July 2005

On the Robustness of Router-based Denial-of-Service (DoS) Defense Systems

Ying Xu
University of Pennsylvania

Roch A. Guérin
University of Pennsylvania, guerin@acm.org

Recommended Citation

Ying Xu and Roch A. Guérin, "On the Robustness of Router-based Denial-of-Service (DoS) Defense Systems", . July 2005.

Postprint version. Copyright ACM, 2005. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in *ACM SIGCOMM Computer Communication Review*, Volume 35, Issue 3, July 2005, pages 47-60.
Publisher URL: http://doi.acm.org/10.1145/1070873.1070878

On the Robustness of Router-based Denial-of-Service (DoS) Defense Systems

Abstract
This paper focuses on "router-based" defense mechanisms, and whether they can provide effective solutions to network Denial-of-Service (DoS) attacks. Router-based defenses operate either on traffic aggregates or on individual flows, and have been shown, either alone or in combination with other schemes, e.g., traceback, to be reasonably effective against certain types of basic attacks. Those attacks are, however, relatively brute-force, and usually accompanied by either significant increases in congestion, and/or traffic patterns that are easily identified. It is, therefore, unclear if router-based solutions are viable in the presence of more diverse or sophisticated attacks. As a result, even if incorporating defense mechanisms in the routers themselves has obvious advantages, such schemes have not seen wide deployments. Our ultimate goal is to determine whether it is possible to build router-based defense mechanisms that are effective against a wide range of attacks. This paper describes a first phase of this effort aimed at identifying weaknesses in existing systems. In particular, the paper demonstrates that aggregate defense systems can be readily circumvented, even by a single attacker, through minor modifications of its flooding patterns. Flow-based defenses fare slightly better, but can still be easily fooled by a small number of attackers generating transient flooding patterns. The findings of the paper provide insight into possible approaches for designing better and more robust router-based defense systems.

Keywords
Security, Denial-of-Service, Router

On the Robustness of Router-based Denial-of-Service (DoS) Defense Systems

Ying Xu and Roch Guérin*
Multimedia and Networking Lab
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA, USA
yingx,guerin@ee.upenn.edu

ABSTRACT

This paper focuses on "router-based" defense mechanisms, and whether they can provide effective solutions to network Denial-of-Service (DoS) attacks. Router-based defenses operate either on traffic aggregates or on individual flows, and have been shown, either alone or in combination with other schemes, e.g., traceback, to be reasonably effective against certain types of basic attacks. Those attacks are, however, relatively brute-force, and usually accompanied by either significant increases in congestion, and/or traffic patterns that are easily identified. It is, therefore, unclear if router-based solutions are viable in the presence of more diverse or sophisticated attacks. As a result, even if incorporating defense mechanisms in the routers themselves has obvious advantages, such schemes have not seen wide deployments. Our ultimate goal is to determine whether it is possible to build router-based defense mechanisms that are effective against a wide range of attacks. This paper describes a first phase of this effort aimed at identifying weaknesses in existing systems. In particular, the paper demonstrates that aggregate defense systems can be readily circumvented, even by a single attacker, through minor modifications of its flooding patterns. Flow-based defenses fare slightly better, but can still be easily fooled by a small number of attackers generating transient flooding patterns. The findings of the paper provide insight into possible approaches for designing better and more robust router-based defense systems.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Security, Performance

*This work was supported by NSF grant ITR00-85930.

Keywords

Denial-of-Service, Router-based Defense

1. INTRODUCTION

With the Internet emerging as the de-facto commercial communications infrastructure, it has increasingly become the target of attacks from a broad range of sources. An important category of such attacks consists of network denial-of-service (DoS) attacks, or bandwidth attacks, that directly target network resources such as link capacity and/or router buffers. Network DoS attacks are usually perpetrated by taking advantages of both network and system vulnerabilities. In particular, attackers exploit end-system security flaws to take control of one or more remote hosts that can then be commandeered to mount coordinated flooding attacks. The network's vulnerability to such attacks is compounded by the "best effort" nature of the Internet that does not impose explicit constraints on the traffic rate of individual hosts. As a result, compromised hosts can often be used to source enough traffic to saturate a target network link.

The potential threats of network DoS attacks have raised concerns among the network community and motivated the development of a broad spectrum of defense solutions (see [20] for a comprehensive survey of various types of DoS attacks and defenses). One family of mechanisms is traceback schemes that attempt to identify the sources of attacking traffic in order to shut them off. Traceback mechanisms require adding a light-weighted packet manipulation function at the participating routers. In particular, every traceback-capable router randomly selects a packet (usually with a small probability) and informs the destination of that packet of its own identity, either via header encapsulation [23] or through an additional ICMP message [6]. Hence, during a high-volume traffic flooding attack, the victim will be able to reconstruct the incoming path of the attack traffic once it has gathered a sufficiently large amount of such information. Although traceback schemes automate the process of tracking down where the flooding traffic comes from, they are not very effective at rapidly stopping an ongoing attack since collecting enough information to identify the source of an attack often takes a long time.

Other more reactive solutions address the issue of *on-line detection and mitigation* of network DoS attacks by relying on certain *pre-defined* "normal" system behaviors, usually in terms of properties relating to local network conditions and/or incoming traffic patterns, to throttle attacks. Any behavior deviating from this "norm", once detected, is automatically classified as malicious and subjected to regulation. "Normal" system behaviors can be defined at the transport layer. For example, [10, 19] identify suspicious connections based on comparing forward and reverse traffic patterns with certain pre-defined "normal" patterns derived from typical TCP connections. In [25, 14], statistics relevant to TCP session setup and teardown specifications were utilized to distinguish malicious traffic from legitimate traffic. Most recently, [26] proposes a framework to extract regular connection behaviors based on analyzing network wide traffic, using statistical techniques borrowed from information theory and data mining. Although checking "normal" behaviors at the transport layer can be effective against attacks targeting end-system resources, e.g., SYN flooding attacks, or detecting malicious intrusion attempts, e.g., port scanning activities, it may not be effective at thwarting bandwidth attacks. This is mainly because behaviors defined at the transport layer often do not specify how link resources such as bandwidth or buffer should be accessed. Therefore, DoS attackers may set up regular (TCP) connections, but can still flood at an abnormally high rate to grab link resources by ignoring network congestion notifications.

