Adaptive Defense Against Various Network Attacks

Cliff C. Zou[1], Nick Duffield[2], Don Towsley[3], Weibo Gong[3]

[1]School of Electrical Engineering and Computer Science

University of Central Florida, Orlando, FL 32816

czou@cs.ucf.edu

[2]AT&T Labs-Research, Florham Park, NJ 07932

duffield@research.att.com

[3]University of Massachusetts, Amherst, MA 01003

towsley@cs.umass.edu, gong@ecs.umass.edu

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. Therefore, a good defense system needs to have a built-in "adaptive defense" functionality based on cost minimization — adaptively adjusting 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 can generate fewer false alarms in normal

This work was supported in part by ARO contract DAAD19-01-1-0610, NSF Grant EEC-0313747, EIA-0080119, ANI-0085848 and CNS-0325868.

situations or under light attacks with relaxed defense configurations, while protecting a network or a server more vigorously under severe attacks.

In this paper, we present concrete 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 various underlying non-adaptive detection and filtering algorithms, which makes it applicable for a wide range of security defenses.

Index Terms

Computer security, adaptive defense, DDoS, SYN flood, Internet worm

I. Introduction

Current Internet and computers are constantly under various attacks: hackers' intrusion, port scan, Distributed Denial-of-Service (DDoS), virus and worm infection, email spam, etc. Many defense methods and systems have been proposed. These systems typically need to detect the on-going attack traffic first, and then block (filter) the attack traffic before it reaches the victims. The attack detection is of crucial importance in such defense systems. When using an imperfect detection system, it can generate "false positives" and "false negatives". A "false positive" means a detection system incorrectly identifies a normal packet/connection/host as an attack whereas a "false negative" means a detection system incorrectly identifies an attack packet/connection/host as a normal one.

A good detection system generates fewer false positives and false negatives. However, these two types of detection errors always conflict with each other when the detection algorithm is fixed: when we adjust the detection parameters to decrease the number of false positives or false negatives, the other one inevitably increases.

Most research has focused on stationary network operation with fixed configurations. However in reality, attack detection systems have to face rapidly changing network condition and attack intensity. Therefore, besides finding a specific detection algorithm, it is equally or more important for us to design an "intelligent" defense system that can automatically adjust its detection and filtering parameters to achieve as best performance as it could under various attack situations.

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 and false negatives at any time. Such a defense system, called an "adaptive defense system", adaptively tunes its configurations according to the attack severity and to the relative cost of erroneous actions. Compared to a traditional 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 of attack traffic, the volume of attack traffic, or other indexes determined by the types of attacks. Denote by θ_t the set of configuration parameters used in the defense system (either a single or a vector parameter). A defense system's false positive cost and false negative cost at time t are functions of κ_t and θ_t. Denote the false positive cost as $C_p(\kappa_t, \theta_t)$, false negative cost as $C_n(\kappa_t, \theta_t)$. The "adaptive defense principle" means that, whenever the attack severity κ_t changes, the defense system will choose the up-to-date optimal configurations θ_t by minimizing the combined cost:

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

The adaptive defense mechanism relies on the fact that we can get a relatively good estimate of the "attack severity". Therefore, it is suitable for defense systems dealing with attacks that involve a large amount of attack traffic, such as DDoS attack, or Internet worm infection, etc. It is not suitable for detecting of a hacker's intrusion since such an attack may only involve one connection with several packets.

In this paper, 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 various underlying non-adaptive detection and filtering algorithms, which makes it applicable for a wide range of security defenses.

The rest of this paper is organized as follows. Section II surveys related work. We present the adaptive defense system design for defending against SYN flood DDoS attack in Section III, and the system design for defending against Internet worm infection in Section IV. After introducing how to design the adaptive defense systems for the above two major network attacks, in Section V we evaluate the performance of these adaptive defense systems through either simulation experiments or real attack traces. Finally we conclude this paper with Section VI.

II. Related Work

Mirkovic *et al.* [14] presented a comprehensive taxonomy of DDoS attack and defense mechanisms. Many DDoS detection approaches, such as the “IP traceback” in [17] and a series of following papers, or the “MULTOPS” [2], try to find the identities of the real attacking sources. However, these approaches cannot filter incoming attack packets without the help from the ISPs that contain those attacking sources. Hussain *et al.* [8] presented a framework to classify DDoS attacks into single-source and multi-source attacks. Such an attack classification helps us to choose an appropriate response mechanism, but it still cannot block attack traffic. [13] and [21] presented methods to detect and block DDoS traffic at the source-end networks. Like the egress filtering for preventing IP spoofing, such a DDoS defense at the source-end lacks incentive for people to deploy and also requires universal deployment to be effective. Keromytis *et al.* [7] presented a secure overlay structure to protect a mission-critical server against DDoS attacks.

In order to detect *and* filter SYN flood packets at the victim end, Kim *et al.* [10] provided a general anomaly detection framework based on the assumption that normal and attack packets have different statistical characteristics. However, the authors did not present the detailed algorithm on how to detect SYN flood attack packets. Jin *et al.* [4] provided a concrete “Hop-Count Filtering” algorithm based on the fact that attackers do not know the real hop-lengthes from their spoofed sources to the target victim, which can filter close to 90% of spoofed IP packets.

There are many papers on worm research and here we only introduce related work for Internet worm defense. Williamson [24] proposed a rate-limiting “throttling” method to constrain infection traffic. “EarlyBird” in [18] and “Autograph” in [9] detect and block worm spreading through identifying the common bit-strings among all infection network traffic of a worm. To prevent internal infection, Staniford [19] presented the segmentation idea to separate an enterprise network into many isolated subnetworks. Jung *et al.* [5][6] 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.* [23] presented a simplified version of TRW algorithm that is suitable for both hardware and software implementation.

Some researchers have discussed the notion of "adaptive" in defense, although appeared in other forms. Lee *et al.* [11] considered various cost factors, including false positive/negative cost, in the process of developing and evaluating 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 during run time. Zou *et al.* [25] mentioned the false positive cost in the worm quarantine defense and introduced briefly the "feedback adjustment" idea with no detailed discussion. Li *et al.* [12] presented a similar adaptive defense formula for DDoS attacks based on statistical filtering. However, they did not present concrete algorithms on how to detect attack traffic. Our work was conducted independently at the same time as [12], and we have present concrete formulation of the adaptive defense for both DDoS attack and Internet worm infection.

III. 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 [14]. The attack packets usually have spoofed source addresses to hide the real attacking sources and also make defense much harder. For simplicity, from now on we call the victim of a SYN flood attack as a server.

