Proceedings of the IASTED International Conference
Communication, Network, and Information Security
October 9–11, 2006, MIT, Cambridge, MA, USA
ISBN CD: 0-88986-636-8

PERFORMANCE STUDIES OF THE SERVER-SIDE ACCESS CONTROL FOR SYN-FLOODING DISTRIBUTED DENIAL OF SERVICE ATTACKS USING REAL SYSTEMS

Hiroshi Fujinoki

Department of Computer Science
Southern Illinois University Edwardsville
Edwardsville, Illinois 62026-1656, USA
E-mail: *hfujino@siue.edu*

Ravi Kumar Boyapati

Department of Electrical Engineering
Southern Illinois University Edwardsville
Edwardsville, Illinois 62026-1656, USA
E-mail: *rboyapa @siue.edu*

ABSTRACT
This paper presents our on-going project on performance evaluation of the major existing solutions based on server-side access control for SYN-flooding distributed denial-of-service attacks using a real network system. Although many solutions have been proposed and implemented, there is no formal performance study that measures and compares the solutions based on server-side access control. The successful connection rate of the existing solutions was measured, compared and analyzed using an experiment test bed developed by LINUX-based PCs. We have tested SYN-cookie, Random Drop and the unmodified TCP in various conditions. We also simulated different types of legitimate clients in the end-to-end signal propagation delay to evaluate the fairness in connections. The results of our experiments showed that SYN-cookie resulted in the perfect (i.e., 100%) connection rate in all the experiments and configurations. Regardless of the length of the end-to-end delay, the connection rate of the unmodified TCP dropped to below 5% for a low request rate of 50 requests per second or more. Random Drop was more effective in improving connection rate than the unmodified TCP if the end-to-end delay was short or when the TCP backlog queue size was increased to more then 300 slots.

KEY WORDS
Network security, access control, denial-of-service attacks, TCP SYN-flood attacks, flash crowd

1. Introduction

Denial of service attacks have been serious security hazards in the recent years. There seems to be no ultimate solution for the attacks primarily because of the following three reasons. First, new variants of the attacks keep emerging and sophistication of the attacks continues to increase. Second, the major trend has shifted to distributed versions, called distributed denial-of-service (DDoS) attacks, where the only distinction from legitimate accesses is intention [1]. Third, lack of provision for built-in security features in the existing network protocols makes implementing robust security enhancements quite difficult. Despite the recent research activities, large commercial network sites tend to be targets of these crimes, resulted in huge losses to the organizations and our social damages from this type of security hazards still keep increasing.

DDoS attacks have been performed in many different ways [2]. Major past DDoS attacks take the forms of bandwidth consumption (such as bot-nets [1] and reflector attacks [3]), exhausting server-side local resources (SYN-flooding), and exploiting known problems in end hosts (ping-to-death) and in the network infrastructure software and hardware configurations [4]. The recent trend is combinations of any of the above attacks (such as DNS DDoS attacks [5]). This project focuses on solutions for SYN-flooding DDoS attacks.

SYN-flooding attacks exploit vulnerability in resource assignment in TCP three-way handshaking that is required each time a new TCP connection is established. TCP three-way handshaking consists of three message exchanges between a client and a server host. First, an initiating client sends a message called, SYN message, which carries an unpredictable sequence number generated by the client (called client sequence number). When a SYN message reaches a server, the server places the SYN message in a queue, called TCP backlog queue and generates its own unpredictable sequence number (server sequence number). The server constructs the second message, SYN-ACK message, which contains the client and server sequence numbers. Then the SYN-ACK message is transmitted to the client. On receiving the SYN-ACK message, the client confirms the willingness for a connection from the server and transmits the third message, which contains the next client sequence number, the server sequence number in the SYN-ACK message and the first payload message to the server. Finally, the server establishes a TCP connection on the arrival of the third message.