An arguably more reliable way to characterize "normal behaviors" is to define rules and criteria at the *IP layer* (*network layer*). IP layer is the de-facto "convergence" layer that provides a common interface for various upper layer protocols to utilize network resources. Monitoring network layer behaviors enables one to directly check whether link resources are accessed in a regular manner. In this paper, we focus on defense systems enforcing network layer behaviors. One family of such systems are *router-based defenses* [18, 17, 12, 13, 21] [1] that rely on incoming traffic patterns and/or local router conditions, e.g., congestion level or queue size, to identify DoS attack(ers). Based on the granularity at which abnormal behaviors are defined, router-based defense solutions can be categorized into aggregate level defenses and flow level defenses. Aggregate level defenses throttle bulks of traffic that appear to be malicious. Specifically, incoming packets are classified into distinct traffic aggregates, i.e., a group of packets sharing some common attributes. Any traffic aggregate matching a pre-defined abnormal behavior will be identified and later regulated. A traffic aggregate is usually defined in a *coarse-grain* manner, so that a small number of aggregates can fully profile traffic from all malicious users. For example, in defense schemes such as the Aggregate Congestion Control and Pushback (ACC-Pushback) [17] proposal, traffic aggregates are defined based on destination address clusters. Traffic aggregates responsible for inducing significant link congestion will be identified and rate-limited. In contrast to aggregate level defenses, flow level defenses directly address how a well-behaving user or a flow[2] should send traffic so that every user can get its "fair" share of network resources. Since TCP users dominate the Internet user community, it is reasonable to define "normal" user behavior based on how a TCP flow consumes link resources. Such an approach is taken in solutions such as the RED preferential dropping (RED-PD) proposal [18], which identifies and regulates any flow transmitting at an abnormally higher rate than a standard TCP flow. Aggregate level and flow level defenses exhibit maximal efficacy in different scenarios. Specifically, flow level defenses are most effective when a few brute-force attackers are present, since their high flooding rates will ensure that they are accurately identified and throttled. However, if the attack traffic is composed of a large number of hosts, flow level defenses may not be effective since the rate of each attack flow may be too small to be distinguishable from a legitimate TCP flow. Even if the attack flows could be recognized, their sheer number will make the overhead of managing them prohibitively large. In comparison, it is possible for an aggregate level defense to throttle brute-force DoS attacks involving a large number of hosts. Specifically, as attacking flows can usually be categorized into a few traffic aggregates, the complexity of handling them will be small. Nevertheless, when the number of attacking flows is reasonable, flow level systems impose more stringent control than aggregate level systems, which do not precisely limit the amount of resources consumed by individual users. For example, the ACC system allows flows sharing a common link to consume different proportions of the link bandwidth, as long as they do not induce severe congestion. In contrast, flow level mechanisms continuously impose a common resource consumption criterion on all flows sharing a link, and thus have the extra benefit of strengthening "fair" sharing of the link capacity among flows.

Despite their documented efficiency against brute-force flooding attacks, it is still unclear whether router-based defenses can be truly successful if deployed in real networks. For example, the behaviors of real-world attackers may evolve from current brute-force attacks and render existing proposals ineffective. Our ultimate goal is *to design a router-based defense system that is robust against a broad range of network attacks*. In order to make progress towards achieving this goal, we first focus on two specific systems, ACC-Pushback and RED-PD that we believe are representative of the two major classes of existing router-based systems, and investigate *why* and *how* they can be defeated. This is the main theme of this paper. In [27], we leverage this understanding to develop a new type of router-based defense system that can successfully thwart a much broader range of attacks than existing ones.

Given our goal of assessing the efficacy of router-based defenses against a broad range of network attacks, we first develop a number of "smart" attacking schemes along what we feel are the most natural and "promising" directions. These schemes represent likely choices for attackers to challenge existing defenses as they evolve their current "brute-force" behaviors. When evaluating aggregate defense systems, we focus on investigating whether attacking traffic can both avoid being detected and still significantly degrade TCP performance. We suspect that this can be achieved with a small number of attackers, or even a single attacking host, via a more "gentle" flooding behavior. As for flow level defenses, our focus is on exploring how the efficiency of the defense system diminishes as the number of attacking hosts increases. Clearly, even the addition of a single host sending normal TCP traffic affects the performance of other users. No flow-level defense system can, therefore, protect TCP users against an unlimited number of malicious hosts. However, taking control of a very large number of end-systems can be difficult, so that attackers will likely prefer using a smaller number of hosts to achieve a given impact. To understand how the number of distinct attacking hosts affects the impact of an attack, we first evaluate, as a benchmark, the effect of a single attacking host that intelligently shapes its traffic. Next, we examine the extent to which defense systems can be defeated by a relatively small number of such "intelligent" attackers. A related but different scenario that we also explore for flow-level defenses, is the availability of *multiple identities* (or source addresses) at a single host. A host that assumes multiple identities (spoof its source address)

[1] Several commercial products with similar goals are also available on the market [1, 2, 4]. These products usually rely on a hybrid set of rules, often covering multiple layers, to detect network anomalies and filter out suspicious traffic. However, due to the complexity of the rule set, many of them can only achieve semi-automatic operation and require network administrators to decide which filters to install. In addition, the details of the algorithms they use to identify anomalies and regulate attackers are not publicized. Thus, we do not study these systems in this paper.

[2] The traffic of an individual user is often represented by a "flow", which is a *fine-grain* traffic descriptor associated with a particular combination of source/destination addresses, port numbers and protocol ID.

makes its attacking traffic appear as if it originates from distinct end-systems. When the number of available spoofed address is large, defense mechanisms can then be fooled into believing that the overall traffic originates from multiple legitimate hosts. As in the multi-host case, potential attackers may have access to only a limited number of addresses because of increased policing by service providers that check for legitimate source addresses at their network boundary. Understanding how an attacker's impact varies with the number of addresses at its disposal is, therefore, of special interest in this context.

We tested the "smart" attacking schemes we developed on both aggregate and flow level DoS defense mechanisms. Our results reveal that existing router-based schemes are indeed relatively easy to defeat, and we are currently applying the understanding gained in this paper towards designing more robust mechanisms [27]. The rest of the paper is structured as follows. In Section 2, we review the defense systems we consider and the techniques they use to combat DoS attacks. We describe the setup we use for evaluation purposes in Section 3, and report on the performance of aggregate and flow level defense systems when attacked by a single "smart" attacker in Section 4 and 5, respectively. Section 6 further explores scenarios involving multiple attackers and attackers that are able to assume multiple identities. Section 7 is devoted to a brief review of several related studies. Finally, we conclude in Section 8 and point out directions for further study.

2. ROUTER-BASED DEFENSE SYSTEMS

In this section, we review the two main families of router-based defense mechanisms we are considering in this initial study. As mentioned before, router-based defense systems can be classified into *aggregate level* and *flow level* systems, based on the granularity at which "abnormal behaviors" are defined. For each type of systems, we highlight the key components they rely on.

2.1 Aggregate Level Defense Systems

We start with aggregate DoS defense systems. The most fundamental design issues for this type of defenses are the definition of aggregates and what is considered malicious behavior. Although there is no consensus regarding what kind of traffic should be viewed as "abnormal", the conventional wisdom is that DoS attacks usually exhibit some unique characteristics. First, although DoS attacks can originate from multiple sources, the traffic usually heads for a common destination, be it a specific host machine or a targeted network domain. Second, in order to generate maximum disruption, the attack traffic often consists of high-bandwidth packet streams that create heavy congestion along their path. From those observations, it is natural to define as abnormal traffic aggregates with a high volume of packets that *both* contribute to severe congestion *and* share a common destination.

