Adaptive Defense Against Various Network Attacks

Cliff C. Zou[1], Nick Duffield[2], Don Towsley[1], Weibo Gong[1]
[1] *University of Massachusetts, Amherst, MA*
[2] *AT&T Labs Research, Florham Park, NJ*

Abstract

In defending against various network attacks, such as Distributed Denial-of-Service (DDoS) attacks or worm attacks, a defense system needs to deal with various network conditions and dynamically changing attacks. In this paper, we introduce an "adaptive defense" principle based on cost minimization — a defense system adaptively adjusts its configurations according to the network condition and attack severity in order to minimize the combined cost introduced by false positives (misidentify normal traffic as attack) and false negatives (misidentify attack traffic as normal) at any time. In this way, the adaptive defense system generates fewer false alarms in normal situations (or under light attacks) with relaxed defense configurations, while protecting a network or a server more vigorously under severe attacks. Specifically, we present detailed adaptive defense system designs for defending against two major network attacks: SYN flood DDoS attack and Internet worm infection. The adaptive defense is a high-level system design that can be built on top of various non-adaptive detection and filtering algorithms, which makes it applicable for a wide range of security defenses.

1 Introduction

The current Internet is constantly under network attacks. Many defense methods and systems have been proposed to deal with these attacks. These systems typically first detect the on-going attack traffic, then block (filter) the attack traffic accordingly. Attack detection is of crucial importance in such defense systems. An imperfect detection algorithm will inevitably generate detection errors in terms of "false positives" and "false negatives". A "false positive" means incorrectly identifying a normal packet (or connection, or host, etc) as an attack whereas a "false negative" means incorrectly identifying an attack as a normal one.

Most research has focused on stationary network operation with fixed configurations. However in reality, attack detection systems have to face rapidly changing network conditions and various attack intensities. Therefore, besides finding a good detection algorithm, it is equally or more important to design an "intelligent" defense system that can automatically adjust its detection and filtering parameters to achieve the best performance possible under every possible attack situation.

We introduce an "*adaptive defense principle*" based on "cost minimization" — a defense system adaptively adjusts its configurations according to network conditions and "*attack severity*" in order to minimize the combined cost introduced by false positives and false negatives at any time. We call such a defense system as an "adaptive defense system". Compared to a traditional non-adaptive defense system, an adaptive defense system generates fewer false alarms in normal situations (or under light attacks) while protecting a network or a server more vigorously under severe attacks.

Denote by κ_t the "attack severity" at time t, which can be the fraction or volume of attack traffic, or other metrics determined by the types of attacks. Denote by θ_t the set of configuration parameters used in the detection algorithm. A defense system's false positive cost and false negative cost at time t, denoted by $C_p(\kappa_t, \theta_t)$ and $C_n(\kappa_t, \theta_t)$ respectively, are functions of κ_t and θ_t. Whenever the attack severity κ_t changes, the adaptive defense system will choose the up-to-date optimal configurations θ_t by minimizing the combined cost:

$$f = \min_{\theta_t}\{C_p(\kappa_t, \theta_t) + C_n(\kappa_t, \theta_t)\} \quad (1)$$

We present concrete adaptive defense systems for defending against two major network attacks: SYN flood DDoS attack, and Internet worm infection. The adaptive defense is a high-level system design that can be built on top of various *non-adaptive* detection and filtering algorithms, which makes it applicable for a wide range of security defenses.

The rest of the paper is organized as follows. Section 2 surveys related work. We present the system design for defending against DDoS attack and Internet worm infection in Section 3 and Section 4, respectively. In Section 5 we evaluate the performance of these two adaptive defense systems. Finally Section 6 concludes this paper.