In SYN-flooding attacks, attackers dump a large number of SYN messages with their source address and port number spoofed. Attacking hosts will never complete the TCP three-way handshaking (i.e., never respond to the SYN-ACK message), which will clog the TCP backlog queue. A server will refuse any new connection by transmitting RST message back to clients for any new SYN message while its backlog queue is full. SYN messages are simply dropped and RST message (which means "reset request" from a server) is sent back to each connecting client. When a client receives RST message from a server, the client must drop the TCP connection. Although suspended SYN messages in the queue will eventually expire and will be removed, no other authorized clients will be able to establish a TCP connection while the backlog queue is full.

SYN-flooding attacks are easy to perform but difficult to defend because of the following reasons: TCP backlog queue is limited up to at most around 30 slots in most of the commercial operating systems. SYN messages are small (this means the attackers do not need high transmission bandwidth) and spoofing IP and TCP headers are easy (spoofing tools are easily accessible in the underground communities). Expiration interval is 75 seconds by default and it can not be short because of legitimate clients with long end-to-end delay and high fluctuations in end-to-end delay in the Internet.

Hardaker categorized the major existing solutions in network based, source based and end-point based solutions [6]. Network based solutions are those that drop or avert attacking traffic before they reach target servers, or hide the location of the servers from attackers. Ingress filtering [7], router rate limiting [8], security network overlay [9], load-balancing using content distribution networks [10], back tracing [3], and remote firewalls [11] belong to this category. Source based solutions try to drop attacking traffic before one is injected to the public Internet domain. Egress filtering [12] is an example. End-point based solutions control inflow of incoming traffic to prevent or mitigate the effect of DDoS attacks. Access control [13], server-side rate limiting [14], and graphic testing and authentication [1] are the examples in this group.

Although each approach has its own strengths and weaknesses, we focus on the end-point based approach due to the following two reasons:

1. The network based solutions require the support in the level of the entire network infrastructure to take the expected effect. Required cost for implementing such a solution will be prohibitively expensive or one takes a long transition delay before complete installation.
2. The source-based solutions will not protect the organizations that install this type of protections. As a common disadvantage to the network based solutions, an attacked host does not have any control to the way it is protected. Source based solutions will not be efficient, especially for DDoS attacks.

In the contrast, end-point based solutions have the following advantages. Since a protection requires only local installation, one will be a quick relief for new attacks. Significance in direct controls under attacks can be seen in some real stories of DDoS attacks [15]. A solution can be flexibly adjusted and tuned, allowing administrators customize a solution for each host or network domain. A solution does not depend on other organizations. Due to lack of dependency to other organizations, long-term running cost will be relatively lower than the solutions in the other two categories.

One of the end-host approaches is server-side access control. A lot of work has been performed and published regarding the performance and efficiency in access control based solutions for SYN-flooding DDoS attacks. Not only individual and corporate projects, some police and security agencies contributed to this field [16]. However, to the best of the authors' knowledge, Lemon's work is the only project studied the performance of SYN-cache and SYN-cookie [17]. Moreover, there is no performance study that covers major access control based solutions for SYN-flooding DDoS attacks using a common experimental foundation. This project developed and performed experiments using a real system to compare the performance of the major access control based solutions for SYN-flooding DDoS attacks to answer the following questions:

- Some of the existing solutions have never been formally tested using a real network. Are they as good as claimed by the papers even in a real environment?
- What are the characteristics in each solution? What are the relative weaknesses and strengths as a protection for SYN-flooding DDoS attacks in real use?
- Are the existing solutions feasible and practical even in a real environment? If not, what are the problems?

The rest of this paper is organized as follows. In Section 2, the major existing protection methods against SYN-flooding attacks that are based on the server-side access control are described and analyzed for their know advantages and disadvantages. Section 3 presents the performance evaluations for the major protection methods using a real network. Our experiment test bed and experiment design are described in the details that allow our audience to reproduce our experiments. Section 4 discusses the conclusions, contributions from this project, as well as possible future work, followed by a list of selected references.

2. Major existing solutions

The major existing solutions based on the server-side access control are, Berkeley cookie, RST-cookie, SYN-cookie, Random Drop, Probabilistic Pre-filtering Random Drop, and SYN-cache.