It is relatively easy to detect whether or not a server is under SYN flood DDoS attack by observing the existence of excessive amount of half-open TCP connections. However, filtering SYN attack packets is very hard: a filtering-based defense system must be able to detect individual incoming SYN attack packet based only on its packet header, even though all fields in the packet header can be arbitrarily modified by attackers.

A. Underlying detection algorithm: extended "Hop-Count Filtering"

The "Hop-Count Filtering" (HCF) presented in [4] is a concrete and promising defense approach for SYN flood DDoS attack. Although attackers can forge any field in the IP header, they cannot change the number of hops an IP packet takes to reach a server. In a nutshell, HCF infers the hop-length of incoming SYN packets to a server based on the Time-to-Live (TTL) value in the IP header, then compares this value with the real hop-length of the source address, which is derived from previously successful connections from the real source. If these two values are different (to certain extend), the HCF determines that the incoming SYN packet is a spoofed attack packet.

Attackers can arbitrarily change the initial TTL of spoofed packets. However, attackers do not know the real hop-counts from their spoofed source addresses to the victim. the authors of [4] proved that attackers cannot increase much the chance of cheating the HCF by forging TTL values; the HCF can identify close to 90% of spoofed SYN packets with little false positive damage [4].

We have fortunately obtained the hop-count collection data from the authors of [4], which contains clients' hop-counts to 46 servers. Most of these servers have measured hop-counts from more than 40,000 clients.

We study the adaptive defense system's performance based on an arbitrary server in the above hop-count collection data. The data provides each server's real hop-count table for all its clients contained in the data. Since we do not have real SYN flood attack trace, we use simulation to generate attack SYN packets mixed together with normal SYN packets to such a server. We use the similar assumptions as the experiments in [4]: we assume that the attacker knows the IP addresses of all clients to the server and generates spoofed SYN packets with randomly selected source addresses among these legitimate clients. We further assume that the attacker knows the hop-count distribution of the server.

Due to memory constraint, HCF cannot save the hop-count of every client in its hop-count table; in addition, the server cannot know the real hop-counts for all its potential clients since some of them seldom send connection requests to the server. To solve these problems, [4] presented a simple 24-bit address aggregation theme, which means that the source IP addresses with the same first 24 bits (/24 network) are grouped together and represented by a single hop-count in the hop-count table. This hop-count value is selected to be the smallest hop-count among observed clients in the same /24 network.

The 24-bit aggregation may not preserve the correct hop-counts for all clients in a /24 network if this network is composed of several longer prefixes networks that reside in different physical places.

In addition, the hop-count table of a server may not be up-to-date for all clients. For the above two reasons, the HCF detection could falsely treat normal packets as spoofed attack packets. In our simulation experiments, we assume that after deriving the hop-count table, 5% of clients change their hop-counts by 1, 1% of clients change by 2, and 0.5% of clients change by 3. In a real implementation, how the hop-count table differs from the real hop-counts of clients can be determined by deploying the HCF detection without packet filtering; the real hop-count of a client is derived from its successful TCP connections with the server.

Three filtering configurations were presented in [4] based on the 24-bit aggregation: "Strict Filtering", "+1 Filtering" and "+2 Filtering". "Strict Filtering" drops packets whose hop-counts do not match those stored in the hop-count table; "+2 Filtering" drops packets whose hop-counts are smaller or greater than 2 compared with the values saved in the hop-count table. In this paper, we extend the HCF to nine filtering configurations, denoted by: "0", "+1", "+1-1", "+2-1", "+2-2", "+3-2", "+3-3", "+4-3" and "+4-4". The "Strict Filtering" and "+1 Filtering" in [4] correspond to "0" and "+1" Filtering, respectively. "$+a-b$" Filtering drops packets whose hop-counts are smaller than b or greater than a compared with the values in the hop-count table.

Denote the "false positive probability" as P_p, which is 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, which is the probability of incorrectly treat an attack packet as a normal packet. For a specific server in the hop-count collection data in [4], 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 of each HCF configuration in terms of P_p and P_n. Fig. 1 shows the detection performance trade-off for the server "net.yahoo.com". Nine small circles in the figure from the left to the right represent the detection performance of "0", "+1", $\cdots$, "+4-4" Filtering, respectively. For other servers, the HCF achieves the similar detection performance.

Fig. 1 shows that the HCF has two correlated parameters, P_n and P_p. Thus the HCF can be treated as only having one adjustable parameter δ[1], which determines the pair of P_n and P_p. In this paper, δ

[1]Of course, we can define an extended HCF with a vector adjustable parameter δ, too. For example, $\delta = (a, b)$ where variable a and b represent the "$+a-b$" Filtering. All the work and discussion in this paper will be the same for such a vector parameter δ.

is defined to vary from 0 to 8: $\delta = 0$ corresponds to the "0" Filtering, $\delta = 1$ corresponds to the "+1" Filtering, $\delta = 2$ corresponds to the "+1-1" Filtering, $\delta = 3$ corresponds to the "+2-1" Filtering, etc.

Now the HCF has nine different operation points, i.e., $\delta = 0, 1, \cdots, 8$. We call it a discrete HCF. If the HCF can be further extended to adjust its parameters continuously within those nine discrete operation points, the adaptive defense system will have more freedom in tuning detection parameters to possibly achieve a better defense performance. We extend the above discrete HCF to derive a continuous HCF based on "probabilistic dropping" in the following.

Suppose the continuous HCF uses a real number δ as its detection parameter, $0 \leq \delta \leq 8$. Denote the integer $m = \lfloor \delta \rfloor$ and a real number $q = \delta - m$. Then, this continuous HCF will accept all packets that should be accepted 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 other 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 such a continuous HCF detection algorithm, both P_p and P_n are piece-wise linear functions of δ. Therefore, P_p is also a piece-wise linear function of P_n as the performance curve shown in Fig. 1.

B. Adaptive defense system design based on a general cost function

We have presented the underlying detection algorithm in the previous section, the next step in designing the adaptive defense system is to find out how to represent the "attack severity" for a SYN flood DDoS attack. 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.

The adaptive defense system updates its HCF detection parameter δ at each discrete time denoted by k $(k = 1, 2, \cdots)$. During the time interval from the discrete time k to $k+1$, suppose the HCF uses the parameter $\delta(k)$, which corresponds to the false positive probability $P_p(k)$ and the false negative probability $P_n(k)$. During this time period, the fraction of packets identified by the defense system as attack packets is denoted by $\pi'(k)$. The real attack fraction during this discrete time interval 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.