2 Related Work

Mirkovic *et al.* [10] presented a comprehensive taxonomy of DDoS attack and defense mechanisms. Many DDoS detection approaches, such as the "IP traceback" [12], or the "MULTOPS" [2], try to find the identities of the real attacking sources. Hussain *et al.* [6] presented a framework to classify DDoS attacks into single-source and multi-source attacks. However, these methods cannot be used directly to block attack DDoS traffic. In order to detect *and* filter SYN flood packets at the victim end, Kim *et al.* [8] provided a general anomaly detection framework. Jin *et al.* [3] provided a concrete "Hop-Count Filtering" algorithm to filter out spoofed attack SYN packets based on packets' TTL values.

For Internet worm defense, Williamson [16] proposed a rate-limiting "throttling" method to constrain infection traffic. "EarlyBird" in [13] and "Autograph" in [7] detect and block worm spreading through identifying the common bit-strings among all infection network traffic of a worm. To prevent internal infection, Staniford [14] presented the segmentation idea to separate an enterprise network into many isolated subnetworks. Jung *et al.* [4][5] presented "Threshold Random Walk (TRW)" detection algorithms to detect and block worm infection based on the excessive number of unsuccessful scans sent by a worm. Weaver *et al.* [15] presented a simplified version of TRW algorithm that is suitable for both hardware and software implementation. Pang *et al.* [11] provided a comprehensive study of the characteristics of the abnormal traffic in the Internet.

Lee *et al.* [9] considered various cost factors, including false positive/negative cost, in the process of developing Intrusion Detection System (IDS). However, such a cost-sensitive design is a static system design method, which does not consider how to dynamically adjust an IDS's configurations according to the attack condition. Our previous paper [17] only briefly mentioned the adaptive defense principle, but never explored it.

3 Adaptive Defense System I: SYN Flood DDoS Attack

"SYN flood" attack is a denial-of-service attack by sending a large amount of SYN packets to a network or a server [10]. The attack packets usually have spoofed source addresses to hide the real attacking sources and also make defense much harder. For simplicity, we refer to the victim of a SYN flood attack as a server.

3.1 Underlying detection algorithm: extended "Hop-Count Filtering"

The "Hop-Count Filtering" (HCF) algorithm presented in [3] is a concrete and promising approach for SYN flood DDoS attack. In a nutshell, HCF infers the hop-length of a connection request source to a server based on the Time-to-Live (TTL) value in the incoming SYN packet IP header, then compares this value with the real hop-length of the client, which is derived from the client's previous successful connections. If these two values are different, HCF determines that the incoming SYN packet is a spoofed attack packet. Since attackers do not know the real hop-counts from their spoofed source addresses to the victim, spoofing the initial TTL values cannot help attack packets to avoid HCF detection as proved in [3]. Due to space constraint, please refer to [3] for the detail of the HCF detection.

Denote the "*false positive probability*" as P_p, the probability of incorrectly dropping a normal packet (or a normal connection, or a normal host for other types of attacks); denote the "*false negative probability*" as P_n, the probability of incorrectly treating an attack as a normal one. Due to memory constraint and Internet routing path changes, the HCF can change its detection strictness by allowing curtain deviation of the observed hop-count value from the value saved in its hop-count table. We run the HCF detection on the simulated normal SYN traffic and spoofed SYN flood traffic, respectively. From the simulation, we derive the detection performance in terms of P_p and P_n under different detection configurations. Fig. 1 shows the detection performance trade-off for the server "net.yahoo.com", whose hop-count data is provided to us by authors in [3]. Nine small circles in the figure from the left to the right represent the detection performance under different configurations. We define a detection sensitivity parameter δ ($0 \leq \delta \leq 8$): each small circle in Fig. 1 from the left to the right corresponds to $\delta = 0$ to $\delta = 8$, respectively.