Berkeley-cookie [18]: Berkeley cookie increases the backlog queue size. The primary advantage is that this solution does not require any change in TCP, application and network setting. However, as the improvement in network speed is outpacing Moore's Law, (Gilder's Law predicted the network transmission rate would double in every 9 to 12 months [19]), this solution will not be a practical option soon in the near future.

RST-cookie [20]: A server will always respond with RST message and drop the request when one receives a new SYN message for the first time. On an arrival of a RST message, a client is expected to retry for the connect ion if the client is a legitimate client. RST-cookie was designed to cope with the classical single-attacker DoS attacks, where an attacking host continuously dumps SYN messages with different spoofed source addresses. The problem in RST-cookie is that servers have to remember all the dropped SYN messages until a RST is replied on the first request. For the servers with wide bandwidth connection, this, itself, can be another target for attacks.

SYN-cookie [21]: SYN-cookie solves the problem in RST-cookie. When a server receives a new SYN message, it replies with a SYN-ACK message without holding the SYN message in the server. Instead, a server calculates a hash key using the identification of the request (such as the client's source IP and port number). A server includes this hash key in its SYN-ACK message and recalculates it when the reply for this SYN-ACK comes back from this client. Then the server recalculates the hash key to find if the hash-key in the reply and the recalculated one match. Only if they match, a TCP connection is established. The primary advantage in SYN-cache is the throughput. Since a server will not keep anything, but only the overhead for calculation of a hash-key and constructing a SYN-ACK message is required, the procedure to establish a TCP connection will not be the bottleneck as long as the processor is fast enough. This will make DoS attacks that exhausts server resources hard to succeed. One of the known problems in SYN-cookie is the lack of full compatibility to the existing implementation of TCP. Since the SYN message in the three-way handshaking is discarded as soon as the SYN-ACK message is transmitted by a server, the information in the TCP header (such as initial open window size, the TCP flags, URG pointer and possible options) will be lost, resulting in an incomplete implementation of the TCP protocol [22].

Random-Drop [23, 24]: The cause of denial of services is that TCP will refuse any new SYN message once its backlog queue becomes full (all the new incoming SYN messages are dropped). RD attacks this problem by randomly dropping a SYN message in the backlog queue to make a space for a new one, hoping that the one dropped be the one issued by an attacker. The problem in this method is that the one randomly dropped can be the one from a legitimate client, and it will be the one from a legitimate client under a flash crowd, resulting in a problem similar to receive live-locking [25].

Probabilistic Pre-filtering Random Drop (PP-RD) [26]: PP-RD is a solution for the problem of the possible preemption of legitimate SYN requests in RD. PP-RD minimizes preempting legitimate requests in the TCP backlog queue by controlling the number of requests that reach the backlog queue when high preemption rate is expected. PP-RD controls the number of requests that reach the TCP backlog queue by randomly pre-filtering incoming SYN messages. The survival rate is dynamically adjusted to minimize preemption using the following formula:

$$K = \frac{-1}{\ln(1-1/q)\cdot(R_{good}+R_{bad})\cdot T} \qquad (1)$$

where R_{good} is the request rate (i.e., number of SYN messages arrived in a second) by legitimate clients and R_{bad} is that of attacker(s). The parameter, q, represents the number of slots in the TCP backlog queue while T is the average response time to the SYN-ACK message by legitimate clients. By controlling the number of SYN messages that will reach the TCP backlog queue, preemption of legitimate clients' SYN messages can be decreased. The primary problem in PP-RD is that minimizing preemption does not necessarily maximize the connection rate for legitimate clients. That is because attacking requests will reach the TCP backlog queue and TCP does not distinguish malicious requests from legitimate ones.

SYN-cache [17]: The concept of SYN-cache was first proposed by Lemon. SYN-cache utilizes the fact that the same clients will repeatedly access a server multiple times in a short time in some network applications, such as web and online database. As long as the target of attacks is the TCP backlog queue, most of the connection requests from attackers will not complete the procedure of the TCP three-way handshaking. The local cache behind the TCP backlog queue will remember the identifications of the clients who successfully establish a connection, assuming that they are legitimate users. Similar to Berkeley Cookie, the primary advantages in this method is completely transparent to the existing TCP and no change is necessary. Possible pathological cases include limitation by the cache hit rate, lack of flexibility in support for general network applications and potentially high overhead in finding cached entries especially for high-speed connections.