Based on the observation data $\pi'(k)$ and the detection parameters $P_p(k)$, $P_n(k)$, in the following we present a formula to derive an unbiased estimate of the attack severity, denoted by $\hat{\pi}(k)$. Suppose during the time interval from the discrete time 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 overall $\pi'(k)N(k)$ 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

$$\pi'(k)N(k) = [1-P_n(k)]\pi(k)N(k) + P_p(k)[1-\pi(k)]N(k) \tag{2}$$

Removing $N(k)$ from both sides, we derive the estimation formula of the real attack fraction $\pi(k)$ as:

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

A more formal proof of the estimation formula is given in Appendix, which shows that the estimate $\hat{\pi}(k)$ is unbiased (when the detection decision of each packet is independent to each other).

Fig. 2 illustrates the architecture of the adaptive defense system. At the end of the discrete time k, based on the parameters $P_n(k)$, $P_p(k)$ used in the last discrete time interval and the observed attack severity $\pi'(k)$, 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 discrete time interval. In the "optimization" module shown in Fig. 2, the adaptive defense system tries to minimize the combined cost of false positives and false negatives by minimizing:

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

where c_p is the cost of incorrectly dropping a normal SYN packet; c_n is the cost of incorrectly accepting an attack packet. When the detection system uses the new detection parameter $\delta(k+1)$ for the DDoS attack with the current attack severity $\hat{\pi}(k)$, $[1-\hat{\pi}(k)]P_p(k+1)$ is the fraction of falsely dropped normal SYN packets, and $\hat{\pi}(k)P_n(k+1)$ is the fraction of attack SYN packets that pass through to the server.

The 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.

It should be noted that the adaptive defense system can be implemented not only on a server itself, but on the server's upstream routers or gateways, as what shown in Fig. 2. The server only needs to update its hop-count table based on successful TCP connections and pass the up-to-date hop-count table to its upstream routers or gateways. In this way, deploying the adaptive defense system does not add any burden to the server under protection.

C. Adaptive defense system design based on a "buffer-aware" performance function

The adaptive defense system design presented above is based on a general cost function (4), which is suitable for a wide range of security defense systems. If the network or the server under protection has its own specific performance function, the adaptive defense system can use such an object-oriented performance function in its optimization.

A server usually has two separated buffers for incoming TCP connections: one for pending connections, which is call "pending buffer" in this paper, 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 as long as the buffer is not overflowed to drop connection requests. Such a server has a specific performance function in dealing with SYN flood attack: to accept as many as possible normal connection requests.

With such a "buffer-aware" performance function, the adaptive defense system should deactivate its filtering functionality as long as the pending buffer is not overflowed. In this way, no normal packets will be dropped due to false positives. When the pending buffer cannot support all incoming SYN packets, the defense system should activate its filtering functionality to drop most attack packets. The defense system will drop more attack packets when its detection component becomes more aggressive, i.e., decreasing P_n; however, an increasing number of normal packets will also be dropped due to the increasing false positive probability P_p introduced when the detection becomes more aggressive (as illustrated in Fig. 1). Therefore, to accept a maximum number of normal SYN packets, the adaptive defense system should filter packets such that the packets passing though can fill up the pending buffer without causing overflow.

Define “sojourn time” as the time period a SYN packet resides in the pending buffer. Denote the average sojourn time of a normal SYN packet as T_1, the average sojourn time of an attack SYN packet as T_2. In most cases, normal clients can quickly set up TCP connections with a server, whereas most attack SYN packets will reside in the pending buffer for a long time, many of which will only be removed from the buffer after they are time-out. In the following, we present the adaptive defense system design considering the different sojourn time of normal and attack packets.

The adaptive defense system still has the same architecture shown in Fig. 2. Suppose the server under protection has a pending buffer of size K, which means the buffer can hold K pending TCP connections at the same time. The discrete time interval is denoted by Δ. At the end of the discrete time k, the defense system knows the number of incoming packets during the last time interval, denoted by $N(k)$, and the estimate $\hat{\pi}(k)$. Based on current attack situation, the adaptive defense system determines whether to activate the filtering functionality and finds the best defense parameters $P_p(k+1)$, $P_n(k+1)$.

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

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

When the defense system activates its filtering functionality with the 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:

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

Therefore, at the end of the discrete 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 detection system keeps running in order to know the attack severity $\pi(k+1)$ in the next time interval.
- 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:

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

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

D. Stability study

When the detection decision of one packet is *independent* from the detection decisions of other packets, we derive (details are shown in Appendix) the statistical property of the estimate of attack severity $\hat{\pi}$ (the formula of $\hat{\pi}$ is in (3)):

$$E[\hat{\pi}] = \pi, \quad Var[\hat{\pi}] = \frac{P_p(1-P_p)(1-\pi) + P_n(1-P_n)\pi}{(1-P_n-P_p)^2} \cdot \frac{1}{N} \tag{8}$$

The above equation shows that the unbiased estimate $\hat{\pi}$ has a statistical error due to limited sampling. When we have fewer samples observed in a discrete time interval (i.e., smaller N), the estimate $\hat{\pi}$ could have more statistical error compared with the real value π.

If the adaptive defense system updates its detection parameters (P_p, P_n) at every discrete time based on either (4) or (7), its parameters will always change as $\hat{\pi}$ changes, even if the real attack severity π does not change. Such kind of performance oscillation is unnecessary and undesirable.

The "adaptive defense" principle means that we change system configurations only when the attack situation changes. The adaptive defense system does not need to update its detection/filtering configurations as long as π does not change or changes within a small range. Therefore, in order to make the defense system stable without the oscillation mentioned above, at the end of discrete time k, the adaptive defense system will update its parameters for the $(m+1)$-th times $(m < k-1)$ *only* when:

$$|\hat{\pi}(k) - \hat{\pi}_m| > \epsilon \tag{9}$$

where $\hat{\pi}_m$ is the attack severity estimated at the m-th update. The parameter ϵ exhibits a trade-off between defense sensitivity and stability: when ϵ increases, the adaptive defense system is less sensitive to small changes in the attack severity, but more stable in dealing with the estimation error (8). In addition, by using (9), the adaptive defense system updates its configurations less frequently, which is a desirable property for many defense systems.

Another approach in stabilizing system performance is to implement a low-pass filtering on the estimate $\hat{\pi}$ to remove its high frequency estimation error. A simple low-pass filter is an Exponentially Weighted Moving Average (EWMA) filter. Denote the filtered estimate of the attack severity as $\hat{\pi}'(k)$. When using the EWMA filter, $\hat{\pi}'(k)$ is:

$$\hat{\pi}'(k) = (1-w) \cdot \hat{\pi}'(k-1) + w \cdot \hat{\pi}(k) \tag{10}$$

where $0 \leq w \leq 1$. As w decreases, the filtered estimate $\hat{\pi}'$ becomes more smooth and stable, but slower in response to the changes of attack severity.

