Maintaining Useful Server Throughput under Load Attacks Using Active NIC Portals*

Onur Demir and Kanad Ghose
Department of Computer Science
State University of New York at Binghamton
Binghamton, NY 13902-6000
{onur, ghose}@cs.binghamton.edu

Abstract— **This paper presents a solution to denial-of-service (DoS) attacks on servers where the server resources are saturated by the repeated request for execution of scripts or the download requests for large files. The existing solutions for coping with DoS attacks, which are primarily based on limiting the traffic rates, are incapable of providing any protection against load attacks, as these attacks do not manifest themselves as heavy bursts of traffic. We present an intelligent gateway based solution for maintaining the useful throughput of the servers under load attacks that uses specific information from the servers to perform dynamic load balancing and dynamic packet filtering. The intelligent gateway is implemented using a dual-ported active network card (NIC). Clients are classified according to their request history, and rate limits are imposed at the gateway for each class according to the level and duration of the attack. Results for a prototype implementation indicate our solution to be an effective deterrent against load attacks.**

Keywords: network security; denial-of- service attacks; web servers; load attacks

I. INTRODUCTION

Web servers are often targets of attacks that saturate server resources and consequently deny the use of these resources to legitimate requests. A particular form of denial-of-service (DoS) attack is to tie up one or more types of resources on the server by repeatedly requesting downloads of large files execution of scripts (such as .cgi files) on the server. We will generically refer to these as "load attacks". One generic solution to load attacks is to apply rate limiting on the requests from identified attackers and suspected clients to keep the server load at an acceptable level. The defending mechanism for protecting the servers against such attacks cannot use any of the existing solutions for defending against DoS attacks that perform naive rate limiting at the routers based on the traffic rate directed towards the server. This is simply because load attacks do not necessarily need to create a large traffic flow towards the servers. For example, a few download requests for large files or for the execution of a complex script file can easily inundate the servers. The mechanism for defending against load attacks must therefore be able to limit requests directed towards the server based on the type of request submitted and based on specific information from the server side. [3][5][7][8][9]

The type of information provided by the server to the rate limiting mechanism for coping with load attacks can be quite varied. In the simplest form, it could be the current loading level information. The server-provided information can also be a function of the service provided by the server and reflect service goals. For example, for a content-server site (like CNN), the service goal may be to maximize the number of clients served, with no preference to any specific client. In contrast, for a web server designed for an on-line auction service (such as eBay), the service goal will be to give preferential service to the bidding clients, when an auction is about to end.

In a general solution for coping with load attacks, the servers should be given a role in the defending mechanism. The mechanisms that defend the server against load attacks should perform traffic request limiting based on information provided from the server such as, but not limited to:

- Server load level for specific types of requests
- IP addresses of suspected attackers (for load attacks, the client IP addresses cannot be spoofed, as a TCP connection has to be set up)
- IP addresses of clients given a preference

Information such as these could be obtained directly or indirectly from the log and session files kept at the server.

A intelligent gateway for a single server or a pool of servers implemented using active network cards (NICs) can be used effectively to regulate traffic directed towards the server(s) and allow the servers to be defended against load DoS attacks in real-time. An active NIC is essentially a network card, often with multiple network interfaces, a programmable processor, DMA controllers and a fair amount of on board memory (flash, as well as RAM). The active NIC based gateway can implement dynamic filtering rules based on information provided by the servers that they defend. The data structures and filtering rules used by the gateway can be held in the RAM of the active NIC and the processing capabilities of general-

* Supported in part by the NSF through award No. EIA 9911099

purpose processor on the active NIC can be well exploited in implementing fast, flexible dynamic filtering solutions. In addition, load balancing functions can be integrated into such a gateway. This paper describes exactly such a solution and evaluates the potentials of the solution from a prototype implementation based on active gateways implemented using dual-ported active NICs.