3. Performance Evaluation

The experiment test bed was constructed with three high-performance desk-top PCs connected by links through a switch (Figure 1). Two of the desktop PCs (ACK generator and the server) had Intel Pentium III, 750MHz and 256MB memory and were connected to the switch by a 100Mbps CAT-6 UTP cable. The request generator had Intel Xeon 1.7GHz and 512MB memory with 64-bit wide PCI interfaces and was connected to the switch using a 1Gbps connection. The operating system used was RedHat LINUX 9 (2.4.20-8 kernel) for all the three hosts. Three feet CAT6 UTP cables were used for all the three links. The switch used to connect the three hosts was CISCO Catalyst 3500-XL switch (an informal performance analysis shows that this switch is capable of forwarding frames in the wire speed for 128-byte or larger frames [27]). The experiment test bed was isolated from other networks to prevent uncontrolled traffic from disturbing our experiment results.

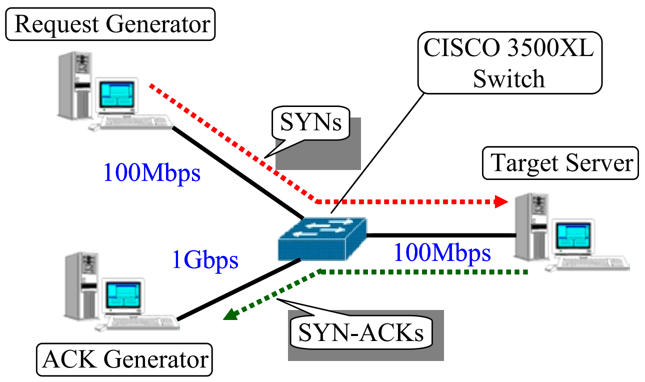

Figure 1 – Experiment test bed network

The request generator issued SYN requests to the server, each of which carried a different source IP address to simulate large population groups of legitimate and attacking clients. We simulated three different groups of legitimate clients (described later) and a group of attackers. Different IP address ranges were used to identify activities and to make statistics in each client group. The server host simulated a host attacked by SYN-flooding DDoS attacks while legitimate clients access to it. The server responded the SYN requests by SYN-ACK messages and those SYN-ACK messages were forwarded to the ACK generator. We let the server to forward all of its SYN-ACK messages to the ACK generator by setting the address of the default gateway router at the server to that of the ACK generator.

The ACK generator recognized the client group for each SYN-ACK from the server, simulated the response time expected for each client in each legitimate client group, and replied to the server with the ACK message to complete the TCP three-way handshaking. The ACK generator simply dropped all SYN-ACK messages that were issued by attackers. Figure 2 shows the internal organization of the ACK generator. The ACK generator had three queues, one for each legitimate client class to simulate the different response time and to collect statistics. We implemented the Request Generator and the ACK Generator using a combination of raw sockets and the promiscuous mode.

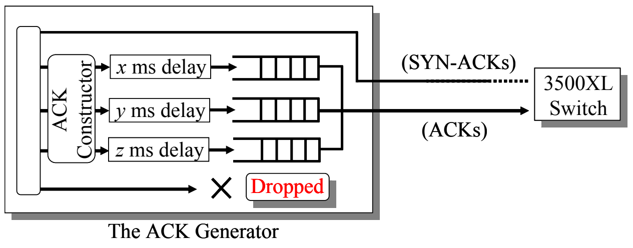

Figure 2 – Internal organization of the ACK generator

We measured various performance factors for the existing end-point based solutions against SYN-flooding attacks. The performance metrics in our experiments was the successful connection rate, which was the ratio of the successfully connected SYN requests to the SYN requests issued by the legitimate clients. The control parameters common to all the tested end-point based solutions are: SYN request rate, backlog queue size and the ratio of the SYN requests generated by attacking hosts and the legitimate clients (we called this parameter “A:L ratio”).