Both (9) and (10) have their own advantage and disadvantage. (9) cannot respond to the attack severity changes that are smaller than ϵ, but it can quickly and correctly respond to a big attack change after just one discrete time interval. On the other hand, (10) can respond to an arbitrarily small change in the attack severity, but it is slow in response to changes of the attack severity due to its low-pass filtering. An adaptive defense system should select one of them in its design based on the specific defense requirement of the system under protection.

IV. Adaptive Defense System II: Internet Worm Infection

Computer “worms” are programs that self-propagate across a network exploiting security or policy flaws in widely-used services [22]. Because worms propagate without any human interference, they can quickly spread out to infect most vulnerable hosts in the Internet before people put up any defense [16][20]. In recent years, most fast-spreading worms, such as Code Red, Slammer, Blaster and Sasser [1], are “scan-based worms”: they propagate through randomly scanning IP addresses to find and then compromise vulnerable computers in the Internet. In this section, we study how to design an adaptive defense system for defending against the infection by scan-based worms.

A. Adaptive defense system based on modified TRW detection algorithm

Because a scan-based 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)” [5][6] detects a worm-infected host based on the difference between the number of successful connections and the number of failed connection requests initiated by the host. Weaver *et al.* [23] presented a simplified version of TRW algorithm that is suitable for both hardware and software implementation. For our adaptive defense system for defending against Internet worm infection, we deploy a modified version of the worm detector presented in [23] as the underlying detection algorithm.

We are mostly interested in protecting enterprise networks from worm infection, which is the first defense step before we can effectively defend worm infection in the global Internet. There are two types of worm infection for an enterprise network: one is the infection initiated from infected hosts in the outside Internet; another is worm infection among internal hosts in the enterprise network. The simplified TRW detector [23] is suitable for detecting both types of attacks. If an outside infected host is detected, the defense system puts the IP address of the detected host in a “blacklist” on the edge routers, gateways, or firewalls to filter any traffic from it; if an internal infected host is detected, the defense system relies on “worm containment”, such as the network segmentation theme in [19], to quarantine the worm from spreading out. To decrease the damage caused by false alarms, the defense system usually only blocks the detected hosts on the destination port(s) used in their failed connection requests.

The modified TRW detector we use in this paper works in the following way: each source host that initiates a connection attempt has an associated non-negative "counter", which has the initial value of zero. This counter decreases by one if the source host initiates a successful connection, and increases by one if the source host initiates an unsuccessful connection request. Multiple connection attempts from a source host targeting the same destination host are treated as one connection attempt (e.g., TCP/SYN retransmission before the timeout). A source host is determined to be an attacker when its counter's value reaches W.

The next step in designing the adaptive defense system is to find out how to represent the "attack severity". An enterprise network has a fraction of unused IP addresses. All connection attempts to these unused 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. The nice property of Z is that, as long as we monitor illegal scans in front of the filter, the number of illegal scans observed by the adaptive defense system is not affected by the filtering action — Z correctly represents the worm attack severity.

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 detected host is blocked. The defense system adaptively updates its detection parameter, W, at each discrete time denoted by k $(k = 1, 2, \cdots)$. Denote by $Z(k)$ the number of illegal scans observed during the time interval from discrete time k to $k+1$; $W(k)$ as the detection parameter used from k to $k+1$. Fig. 3 shows that at the end of discrete time k, the adaptive defense system derives the optimal $W(k+1)$ to use for the next discrete time interval based on the current attack severity $Z(k)$. The "optimization" module derives $W(k+1)$ based on the performance function:

$$f = \min_{W(k+1)} \{c_p \cdot \frac{1}{W(k+1)} + c_n \cdot W(k+1) \cdot Z(k)\} \tag{11}$$

where c_p is the cost factor of false positives and c_n is the cost factor of false negatives (damage caused by not detecting an infected host quickly). As the detection parameter W increases, an infected host is able to send more scans and possible infection traffic before it is detected and blocked; but fewer benign hosts would be incorrectly blocked due to false positives. Therefore, $W(k)$ describes the trade-off between false positives and false negatives of the detection algorithm: $c_p/W(k+1)$ corresponds to false positive cost and $c_n \cdot W(k+1) \cdot Z(k)$ corresponds to false negative cost.

B. Adaptive defense system based on "probabilistic marking" detection algorithm

The "Threshold Random Walk" detection algorithm [5] is mathematically solid, and is shown to be effective [6][23]. However, it has the following weaknesses: First, it requires to monitor all legitimate connections, which makes it hard to be deployed by ISPs for the privacy issue of their customers; Second, upon receiving a worm's connection request, the detection algorithm usually needs to wait for a timeout in order to know that the connection request is failed, which makes it slow in response to fast-spreading worms.

In this paper we present a simple alternative worm detection algorithm, called "probabilistic marking", based on illegal scans only (scans to unused IP space): when the detector receives an illegal scan from a source host, the detector treats the source host as an attacker with a probability P. Consecutive illegal scans from the same source targeting the same destination host are treated as one illegal scan (e.g., due to TCP/SYN retransmission).

Such a detection algorithm does not need to wait for a response or a timeout to make its detection decision. If a network has a large fraction of unused IP space, which is the case for many U.S. networks, the probabilistic marking is an effective detection algorithm for fast-spreading worms since it can quickly detect and block worm-infected hosts. In addition, the probabilistic marking detection algorithm does not need to monitor any legitimate traffic, which makes it easier to be deployed on high-speed links and also free from privacy issues.

If the probability P is fixed, an infected host will be detected after the detector observes X illegal scans from the host, where X is a geometric distributed random variable. In the adaptive defense system, P is the adaptive variable and is denoted as $P(k)$ for the value used during the time interval from the discrete time k to $k+1$. The adaptive defense system has the same architecture as shown in Fig. 3 by replacing $W(k+1)$ to $P(k+1)$.

Upon receiving an illegal scan, the probabilistic marking detector has the probability $P_p = P$ and $P_n = (1 - P)$ to incorrectly mark the scan's source host. Therefore, one reasonable performance function f is:

$$f = \min_{P(k+1)} \{c_p \cdot P(k+1) + c_n[1 - P(k+1)]Z(k)\} \tag{12}$$

where the first item corresponds to false positive cost and the second item corresponds to the false negative cost.

One interesting property of this performance function is that it is linear in terms of the variable $P(k+1)$. Therefore, $P(k+1)$ has only two possible optimal values, denoted by $P^*(k+1)$:

$$P^*(k+1) = \begin{cases} 0, & \text{if} \quad Z(k) \leq c_p/c_n \\ 1, & \text{if} \quad Z(k) > c_p/c_n \end{cases} \tag{13}$$

From (13), we can see that such an adaptive defense system is in fact a traditional "*on-off*" defense system with fixed parameters: it activates filtering to block any host that sends illegal scans when the monitored $Z(k)$ is over a predefined threshold H, and deactivates filtering when $Z(k) < H$.