Load attacks are carried out by sending numerous requests from one or a large number of clients, possibly compromised, in a continuous manner. The attackers have to use real IP addresses to make a server request; spoofing is not possible. Load attacks on servers are not generally a consequence of any security problems or with the setup of the server. Such attacks can be effective simply because server resources are finite in nature. In theory, the servers can be configured to limit the requests at the server and thus prevent resources from getting saturated. Such a solution can itself end up taxing the resources at the server. Using a separate gateway to limit the requests is a better solution, as the server resources are not used up in the rate limiting process or in the process of limiting requests based on the client's address. The bulk of the processing resources needed for constructing the dynamic packet filtering rules and to perform the filtering resources are within the active NIC and its host machine.

A single active NIC-based intelligent gateway may provide the necessary protection for a single server or a limited number of servers. The processing capabilities of a single gateway may not be adequate for protecting a large server pool. In such cases, multiple intelligent gateways need to be used, with each gateway controlling access to a limited number of servers. This, in fact, is the manner in which the solution proposed in this paper scales up to handle large sever pools. As in the case of large server farms, a front-end load-balancing/directing switch may be used to direct the incoming traffic to a specific intelligent gateway and the servers that are accessible through that gateway.

II. Locally Distributed Web Server Architecture

A locally distributed web server cluster refers to a web site that uses a pool of servers within a local network. Figure 1 depicts our solution for such a server. The dual-ported active NIC based gateway acts as an interface to the web server. The web server cluster provides a single IP (virtual IP, VIP) address to the Internet, which is assigned to the incoming port of active NIC. All admitted client traffic goes through the active NIC portal towards the server pool through the second interface on the active NIC. Client responses from the server use a different path as shown, bypassing the gateway.

The active NIC is responsible for distributing inbound packets to the servers after subjecting them to a filtering rule. The active NIC can use fixed or dynamic routing schemes according to types and sources of the incoming packets. In this paper, we have used a static, network layer routing to select the destination of the packets. The incoming packet headers are modified by the gateway, which changes the VIP with the IP address of the selected server machine. When the server machine replies the request it uses VIP as the source IP.

The host, where active NIC is mounted (called the active NIC host), runs a daemon called the control agent. The control agent periodically collects information from server agents that run on the servers. The control agent uses this information to determine the dynamic packet filtering rules that have to be deployed on the gateway and updates the existing filtering rule set on the active NIC. Keeping the control agent on the active NICs host significantly eases the processing load on the active NIC.

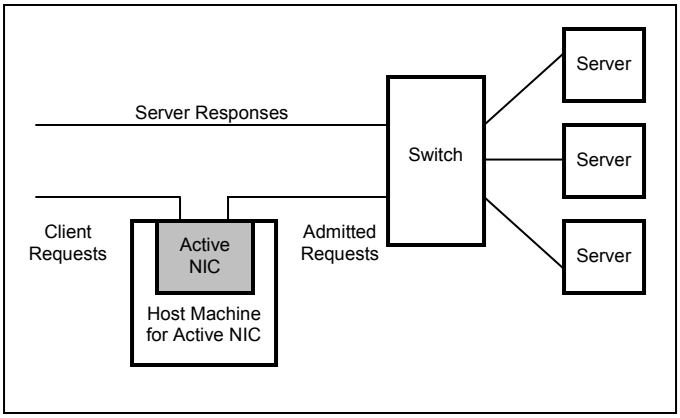

Figure 1. Active NIC Gateway

In our prototype implementation, we have used a Ramix PMC 694 active NIC with dual 100 Mbits/sec Ethernet interfaces, two autonomous DMA controllers, a 233 MHz. Power PC CPU and 32 Mbytes of RAM and 8 Mbytes of Flash memory [1]. The Ramix PMC 694 is a PCI card. The primary packet filter used on the active NIC gateway is based on the well-known BPF+ filter [23]. The filter uses an additional trie data structure maintained within the on-board RAM to hold IP addresses that have to be considered for filtering.

The Ramix card runs RTEMS, a real-time operating system based on BSD UNIX and implements a TCP/IP stack to allow for TCP offloading. A proprietary library is used for communicating from the host PC to the PMC 694; this interface is not critical to the performance of our scheme. The experiments with the packet filter shows that the card can handle up to 88, 495 packets per second.