The SYN request rate was the number of SYN requests issued to the server host and it was controlled by the interval between two successively generated SYN requests. Each interval between two SYN requests was generated based on a given SYN rate (e.g. 1,000 SYN/sec interval =1ms). The TCP backlog queue size was the maximum number of slots the TCP backlog queue had.

To control A:L ratio, we randomly assigned attacking and legitimate clients using the uniform random number generator. The A:L ratio was controlled in the following way. For the A:L ratio of 1:4, uniform random numbers were generated with a range between 0.0 and 1.0. For any random number between 0.0 and 0.2, we marked one as a SYN message generated by an attacker while for the numbers above 0.2, we marked one as a SYN from a legitimate client.

We used another parameter, the round trip time (RTT), to simulate the delay for the client reply to SYN-ACK messages (generated by the server) that come back to the server. RTT was controlled to simulate different types of legitimate clients (described in the next paragraph). To model various RTT, fixed amount of delay was inserted before the ACK message was transmitted back to the server at ACK Generator. The amount of inserted delay depended on the type of legitimate clients (e.g., 70ms for the fast, 300ms for the slow, and 900ms for the congested clients).

Profile of legitimate client demography can vary in each web site. For example, for a web site, majority of customers may have high-speed connections, such as direct Ethernet connection, cable or DSL modems, while the opposite is also quite possible. Not only for connection speed, the end-to-end delay can greatly vary. For example, some customers use high-delay satellite connections or some may be located at the other side of the earth. Fortunately, all the above variations can be summarized by single parameter, the round-trip time, as long as the performance of access control against SYN flooding DDoS attacks is concerned. To simulate different types of legitimate clients, we defined three legitimate client groups. Class-F (Fast) legitimate clients are those that have a high-speed connection and are located close (in link distance) to the server (the end-to-end signal propagation delay is short). Class-S (Slow) clients are those who do not have access to a high-speed connection and/or are located far from the server. Class-G clients are those whose connection goes through a congested links. The exact parameter settings for the base configuration for the common parameters are shown in

Table 1.

Table 1 - Parameter settings for the base configuration

Control Parameters	Base Configuration
Attacking:Legitimate Ratio	1:1 (A:L)
Legitimate Client Demography	2:1:1 (F:S:G)
SYN Request Rate	1,000 SYN/seconds
TCP Backlog Queue Size	64 SYN slots
TCP SYN-RECEIVED timer	90 seconds
Average RTT (for class F, S, G)	70ms, 300ms, 900ms
Number of SYNs Simulated	10,000 SYN requests

The exiting solutions tested were, SYN-cookie, and Random Drop (RD). The existing TCP (without any protection) was also tested as a reference. The existing solutions were implemented in the following way. SYN-cookie and RST-cookie have an existing implementation available for RedHat LINUX. We used those existing implementations for our experiments. We implemented RD by modifying the LINUX 2.4.20-8 kernel. The source codes for our implementation of the three solutions are available [29]. The following experiments were performed.

Experiment #1 (“**SYN-Rate Experiment”**): In this experiment, the SYN request rate was changed in the base configuration to observe the successful connection rate for the existing solutions. The SYN rate was changed from 12.8 to 1,250 requests per second (the tested SYN rates: 12.8, 64, 128, 250, 400, 600, 770 and 1,250). The rate was controlled by adjusting the average interval time between two SYN requests at the request generator. The same experiment was repeated 20 times and the results were averaged (same for the other experiments).

Experiment #2 (“Backlog Queue Size Experiment”): The backlog queue size was changed while all the other parameters in the base configuration remained same. The backlog queue size was changed from 64 to 1,024 slots. Note that this was expected not to have any impact to SYN-cookie, since SYN-cookie does not logically use the TCP backlog queue.

