Mitigating DoS Attack Through Selective Bin Verification

Micah Sherr[†], Michael Greenwald[‡], Carl A. Gunter[⋆], Sanjeev Khanna[†], and Santosh S. Venkatesh[†]

[†] School of Engineering and Applied Science, University of Pennsylvania
{msherr,sanjeevk,venkatesh}@seas.upenn.edu

[‡] Bell Labs
greenwald@research.bell-labs.com

[⋆] Department of Computer Science, University of Illinois at Urbana-Champaign [*]

Abstract

Despite considerable attention from both the academic and commercial communities, denial-of-service (DoS) attacks represent a growing threat to network administrators and service providers. A large number of proposed DoS countermeasures attempt to detect an attack in-progress and filter out the DoS attack packets. These techniques often depend on the instantiation of sophisticated routing mechanisms and the ability to differentiate between normal and malicious messages. Unfortunately, neither of these prerequisites may be practical or possible.

We propose and evaluate a defense against DoS attacks which we call selective bin verification*. The technique shows promise against large DoS attacks, even when attack packets are able to permeate the network and reach the target of their attack. We explore the effectiveness of our technique by implementing an experimental testbed in which selective bin verification is successfully used to protect against DoS attacks. We formally describe the mathematical properties of our approach and delineate "tuning" parameters for defending against various attacks.*

1. Introduction

The increasing number and severity of denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks have become a growing annoyance to network administrators and service providers [7]. Consequently, there has been a significant body of work in which countermeasures are proposed to alleviate or eliminate the effects of these attacks. Typically, human operators defend against DoS by augmenting their networks with additional equipment or by employing some filtering mechanism that prevents malicious packets from reaching their target [5].

While these approaches may be appropriate for several types of DoS attacks, they unfortunately are not always applicable or practical. The deployment of additional computational and network resources may be technologically or financially infeasible. Techniques that discriminate between normal and malicious traffic assume that these two traffic streams are somehow distinguishable. For some protocols, it may be prohibitively expensive to reliably make this distinction.

In this paper, we study the effectiveness of a novel countermeasure to DoS called *selective verification*. First introduced in [4], our technique allows the receiver to tolerate large DDoS attacks while providing service to legitimate senders. Unlike more orthodox approaches in which success is measured by the ability to prevent bogus requests from entering the targeted network, selective verification is effective even when attack packets reach the receiver.

In particular, this paper investigates a class of selective verification called *selective bin verification*. While previous work [4] has illustrated the ability of *sequential selective verification* to protect authenticated broadcasts from DoS, we show that bin verification is an effective technique to protect point-to-point protocols.

The remainder of the paper is organized as follows. In the following section, we discuss related work. In Section 3, we describe the selective bin verification protocol. To test the efficacy of our approach, we describe and analyze various selective bin verification experiments in Section 4. We conclude and discuss areas of future research in Section 5.

[*]In: Workshop on Secure Network Protocols (NPSec), November 2005, Boston, Massachusetts.

2. Related Work

Much prior work has investigated the problem of protecting against DoS. Existing defenses generally fall into two categories: those that mitigate the attack by adding additional resources and those that attempt to differentiate between legitimate and malicious (or anomalous) traffic. While the former approach is often used in practice to mitigate the effects of an ongoing attack [2], most of the academic literature has focused on the latter category.

In particular, automated detection and response mechanisms present promising defenses. Network intrusion detection systems [8] are effective at detecting certain types of DoS attacks. However, approaches that rely on signature or anomaly detection make the assumption that the attack packets are distinguishable from the legitimate traffic. While this may be true for certain attacks (e.g., "smurf" attacks [3]), other DoS techniques merely inundate a service with requests that appear valid.

Additional work has focused on curbing DoS attacks at the source [5, 6]. Here, a router detects an anomalous traffic flow and actively responds to the attack. However, since distributed DoS attacks are often carried out by unwitting zombie hosts that may be dispersed throughout the Internet, this task is non-trivial without the widespread adoption of cooperating DoS-aware routers.