For an on-off defense system, the activation threshold H is usually selected based on experiments or people's experiences. The adaptive defense system with the performance function (12) provides an explanation and guideline on how to choose an appropriate threshold by letting $H = c_p/c_n$. Although we still need to choose the constants c_p, c_n based on experiments or experiences, they have more concrete meanings than the abstract threshold H, and hence, easier to choose.

In (12), the constant c_p means that the false alarm cost is selected to be proportional to the number of blocked benign hosts. If we believe that the false alarm cost increases more quickly than the increase rate of the number of blocked benign hosts, c_p should be a function of P. A simple function of such a relationship could be $c_p = c \cdot P$ where c is a constant. Extended from (12), the adaptive defense system will have the following performance function f:

$$f = \min_{P(k+1)} \{c \cdot P^2(k+1) + c_n[1 - P(k+1)]Z(k)\} \tag{14}$$

The optimal $P^*(k+1)$ for the above performance function could be any value between 0 and 1.

By using such a performance function, the adaptive defense system is able to tune its parameter P smoothly between 0 and 1. Therefore, we prefer to use (14) in the adaptive defense system design.

For the stability issue of the adaptive defense system, we can use the similar approaches as what explained in Section III-D by replacing $\hat{\pi}(k)$ with $Z(k)$. First, the adaptive defense system updates its detection parameters at the end of discrete time k only when:

$$|Z(k) - Z_m| > \epsilon \tag{15}$$

where Z_m is the number of observed illegal scans in the last update.

Second, the adaptive defense system can also use filtered $Z(k)$, denoted by $Z'(k)$, in its optimization, where

$$Z'(k) = (1 - w) \cdot Z'(k-1) + w \cdot Z(k) \tag{16}$$

V. Evaluation

We have presented how to design the adaptive defense system for two major classes of attacks in the previous two sections. In this section, we evaluate the performance of these adaptive defense systems based on either simulation experiments or real attack traces.

A. Defense against SYN flood DDoS

We study the performance of using the adaptive defense system in protecting the server "net.yahoo.com", which hop-count information is included in the data provided by the authors in [4]. For this server, the detection trade-off in terms of P_n vs. P_p is shown in Fig. 1 (based on 24-bit HCF filtering).

1) General cost function: First, we study the adaptive defense system with the general cost function (4). During the entire simulation, we assume the server receives a constant 1,000 normal SYN packets within each discrete time interval Δ (such as 30 seconds). The spoofed SYN flood attack varies its attack intensity as shown in the top figure in 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 figure in Fig. 4 shows how the adaptive defense system automatically tunes its detection parameter δ. This figure shows that as the DDoS attack becomes more severe, the adaptive defense system will implement more aggressive detection and filtering by decreasing δ, i.e., decreasing P_n to reduce false negatives at the cost of increasing P_p accordingly. When attack intensity recedes, the adaptive defense system returns back to the normal conservative detection and filtering status.

In this simulation experiment, $c_p/c_n = 2$; the absolute values of c_p and c_n do not matter because of the performance function (4). If we select a different ratio between c_p and c_n, the adaptive defense system will correspondingly become more aggressive or more conservative as the attack severity changes. In a real implementation, we need to rely on experiments, the server's defense requirement, and people's experiences to select a pair of suitable cost factors c_p and c_n.

The detection trade-off curve in Fig. 1 shows that P_n and P_p are piece-wise linear functions of δ. In addition, the performance function (4) is a linear function of P_n and P_p. Therefore, the performance function (4) is a piece-wise linear function of δ with the inflexion points at the integer values of δ. This is the reason why the optimal δ chosen by the adaptive defense system are always integer values as shown in Fig. 4.

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

The adaptive defense system uses (9) to determine when to update its detection parameter. In the above experiment, $\epsilon = 0.02$ and the adaptive defense system only updates 76 times (the simulation lasts 1000 discrete time).

Because the variance of $\hat{\pi}(k)$ is inversely proportional to the number of received packets in a discrete time interval as shown in (8), we have not used the low-pass filter (10) on $\hat{\pi}(k)$ in the experiment since the defense system observes a relatively large number of SYN packets in each discrete time interval. Therefore, Fig. 4 shows that the adaptive defense system can quickly updates its parameter when the attack changes. If an adaptive defense system can only observe a small number of SYN packets in a discrete time interval, it can use the low-pass filter (10) to stabilize its estimated attack severity $\hat{\pi}$.

2) Buffer-aware performance function: Next, we study the adaptive defense system based on the buffer-aware performance function (7). The discrete time interval $\Delta = 30$ seconds, which is the same as the previous simulation. We assume that normal SYN packets have the average sojourn time $T_1 = 3$ seconds in the pending buffer; attack packets have the average sojourn time $T_2 = 25$ seconds (most attack packets will stay in the buffer until time-out). We assume that the pending buffer can support $K = 200$ connection requests at the same time.

The SYN flood attack follows the same dynamics as what used in previous experiment. Fig. 6 shows the attack scenario and how the adaptive defense system adjusts its detection 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.

3) Performance comparison: Since the second adaptive defense system (the one based on (7)) explicitly optimizes its performance to allow the server to accept the maximum number of normal packets, its performance, in terms of accepted normal traffic, should be better than the first adaptive defense system (the one based on the general cost function (4)). 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 [4]. Fig. 7(a) and Fig. 7(b) show the defense performance in terms of the number of accepted normal SYNs for these two adaptive defense systems, respectively.

Compared with the fixed-parameter defense, at the normal situation when the server is under no attack or light attack, both adaptive defense systems have a much smaller false positive probability P_p and can accept almost all normal incoming SYN requests. When the server is under heavy SYN flood attack, both adaptive defense systems will implement the most aggressive detection and filtering actions ($\delta = 0$), and hence, be able to accept more normal requests than the fixed-parameter defense system. Therefore, an adaptive defense system achieves a better performance than a fixed-parameter defense system both in normal situations and under severe 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.

The results shown in Fig. 7 do not mean that the adaptive defense system based on the general cost function (4) is worse than the other one. The defense system based on (4) requires no congestion information to be provided by the server under protection. A separated "access control" system can be implemented in between of the server and general adaptive defense system to insure that the server has a good response time for all accepted connection requests.

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).

B. Defense against Internet worm infection

In this section, 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. Because of the simple and distinct signature of Slammer worm — each scan packet contains 376 bytes payload [15] — we know that all such UDP packets in the trace belong to Slammer.