We extend this discrete HCF to a continuous HCF based on "probabilistic dropping". Suppose the continuous HCF uses a real number δ as its detection parameter, $0 \leq \delta \leq 8$. Denote an integer $m = \lfloor \delta \rfloor$ and a real value $q = \delta - m$. Then, this continuous HCF will accept all packets acceptable by the discrete HCF with $\delta_1 = m$ while drop all packets that should be dropped by the discrete HCF with $\delta_2 = m + 1$. For the remaining packets that should be dropped by the δ_1 HCF but accepted by the δ_2 HCF, the continuous HCF accepts them with the probability q. In this continuous HCF, both P_p and P_n

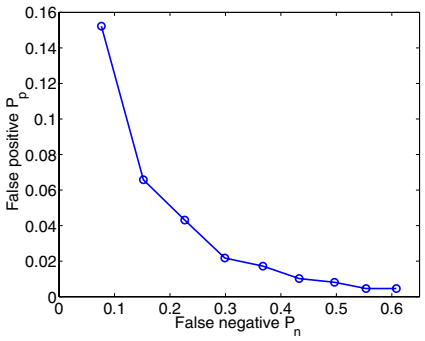


Figure 1: HCF detection performance under different configurations

are piece-wise linear functions of δ. Therefore, P_p is also a piece-wise linear function of P_n as shown in Fig. 1.

3.2 System design based on a general cost function

Denote by π the fraction of attack packets among all incoming SYN packets. π naturally exhibits the relative attack intensity compared to the normal payload of a server, and hence, we use π to represent the "attack severity" of a SYN flood DDoS attack.

The adaptive defense system updates its HCF detection parameter δ periodically at each discrete time denoted by k ($k = 1, 2, \ldots$). During the time interval from k to $k+1$, the HCF implements $\delta(k)$, which corresponds to the pair of $P_p(k)$ and $P_n(k)$. During this time period, the fraction of incoming packets identified by the defense system as attack packets is denoted by $\pi'(k)$, while the real attack fraction is denoted by $\pi(k)$.

$\pi(k)$ differs from the observed value $\pi'(k)$ because: (1) the limited samples within a discrete time interval introduce an observation statistical error; and (2) some attack packets are not counted in $\pi'(k)$ due to false negatives whereas some normal packets are counted in $\pi'(k)$ due to false positives.

In the following we derive an unbiased estimate of the real attack severity, denoted by $\hat{\pi}(k)$. Suppose during the time interval from k to $k+1$, the attack severity $\pi(k)$ does not change and the defense system receives $N(k)$ SYN packets. Then, $\pi(k)N(k)$ packets are attack packets while the remaining $[1-\pi(k)]N(k)$ are normal ones. The defense system drops $\pi'(k)N(k)$ packets, among which $[1 - P_n(k)]\pi(k)N(k)$ are real attack packets and the remaining $P_p(k)[1 - \pi(k)]N(k)$ are falsely dropped normal packets. Therefore, we have

$$\pi'(k)N(k) = [1-P_n(k)]\pi(k)N(k)+P_p(k)[1-\pi(k)]N(k)$$

Removing $N(k)$ from both sides yields

$$\pi'(k) = [1 - P_n(k)]\pi(k) + P_p(k)[1 - \pi(k)] \quad (2)$$

From (2), we derive the estimation formula of $\pi(k)$ as:

$$\hat{\pi}(k) = \frac{\pi'(k) - P_p(k)}{1 - P_n(k) - P_p(k)} \quad (3)$$

Through statistical analysis, we find $E[\hat{\pi}] = \pi$; hence $\hat{\pi}(k)$ is an unbiased estimate.

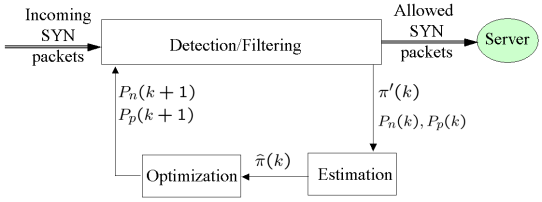


Figure 2: Adaptive defense system architecture

Fig. 2 illustrates the architecture of the adaptive defense system. At the end of time k, the adaptive defense system first uses (3) to derive an estimate $\hat{\pi}(k)$ of the real attack severity, then finds the "optimal" detection parameters $P_n(k+1)$, $P_p(k+1)$ (i.e., $\delta(k+1)$) for use in the next time interval. The "optimization" module tries to minimize the combined cost of false positives and false negatives by minimizing:

$$f = \min_{\delta(k+1)} \{c_p[1-\hat{\pi}(k)]P_p(k+1) + c_n\hat{\pi}(k)P_n(k+1)\} \quad (4)$$

where $[1 - \hat{\pi}(k)]P_p(k+1)$ is the fraction of falsely dropped normal packets and $\hat{\pi}(k)P_n(k+1)$ is the fraction of attack packets that pass through to the server.