An important advantage of selective bin verification is that it reduces the effects of a DoS attack even in the case where the attack fails to be detected. It is a software-based approach and requires no hardware modifications. While our technique is not a panacea to the DoS problem, its use in conjunction with existing approaches may significantly reduce the effects of many DoS attacks. In the section that follows, we describe our technique more formally.

3. Selective Bin Verification

3.1. Sequential Selective Verification

Selective verification has previously been introduced as an effective approach to thwarting DDoS attacks [4]. Specifically, the focus of the prior work was to alleviate the effects of a DoS attack against an authenticated broadcast stream. The authors considered a protocol in which the receiver must exert significant processing resources to authenticate a message (for example, carry out a public-key operation). Furthermore, they imposed the restriction that the attack traffic attempted to overwhelm the computational resources of the receivers but was not able to prevent all legitimate traffic from being delivered. Under such an attack model, there is a disparity between the bandwidth used by the sender and by the attacker. While the sender broadcasts only the legitimate stream, the attacker must flood the network with superfluous packets in order to mount an effective attack.

Selective verification exploits this asymmetry to protect the broadcast message stream from a DoS attack. Rather than process all arriving packets, a host using selective verification processes only a subset of the received packets. To ensure that legitimate messages are not lost, the sender transmits multiple copies of each message.

Upon first inspection, selective verification appears to worsen the DoS attack by increasing the amount of traffic received by the receiver. However, due to the asymmetry in the cost of sending versus processing messages, the large reduction in processing cost outweighs the increase in network traffic during an attack. For the sender, sending duplicate copies of a packet is cheap. Conversely, the receiver must exert significant computational resources to process messages. To alleviate this potential high cost, the receiver randomly drops packets with probability p (randomly dropping packets is, again, a cheap operation). On average, the receiver still may process multiple copies of legitimate messages (nonces may be used to quickly discard duplicates), however, DoS packets are also discarded with probability p. Thus, to sustain the same level of attack, the adversary must increase his traffic by a factor of $1/p$. While the legitimate senders must also send multiple copies of their messages, senders use only a small fraction of the receiver's bandwidth during a DoS attack and transmitting duplicates adds little overhead to either the sender or the receiver. The primary cost of using selective verification is therefore borne by the attackers.

3.2. Selective Bin Verification

In this work, we explore the effectiveness of using selective bin verification to defend against DoS attacks. Here, we consider a typical client-server system in which multiple independent clients (senders) simultaneously access a single server (the receiver). Concurrently, the system may be subject to a DoS or DDoS attack in which the receiver is inundated with seemingly valid requests.

Like sequential verification, bin verification operates by processing only a randomly-selected subset of received messages. However, unlike sequential verification in which all messages are stored in the same queue, bin verification utilizes a set of n bins, labeled $0, \ldots, n-1$. When a legitimate sender transmits her request, she sends c copies of her message. Each copy includes a bin identifier (b) and successive copies are numbered sequentially starting from a randomly chosen initial point. Upon receiving a copy of the request, the receiver queues the message in bin $(b \bmod n)$. Each sender hence "stripes" the n bins with packet copies. If the sender packets are lost with probability l, then the average number of sender packets per bin is $(1-l)c/n$.

Attack packets are processed and assigned to bins in an identical fashion though, conservatively, we do not assume that attack packets are subject to loss.

After some fixed amount of time (which we call the *collection interval*), the receiver chooses k bins, where $k \leq n$. For example, a simple (and effective) technique is to choose the k bins that contain the least (but non-zero) number of queued messages. Regardless of how the bins are selected, the receiver proceeds by processing each message in the k bins. For each valid request (e.g., messages that are successfully authenticated through a digital signature), the receiver responds to the request according to the particular protocol being carried out[1].

3.3. Sequential vs. Bin Verification

While sequential verification is effective at protecting an authenticated broadcast stream against DoS, bin verification is more appropriate for protocols in which multiple senders simultaneously send requests to a single receiver. In the latter case (which is the impetus of this paper), the binning technique increases the probability that a sender and the receiver will be able to participate in a successful exchange.