III. Choosing the Classes and Rates

To defend against load attacks in real time, we classify client IP addresses dynamically into four groups based on the usage history of each client. The server agents gather the information used for such classifications on a regular basis and pass such information to the control agent on the active NIC's host on a regular basis. The final decision for deploying rate limiting and the dynamic alteration of the packet-filtering rule at the gateway is left to the control agent.

The data structure used to keep track of client IP addresses is PATRICIA tries [25]. The control agent, the server agent and the active NIC all use this data structure. The IP addresses of the clients constitute the keys in this data structure. Each

entry has a time stamp for last access time. Entries are aged according to this time stamp, and eventually removed from the data structure when the last access time becomes older than 1 hour.

The four categories of IP addresses used in our implementation are as follows:

Green Addresses: These class of IP addresses are assumed unknown to the servers. They have not submitted any requests for the last 1-hour period.

Red Addresses: The number of requests for a specific server resource from clients with a red IP addresses have crossed a pre-specified limit. Generally, two types of limits are used. The first one is the total number of bytes requested in file downloads. The clients that continuously request large files can be easily isolated using this limit. The second type of limit is the number of requests for a specific service. The services that do not consume huge amounts of network bandwidths but instead expend a considerable amount of CPU time and memory (e.g. CGI scripts) can be easily controlled by this limit. There are also additional ways to identify red addresses, such as the number of access errors, the source domain, and severe access violations. Clients IP addresses are reclassified as red when the client's requests exceed the specified limits.

Amber Addresses: This class of IP corresponds to clients that have used the web server within their individual limits.

Preferred Addresses: This optional class of IP addresses is specific to the web server application. The server can choose the set of preferred addresses based on login information, region, domain, or any other criteria. Another way of choosing preferred addresses can be a static list of trusted hosts or domains.

After classifying the addresses into groups, the control agent transfers the corresponding filter rule updates to the active NIC gateway.

IV. THE LOAD ATTACK DEFENSE POLICY

There can be different choices in deciding what client IP address class, as defined earlier, has to be given admission preference to the servers. In this paper we show the effects of the two different types of preferences. According to the preference, during the load attack, rate limits are applied to the classes. The rate limits start with higher vales and decreases progressively if the attack continues to be sustained or when the attack volume increases.

The onset of a load attack can be detected by any server agent from the local loading information. Any server agent can generate an attack alert when CPU load value goes beyond a threshold value. We have used /proc/loadavg file to obtain the CPU load level. These variables are specific to the servers and the proper values of these thresholds are determined experimentally. There are other ways of raising alerts and their discussion is beyond the scope of this paper.

The two specific preferences that we have studied in this paper are shown in the Table 1. The table shows how the defending mechanism limits the rates during a load attack for each of the two preferences. The various specific values shown in this table (percentages and durations) are what we have used in the experimental evaluations reported here. They can, of course, be tuned to optimize the overall performance of the protection scheme.

As seen in the table rate limiting only applied to excess traffic for each given class of requests. Each class has a limit on the number of requests per second and a rate limit is applied when this rate is beyond thresholds.

V. EXPERIMENTAL SETUP

The servers participating to the evaluation system are Pentium IV PCs running a modified version of Linux kernel 2.4.18. We have used two switches and constructed two subnets in 100 Mbits/sec Ethernet. The server pool constitutes one subnet and the attacker and client machines form another subnet (representing the outside world). The active-NIC is positioned as a gateway with its two ports connected to the two subnets. Multiple addresses are assigned to network interfaces of client and attacker machines to extend the IP range. A simple command line http loader utility is used to generate get the pages. For each request, clients are able to select an IP assigned to the interface. The incoming pages are not displayed on a browser at the client; they are only inspected for the checksum and disposed of whenever they are completely downloaded.