In this paper, we concentrate on a representative aggregate level defense system, the ACC-Pushback proposal [17], that relies on a similar definition of abnormal traffic. The ACC-Pushback solution can be divided into two components, the basic local ACC mechanism and the extra Pushback mechanism. The initial DoS detection and mitigation are undertaken by the local ACC mechanism. As illustrated in Figure 1, the local ACC system is built on the RED queue management scheme [9], and counts the number of random packet drops generated by the RED queue. If the observed K-second link loss rate exceeds a given threshold p_{high}, the ACC system decides that the monitored link is under attack and activates an identification agent that starts searching for malicious traffic. Specifically, packets are clustered into traffic aggregates based on their destination addresses[3]; traffic aggregates that contribute the largest amount of packet drops are identified as potentially malicious. One important configuration knob of the local ACC is p_{high}, which needs to be chosen so as to achieve a reasonable trade-off between mis-detections (false negatives) and false alarms (false positives). In [17], p_{high} is set at 10% by default to ensure that the detection mechanism only functions when there is sufficient proof of ongoing attacks. It is not difficult to see that such a mechanism is predicated on the assumption that DoS attacks are always accompanied by heavy congestion. In the following, we show that an attacker employing an alternative flooding strategy can successfully evade the ACC defense by not triggering large amount of packet losses, while still causing significant disruption to TCP flows..

After a traffic aggregate is classified as malicious, it is rate-limited in a pre-filter (See Figure 1). The goal of the pre-filter is to bring the link loss rate back down to a value lower than p_{high}. In addition to the local pre-filter, the regulation of aggregates identified as malicious can also be propagated to upstream routers through the Pushback mechanism. Specifically, the aggregates are split into smaller streams (aggregates) based on where they are coming from. Pre-filtering is then performed on the largest traffic streams at upstream routers. In many scenarios, the Pushback mechanism provides better discrimination between attacking and normal traffic. For instance, if the attack traffic only comes from a few incoming links, legitimate traffic from other directions will not be penalized even if they all share the same set of destinations. In spite of such extra benefit, we will not explicitly evaluate the performance of the Pushback mechanism, as our goal is to assess the extent to which the local ACC mechanism can fail to *detect* malicious traffic.

2.2 Flow Level Defense Systems

Flow level defense systems take a more forceful approach, as they rely on an a priori definition of the "normal" behavior of individual flows, which they constantly strive to enforce by identifying flows that deviate from this norm and forcing them into conformance. A commonly used definition of good behavior is that of fair[4] sharing of resources among flows. This approach is embodied in defense systems that utilize *preferential dropping* mechanisms to control resource consumption, including proposals such as: Core-Stateless Fair Queueing (CSFQ) [24], Flow Random Early Detection (FRED) [16], CHOKe [22], Approximate Fair Dropping (AFD) [21], and RED preferential dropping (RED-PD) [18]. These systems estimate the resource usage of a certain set of flows, and accordingly assign them dropping probabilities.

In this paper, we focus on one representative flow level defense mechanism, namely, the RED-PD proposal [18]. The goal of the RED-PD system is to ensure that the transmission rate of each user does not exceed the throughput of a *standard* TCP flow under the same network condition, i.e., to enforce TCP-friendliness. Specifically, the throughput of a standard TCP flow is well-known to be roughly equal to:

$$f(r,p) = \frac{\sqrt{1.5}}{r\sqrt{p}} \qquad (1)$$

where r and p are its round-trip time and loss rate, respectively [8]. Hence, when the loss rate is p, the RED-PD system attempts to enforce a maximum flow rate of $f(r,p)$.

Enforcing this constraint could be achieved by measuring the bandwidth of individual flows and comparing these values to $f(r,p)$.

[3]See [17] for the detailed clustering algorithm.

[4]Fairness can have many interpretations, such as max-min fairness or TCP-friendliness. In this paper, we assume TCP-friendliness as our fairness criterion.

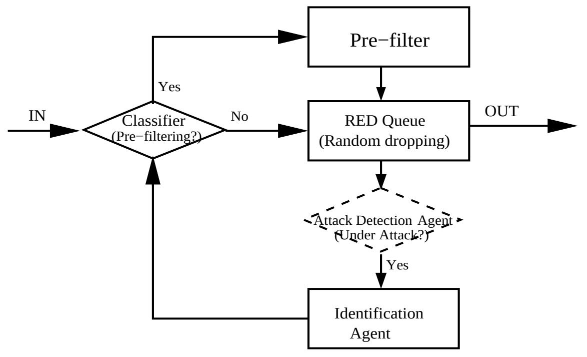


Figure 1: Architecture of Router-based Defense

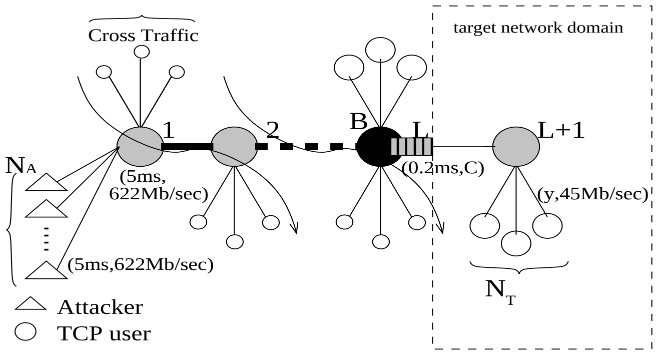


Figure 2: System Topology

This approach is adopted in the CSFQ and FRED proposals. However, tracking all individual flows can represent a significant overhead when the number of flows is large. To avoid such burden, the RED-PD system relies on packet losses to *sample* the incoming traffic. The RED-PD architecture is similar to that of Figure 1, with the difference that it does not involve an attack detection component. Instead, the random drops of the RED queue are directly fed to the identification agent, where they are used to estimate the bandwidth consumption of individual flows.

The algorithm used by the RED-PD system to identify a TCP-unfriendly flow is based on a simple rationale, which assumes that the number of packet losses of a flow is proportional to its number of transmitted packets. Specifically, let $T(r,p)$ be equal to $\frac{r}{\sqrt{1.5p}}$, i.e., the average duration of the period in which a standard TCP flow loses one packet. The RED-PD system collects a trace of packet drops, and scans this trace every $K * T(r,p)$ seconds. Any flow that contributes more than K losses in one such scanning period is deemed TCP-unfriendly. The actual RED-PD design divides a scanning period into M lists, each equal to $\frac{K}{M}T(r,p)$ seconds. Any flow that contributes losses in at least K out of M lists in the *current* scanning period is marked as non-conformant at the end of this period. According to [18], using M lists provides a better tolerance to bursty traffic and can reduce the chances of wrongly punishing short TCP flows or TCP friendly flows that respond to congestion relatively slowly. These flows might remain non-conformant for some time before responding to congestion notifications. Hence, quickly marking them as TCP-unfriendly would increase the probability of false alarms. Once a flow has been identified as non-conformant, its traffic is sent to the pre-filter, where it is subjected to additional dropping before entering the RED queue. The dropping probability applied to a specific non-conformant flow, denoted as p_d, is adjusted adaptively (See Fig.7 and Fig.8 in [18]). Specifically, after a flow is identified, its dropping probability p_d will be increased by an amount equal to p_Δ. p_d will keep on increasing until the remaining traffic of this flow is deemed conformant in a subsequent scanning period. After that, p_d will be decreased as long as the flow continues to be TCP-friendly. Once p_d is less than p_{min}, the flow is released from the pre-filter and not subjected to any further regulation.