This Slammer trace recorded 5 minutes duration of Slammer infection traffic. The top figure in Fig. 8 shows the number of Slammer UDP packets received in each second. The discrete time interval is set to be one second in the adaptive defense system design. Thus this figure shows the dynamic of $Z(k)$. The monitored /16 network has two Internet connections. At the 150 seconds shown in Fig. 8, one of these two connections went down, which is the reason why the monitored Slammer scans dropped suddenly. At the 217 seconds, one internal computer in the /16 network was infected and began to send out a large amount of infection traffic, which caused the monitored Slammer scans dropped suddenly for the second time.

1) Modified TRW detection algorithm: The bottom figure in Fig. 8 shows how the adaptive defense system responds to the attack changes by adjusting its detection threshold $W(k)$. In this experiment, $c_p/c_n = 1000$. When $W(k) = 1$, the system is taking the most aggressive defense action: any host will be blocked as soon as the defense system observes one illegal scan from it (only blocking the host for the port used in its illegal scans).

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).

As shown in Fig. 8, the optimal detection parameter $W(k)$ in the adaptive defense system changes frequently since we haven't use any stability method. If we want the adaptive defense system to have a more stable configuration, we can use (15) to update the detection parameter W only when the change of attack intensity exceeds a threshold ϵ. When we choose $\epsilon = 30$, Fig. 10 shows how the adaptive defense system updates its detection parameter. Comparing with the previous one shown in Fig. 8, we can see that the defense system in Fig. 10 is more stable, but it has a small delay in response to the increases of worm scan traffic at the beginning.

For the evaluation of false positives, [5] and [23] 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.

2) "Probabilistic marking" detection algorithm: For the probabilistic marking detection algorithm and its adaptive defense system (14), Fig. 11 shows how the defense system responds by adjusting the detection parameter $P(k)$. In this experiment, $c/c_n = 200$. When $P(k) = 1$, the system is taking the most aggressive defense action: any host will be blocked as soon as the defense system observes one illegal scan from it.

VI. 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. Actually, this basic “adaptive defense” idea has already been used in many other areas, such as the epidemic disease control in the real world [25], the five-level terrorism alert system [3], etc. The major issue is to find out how to use this basic principle to design a practical defense system. In this paper, we present concrete adaptive defense systems to defend against two major network attacks: SYN flood DDoS attack and Internet worm infection. The adaptive parameter update includes simple estimation and optimization, thus the computational overhead is very small.

The adaptive defense is a high-level system design and there are many good but non-adaptive detection and filtering algorithms. Therefore, we believe the adaptive defense 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 to refine the adaptive defense design. First and most importantly, the adaptive mechanism requires the knowledge of the detection trade-off curve in terms of false positives vs. false negatives. We can obtain such a detection performance curve based on past attacks and simulations. However, a new attack that has a different statistical pattern will have a different detection trade-off. In this case, the adaptive defense system will produce suboptimal defense due to the non-accurate detection trade-off curve used. We will study how to improve the defense performance in this case. One possible way is to continuously update and derive the correct detection trade-off for a new ongoing attack based on the observed attack and detection results.

In addition, we need to further study how to choose the cost factors c_p and c_n quantitatively according to the defense requirements. Third, 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. And finally, 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 have present the primary study on how to improve system robustness in Section III-D. We plan to further study this system robustness issue.

Acknowledgement

We gratefully thank Eric Cronin and Anthony Kurc for sharing their hop-count collection data. We also gratefully thank Andrew Daviel from TRIUMF, Canada for sharing his monitored Slammer trace.

Appendix: Estimation of the attack severity π

Assume that the detection system makes detection decision for each packet independently with the false positive probability P_p and the false negative probability P_n. Suppose within a discrete time interval, the detection system observes N incoming SYN packets, among which N_1 packets are attack packets while the other N_2 packets are normal packets. Since the real attack severity is π, $N_1 = \pi N$ and $N_2 = (1-\pi)N$.

Let the binary random variable $X_i = 1$ when the i-th attack packet is detected, and $X_i = 0$ when this attack packet is not detected ($i = 1, 2, \cdots, N_1$). Let the random variable $Y_j = 1$ when the j-th normal packet is incorrectly detected as an attack while $Y_j = 0$ when this normal packet is correctly treated as a normal one ($j = 1, 2, \cdots, N_2$). Thus we have

$$E[X_i] = 1 - P_n, \qquad Var[X_i] = P_n(1-P_n)$$
$$E[Y_j] = P_p, \qquad Var[Y_j] = P_p(1-P_p)$$

The observed attack severity, π', is derived by:

$$\pi' = \frac{X_1 + X_2 + \cdots X_{N_1} + Y_1 + Y_2 + \cdots Y_{N_2}}{N}$$

The mean value of π' is:

$$E[\pi'] = \frac{N_1 \cdot E[X_i] + N_2 \cdot E[Y_j]}{N} = P_p + (1 - P_n - P_p)\pi$$

which means the estimation formula for attack severity is:

$$\hat{\pi} = \frac{\pi' - P_p}{1 - P_p - P_n}$$

$\hat{\pi}$ is an unbiased estimate of π, i.e.,

$$E[\hat{\pi}] = \pi$$

The variance of the estimate $\hat{\pi}$ is:

$$Var[\hat{\pi}] = \frac{Var[\pi']}{(1 - P_n - P_p)^2} = \frac{N_1 Var[X_i] + N_2 Var[Y_j]}{N^2(1 - P_n - P_p)^2} = \frac{P_p(1 - P_p)(1 - \pi) + P_n(1 - P_n)\pi}{(1 - P_n - P_p)^2} \cdot \frac{1}{N}$$

References

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

[2] T. M. Gil and M. Poletto. MULTOPS: a data-structure for bandwidth attack detection. In *Proceedings of USENIX Security Symposium*, August 2002.

[3] US homeland security advisory system. http://www.dhs.gov/dhspublic/display?theme=29.

[4] C. Jin, H. Wang, and K. G. Shin. Hop-count filtering: an effective defense against spoofed DDoS traffic. In *Proceedings of 10th ACM Conference on Computer and Communications Security*, October 2003.

[5] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 211–225, May 2004.

[6] J. Jung, S. E. Schechter, and A. W. Berger. Fast detection of scanning worm infections. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 59–81, September 2004.

[7] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proceedings of ACM SIGCOMM*, August 2002.

[8] A. Keromytis, V. Misra, and D. Rubenstein. A framework for classifying denial of service attacks. In *Proceedings of ACM SIGCOMM*, August 2003.

[9] H. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of 13th USENIX Security Symposium*, August 2004.