TABLE I. DEFENSE POLICY

Sequence	Preference for Green Address	Preference for Preferred/Trusted Address
No attack detected	No rate limiting	No Rate limiting
First 30 seconds into attack	Limit all the traffic from the red address class; Allow 90% of excess amber class traffic	Limit all traffic from red address class, Allow 90% of Amber Traffic
Next 60 seconds into the attack (if attack continues)	Limit all the traffic from red address class: Allow 70% of excess amber class traffic	Limit all the traffic from red address class; Allow 70% of amber class traffic, limit 90% of green class traffic
Another 60 seconds into the attack (if attack continues)	Limit all the traffic from red address class: Allow 50% of excess amber class traffic	Limit all the traffic from red address class: Allow 50% of amber class traffic, limit 10% of green traffic
A further 60 seconds into the attack (if attack continues)	Allow only preferred/trusted addresses green address traffic and block any excess traffic	Allow only preferred/trusted addresses and block any excess traffic

VI. EXPERIMENTAL RESULTS

In the experiments reported in this paper, we used different server update frequencies (abbreviated as **UF**) - the frequency of data update from servers to control agent - to see the effect of the frequency of updating information about clients, rules and limits. The effect of allowed rate limits (abbreviated as

RL) is also studied in the experiments. Rate limits indicate permissible resource usage by the clients. The clients whose resource usage exceeds their specified RL value are considered as attackers.

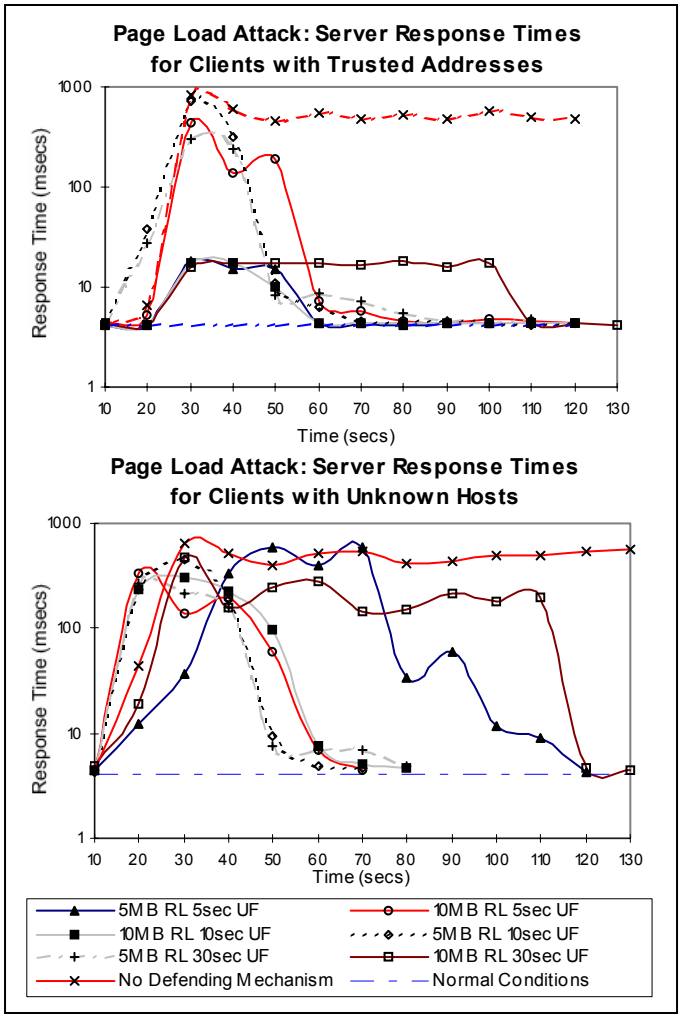

Figure 2. Response times for trusted and unknown addresses during page load attack with different server update frequencies (UF) and rate limits

In the first set of experiments, we evaluated the connection times of the trusted and unknown address classes under different types of attacks. We used three attack types in the experiments. The first type of the attack is a page load attack. In this attack, the attackers request the download of large files to degrade the performance in a high frequency. A coordinator starts the attackers at the same time. The second type of the attack is a CGI load attack. In this type of the attack, the attackers request the execution of a CGI script, which uses a fair amount of CPU time. A moderately high script execution time was used for the experiments. The third type of the attack is a mix of both attack types, combining download attacks with CGI attacks.