Experiment #3 (“Attacking:Legitimate Ratio Experiment”): The request rate of attacking to legitimate requests was changed from (A:L = 6:1) to (0:1). The request rate of (0:1) means a flash crowd where all requests were from legitimate clients, while (6:1) means approximately 86% of the incoming SYN requests were issued by attacking hosts. By performing the experiment for various A:L ratio, the effectiveness of each of the tested solutions to flash crowds, low-density attacks, and high-density attacks was studied.

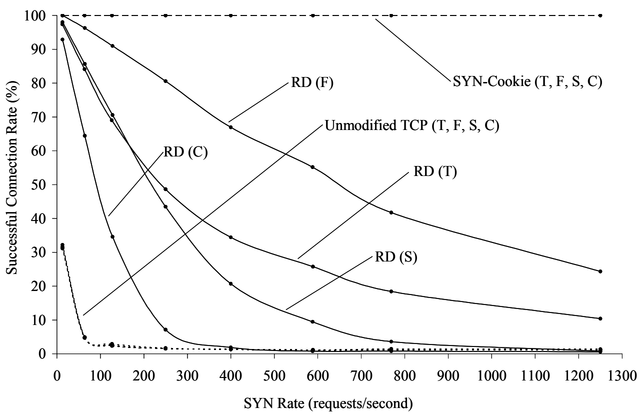

Figure 3 – Observed connection rates in the SYN-rate experiments (Experiment #1)

Figure 3 shows the successful connection rates for the three existing solutions for various SYN request rates (Experiment #1). SYN cookie resulted in 100% connection rate for all the three classes of the legitimate clients and for the SYN rate up to 1,250 requests/second. On the other hand, the figure shows that the connection rate for the unmodified TCP dropped to below 5% for a relatively light request rate (at 64 requests/second) regardless of the client classes (F for fast, S for slow, C for congested and T for all clients). A strong correlation was observed between the required RTT and the successful connection rate in RD. The declining slope ((decrease in connection rate in percentage)/(absolute increase in the number of requests per second)) before “knee” was 0.06 between 12.8 to 1,250 slots for the fast clients (calculated as 100 (as % at 12.8 slots) – 24 (as % at 1,250 slots) / (1,2590-12.8)), 0.19 between 12.8 to 400 slots for the slow clients, and 0.36 between 12.8 to 200 slots for the congested clients respectively.

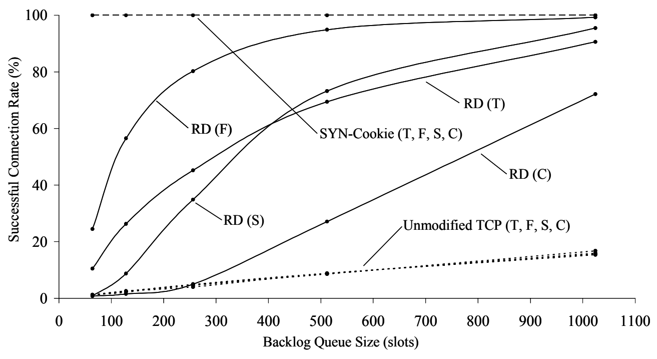

Figure 4 – Observed connection rates in the backlog queue size experiments (Experiment #2)

Figure 4 visualizes the results of the backlog queue size experiment (Experiment #2). The connection rate was measured for first (F), slow (S), congested clients (C) and all clients (T). SYN-cookie resulted in 100% connection rate for all the three legitimate client classes and all the tested backlog queue size. In RD, the connection rate for the fast clients increased significantly (24.5% at 64 slots to 94.9% at 512 slots) while it resulted approximately in a steady linear growth for the congested clients (0.8% at 64 slots to 72.1% at 1,024 slots). It was observed that RD did not effectively improve the connection rate for small backlog queue (64 to 256 slots), while it resulted in significant improvement for the fast clients. The improvement in the unmodified TCP was linear but slower than that of RD. From the figure, it can be seen that the unmodified TCP was not efficient in utilizing a large TCP backlog queue compared to RD.