Consider a configuration in which we have n bins and m senders. Suppose each sender sends n copies. In the absence of network loss, we are guaranteed that by choosing a single bin we satisfy all m senders. The receiver's load is therefore m, or 1 packet per sender. On the other hand, in sequential verification, in order to generate an expected load of 1 packet/sender, the receiver needs to discard packets with probability $(1 - 1/n)$. Then the probability that none of the packets of a sender are received is roughly $1/e$. Thus, approximately m/e senders will have none of their packets received. In other words, with a comparable work load, we expect only 63.21% of the senders to now succeed as opposed to 100% with binning.

The calculations remain unchanged in the presence of a DoS attack. To scale down the attack by a factor of n, the receiver can inspect one of his n bins. Here, the load on the receiver is $m + \gamma/n$, where γ is the number of received attack messages. Assuming no network loss, all sender requests will be processed.

3.4. Memory Requirements of Selective Verification

For bin verification to be most effective, the receiver must have sufficient memory to enqueue all arriving messages during the collection interval. With diminishing prices for memory and increasing speeds of hard disks (i.e., for virtual paging), we believe that processing power rather than memory is more often the scarce resource. (Many commodity hard disks have transfer rates exceeding 100Mb/s and could therefore be used to buffer packets when memory becomes unavailable.) Moreover, for many protocols, particularly those that rely on expensive cryptographic operations and are thus more susceptible to DoS (e.g., X.509 [1] and SSL/TLS), the bottleneck is the receiver's computational resources. For example, according to OpenSSL benchmarks we conducted on a 2.4GHz Pentium IV, the maximum number of signatures that can be computed per second using a 2048-bit key is only 34.4 for RSA and 122.0 for DSA, far below the number of requests that can be transmitted in one second using a 100Mb/s connection.

While bin verification is effective for computationally intensive protocols, sequential verification is more appropriate for protecting against attempts to overwhelm the memory resources of the receiver. In such cases, the receiver can discard packets with a probability sufficient to ensure that the nondropped packets can be buffered in memory.

In the remainder of this paper, we make the assumption that there is sufficient memory to hold all incoming messages. That is, we concentrate on attacks in which processing power is the scarce resource and show how bin verification can alleviate the effects of these attacks.

4. Experiments

We have conducted a series of experiments to verify the ability of selective bin verification to mitigate the effects of DoS. In each experiment, multiple senders attempt to carry out a modified version of the X.509 two-pass protocol [1].

4.1. Modified X.509 Two-pass Protocol

(1)	$A \rightarrow B$	:	$cert_A, D_A, S_A(D_A)$
(2)	$B \rightarrow A$	:	`OK`

where

$cert_A$	is a certificate from a certificate authority used for authenticating A's public key
D_A	is defined as $(r_A, B, P_B(k_1))$
$P_B(k_1)$	denotes the encryption of k_1 using B's public key
$S_A(D_A)$	denotes the signing of D_A using A's private key
r_A	is a nonce

Figure 1. Modified version of the X.509 protocol

The goal of the X.509 protocol is for a sender to securely transmit a symmetric key, k_1, to a receiver. As shown in Figure 1, a sender encrypts k_1 using the receiver's public key and signs the result using its private key. The message is then sent to the receiver, where the receiver ensures that

[1] Since multiple copies of a message may appear in the k bins, request identifiers may be used to both speed the processing of redundant messages and prevent non-idempotent requests from being executed multiple times.

the nonce is fresh, validates the signature (using the supplied certificate), and decrypts $P_B(k_1)$. If the receiver is successful in learning k_1, it transmits an `OK` message to the sender.

It is worthwhile to note that the cost of conducting the protocol is quite high for the receiver. For a request that contains a fresh nonce, the receiver must conduct an expensive signature check. An adversary who uses fresh nonces but sends random bits for D_A and $S_A(D_A)$ forces the receiver to perform this operation. Thus, the protocol's high cost causes the receiver to be particularly vulnerable to DoS. We therefore use the modified X.509 protocol to measure the ability of bin verification to mitigate the effects of DoS.

4.2. Experimental Setup

The X.509 server (the receiver) was implemented using OpenSSL version 0.9.7. The server operated on a 800MHz Pentium III machine. Request initiators (senders) ran on a 2.4GHz Pentium IV and were constructed using POSIX threading and OpenSSL libraries. The sender software was capable of running approximately 80 concurrent connections with the receiver. In the cases where a greater number of senders were needed, a 700MHz Pentium III was also used. DoS attacks initiated from a 700MHz Pentium III computer. All machines ran Linux 2.4 and were connected using 100Mbps Ethernet network interfaces and switches.

