the queue becomes full, such that:

$$d = \frac{B - B_0}{\delta_A + \delta_N - \delta_p}$$

Where, d p is the average rate of requests processed on the target system. After d seconds, the queue remains full if dA þ dN dp.

F. Minimize Attack Visibility

According to the previous stealthy attack definition, in order to reduce the attack visibility, Conditions (2) have to be satisfied. Therefore, through the analysis of both the target system and the legitimate service requests (e.g., the XML document structure included within the HTTP messages), a patient and intelligent attacker should be able to discover an application vulnerability (e.g., a Deeply-Nested XML vulnerability), and identify the set of legitimate service request types #k u (Cond. (2.c2)), which can be used to leverage such vulnerability. For example, for an X-DoS attack, the attacker could implement a set of XML messages with different number of nested tags nTi ¼ 1; . . . ; NT.

The threshold NT can be either fixed arbitrarily, or possibly, estimated during a training phase, in which the attacker injects a sequence of messages with nested XML tags growing, in order to identify a possible limitation imposed by a threshold-based XML validation schema. A similar approach can be used to estimate the maximum message rate dT with m which injecting the service requests ’i.

The attacker has to define the minimal number p of flows fA characterized by malicious requests injected with: an average message rate lower than dT, in order to evade rate-controlling- and time- window-based detection mechanisms (Cond. (2.c1)), and a polymorphic pattern described in the next section), in order to evade low-rate detection mechanisms such that maximize the functions PA and CA

G. Attack Effect Estimation

During the attack, in order to determine if the current flows fA are generating a service degradation, the Meter injects a flow fM of requests ’i overlapped to the attack flows fA, and estimates the service time tS to process each message ’i on the target system. In particular, if they assume that the flow fM is not limited by a network bottleneck, and the network latency is negligible, then, we can approximate tS with the response time of the target application.

Therefore, during a training phase, the attacker can estimate an approximation of the actual distribution of the response time tR, for each message of type #k u, and then, uses it to evaluate the service degradation achieved. Since the actual response time distribution may have a large variance during the attack, the estimation model has to be in charge of identifying significant deviations.

Therefore, supposing that mRð#kÞ and sRð#kÞ are the mean and standard deviation of the response time tR for the messages type #k, empirically estimated during the training phase, the Meter can adopt the following Chebyshev's inequality to compute deviation of the service time tSð'iÞ during the attack:

$$p(|t_S(\varphi_i) - \mu_R(\vartheta_k)| \geq \lambda * \sigma_R(\vartheta_k)) \leq \frac{1}{\lambda^2} \quad with \quad \varphi_i \in \vartheta_k.$$

The Chebyshev's inequality establishes an upper bound for the percentage of samples that are more than standard deviations away from the population mean. The Chebyshev's inequality can be used to compute an upper limit (an outlier detection value) $\varsigma(\vartheta_k) = \mu_R(\vartheta_k) + \lambda * \sigma_R(\vartheta_k)$ beyond which the sample tS can be considered to be an outlier.

H. Wraps Algorithm Steps

WRAPS grants a client greater privilege to access its service by assigning to it a secret fictitious URL called privilege URL with a capability token embedded in part of the IP and port number fields. Through that URL, the client can establish a privileged channel with that website even in the presence of flooding attacks.

A client may obtain a privilege URL either directly from the target website or indirectly from the website's trusted neighbors. A website offers a client a privilege URL if the client is referred by one of the site's trusted neighbors, or is otherwise qualified by the site's policies that are used to identify valued clients, for example, those who have paid or who are regular visitors.

A qualified client will be redirected to the privilege URL generated automatically using that client's identity, service information, and a server secret. A privilege URL leads its holder to the target website through a protection mechanism which protects the website from unauthorized flows. The border of this mechanism is the site's ISP's edge routers, which classify traffic into privileged and unprivileged flows, and translate fictitious addresses in privilege URLs into the website's real address. Within the protection perimeter, routers protect privileged traffic by dropping unprivileged packets during congestion.

A neighbor website refers a trusted client to the target website's privileged service. The referral is done through a simple proxy script running on the referrer site, from which the client acquires a redirection instruction leading to the privilege URL. WRAPS specially detects the request is generated through click events by human or through programmatically.

1. Receive a request.
2. Check IP Address in blocked list.
3. Check Requested URL of importance against attack. i.e., the document or web page is required to be checked for attack.
4. If the count of requests is found to be reached to allowed limit in a specified period, then redirect the request to access denied page
5. The last request time is stored again so that the successive requests' time are checked for request count.