When testing the RED-PD system, we focus on evaluating the performance of its detection and traffic regulation mechanisms. It is not difficult to see that these mechanisms are most effective against a few high rate attackers, since each will experience a relatively large number of packet losses, and therefore be quickly detected. However, as we shall see, this situation changes when an attacker adopts a more adaptive attacking strategy.

3. EVALUATION SETUP

We carry out our investigation using the topology of Figure 2, which consists of L links shared by a number of TCP flows and a set of malicious hosts. The first $L-1$ links are backbone (non-bottleneck) links, while the Lth link is the edge (bottleneck) link[5]. Each backbone link has a capacity of 622 Mb/sec and a propagation delay equal to 5 ms, and carries cross traffic generated by 10 long-lived TCP flows. The bottleneck link has a capacity of C Mb/sec and a propagation delay of 0.2ms, shared by N_T long-lived TCP users. The access links of all TCP users have a speed of 45 Mb/sec and a delay of y ms. We do not impose any window size limitations on the TCP flows, so that their traffic can grow to consume the full link bandwidth. In addition, there are N_A attacking sources sending traffic across both the backbone and the bottleneck links. We assume that each attacking source is connected over a high-speed access link of, say 622 Mb/sec and with a 5 ms delay. In our experiments, parameters such as L, N_A, N_T, C, y can all be varied. Unless stated explicitly, we use the following default setting: $L = 1$ (single link), $N_A = 1$ (single attacker), $N_T = 32$, $C = 45$Mb/sec, $y = 10$ms. We also assume that the queue at the bottleneck link is a RED queue, with a total size of 1000 packets. The min_{th}, max_{th} and q_{weight} of the RED queue are set to 5 packets, 200 packets, and 0.002, respectively. The specific RED parameter setting was found to have minor impact on the performance of the defense mechanisms.

Though simple, Figure 2 is representative of how a private network, e.g., an enterprise or a campus network, connects to the Internet. In such configurations, the link connecting the private network to the backbone is usually the bottleneck link and a potential target for external attackers. An attacker can flood this link simply by sending packets destined for any machine inside the target network.

4. PERFORMANCE OF AGGREGATE LEVEL DEFENSES

In this section, we explore the performance of aggregate-based defense systems, particularly, the ACC system, when under “intelligent” DoS attack.

4.1 Bandwidth Soaking Attack

4.1.1 Intuition and Attacking Behavior

Current DoS attacks are typically brute-force, with attackers flooding the target link with more traffic than it can carry. This triggers persistent congestion and high link loss rate. As demonstrated

[5]In many settings, the speed of the edge (bottleneck) link can still be quite large, e.g., over 100Mb/sec

```
Send probing packets at a fixed rate of $r_{pro}$
foreach received feedback packet $i$
        Extract sequence number $seq_i$
        $N_l^i = seq_i - seq_{i-1} - 1$
        if $N_l^i > 0$ $R = R - N_l^i r_{decr}$
        $R = R + r_{incr}$
($N_l^i$: Number of lost packets
    indicated by feedback packet $i$)
end
```

Table 1: Bandwidth Soaking Attacker

in [17], an ACC system deployed on the target link can easily detect such an attack. Unless the attack traffic is highly distributed across the network, it can be contained either locally or by pushing back to an upstream router close to the attacking sources. In this section, we assess the performance of an ACC system in the face of a "gentler" type of DoS attackers. Instead of directly blasting at its highest possible rate, this type of attackers *gradually* increase the rate at which they flood the target link until the loss rate *reaches* a certain value p_{thres} that corresponds to only modest congestion, i.e., is below the activation threshold of the ACC system. The attacker then tries to *stabilize* the link congestion at that level using only small rate adjustments. Intuitively, we expect that gradual rate increases can help an attacker slowly increase congestion without ever creating any abnormal traffic patterns seen at the aggregate level. Specifically, as the attacker increases its rate by a small amount, the link loss rate also increases, but only slightly, since the increase in the total input rate is also small. The nature of TCP congestion control makes in turn normal TCP users reduce their own transmission rates. Hence, the slight increase in the rate of the flooding traffic is compensated for by the lost throughput on the part of TCP users. As this process continues, the attacker keeps grabbing more and more bandwidth from the TCP users. When the link loss rate ultimately reaches p_{thres}, the attacker will often[6] have grabbed a large fraction of the link bandwidth away from the TCP users without ever incurring sufficiently heavy losses to trigger detection.

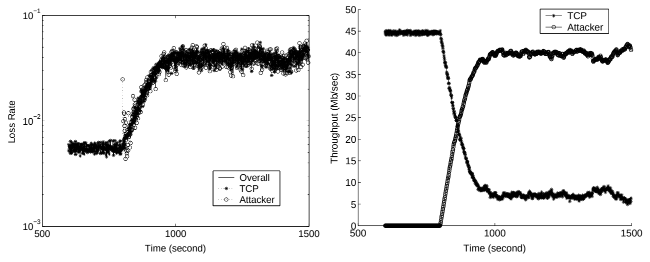
We use the name "bandwidth soaking" to capture the behavior of this type of attacker. To achieve its goal, a "bandwidth soaking" attacker needs to be aware of the congestion level at the target link and adjust its flooding rate accordingly. Next, we present a prototype design that realizes this using a simple rate adaptation mechanism.

4.1.2 Attacking Strategy

A "bandwidth soaking" attacker sends out two types of packets: *probing* packets and *flooding* packets. We assume that probing packets are sent at a fixed rate r_{pro} and contain increasing sequence numbers. Each probe triggers one feedback packet carrying the same sequence number (more on how to realize this later), so that the number of lost probes can be measured at the source. This information can then be used to control the rate of the flooding packets, whose main purpose is to load the target link. Table 1 describes the algorithm used to adjust the total transmission rate R, i.e., the flooding rate plus probing rate. R is adjusted in an additive-increase additive-decrease (AIAD) manner that provides for relatively smooth rate variations.

In order to drive the loss rate of the bottleneck link up to p_{thres}, the ratio between rate increases (r_{incr}) and decreases (r_{decr}) used

[6]Clearly, this is a function of the initial level of congestion on the link.