All communication occurred over TCP/IP. Network loss was simulated at the sender by failing to send a request with probability l, which we call the *loss probability*. We chose to use TCP rather than UDP in order to more accurately control the loss rate visible to the application layer. DoS messages from the attack host experienced no loss.

The receiver software utilized 20 bins and a six second collection interval. Incoming messages were enqueued into a bin using the modulo technique described in Section 3.2. In all experiments, the collection interval was sufficient to capture all messages transmitted by the senders. A simulated loss probability of 0.20 was experienced by the senders. To compensate, each sender sent 25 copies of her requests. Each sender chose uniformly at random an initial bin from $[0, 19]$ and proceeded by sending subsequent copies in a round-robin fashion.

After the collection interval, the receiver selected three bins. With the exception of the experiment described in Section 4.5, the three non-empty bins with the smallest number of messages were chosen. The receiver processed all messages in each bin (i.e., it verified the freshness of the nonce, validated the signature, and in the case of legitimate messages, decrypted $P_B(k_1)$). If the message passed all checks, the receiver sent an `OK` message to the corresponding sender.

4.3. Failure Rate of Bin Verification

A sender's message may fail to be processed due to the simulated network loss probability (fixed in our experimentation at 0.20). After the collection interval has expired, the receiver processes messages only from a small subset of the nonempty bins. It is therefore possible that due to the loss probability, no copies of a particular request are present in any of the selected bins. We call this a *failure* and the fraction of unique sender requests (as opposed to sender copies) that end in failure the *failure rate*. Here, we show that by checking just a small number of bins, the failure rate becomes quite low.

For simplicity, we consider a bin selection algorithm in which the receiver chooses k random bins rather than the k smallest. Although this strategy is not optimal from the receiver's point of view (see Section 4.5 for an analysis of different DoS attacker techniques), it serves as a simple and useful baseline for understanding the anticipated loss rate of selective bin verification. Let l be the loss probability, c be the number of copies sent by a sender, and n be the number of bins used by the receiver. Additionally, we assume that the sender stripes her request across the n bins. We can then calculate the probability, f', that a randomly chosen bin does not include a message from the sender as:

$$f' = \left(\frac{c \mod n}{n}\right) l^{\lceil \frac{c}{n} \rceil} + \left(1 - \frac{c \mod n}{n}\right) l^{\lfloor \frac{c}{n} \rfloor} \leq l^{\lfloor \frac{c}{n} \rfloor} \tag{1}$$

Since the probability that messages arrive in a given bin are independent, the failure rate, f, when k bins are examined is therefore

$$f \leq l^{\lfloor \frac{c}{n} \rfloor k} \tag{2}$$

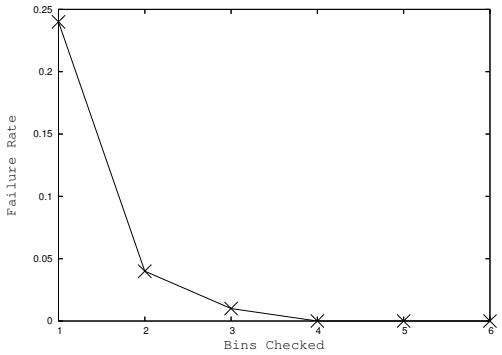

Figure 2. Failure Rate vs. Bins Checked (100 senders)

To verify our analysis, we conducted an experiment using the X.509 testbed. For this experiment, 100 senders simultaneously sent X.509 requests. Here, however, the re-

ceiver used a more intelligent strategy and chose the smallest (rather than random) three bins. Although the failure rate is slightly higher for this technique as compared to the expected value calculated using Equation 2, the rate becomes quite negligible when checking a small number of bins with a sufficiently large number of senders. This trend can be seen in Figure 2, with the failure rate approaching 1% after only three bins are checked.