IPClassifier classifies all inbound packets into three categories: packets addressing the website's privilege port which are dropped, TCP packets which are forwarded to IPVerifier, and other packets, such as UDP and ICMP, which are forwarded to the normal forwarding path.

IPVerifier verifies every TCP packet's capability token embedded in the last octet of the destination IP address and the 2-octet destination port number. Verification of a packet invokes the MAC over a 5-byte input and a 64-bit secret key. The packets carrying correct capability tokens are sent to IPRewrite, which sets a packet's destination IP to that of the target website and destination port to port. WRAPS overcome the drawbacks through checking the HTTP_REFERER property in Request. If the value is null, it is clear that the page is requested programmatically by an application.

WRAPS differs from overlay-based approaches in several important ways. WRAPS, Fig 3.2 however, asks only referral websites to offer a very light- weight referral service, which allows WRAPS to take advantage of existing referral relationships on the web to protect important websites. WRAPS also alters neither protocols nor client software. WRAPS does not change packets routing paths and thus avoids these overheads.

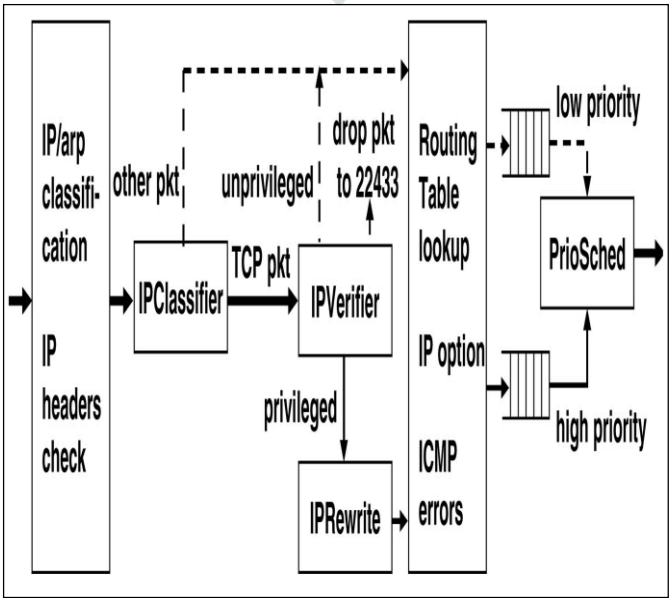

Fig 3.2. WRAPS elements on a Click packet forwarding path

IV EXPERIMENTAL RESULTS

The following **Table 4.1** describes experimental result for existing system secure transmission Services analysis. The table contains number of time slot interval and given time interval to calculate average numbers of send transmission services details are shown

S.NO	NUMBER OF WEBSITES TIME SLOT (M)	RATIO OF SECURE TRANSMISSION SERVICES
1	10	0.43
2	20	0.52
3	40	0.61
4	60	0.69
5	80	0.74
6	100	0.80
7	120	0.86
8	140	0.90
9	150	0.93
10	160	0.97

Table 4.1HitRate-Performances Analysis

The following **Fig 4.1 d**escribes experimental result for existing system secure transmission Services analysis. The figure contains number of time slot interval and given time interval to calculate average numbers of send transmission services details are shown

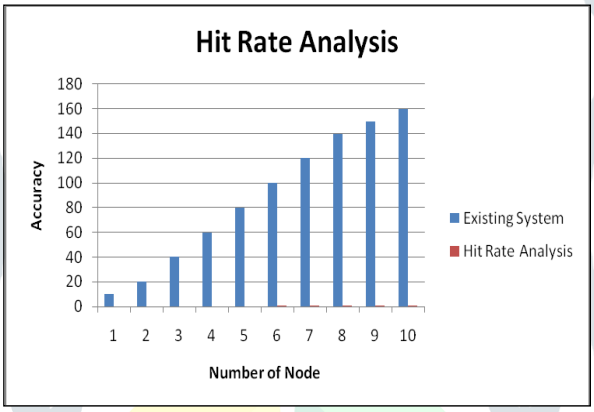

Fig 4.1HitRate-Performances Analysis

V. CONCLUSION

The Secure Overlay Service system needs to increase the server speeds or number of servers to balance the client's request. DDoS attack is a critical threat to current Internet. Recently too many technologies of the detection and prevention have developed, but it is difficult that the IDS distinguishes normal traffic from the DDoS attack.