Those two cost factors, c_p and c_n, have concrete physical meanings: they represent the cost of incorrectly dropping (accepting) a normal (attack) SYN packet, respectively. In some cases, they can be chosen as constants whereas in other cases they should be functions of the attack severity. For example, while a server can tolerate a small number of false negatives, beyond some point, the received attack traffic will severely consume the system's resources. Whether to choose constant or functional cost factors should be determined by the specific defense requirement and experiences from security staffs.

3.3 "Buffer-aware" performance function

The general cost function (4) is suitable for a wide range of security defense systems. If a server has a specific requirement, however, an object-oriented performance function would achieve a better defense performance.

A server usually has two separate buffers for incoming TCP connections: one for pending connections called

"pending buffer"; another for connections that have been established. The pending buffer is susceptible to SYN flood DDoS attack. Suppose the server's performance is not affected by the number of pending connections in the pending buffer so long as the buffer is not overflowed. Such a server has a specific performance objective: to accept as many as possible normal connection requests.

Define "sojourn time" to be the time period a SYN packet resides in the pending buffer. Denote the average sojourn time of a normal SYN packet as T_1 and the average sojourn time of an attack SYN packet as T_2. The adaptive defense system updates its parameters periodically at each discrete time k and the time interval is denoted by Δ. The defense system still has the same architecture as shown in Fig. 2. Suppose the server has a pending buffer that can hold K pending TCP connections at the same time. At the end of discrete time k, denote the number of incoming packets during the last time interval as $N(k)$.

If the defense system deactivates its filtering functionality and allows all $N(k)$ packets to pass through, the buffer size requirement, denoted by B^0, is:

$$B^0 = \frac{T_2}{\Delta}\hat{\pi}(k)N(k) + \frac{T_1}{\Delta}[1-\hat{\pi}(k)]N(k) \qquad (5)$$

If the defense system activates its filtering functionality with parameters $P_p(k+1)$, $P_n(k+1)$, then $P_n(k+1)\pi(k)N(k)$ attack packets and $[1-P_p(k+1)][1-\pi(k)]N(k)$ normal packets will pass through the detection/filtering module to reach the server. Thus the buffer size requirement, denoted by B^1, is:

$$B^1 = \frac{T_2}{\Delta}P_n(k+1)\hat{\pi}(k)N(k) + \frac{T_1}{\Delta}[1-P_p(k+1)][1-\hat{\pi}(k)]N(k) \qquad (6)$$

Therefore, at time k, the adaptive defense system should choose its defense parameters $P_p(k+1), P_n(k+1)$ for the next time interval according to:

- If $B^0 < K$, deactivate the filtering functionality (the server's pending buffer will not overflow anyway).
- If $B^0 \geq K$, activate the filtering functionality and choose the optimal $P_p(k+1), P_n(k+1)$ (i.e., $\delta(k+1)$) by minimizing:

$$f = \min_{\delta(k+1)} |B^1 - K| \qquad (7)$$

  In this way, the server can accept the maximum number of normal SYN packets.

Basically, (7) tries to minimize the cost caused by over-filtering ($B^1 < K$) or under-filtering ($B^1 > K$).

4 Adaptive Defense System II: Internet Worm Infection

In this section, we study how to design an adaptive defense system for defending against a fast spreading Internet worm, such as Code Red, Slammer and Blaster [1].