For all of our experiments, we have used three host machines to exclusively run the attacking clients. Each such host uses 80 IP addresses as aliases, mimicking 80 different attacking hosts. A request targeting a server is formed by choosing a randomly-generated client address. We have used 20 different IP addresses for the clients with unknown IPs. For the experiments, the unknown IP addresses are never marked as amber or red. There are also 20 different trusted IP addresses, which have priorities over the other classes as described earlier. Three identical web servers, defended by a single active NIC gateway, are used as targets of the load attacks.

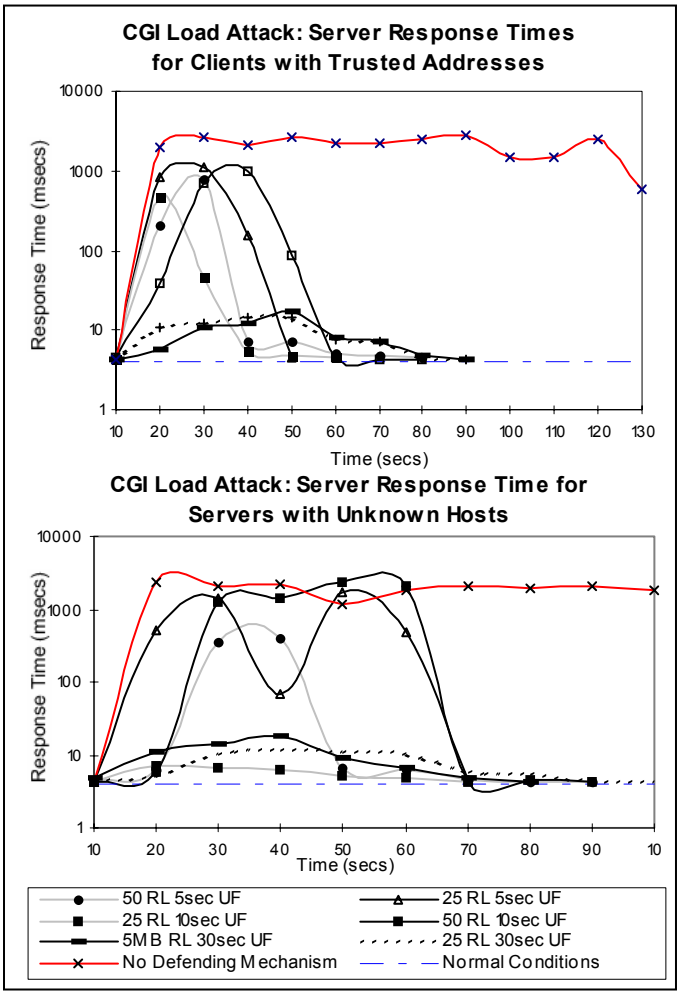

Figure 3. Response times for trusted and unknown addresses during CGI load attack with different server update frequencies (UF) and rate limits

We measured the response times for clients with trusted and unknown IP addresses. The normal response time and the response time under attack without any defending mechanism is shown in all of the graphs for reference. In the experiments, the attack traffic is started ten seconds after starting the measurements. The experiments show different cases of update frequencies (UF) and, rate limits (RL). We have used UF values of one update in every 5, 10 and 30-second periods. Rate limits (RL) are the limits for one IP to be identified as red. For page load attacks, the rate limit is the total number of bytes downloaded per server in one hour. For the CGI load attacks,

the rate limit is the number of execution requests of CGI script files in one hour. For the mixed attack, both limits are used. We have used 5 MB and 10 MB per server per hour as rate limit for page attacks, and 25- 50 requests per hour for CGI script execution as rate limit for CGI attacks.