[10] Y. Kim, W. Lau, M. Chuah, and H. Chao. Packetscore: Statistical-based overload control against distributed denial-of-service attacks. In *Proceedings of the IEEE INFOCOM*, March 2004.

[11] W. Lee, W. Fan, M. Miller, S. Stolfo, and E. Zadok. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security*, 10(1,2), 2002.

[12] Q. Li, E.-C. Chang, and M. Chan. On the effectiveness of ddos attacks on statistical filtering. In *Proceedings of the IEEE INFOCOM*, March 2005.

[13] J. Mirkovic, G. Prier, and P. Reiher. Attacking DDoS at the source. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)*, Novemeber 2002.

[14] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2), 2004.

[15] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer worm. *IEEE Magazine on Security and Privacy*, 1(4), July 2003.

[16] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of the IEEE INFOCOM*, March 2003.

[17] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of ACM SIGCOMM*, August 2001.

[18] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, December 2004.

[19] S. Staniford. Containment of scanning worms in enterprise networks. *Journal of Computer Security*, to appear.

[20] S. Staniford, V. Paxson, and N.Weaver. How to own the Internet in your spare time. In *Proceedings of USENIX Security Symposium*, pages 149–167, August 2002.

[21] H. Wang, D. Zhang, and K. G. Shin. Detecting SYN flooding attacks. In *Proceedings of the IEEE INFOCOM*, June 2002.

[22] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In *Proceedings of ACM CCS Workshop on Rapid Malcode (WORM'03)*, pages 11–18, October 2003.

[23] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *Proceedings of 13th USENIX Security Symposium*, pages 29–44, August 2004.

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

[25] C. C. Zou, W. Gong, and D. Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *Proceedings of ACM CCS Workshop on Rapid Malcode (WORM'03)*, pages 51–60, October 2003.

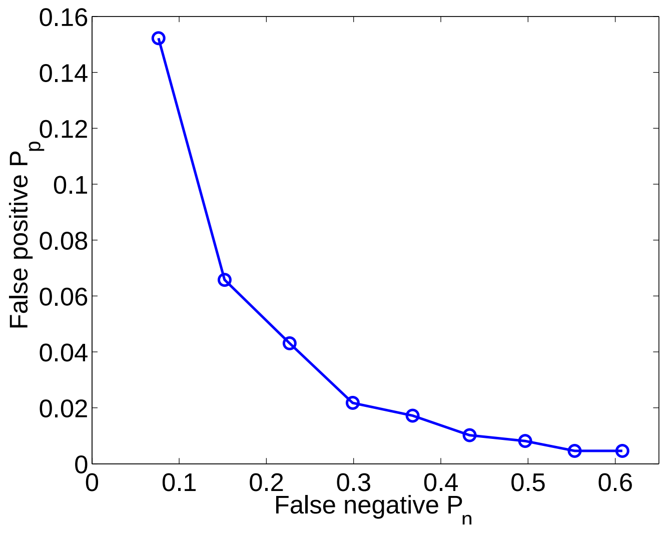


Fig. 1. HCF detection performance under different configurations

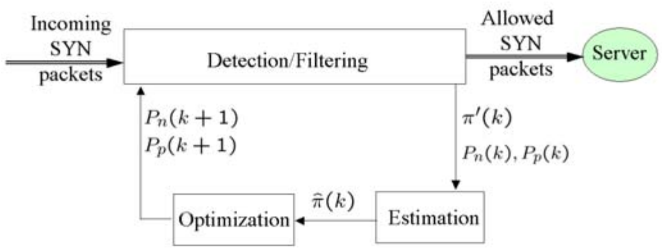


Fig. 2. Adaptive defense system architecture

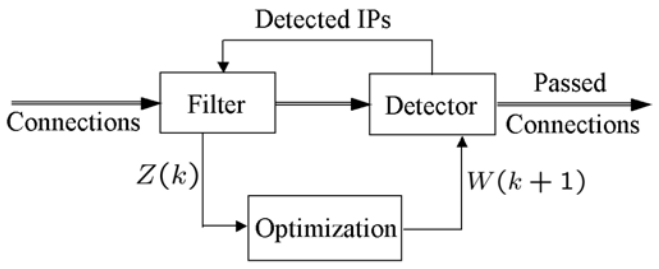


Fig. 3. Adaptive defense system architecture for defending against Internet worm infection

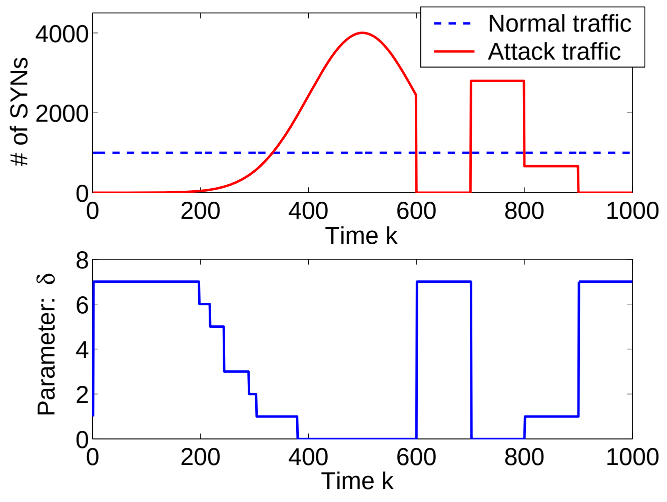


Fig. 4. SYN flood attack scenario and the response by the adaptive defense system based on the general cost function (4)

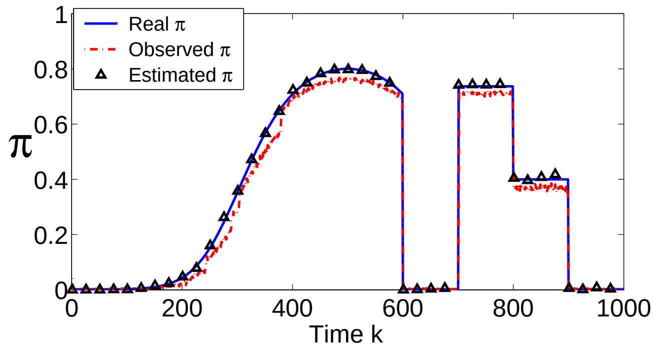


Fig. 5. Verification of the estimation formula (3)