Because a scanning worm blindly scans IP space to find targets, a worm-infected host has a much lower probability to set up successful connections than a benign host. "Threshold Random Walk (TRW)" [4][5] detection is based on the fact that a worm-infected host sends out many more failed connection requests than successful requests. Weaver *et al.* [15] further simplified the TRW algorithm for hardware implementation. We deploy a modified version of the worm detector presented in [15] as the underlying detection algorithm.

Our modified TRW detector works in the following way: each source host that initiates a connection is assigned a non-negative "counter" with the initial value of zero. This counter (if not equals to zero) decreases by one if the source host initiates a successful connection, and increases by one if the source initiates a failed connection. Multiple connection attempts from a source targeting the same destination are treated as one connection attempt (e.g., TCP/SYN retransmission before the timeout). A source host is determined to be infectious when its counter reaches a threshold W.

The next step is to represent the Internet worm "attack severity". An enterprise network has a fraction of unused IP addresses. All connection attempts to these addresses, which are called "illegal scans", will always fail. We use the number of illegal scans observed in a discrete time interval, denoted by Z, to represent the attack severity.

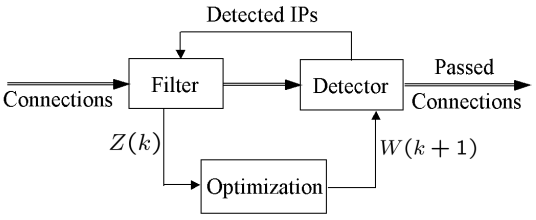


Figure 3: Adaptive defense system architecture for defending against Internet worm infection

Fig. 3 illustrates the architecture of the adaptive defense system. Whenever the "detector" detects an infected host, it sends the host IP to the "filter" where further scanning traffic from the host will be blocked. Denote by $Z(k)$ the number of illegal scans observed from time k to $k+1$ and $W(k)$ the detection parameter used from k to $k+1$. Fig. 3 shows that at the end of time k, the system derives the optimal $W(k+1)$ to use for the next time interval based on the current attack severity $Z(k)$.

The "optimization" module derives $W(k+1)$ based on the performance function:

$$f = \min_{W(k+1)} \{c_p \cdot \frac{1}{W(k+1)} + c_n \cdot W(k+1) \cdot Z(k)\} \quad (8)$$

As W increases, an infected host is able to send more scans before it is detected and blocked; but fewer benign hosts would be incorrectly blocked. Therefore, $W(k)$ describes the trade-off: $c_p/W(k+1)$ corresponds to the false positive cost and $c_n \cdot W(k+1) \cdot Z(k)$ corresponds to the false negative cost.

5 Evaluation

In this section, we evaluate the defense performance of the above two adaptive defense systems based on either simulation experiments or real attack traces.

5.1 Defense against SYN flood DDoS

5.1.1 General cost function

First, we study the adaptive defense system with the general cost function (4). We assume that during each time interval Δ (e.g., $\delta = 30$ seconds), the server receives 1,000 normal SYN packets. The spoofed SYN flood attack varies its attack intensity as shown in the top graph of Fig. 4. The simulated SYN flood attack includes two types of attack dynamics: (1) attacking traffic gradually increases its intensity (from time 0 to 500); and (2) all distributed attacking hosts begin to send attacking packets at the same time (from time 700 to 800). The bottom graph of Fig. 4 shows how the adaptive defense system automatically tunes its detection parameter δ (In this experiment, $c_p/c_n = 2$).

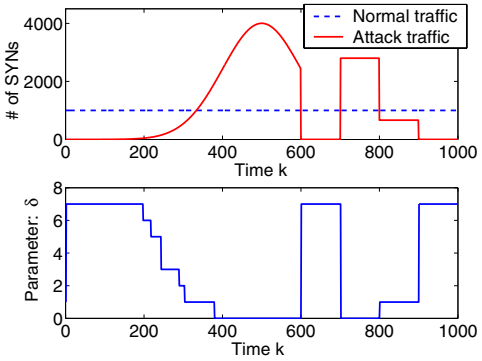


Figure 4: SYN flood attack scenario and the defense system response based on the general cost function (4)

To verify the attack severity estimation (3), we plot the real value $\pi(k)$, the observed value $\pi'(k)$ and the estimated value $\hat{\pi}(k)$ as functions of time k in Fig. 5. This figure clearly shows that Eq. (3) provides accurate estimation results at any time.