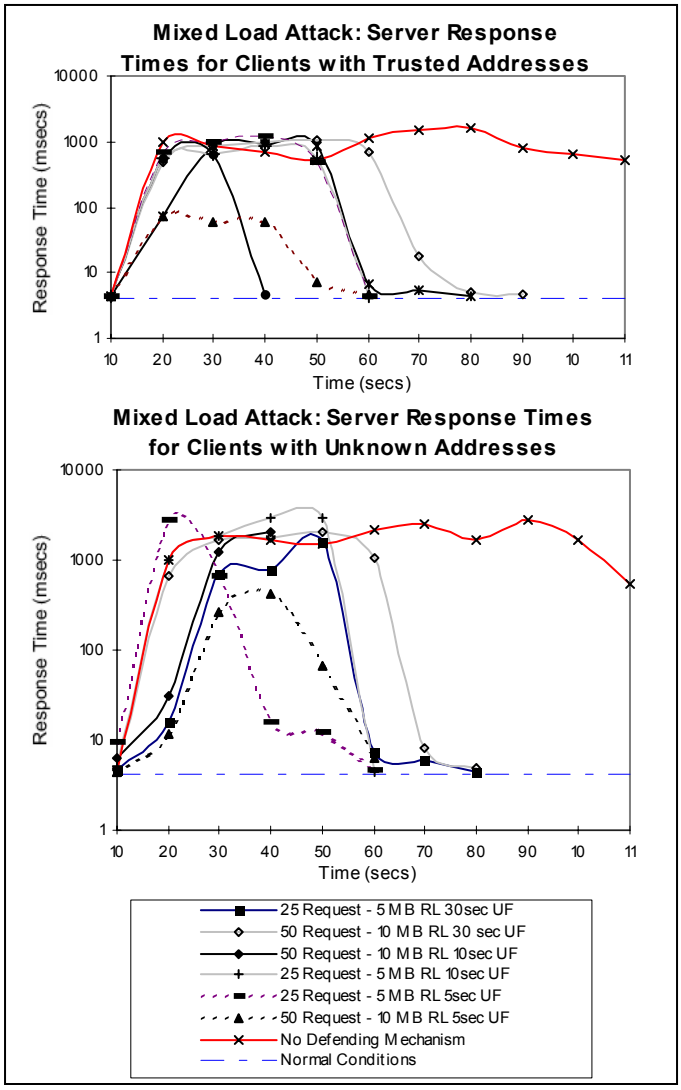

Figure 4. Response times of trusted and unknown addresses during mixed load attack with different server update frequencies (UF) and rate limits

Figures 2 to 4 show the response times of the trusted and unknown client hosts, when the defending mechanism gives preference to trusted addresses. Notice that the defending mechanism severely reduces the adverse effect of the attacks in at most 30-40 seconds, depending on the values used for UF and RL. By at most 120 seconds, the effects of the attacks are virtually eliminated. Furthermore, the server response time for the trusted hosts is impaired by a noticeably lower degree, compared to hosts with unknown IPs, during the attacks. All of the goals of the defending mechanism are thus successfully met. Also notice that, the red class addresses are blocked whenever they are identified. The identification can take 20-30 seconds depending on the frequency of the attacker's requests. In general smaller update frequencies give better results, but especially for CGI attacks where the server load is an important factor, there is no noticeable difference. This result suggests that server loading is a factor in the choice of the UF.