The DoS threats could be mitigated through exploring the enormous interlink age relationshIPs among the websites themselves. The design and implementation of WRAPS, a web referral infrastructure for privileged service, and empirically evaluated its performance. WRAPS enables clients to evade very intensive flooding attacks

Thus the automated generated code, which is unique for each message is attached and sent. The administrator verifies the code and checks the IP address details when there is a mistrusted user. The hacker users were requested to provide the authentic details and those details are verified with the interfaces connected to the server.

When the user did not use the service for a long period, then the user was removed based on the proposed system. Denial-of-service attacks are designed to shut down or render inoperable a system or network. The goal of the denial-of-service attack is not to gain access or information but to make a network or system unavailable for use by other users. It is called a denial-of-service attack, because the end result is to deny legitimate users access to network services. Such attacks are often used to exact revenge or to punish some individual or entity for some perceived slight or injustice. Unlike real hacking, denial-of-service attacks do not require a great deal of experience, skill, or intelligence to succeed. Committers of DoS attacks typically target sites or services hosted on high-profile web servers such as banks, credit card payment gateways, and even root name servers. The term is generally used with regards to computer networks, but is not limited to this field, for example, it is also used in reference to CPU resource management.

REFERENCES

[1] X. Wang and M. Reiter, "Wraps: Denial-of-Service Defense through Web Referrals," Proc. 25th IEEE Symp. Reliable Distributed Systems (SRDS), 2006.

[2] J. Wu and K. Aberer. Using siterank for p2p web retrieval. Technical Report C/2004/31, SwissFederal Institute of Technology, Lausanne, Switzerland, March 2004.

[3] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In Advances in Cryptology EUROCRYPT 2003. Springer-Verlag, 2003.

[4] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The click modular router. ACM Transactions on Computer Systems, 18(3), August 2000.

[5] E. Kohler. The Click modular router. MIT, November 2000. PhD paper.

[6] A. Yaar, A. Perrig, and D. Song. An endhost capability mechanism to mitigate DDoS flooding attacks. In Proceedings of the IEEE Symposium on Security and Privacy, May 2004.

[7] T.Anderson, T.Roscoe, and D.Wetherall. Preventing internet denial-of-service with capabilities. In Proceedings of Workshop on Hot Topics in Networks (HotNets-II), November 2003.

[8] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2003.

[9] L. von Ahn, M. Blum, N.J. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," Advances in Cryptology—EUROCRYPT '03. SpringerVerlag, 2003.

[10]. Benny Pinkas and Tomas Sander. Securing Passwords Against Dictionary Attacks. In Proceedings of the ACM Computer and Security Conference (CCS' 02), pages 161-170. ACM Press, November 2002.

[11]T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet Denial-of-Service with Capabilities," Proc. Second Workshop Hot Topics in Networks (HotNets '03), Nov. 2003.

[12] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial of Service Activity. In Proc. Usenix Security Symposium 2001.

[13] D. Moore, C. Shannon, and J. Brown. Code Red: A Case Study on the Spread and Victims of an Internet Worm. In Proc. Internet Measurement Workshop 2002.

[14] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The Spread of the Sapphire/Slammer Worm. http: /www.cs.berkeley.edu/~nweaver/sapphire/, Jan. 2003.

[15] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. ontrolling High Bandwidth Aggregates in the Network. Computer Communications Review, 32(3), July 2002.

[16] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In Proc. ACM SIGCOMM 2002.

[17] D. Andersen. Mayday: Distributed Filtering for Internet Services. In Proc. of USITS 2003.

[18] P. Barford, J. Kline, D. Plonka, and A. Ron. A Signal Analysis ofNetwork Traffi c Anomalies. In Proc. Internet Measurement Workshop 2002.

[19] A. Hussain, J. Heidemann, and C. Papadopolous. A Framework for lassifying Denial of Service Attacks. In Proc. ACM SIGCOMM 2003.

[20] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In Proc.IEEE Infocom 2003.

[21] J. Ioannidis and S. Bellovin, "Implementing Pushback: Router-Based Defense against DDoS Attacks," Proc. Symp. Network and Distributed System Security (NDSS), 2002.

[22]S. Floyd and K. Fall, "Promoting the Use of End-to-End ongestion Control in the Internet," IEEE/ACM Trans. Networking,Aug. 1999.

[23] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling High-Bandwidth Flows at the Congested Router," Proc. Ninth IEEE Int'lConf. Network Protocols (ICNP '01), Nov. 2001.

.