(a) Dynamics:RED (b) Traffic Rates: RED

Figure 3: Bandwidth Soaking Attacker: RED Queue

by the algorithm should satisfy:

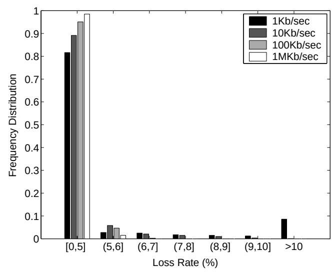
$$\frac{r_{decr}}{r_{incr}} = \frac{1}{p_{thres}} - 1 \qquad (2)$$

The rationale behind this relation is best understood through an example. Consider a scenario where the current loss rate is p, and further assume that the loss rate experienced by the probing traffic is also p. Then, for every K probing packets, the number of observed feedback packets is approximately $K(1-p)$, and the corresponding net rate adjustment, denoted as Δ_K, is equal to $K[(1-p)r_{incr} - p \cdot r_{decr}]$. It can be easily seen that when (2) holds, Δ_K is positive if p is smaller than p_{thres}, and negative otherwise. Hence, the above rate adaptation algorithm forms a feedback loop that attempts to stabilize the link loss rate at p_{thres}. As we will see in most of our experiments, a "bandwidth soaking" attacker can indeed keep the link loss rate closely around this target.

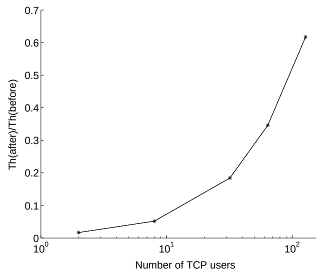
Implementing the above scheme can be done relatively easily and we briefly outline possible options. Because probing packets must generate feedback, ICMP, TCP SYN and TCP FIN are possible packet types. In contrast, flooding packets should generate as little feedback as possible, in order not to congest the reverse path. Reverse path congestion is undesirable, since it could cause extra lost feedback packets, which would in turn trigger unnecessary decreases in the flooding rate. This can be achieved by corrupting the IP checksum of flooding packets or simply by using TCP RST packets, as many operating systems quietly drop such packets. In addition, the probing rate r_{pro} needs to be kept low, not only to avoid reverse path congestion, but also because it is common for firewalls to rate limit ICMP, TCP SYN or TCP FIN packets. Too high a probing rate would, therefore, translate into additional losses that are not caused by congestion, which would again affect the accuracy of the rate adjustment mechanism.

4.1.2.1 Illustration.

We integrated the basic "bandwidth soaking" attacking strategy into the NS2 [3] simulator, and tested its performance using the topology of Figure 2 together with the default parameter setting specified in Section 3. Unless stated elsewhere, we used $p_{thres} = 4\%$, a value that is well below the 10% default detection threshold of the ACC system. In addition, r_{incr} was set to $1\% C/r_{pro}$, so that the attacker can grab an additional 1% of the link capacity every second after the attack starts. In this experiment, the probing rate r_{pro} was also set to 1% of the link capacity. The experiment lasted for 2000 seconds, and the attacker initiated its attack at the 800th second. As our focus is on the detection component of the local ACC mechanism, we did not activate its regulation mechanism, so

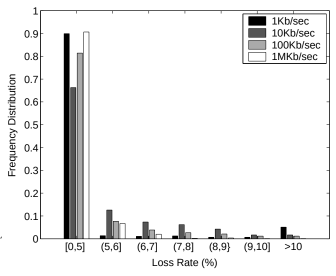


(a) Impact of Probing Rate

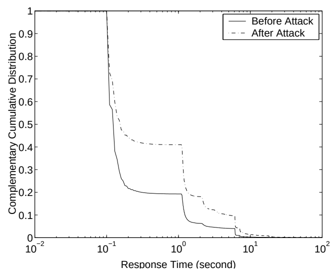


(b) TCP Throughput Change

Figure 4: Multiplexing Long-lived TCP flows



(a) Impact of Probing Rate



(b) Response Time Increase

Figure 5: Mixed Traffic Scenario

that it actually behaved like a standard RED queue. Figure 3 reports the results of our experiment.

Figure 3(a) and 3(b) plot the 1-second loss rates and the evolution of the throughput of both TCP users and the attacker. As shown in figure 3(a), the attacker has successfully driven the link loss rate up to the 4% target. Moreover, the 1-second loss rate indeed stabilizes around 4%, with only small fluctuations (about 1%). As a result, the attacker remains undetected by the ACC mechanism. Moreover, the loss rate of TCP flows has increased from initially less than 1% to 4%, which translates into a significant reduction in their throughput. This can be observed in Figure 3(b), in which the TCP throughput gradually decreases by more than 80%. Meanwhile, the attacker's transmission rate continuously increases and when the system stabilizes, it has grabbed a major fraction of the total link capacity.

4.2 ACC Defense versus Bandwidth Soaking Attacker

The previous section demonstrates that the ACC defense can be defeated even by a single "bandwidth soaking" attacker, at least in *one* specific setting. We now evaluate whether this remains true across a *range* of configurations that involve varying parameters such as the probing rate r_{pro}, the bottleneck link capacity C, the number of TCP users aggregated N_T and the number of links L. We assess the efficacy of the defense system based on two different measures, namely, the likelihood that it detects a malicious traffic aggregate and the degree of protection it provides to TCP flows. An efficient defense scheme should detect attacking traffic with high probability, and prevent significant reductions in TCP throughput. As in the previous experiment, the rate-limiting function of the ACC system is turned off when evaluating how often an attack is detected. However, rate-limiting is turned on when evaluating TCP throughput degradations. Two different environments are considered: one involving only long-lived TCP flows and one involving both long-lived and short-lived TCP flows.

4.2.1 *Long-lived TCP Environment*

We first consider a scenario where normal users are all long-lived TCP flows. The default parameter setting given in Section 3 is again adopted as a baseline configuration, and we vary one parameter at a time to explore the impact of those changes.

4.2.1.1 *Probability of detection.*

We found that the ACC defense cannot detect a "bandwidth soaking" attacker unless the attacker uses a very small probing rate r_{pro}. This is demonstrated in figure 4(a), in which the frequency distribution of the 1-second link loss rate p is plotted, for different values of r_{pro}. As can be observed, when r_{pro} is very low, the probability that a "bandwidth soaking" attacker is detected is non-negligible, as link congestion frequently exceeds the 10% threshold. For example, if $r_{pro} = 1Kb/sec$, p is higher than 10% in more than 15% of the intervals. Intuitively, this is because a lower probing rate results in slower responses to changing network conditions. If r_{pro} is too small, the attacker cannot always detect a queue overflow event in a timely fashion, and thus may trigger significant congestion before reducing its flooding rate. Nevertheless, as shown in figure 4(a), the attacker can evade detection completely if r_{pro} is greater than $100Kb/sec$ (0.2%C), as in these cases the link loss rate remains below 7%. A number of other configurations with different combinations of C, N_T and L values were also tested, and did not result in significant differences, i.e., a probing rate of about 0.2%C consistently allowed the attacker to avoid detection.