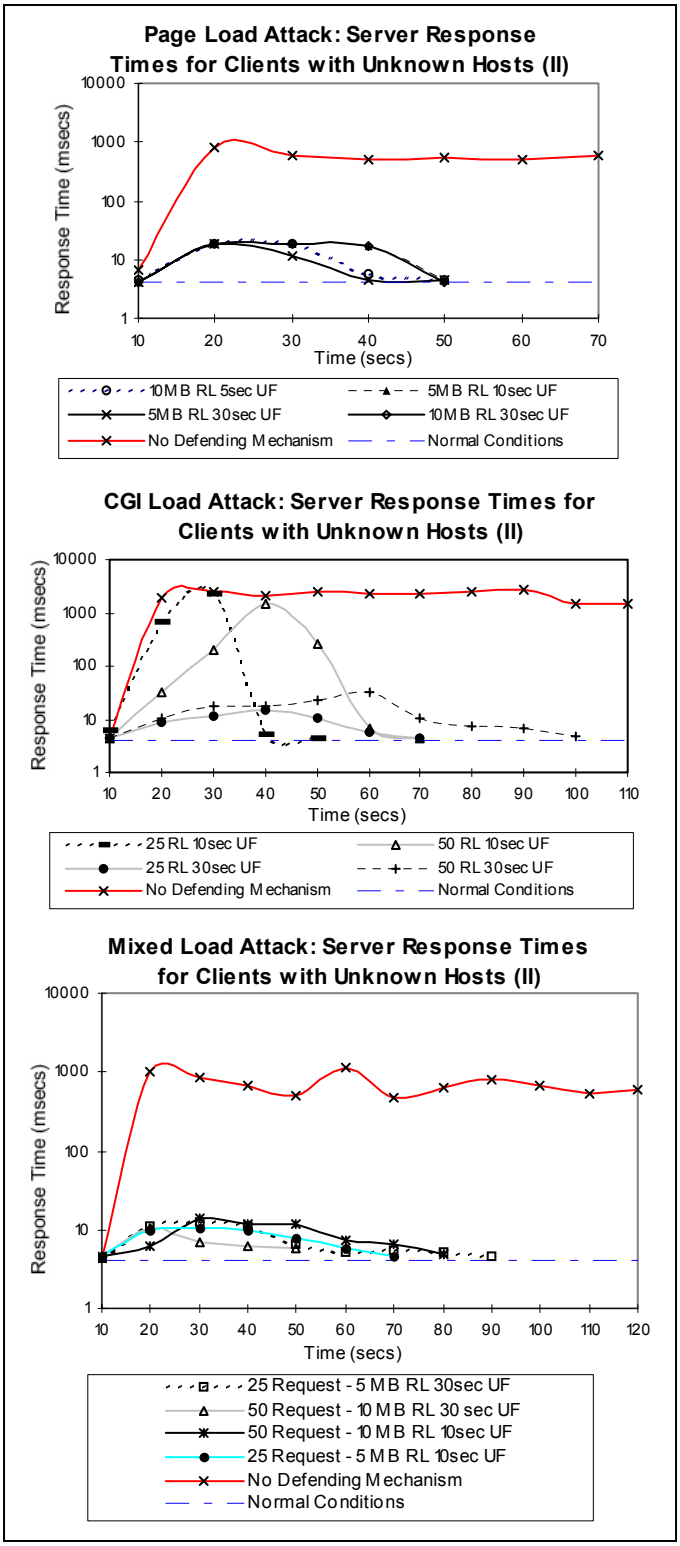

Figure 5. Response times of unknown addresses during page load, CGI, and mixed attack with different update frequencies (UF) and rate limits when the defending mechanism prefers unknown client host addresses

The second set of experiments demonstrates the performance of the clients when the defending system prefers unknown addresses. The results are shown in the Figures 5 through 10. In these experiments, the system identifies all the attackers within 60 seconds at most. All adverse effects of attacks are eliminated after the first minute of the attack.

Note that the identification of *individual* attackers depends on the frequency and the extent of their requests. The attack can be effective if the number of simultaneous requests is more than what the servers can handle. If the number of attackers is small, they have to make recurrent requests to load the server. This will lead to their early identification. If the number of attackers is large, then the identification takes longer.

VII. CONCLUSIONS

We presented a mechanism for maintaining useful server throughput under load attacks. Such load attacks are difficult to detect using existing mechanisms that only sense and detect a high traffic rate directed towards a server. Load attacks can be caused by only a few requests that can inundate the server resources. Our solution is in the form of an intelligent gateway that performs dynamic packet filtering to determine requests that can be admitted. The packet filtering rules are constructed dynamically to cope with the evolving service goals and traffic dynamics during the attack, making use of specific information from the servers. The intelligent gateway is effectively a smart firewall whose computational and storage resources are well-beyond that of traditional firewalls. The combined resources of the active NIC and the host machine that accommodates this NIC are deployed to implement fast packet filtering for coping with load attacks. We show two specific filtering strategies – one that favors specific trusted clients during an attack and another that prefers clients that have not generated an excessive rate of requests. Our experimental results show that our solution can cope with load attacks and is capable of isolating the traffic from offending clients quickly. At the same time our solution permits the servers to provide a reasonable level of service to preferred clients during a load attack. We also briefly discuss how our solution can be scaled up to handle larger server pools.