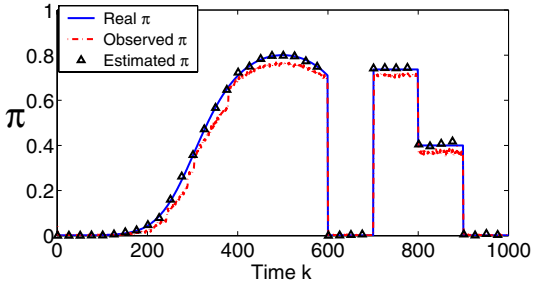


Figure 5: Verification of the estimation formula (3)

5.1.2 Buffer-aware performance function

Next, we study the adaptive defense system based on the buffer-aware function (7). $\Delta = 30$ seconds as used in previous experiment. We assume that normal SYN packets have the average sojourn time $T_1 = 3$ seconds in the pending buffer; attack packets have $T_2 = 25$ seconds (since most attack packets will stay in the buffer until time-out). The pending buffer is assumed to be able to support $K = 200$ connection requests at the same time.

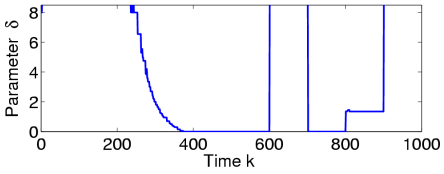
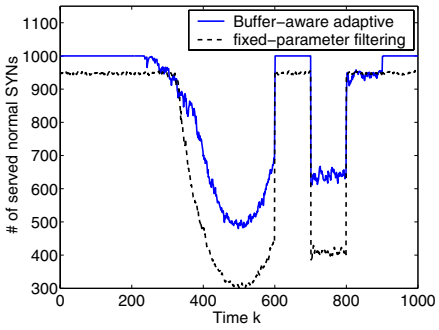


Figure 6: Adaptive defense system response based on the buffer-aware performance function (7)

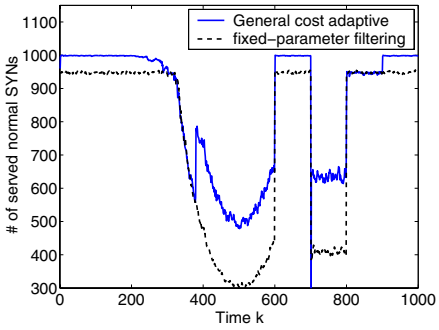
The SYN flood attack follows the same dynamics as the experiment shown in Fig. 5. Fig. 6 shows how the defense system adjusts its parameter δ. $\delta > 8$ means the defense system deactivates its filtering functionality. This figure and previous Fig. 4 show that both adaptive defense systems have the similar responses. The difference is that the adaptive defense system here has a continuously changing optimal δ since (7) is a non-linear function of δ (while (4) is a linear function of δ).

5.1.3 Performance comparison

In the above two experiments, we also obtain the information of accepted normal packets. To see the adaptive defense performance, we also conduct a baseline experiment where a fixed-parameter HCF with $\delta = 1$ is deployed, which is the recommended setting in [3]. Fig. 7(a) and Fig. 7(b) show the defense performance in



(a). Based on buffer-aware function (7)



(b). Based on general cost function (4)

Figure 7: Performance of adaptive defense systems compared with the fixed-parameter system

terms of the number of accepted normal SYNs for these two adaptive defense systems, respectively.

Compared with the fixed-parameter defense, both adaptive defense systems accept more normal connection requests either under very light attacks or under heavy attacks. The fixed-parameter defense system uses a set of settings that is optimal only for a specific attack condition, which is not suitable for a real implementation where people expect a defense system to work well under various network conditions.