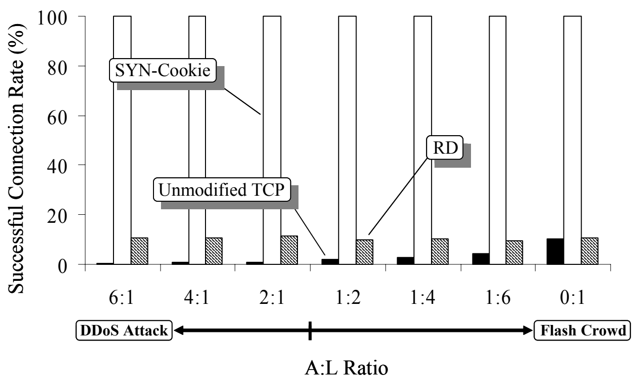

Figure 5 – Observed connection rates in the attacking: legitimate ratio experiments (Experiment #3)

Figure 5 shows the results of the attacking:legitimate ratio experiments (Experiment #3). Similar to the previous two experiments, SYN-cookie resulted in 100% connection rate for all the three legitimate client classes and all configurations of A:L ratio tested. The unmodified TCP resulted in a low connection rate where attackers were the majority (6:1), while the connection improved as the population ratio of the attackers to the legitimate clients reduced to a flash crowd (0:1). These results demonstrated the inefficiency of the existing TCP against SYN flooding denial of service attacks. RD resulted in a steady connection rate of around 10% regardless of the A:L ratio, while the connection rate was in the same level as the unmodified TCP for the pure flash crowd (0:1).

4. Conclusion and Future Work

This paper presents our on-going project that focuses on performance evaluation of the major existing server-side access controls for SYN flooding distributed denial of service attacks using a real network system. We have tested SYN-cookie, Random Drop (RD) and the unmodified TCP to observe the successful connection rate for different SYN request rate, backlog queue size and attacking:legitimate client ratio to understand the relative efficiency of SYN-cookie and RD compared to the unmodified TCP. We also simulated different classes of legitimate clients in the round-trip end-to-end delay to evaluate the fairness in the existing solutions.

We observed that SYN-cookie resulted in the perfect connection rate (i.e., 100%) in all the experiments performed in this project. Those results suggested that the only problem in SYN cookie is lack of support for TCP options in the first SYN message in the TCP three-way handshaking. Regardless of the types of the legitimate client classes, the connection rate of the unmodified TCP dropped to below 5% at a low request rate of 50 requests/second. The result of Experiment #3 revealed that the unmodified TCP handled flash crowds better than SYN-flood DoS attacks. Results of Experiment #2 suggested that the unmodified TCP will not efficiently take advantage of a larger TCP backlog queue.

Random Drop was more effective in improving connection rate than the existing TCP when the end-to-end delay was short (Experiment #1) or when the TCP backlog queue size increased (Experiment #2). Although Random Drop maintains the full compatibility to the existing TCP implementation, it did not effectively protect a server in our worst-case scenarios, such as high request rate, small backlog queue, and high-density denial-of-service attacks. Currently, we are conducting Experiment #4 and implementing RST cookie, PP-RD, and Cached GT-RD [28] for the test bed described above.

References

[1] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur Bergerm "Botz-4-Sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds," *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, May 2005.

[2] A. Hussain, J. Heidemann, and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks," *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 99-110, 2003.

[3] S. Savage, D. Wetherall, A. Karlin and T. Anderson, "Practical Network Support for IP Traceback," *Proceedings of ACM SIGCOMM*, pp. 295-306, 2000.

[4] S. Cheung and K. Levitt, "Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection," *Proceedings of New Security Paradigms*, pp. 23-26 September 1997.

[5] K. Rikitake, K. nakano, H. Nogawa, and S. Shimojo, "T/TCP for DNS: A Performance and Security Analysis," *Journal of Information Processing Society of Japan*, vol.44, no. 8, pp. 2060–2071, August 2003.

[6] Wes Hardaker, Darrell Kindred, Ron Ostrenga, Dan Serene, and Roshan Thomas, "Justification and Requirements for a National DDoS Defense Technology Evaluation Facility," *Network Association Laboratories Report* #02-052, 2006.

[7] P. Ferguson, and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing"; *RFC-2267*; January 1998.