4.2.1.2 *TCP performance change.*

When the ACC system fails to detect a "bandwidth soaking" attacker, it lets all the attacking traffic in and offers virtually no protection to TCP users. The resulting TCP throughput reduction depends in part on the number of TCP flows on the bottleneck link, and on the evolution of the overall loss rate because of the attacking traffic. The number of TCP flows competing for resources determines the initial loss rate, while the loss target selected by the attacker identifies the relative loss increase that the TCP flows will experience. We illustrate this in figure 4(b), which plots the ratio of the TCP throughput after and before the attack, when the number of TCP users N_T increases from 2 to 128. From the figure, we see that performance degradations are greater when N_T is smaller. Specifically, the TCP throughput is reduced by more than 80% when N_T is smaller than 30. However, when N_T is larger than 100, the throughput reduction is only about 40%. This is because the larger number of TCP flows resulted in a higher initial loss rate, and hence a smaller relative increase when the link loss rate reached the attacker's 4% target. For the same reason, the throughput reduction achieved by an attack also varies as a function of the bottleneck capacity C. In particular, higher speed links are more susceptible to larger decreases.

4.2.2 *Mixed Traffi c Environment*

The second scenario we consider is a mixed traffic environment where the bottleneck link carries both long-lived and short-lived TCP flows[7], together with a number of ON-OFF UDP sources. This

[7]The short-lived TCP traffic was generated using the NS2 WEB

is probably a more realistic traffic mix, with long-lived TCP, short-lived TCP and UDP flows representing traffic generated by FTP file transfer applications, web browsing applications and real-time streaming applications. We set the number of long-lived TCP flows to 32, and tune the load of the short-lived TCP and UDP traffic so that they constitute roughly 30% and 10% of the total traffic, respectively. The long-lived TCP flows contribute the remaining 60%. The other system parameters are again set to their default values. When the attacker is absent, such a configuration yields an overall loss rate approximately equal to 1%.

4.2.2.1 Probability of detection.

We first evaluate the probability of detecting the attacker. As with the previous experiment, we evaluate this based on the distribution of the 1-second loss rate for different values of r_{pro}, as plotted in figure 5(a). One can observe that an attacker can be detected with a slightly higher probability in this case. In particular, even probing rate as high as 1Mb/sec cannot completely eliminate periods during which losses exceed 10%. This is mainly due to the presence of the short-lived TCP flows, which generate burstier traffic and are less responsive to variations in congestion levels than long-lived TCP flows. An off-line inspection of the traffic dynamics indicates that the overall rate of the short-lived TCP flows indeed varies significantly over short periods of time. Because neither long-lived TCP flows nor attacker are able to react as fast, this often results in a high instantaneous link loss rate. However, even taking into account those transient excursions above the 10% loss level, the probability that the attack is detected remains small (the loss rate exceeds 10% in less than 2% of the time), as long as r_{pro} is larger than 100Kb/sec.

4.2.2.2 TCP performance change.

Although the defense system can occasionally detect attackers, the detection probability is so small that it is unlikely to greatly reduce the level of disruption an attacker can create. This was verified in our experiments. Specifically, when the ACC regulation is turned on, the overall loss rate of all long-lived TCP flows still increases from 1% to about 4%, making their throughput decrease by more than 70%. The loss rate of the UDP flows also grows by a similar amount. As for short TCP sessions, we anticipate that their performance, measured in terms of session response time will also be adversely affected because of the higher loss rate. We demonstrate this in figure 5(b), which plots the complementary cumulative distribution function (CCDF) of the response time of short-lived TCP sessions before and after the attack. As we can see, more than 40% of the short-lived sessions take more than 1 second to complete their transaction, while this number was below 20% before the attack started. Moreover, 10% of the short-lived sessions experience response times in excess of 6 seconds, while this was the case for only 2.5% of them before the attack. On average, the transmission delay across all the TCP sessions increases by approximately 133% after the attack.

4.3 Comments and Discussions

We have shown that a simple progressive rate increase strategy, "bandwidth soaking", allowed attackers to avoid detection by an ACC-type system. In [28][Appendix I], we further analyze the vulnerability of the ACC defense to a broader range of attacking strategies, where the attacker's rate adaptation algorithm belongs to the broad family of so-called "binomial" rate adaptation mechanisms [5]. By analyzing the behavior of the stabilized system, we establish that the ACC defense system is most vulnerable to a particular subset of binomial algorithms. The "bandwidth soaking" mechanism, represents one specific algorithm within this subset.

Our findings indicate that aggregate level schemes are unlikely to offer a robust solution against broad range of attacks. In particular, they allow situations where although the overall congestion level appears to be "normal", the sharing of link bandwidth among flows is quite "unfair". As we illustrated, a "bandwidth soaking" attacker can gradually nudge out legitimate TCP users and seize a large amount of the link bandwidth without creating significant losses. Obviously, the likelihood a "bandwidth soaking" attacker can avoid detection heavily depends on the specific p_{thres} it chooses as compared with p_{high} configured at the targeted ACC defense. If they are too close, the attacker will likely be identified. However, p_{high} cannot be set to a very low value, as this would also increase the probability of false alarms. For example, setting p_{high} close to 1% (the case in 4.2.2) would cause some legitimate flows to be frequently identified and punished. Lacking the knowledge of the range of "regular" link loss rate during overload, it is expected that in practice p_{high} would be set to a large value, e.g., $> 5\%$, to ensure sufficient transparency to TCP flows. In this case, there should still exist a wide range of p_{thres} that allows an attacker to remain undetected. In fact, in an extreme case where p_{high} is set to zero, the ACC system would be constantly scanning for attackers, and would now have to rely on a more precise description of what consists an abnormal behavior to reduce false alarms. In other words, simply classifying as malicious the aggregates losing most packets would result in the *constant* unwarranted penalization of certain flows. Instead, a finer-grain, e.g., at the flow level, description of normal behavior would be needed. In the next section, we consider a specific flow level defense system, namely, the RED-PD system. By directly addressing traffic anomalies at the flow level, the RED-PD system is expected to yield a greater detection capability, especially in the presence of attackers such as the "bandwidth soaking" attacker that ends up with an abnormally large bandwidth usage.

5. PERFORMANCE OF FLOW LEVEL DEFENSES: SINGLE ATTACKER

In this section, we evaluate the effectiveness of the RED-PD system against "intelligent" DoS attacks. Our first step is to validate that the RED-PD system can indeed thwart a single "bandwidth soaking" attacker.