Fig. 7 also shows that we do not need to design a very accurate adaptive defense system in order to improve the performance of an underlying non-adaptive detection algorithm. As long as we use the adaptive defense principle to adjust a system's settings, the defense performance will be improved more or less. In fact, we run the experiment shown in Fig. 4 many times with different values of c_p and c_n, the adaptive defense system always improves its performance compared with the fixed-parameter system in terms of the number of accepted normal requests (similar results as shown in Fig. 7).

5.2 Defense against worm infection

In the following experiments, we use a monitored Slammer propagation trace to study the performance of the adaptive defense system. The trace is a tcpdump data containing all UDP packets (targeting at port 1434) received by a /16 network. The top graph of Fig. 8 shows the number of Slammer UDP packets received during each second, which is $Z(k)$ as we use one second for the discrete time interval. The monitored /16 network has two Internet connections. At 150 seconds, one connection went down and caused the monitored Slammer scans dropped suddenly. At 217 seconds, one internal computer was infected and its scanning traffic caused local congestion, and hence, the monitored Slammer scans dropped suddenly for the second time.

The bottom graph of Fig. 8 shows how the defense

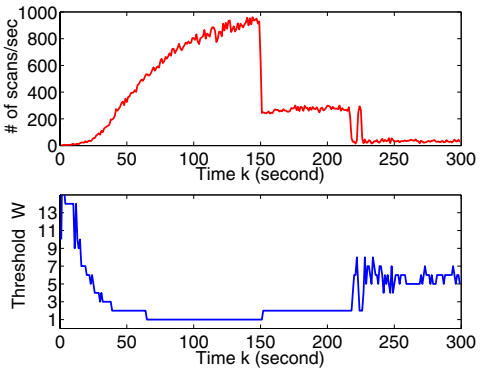


Figure 8: Slammer attack and the response by the adaptive defense system based on TRW detection

system responds to the attack changes by adjusting $W(k)$ (in this experiment, $c_p/c_n = 1000$). $W(k) = 1$ is the most aggressive defense that the system can operate: any host will be blocked (on the suspicious port only) as soon as one illegal scan from it is observed.

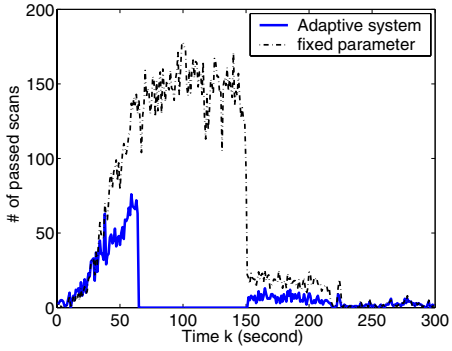


Figure 9: Worm scans passing the defense system

Fig. 9 shows the number of worm scans entering the /16 network — the other worm scans are blocked by the defense system (the peak level of original worm scans is

1000 per second). For comparison, we also show in this figure the case of a fixed-parameter system where $W = 4$. Note that since the number of *vulnerable* computers in a local network is usually much smaller than the number of addresses allocated to the network, only a very small percentage of passed worm scans could possibly cause infection. Of course, if we are very concerned with the worm infection, we can increase the ratio of c_p/c_n to make the defense system quickly updates its threshold $W(k)$ to 1 when $Z(k)$ increases (at the cost of increasing the number of falsely blocked normal hosts).

For the evaluation of false positives, [4] and [15] have used real network traces to show that the "Threshold Random Walk" algorithm has very limited false positives (most of those falsely detected hosts are web crawlers or proxies). Since our adaptive worm defense system uses the similar underlying detection algorithm, we do not repeat such an evaluation here.