REFERENCES

[1] Ramix Inc., Intelligent Ethernet Adapter Product Guide, available at: http://www.ramix.com

[2] Oliver, R., "Countering syn Flood Denial-of-service Attacks", invited presentation at the Usenix Security Conference, 2001. Presentation slides at: http://www.usenix.org/events/sec01/invited talks/oliver.pdf, August 29 2001.

[3] Bellovin, S.M., "ICMP Traceback Messages", Internet Draft: draftbellovin- itrace-00.txt, 2000.

[4] Check Point Software Technologies, http://www.checkpoint.com/

[5] Mirkovic, J., Martin, J. and Reiher, P. , "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms", UCLA Computer Science Department, Technical report #020018, 2001.

[6] Ricciuli, L., Lincoln, P., and Kakkar, P.. "TCP SYN Flooding Defense", in Proc. of the Simulation Multiconference, 1999

[7] Dittrich, D., "The DoS Project's "trinoo" distributed denial of service attack tool". October 21, 1999, http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt - visited 4.16.2003.

[8] Dittrich, D., "The 'Tribe Flood Network' distributed denial of service attack tool". October 21, 1999, http://staff.washington.edu/dittrich/misc/tfn.analysis.txt - visited 4.16.2003.

[9] Dittrich, D., "The 'stacheldraht' distributed denial of service attack tool". December 31, 1999, http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt visited 4.16.2003.

[10] Myrinet Product information at: http://www.myri.com

[11] Trebia Networks, SNP-1000i Dual Port TCP Offload Engine Product Literature, 2003.

[12] Adaptec Corpn., ANA.7711 TCP/IP Offload Adaptor Product Info, 2003.

[13] Ghose, K., Melnick, S., Gaska T., et al. The Implementation of Low Latency Communication Primitivesin the SNOW Prototype, In Proc. of the 26.th. Int.l. Conference on Parallel Processing (ICPP), 1997, pp.462.469

[14] Bunitinas, D., Panda, D.K., and Sadayappan, P., Fast NIC Based Barrier Over Myrinet/GM., in Proc. Int.l Parallel and Distributed processing Symposium, 2001

[15] Krishnamurthy, R, Schwan, K. et al, A Network Co-processor Based Approach to Scalable Media Streaming in Servers., in Proc. Int.l. Conf. on Parallel processing (ICPP), 2000

[16] Noronha, R. and Abu Ghazaleh, N., .Early Cancellation: An Active NIC Optimization for Time Warp., in Proc. PADS 2002.

[17] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In Proceedings of the International World Wide Web Conference, pages 252--262. IEEE, May 2002

[18] Kargl, F., Maier, J., Weber, M. and Schlott, S., "Protecting web servers from distributed denial of service attacks," In Proceedings of 10th International World Wide Web Conference, May 2001

[19] Lemon, J. "Resisting SYN Flooding DoS Attacks with a SYN Cache", Proceedings of USENIX BSDCon'2002, February 2002.

[20] Ferguson, P. and Senie, D., "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing", RFC 2267, January 1998.

[21] Garg, A. and Narasimha Reddy, A.L., "Mitigating denial of service attacks using QoS regulation," Texas A & M University Tech report, TAMU-ECE-2001-06

[22] Martyn Williams, IDG News Service, 02/09/00 , EBay, Amazon, Buy.com hit by attacks,

[23] http://www.nwfusion.com/news/2000/0209attack.html

[24] Begel, A., McCanne, S., and Graham, S.L., "BPF+: Exploiting Global Dat flow Optimization in a generalized Packet Filter Architecture", in Proc. of SIGCOMM 99, 1999.

[25] Gil, T.M. and Poletteo, M., "MULTOPS: a data-structure for bandwidth attack detection", Proceedings of USENIX Security Symposium'2001, August 2001.

[26] D.R. Morrison. Patricia--practical algorithm to retrieve information coded in alphanumeric. Journal of ACM, 15(4):514--534, Jan 1968.

[27] Jozsef Kadlecsik, Harald Welte, James Morris, Marc Boucher, and Rusty Russell. Iptables, http://www.iptables.org