[8] J. Ioannidis and S. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," *Proceedings of Network and Distributed Systems Security Symposium*, pp. 79-86, February 2002.

[9] Angelos Stavrou, Debra L. Cook, William G. Morein, Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein, "WebSOS: An Overlay-based System for Protecting Web Servers from Denial of Service Attacks," *The International Journal of Computer and Telecommunications Networking*, vol. 48, no. 5, pp. 781-807, 2005.

[10] J. Jung, B. Krishnamurthy, and M. Ravinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," *Proceedings of the International World Wide Web Conference*, pp. 252-262, 2002.

[11] A. Keromytis, S. Ioannidis, M. Greenwald and J. Smithy, "The STRONGMAN Architecture," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 17-31, May 1999.

[12] C. Brenton, "What is Egress Filtering and How can I Implement it?," *The SANS Institute,* URL:*http://www.sans.org/rr/paper.php?id=1059,* February 2000.

[13] T. Peng, C. Leckie and K. Ramamohanarao, "Protection from Distributed Denial of Service Attack Using History-based IP Filtering," *Proceedings of the IEEE International Conference on Communications*, pp. 1-5, 2003.

[14] X. Chen and J. Heidemann, "Flash Crowd Mitigation via an Adaptive Admission Control Based on Application-Level Measurement," *Technical Report ISI-TR-557, University of Southern California/ Information Sciences Institute*, May 2002.

[15] Denial of Service Investigation & Exploration Pages, *Gibson Research Corporation*, URL: *http:// grc.com/dos/intro.htm*, June 2005.

[16] "Regarding DoS/DDoS Protections," *@Police Cyber Forth Security Report*, *Japan National Police Agency*, March 2004. URL: *http://www. cyberpolice.go.jp/server/rd_env/pdf/DDoS_Inspec tion_2.pdf*

[17] Jonathan Lemon, "Resisting SYN Flood DoS Attacks with a SYN Cache," *Proceedings of USENIX BSDCon*, pp. 89-98, April 2002.

[18] Berkeley Software Design Inc., "BSDI Releases Defense for Internet Denial-of-Service Attacks," URL: *http://www.bsdi.com/press/19961002.html*, October 1996.

[19] P. Markatos, "Speeding-up TCP/IP: Faster Processors Are not Enough," *Proceedings of the 21st IEEE International Performance, Computing, and Communications Conference*, pp. 341-345, April 2002.

[20] E. Shenk, "Another new thought on dealing with SYN flooding," URL: *http://www.wcug.mmu.edu /lists/netdev/199609/msg00171.html*, September 1997.

[21] D. J. Bernstein, "SYN Cookies," URL: *http://cr.yp. to/syncookies.html*, September 1996.

[22] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," *Internet Draft*, *The Internet Society*, April 2006.

[23] L. Ricciulli, P. Lincoln, and P. Kakkar, "TCP SYN Flooding Defense," *Communication Networks and Distributed Systems Modeling and Simulation*, 1999.

[24] Sun Microsystems, "SUN's TCP SYN Flooding Solutions", URL: *http://ciac.llnl.gov/ciac/bulletins/ h-02.shtml*, October 1999.

[25] Jeffrey C. Mogul and K.K. Ramakrishnan, "Eliminating Receive Livelock in an Interrupt-driven Kernel," *Proceedings of the USENIX Annual Technical Conference*, pp. 99-111, 1996.

[26] A. Cox, "Linux TCP Changes for Protection against the SYN attacks," URL: *http://www.wcug.wwu.edu/ lists/netdev/199609/msg00091.html,* September 1996.

[27] "Catalyst 3500XL Switch Architecture", White Paper, Cisco Systems, Inc., URL: *http://www.nasi.com /pdfs/cisco3500xldatasheet.pdf,* April 2006.

[28] Hiroshi Fujinoki, "Cached Guaranteed-Timer Random-Drop against TCP SYN-flood Attacks and Flash Crowds," *Proceedings of the IASTED International Conference on Communication, Network, and Information Security*, pp. 162-169, November 2005.