6 Conclusion

To defend against various network attacks, we introduce an "adaptive defense" principle based on cost minimization — a defense system adaptively adjusts its configurations according to the network condition and attack severity in order to minimize the combined cost introduced by false positives and false negatives at any time. In this paper, we present concrete system designs to defend against two major network attacks: SYN flood DDoS attack and Internet worm infection. The adaptive parameter update includes very simple estimation and optimization, thus the computational overhead is very small. The adaptive defense is a high-level system design that can be built on top of various non-adaptive detection and filtering algorithms, which makes it applicable for a wide range of security defenses.

There are still many work to do on the adaptive defense design. First, we want to further study how to choose the cost factors c_p and c_n quantitatively according to the defense requirements. Second, in order to understand accurately the impact of false positives/negatives, we plan to evaluate the adaptive defense system based on real monitored traces that include both attack and normal traffic. Third, when defense settings are adaptive, attackers might be able to influence the detection in such a way as to deny service to legitimate traffic. We plan to further study this system robustness issue.

Acknowledgements

We gratefully thank Eric Cronin and Anthony Kurc for sharing their hop-count dataset, and Andrew Daviel from TRIUMF, Canada for sharing his monitored Slammer trace. This work was supported in part by ARO contract DAAD19-01-1-0610, NSF Grant EEC-0313747, EIA-0080119, ANI-0085848 and CNS-0325868.

References

[1] CERT. CERT/CC advisories. http://www.cert.org/advisories/.

[2] GIL, T. M., AND POLETTO, M. MULTOPS: a data-structure for bandwidth attack detection. In *Proceedings of USENIX Security Symposium* (August 2002).

[3] JIN, C., WANG, H., AND SHIN, K. G. Hop-count filtering: an effective defense against spoofed DDoS traffic. In *Proceedings of 10th ACM Conference on Computer and Communications Security* (October 2003).

[4] JUNG, J., PAXSON, V., BERGER, A. W., AND BALAKRISHNAN, H. Fast portscan detection using sequential hypothesis testing. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 2004).

[5] JUNG, J., SCHECHTER, S. E., AND BERGER, A. W. Fast detection of scanning worm infections. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)* (September 2004).

[6] KEROMYTIS, A., MISRA, V., AND RUBENSTEIN, D. A framework for classifying denial of service attacks. In *Proceedings of ACM SIGCOMM* (August 2003).

[7] KIM, H., AND KARP, B. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of 13th USENIX Security Symposium* (August 2004).

[8] KIM, Y., LAU, W., CHUAH, M., AND CHAO, H. Packetscore: Statistical-based overload control against distributed denial-of-service attacks. In *Proceedings of the IEEE INFOCOM* (March 2004).

[9] LEE, W., FAN, W., MILLER, M., STOLFO, S., AND ZADOK, E. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security 10*, 1,2 (2002).

[10] MIRKOVIC, J., AND REIHER, P. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review 34*, 2 (2004).

[11] PANG, R., YEGNESWARAN, V., BARFORD, P., PAXSON, V., AND PETERSON, L. Characteristics of Internet background radiation. In *Proceedings of the Internet Measurement Conference (IMC)* (October 2004).

[12] SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. Practical network support for IP traceback. In *Proceedings of ACM SIGCOMM* (August 2001).

[13] SINGH, S., ESTAN, C., VARGHESE, G., AND SAVAGE, S. Automated worm fingerprinting. In *Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)* (December 2004).

[14] STANIFORD, S. Containment of scanning worms in enterprise networks. *Journal of Computer Security* (2003).

[15] WEAVER, N., STANIFORD, S., AND PAXSON, V. Very fast containment of scanning worms. In *Proceedings of 13th USENIX Security Symposium* (August 2004).

[16] WILLIAMSON, M. M. Throttling viruses: Restricting propagation to defeat mobile malicious code. In *18th Annual Computer Security Applications Conference* (December 2002).

[17] ZOU, C. C., GONG, W., AND TOWSLEY, D. Worm propagation modeling and analysis under dynamic quarantine defense. In *Proceedings of ACM CCS Workshop on Rapid Malcode (WORM'03)* (October 2003).