5.1 RED-PD Defense versus Bandwidth Soaking Attacker

Recall that the RED-PD system periodically scans the loss trace and in every scanning interval marks as TCP-unfriendly flows losing more packets than a "regular" TCP flow, i.e., experiencing losses in at least K out of M scanning lists in a scanning interval (period). A "bandwidth soaking" attacker needs to grab a large portion of the link capacity in order to be disruptive, and for a given loss rate this is likely to generate many lost packets evenly distributed over time. As a result, the attacker will have losses in most scanning lists during a RED-PD scanning period, which should ensure that it is detected and, therefore, mostly filtered out. To verify this hypothesis, we conducted a number of experiments, which showed that when a single "bandwidth soaking" attacker targeted a RED-PD link, long-lived TCP flows were able to retain on average about 95% of their initial throughput, while the response time of short-lived TCP flows grew only by less than 15%.

5.2 Periodical Blasting Attacker

traffic generator.

5.2.1 Intuition and Basic Behavior

Although the RED-PD system can successfully throttle slowly increasing attacking traffic, it may not perform well in the presence of more rapid adaptation patterns. In this sub-section, we examine the potential damage of a highly transient attacker that deliberately alternates between burst (ON) and idle (OFF) periods to avoid detection. Intuitively, in order for such an attacker to be effective, it would blast traffic long enough to create congestion, switch to idle before incurring enough losses to be detected, and remain silent until congestion had subsided at which point the attack cycle could resume. By creating sufficient congestion, the attacker would ensure that TCP flows experience an increase in losses, and by remaining silent for a sufficient amount of time, the attacker would presumably minimize its chances of being detected and filtered. We use the name "periodical blasting" to capture the ON-OFF behavior of this type of attacker. The main challenge with configuring a "periodical blasting" attacker, is in determining how long to stay ON and how long to stay OFF, so as to create congestion while avoiding regulation.

5.2.2 Attacking Strategy

We decompose the activity of a "periodical blasting" attacker into a series of ON-OFF periods. In each ON period, the attacker blasts at a rate R for a duration of b. Once the attacker turns OFF, it remains silent for a fixed amount of time I. The full specification of such an attacking scheme calls for determining how all three parameters $< R, b, I >$ are chosen.

Because the RED queue randomly drops incoming packets, it is highly likely that as congestion increases while an attacker is ON, the attacker itself would experience losses. The attacker would then be flagged as having incurred losses in the associated scanning lists maintained by the RED-PD system. In order to avoid detection, an attacker should stop transmitting before it overlaps with K scanning lists. From an attacking strategy standpoint, this means that the attacker will want to send as many packets as it can before turning itself OFF, in order to leave behind the highest possible level of congestion. This calls for using a reasonably high transmission rate R, i.e., larger than the link capacity C. An added benefit of driving congestion up reasonably fast, is that it leaves little time for TCP flows to react and adapt their rate, so that they are likely to encounter severe congestion, which then translates into substantial throughput reductions. In this section, we mainly test a value of $R = 4.5 \times C$.

When it comes to selecting values for b and I, it is useful to consider the two separately. In particular, the criteria for selecting the ON period duration are that it needs to be long enough to create sufficient congestion, but not so long that the attacker contributes losses in more than K scanning lists. In contrast, an OFF period only needs to be long enough to ensure that the attack does not resume while the RED-PD system is still within the same scanning interval. Based on the expression of $T(r,p)$ given in Section 2.2, and considering all possible phases between an ON period and a RED-PD scanning interval, the above conditions require that:

$$b \leq \frac{(K-2)Kr}{M\sqrt{1.5p}} \quad (3)$$

$$b + I \geq \frac{Kr}{\sqrt{1.5p}} \quad (4)$$

Note that Equations (3) and (4) represent a set of *sufficient* conditions for an attacker to avoid detection and thus regulation. It is still possible for an attacker to circumvent the RED-PD defense using a larger b or a smaller I. This can happen in two cases. First, even if the OFF period I is not long enough to prevent multiple ON periods from falling within one scanning interval, the attacker may still avoid detection if its ON period b is small enough. For example, if $b \leq \frac{KT(r,p)}{M}$, the attacker will not be detected as long as it appears at most $\lfloor \frac{K-1}{2} \rfloor$ times in each scanning interval. However, I cannot be too small, since otherwise burst periods will appear in every scanning list, ensuring that the attack traffic is detected. Conversely, if an ON period overlaps with K or more scanning lists in a particular scanning interval, the attacker will be detected but not necessarily affected. This is because the RED-PD system starts filtering only in the next scanning interval. Then, if the OFF period I extends over a sufficiently large number of scanning intervals, the attacking flow will ultimately be removed from the monitored list without having had any of its packets filtered. Nevertheless, note that if an ON period spans multiple scanning intervals, the attacking traffic will be heavily filtered no matter how large the idle period is, since every ON period will have a major fraction of its traffic subjected to pre-filtering. As we will see later, a large I is especially important for evading the RED-PD system, especially in cases where completely avoiding detection is not possible.

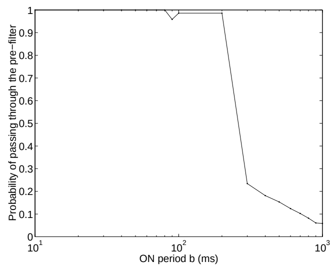
From the perspective of the attacker, a smaller b and a larger I are always "safer." However, they also mean a smaller impact. There exists, therefore, a tradeoff between maximizing impact and avoiding detection. Next, we show that as long as the attacker manages to elude pre-filtering, its impact is not very sensitive to the exact values of b and I.

5.3 RED-PD Defense versus Periodical Blasting Attacker

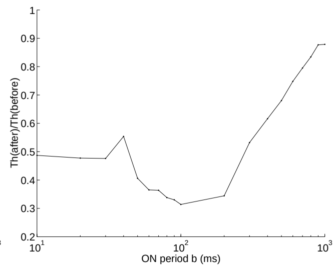
5.3.1 Impact of ON Period Duration

In Figure 6, we illustrate the impact of a "periodical blasting" attacker for different durations of its ON period b. We fix the OFF period duration I at a reasonably large value, i.e., $I = 1600$ msec, while varying b from 10 msec to 1000 msec. Unless stated otherwise, the RED-PD parameters used in this and other experiments were set to their default values, i.e., $r = 40$ msec, $K = 3$ and $M = 5$. Each data point on the figure corresponds to the average value computed over ten independent experiments. Figure 6(a) shows the fraction of the attacking traffic passing the RED-PD pre-filter, as a function of the ON period duration. From the figure, we observe that when the ON period duration is relatively small, i.e., $b \leq 200$ msec, the RED-PD system can only filter out negligible amount of the attack traffic. In contrast, when the ON periods last longer than 300 msec, the RED-PD defense can effectively identify and throttle the attacker. As mentioned before, this is because when the ON period is sufficiently long, the attacker's traffic is filtered for a major portion of its ON period. Figure 6(b) further plots the variations of the total TCP throughput. As expected, when $b \geq 300$ msec, TCP throughput is hardly affected. In contrast, when $b \leq 200$ msec, the RED-PD system failed to provide effective protection to TCP flows. In particular, an ON-period duration of just 10 msec is sufficient to induce a TCP throughput reduction of more than 50%. This is mainly due to the high flooding rate of the attacker, which enables it to dump a very large amount of data in a short amount of time and thus instantly drive the link loss rate to a high level. An off-line inspection of the traces reveals that this sudden increase in loss rate forced a large fraction of the TCP flows to enter a time-out state. Increasing b further introduces only minor differences, as most of the additional attacking packets would be dropped at the target link because of the already high loss rate.