4.4. Protection against DoS Attack

In this experiment, we show the efficacy of our selective bin verification approach against DoS attacks. We use as our metric the number of *inspections* carried out by the receiver. Here, we define an inspection to occur when the receiver processes a message, regardless of whether the message is legitimate, a duplicate, or invalid.

We compare the results of our binning technique against a straightforward implementation in which every message is considered. Let a be the attack rate (i.e., the number of attack messages arriving at the receiver per second), i be the collection interval (measured in seconds), and n be the number of bins. When binning is not used and each sender sends only one request, then the expected number of inspections required for m senders is

$$\texttt{insp}_{\text{nb}} = ai + m(1-l) \tag{3}$$

where l is the loss probability of the sender.

For the binning technique, the m senders are expected to cause $mc(1-l)(\frac{k}{n})$ inspections, where c denotes the number of copies sent by each sender and k is the number of bins inspected by the receiver. Since the receiver only inspects k bins, the contribution of the DoS is $ai(\frac{k}{n})$. Thus, the expected number of inspections is

$$\texttt{insp}_{\text{b}} = \frac{k(mc(1-l)+ai)}{n} \tag{4}$$

Thus, when the attack rate a is much larger than n and $m = O(n)$, then the attack rate is diminished by approximately a factor of k/n. For a receiver with 20 bins and $k = 3$, we would thus expect to process roughly $3/20$ of the DoS packets. Furthermore, we note that Equation 4 is an upper-bound on the number of inspections since the receiver selects the k-smallest bins rather than k bins at random.

We conducted an experiment to confirm our analysis. Figure 3 shows the number of inspections for the binning and non-binning approaches. 50 senders simultaneously sent requests in both the non-binning and binning trials. For the non-binning trials, each sender sent only one copy of her message. As in all binning experiments, a six second collection interval was used. In addition, an adversary flooded the receiver with requests that appeared valid but that contained invalid signatures. These extraneous messages were striped

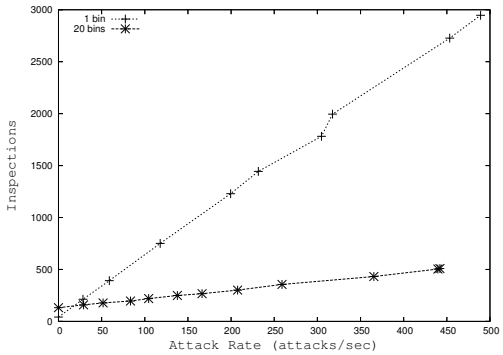

Figure 3. Inspections vs. Attack Rate

across all bins. Although both are linear, the slope resulting from the binning approach is significantly less than that of the non-binning technique. While the specific results shown in the Figure are a function of our chosen parameters (e.g., 50 senders, 20 bins, and 3 selected bins), we believe that our configuration is appropriate for real-world deployment and is effective at alleviating DoS.

4.5. Bin Selection Algorithm

In the previous experiments, the receiver processed messages in the three smallest but non-empty bins. We call this technique the *k-smallest approach*. With a small number of senders, this approach can lead to a higher failure rate. For example, if a small number of senders each transmits 25 copies of their requests across the 20 bins with some loss probability, the selection strategy will favor bins that contain no messages from some of the senders. As a result, the failure rate for each sender is relatively high. With a large number of senders, the distribution of bin loads as a fraction of the total number of senders becomes more uniform, and hence the failure rate decreases.

To counter this effect and increase the probability that each sender's request is processed, we propose the *α-random approach* for selecting k bins. Here, the receiver chooses the $k - \alpha$ smallest but non-empty bins, where $1 \leq \alpha < k$. Let $\mathcal{B}$ be the set of remaining, non-selected bins. The receiver proceeds by computing the mean bin load (i.e., the total number of received messages divided by the number of bins). It then uniformly at random chooses α bins from the subset of $\mathcal{B}$ which consists of bins that contain less than the mean bin load.