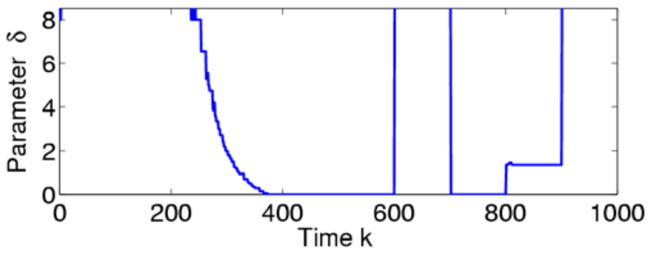
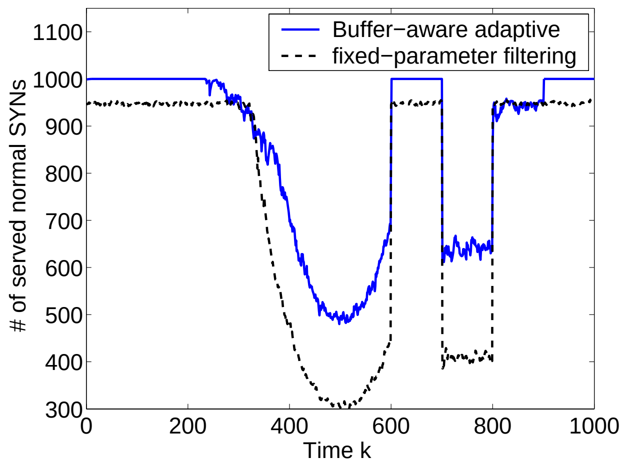
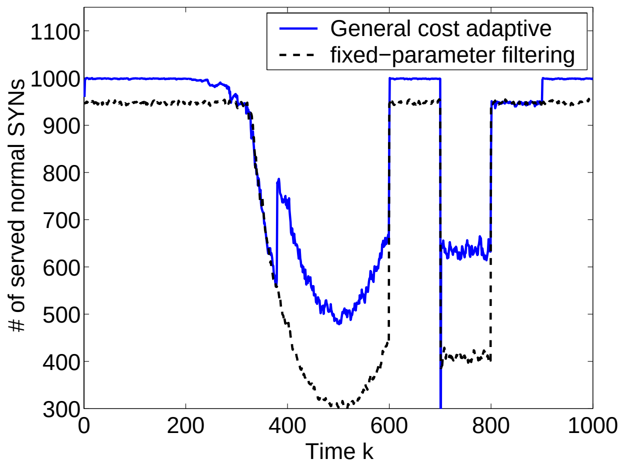


Fig. 6. The response by the adaptive defense system based on the buffer-aware performance function (7)



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



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

Fig. 7. Performance of adaptive defense systems compared with the fixed-parameter system

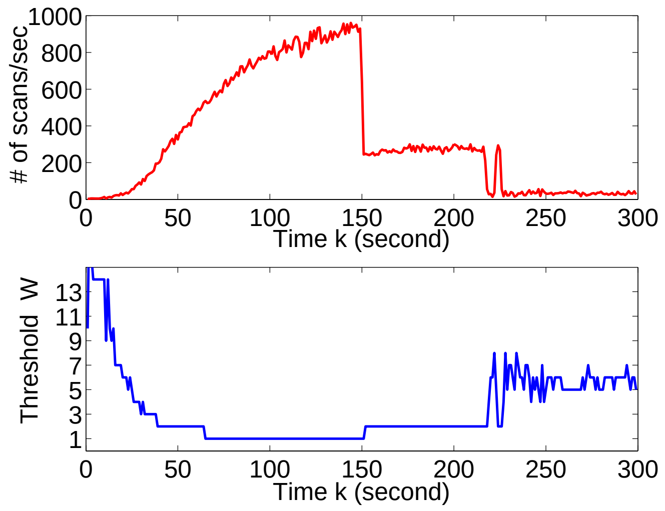


Fig. 8. Slammer attack and the response by the adaptive defense system based on TRW detection

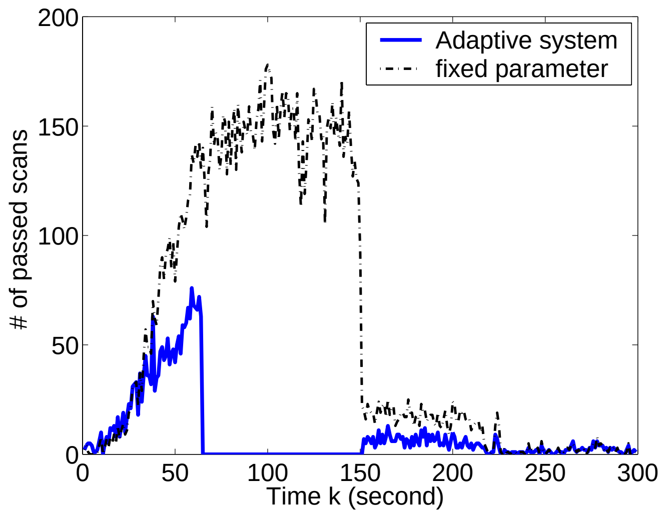


Fig. 9. Worm scans passing the defense system

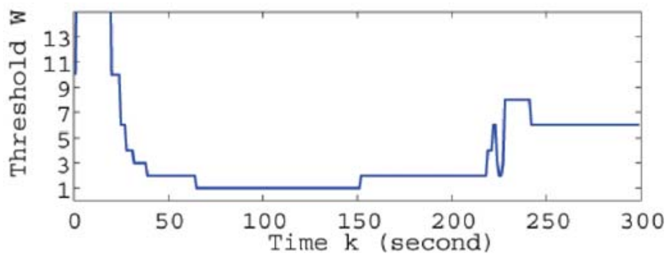


Fig. 10. TRW-based adaptive defense system by using (15) for stability

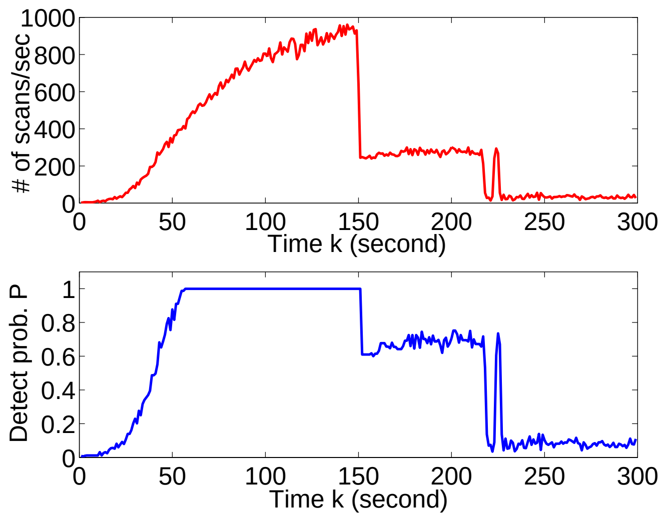


Fig. 11. Slammer attack and the response by the adaptive defense system based on probabilistic marking detection