Figure 4 illustrates the failure rates for both bin selection strategies. As before, we let $k = 3$, and in the case of the α-random selection algorithm, assign $\alpha = 1$. As can be seen in the Figure, the failure rate is significantly higher for the k-smallest selection technique with a small number of senders (e.g., less than 10). This effect is less pronounced

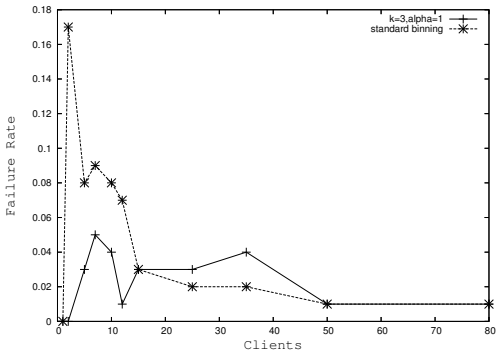

Figure 4. Failure Rate vs. Number of Senders

for the α-random approach.

4.6. Subset Attack

In Section 4.4, we described how selective bin verification can reduce the effect of DoS when the adversary stripes his attack across all bins. Here, we show that our binning technique is resilient even when the attacker constrains his attack to a chosen subset of bins. As before, we assume that the receiver uses the k-smallest approach. Furthermore, we denote the attacker's measure of success as the number of inspections he can force the receiver to undergo.

Theorem: The contribution of inspections due to DoS is maximized when the attack is evenly distributed across all n bins.

Proof. Let $L(\sigma)$ be the total number of adversary packets in the S smallest bins, where σ is the attacker's chosen distribution function such that $\sigma(i)$ denotes the number of DoS packets that the adversary sends to bin i. Also, let $\overline{\sigma}$ be the equal distribution (i.e., $\forall i, j, |\overline{\sigma}(i) - \overline{\sigma}(j)| \leq 1$). For simplicity, we consider the case where $\sum_{i=0}^{n-1} \sigma(i)$ is a multiple of n (that is, $\forall i, j, \overline{\sigma}(i) = \overline{\sigma}(j)$). Since the k-smallest bins can never contain more messages than k times the average bin load (i.e., $\overline{\sigma}(i)$), then $\forall \sigma, L(\sigma) \leq L(\overline{\sigma})$. □

Lastly, we note that the senders' requests do not affect the efficacy of the attacker's chosen bin distribution technique. Legitimate senders evenly distribute their message copies among the n bins. Hence, the contribution of the senders' requests is constant and independent of the manner in which the DoS packets are distributed.

5. Conclusions and Future Work

Existing research concerning DoS has focused almost exclusively on preventing malicious packets from reaching their intended targets. While such research is worthwhile and is useful for thwarting particular classes of attack, these approaches alone are not sufficient to protect service providers. Particularly for protocols in which the receiver has a high processing cost, traditional approaches fail to protect against attacks in which the adversary can overwhelm the receiver by sending malicious messages that appear entirely valid.

In this paper, we have investigated the ability of selective bin verification to protect services from attack, even when the DoS flood reaches the receiver. We show that bin verification is a promising technique for mitigating the effects of even large DoS attacks. Furthermore, we have analyzed the ability of bin verification to protect against attack under different configurations and attack strategies.

There are several interesting areas of future research relating to selective bin verification. Currently, the technique requires communication overhead even in the absence of an ongoing attack. Thus, an appropriate extension incorporates intrusion detection. For example, bin verification could be deactivated in the steady state and automatically enabled during an attack. A formal analysis of which protocols may best benefit from selective verification is another useful future research area. Finally, solutions that employ selective verification in concert with network-based DoS defenses are worthy of future study.

Acknowledgements

This work was partially supported by ONR Grant N00014-02-1-0715.

References

[1] ITU-T (CCITT) recommendation X.509, Mar. 1988. Also ISO 9594-8.

[2] S. Berinato. How a bookmaker and a whiz kid took on an extortionist - and won. *CSO Magazine*, May 2005. `http://www.csoonline.com/read/050105/extortion.html`.

[3] CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks, Jan. 1998.

[4] C. A. Gunter, S. Khanna, K. Tan, and S. S. Venkatesh. Dos protection for reliably authenticated broadcast. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2004.

[5] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium*, February 2002.

[6] P. Jelena and M. Greg. Attacking DDoS at the source, 2002.

[7] D. Moore, G. Voelker, and S. Savage. Inferring internet denial of service activity. In *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., Aug. 2001. USENIX.

[8] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2435–2463, 1999.