5.3.2 Impact of OFF Period Duration

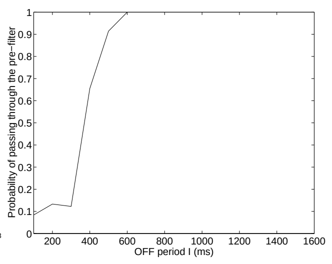


(a) Prob.of passing through

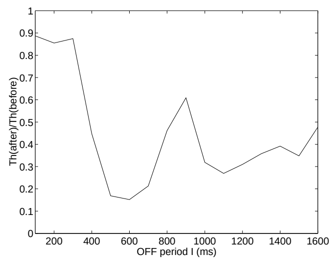


(b) TCP Throughput Change

Figure 6: On the Impact of ON Period Durations



(a) Prob.of passing through



(b) TCP Throughput Change

Figure 7: On the Impact of OFF Period Durations

Figure 7 investigates the effect of changing the duration of the attacker's OFF-period I. Specifically, we chose an ON period of 20 msec, while varying the OFF period I from 100ms to 1600ms. As the duration of every ON period is roughly equal to the minimum possible length of a scanning list[8], an attacker will remain undetected as long as its OFF period is not too short. This can be verified in Figure 7(a), which shows that once the attacker selects a large enough OFF period, i.e., $I \geq 500$ msec, it can consistently elude the RED-PD defense. However, if an aggressive attacker resumes its attack too quickly, i.e., selects an OFF period $I < 400$ msec, it will be detected and rate-limited by the RED-PD system. Therefore, as can be seen from Figure 7(b), it only has minimal impact on TCP performance. In contrast, when $I \geq 500$ msec the throughput of TCP flows decreases dramatically, as *all* attacking traffic manages to pass through the pre-filter and affect TCP traffic. Obviously, increasing I further only extends the length of the attack cycle, which means that TCP flows are affected less frequently. However, this effect is relatively gradual, and even when I is 1600 msec, TCP throughput reduction still exceeds 50%. Note that figure 7(b) also identifies two dips in the TCP throughput curve at $I = 600$ms and $I = 1100$ msec, respectively. This is because at these two values, the attack cycle synchronizes with the 1-second TCP minimum timeout duration, so that many TCP flows are periodically affected and never recover from their time-out state[9].

5.4 Comments and Discussions

In summary, while a RED-PD defense system can readily handle a "bandwidth soaking" attack, it is vulnerable to the so called "periodical blasting" attacks that rely on ON-OFF traffic patterns. In particular, Section 5.3.1 and 5.3.2 demonstrated that even without careful selection of its ON and OFF periods, an attacker can have a substantial impact on TCP throughput. The only requirement is to select a reasonably short ON-period and a sufficiently large OFF-period, and the results are relatively insensitive to the choice of specific values.

The failure of the RED-PD system with ON-OFF patterns is because they are able to exploit inherent weaknesses in *both* the detection *and* the regulation mechanisms it uses. In particular, the use of scanning lists and scanning intervals allows short periods of intense losses to be overlooked. Even when attackers are occasionally detected, no immediate actions are taken. Consequently, all TCP flows significantly reduce their transmission rate. Although they are able to slowly build it back up once the attacker turns silent, it still translates into a substantial drop in overall throughput, as the attack cycle is allowed to repeat periodically. This being said, it should be noted that the RED-PD system still performs better than an aggregate defense system. As we recall from Section 4, an aggregate defense system allowed TCP throughput drops in excess of 80% in the presence of a single "bandwidth soaking" attacker. In contrast, RED-PD manages to shut-off "bandwidth soaking" attackers, and in many cases limits TCP throughput decreases to less than 55% in the presence of a single "periodical blasting" attacker. However, as we shall see in the next section, the situation can rapidly turn bleaker. In particular, TCP flows will almost be shut-off when attackers are capable of enrolling even a modest number of hosts (or identities), and merely relying on other RED-PD configurations will hardly fix the problem.

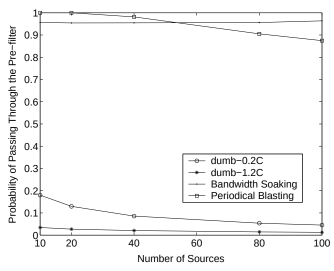
6. PERFORMANCE OF FLOW LEVEL DEFENSES: MULTIPLE ATTACKERS

In this section, we further evaluate the performance of flow level defenses in scenarios where an attacker is capable of compromising multiple host machines to mount distributed denial-of-service (DDoS) attacks. Intuitively, defending against an arbitrarily large number of host machines is beyond the capability of any flow level defense system. However, as mentioned earlier, the growing deployment of protective measures and the increasing scrutiny of network and system administrators can make it more difficult for an attacker to compromise a large number of hosts. Hence, it is reasonable to assume that an attacker will want to launch an attack using the smallest possible number of host machines. Our goal is, therefore, to assess how well flow based defenses can protect legitimate users against increasing numbers of attackers. Ideally, the defense system should limit the impact of attackers to a level that is no more than that of a corresponding number of regular TCP flows, *independent of* the flooding strategy they use. As a result, the throughput of TCP flows should only diminish gradually as the number of attacking hosts grows. Exploring the extent to which this is the case for a RED-PD defense, is our focus in this section.

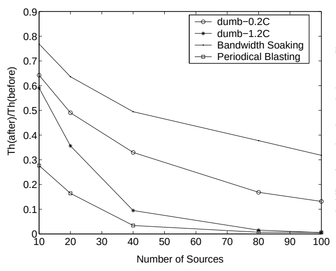
In addition to scenarios involving multiple hosts, we also explore cases where an attacker uses multiple identities (spoofed or legal IP source addresses) in an attempt to disguise itself as multiple distinct users. Since most flow-level defense systems, including the RED-PD system, classify data packets into users/flows based on a combination of source and destination addresses, using multiple source addresses enables an attacker to fool those systems into be-

[8]The minimal scanning list duration $\frac{K}{M}T(r,p)$ is achieved when the link loss rate is 1. Under the default RED-PD configuration, this corresponds to $\frac{3}{5}T(0.04, 1) \approx 20$msec.

[9]Such synchronization phenomenon was first reported and analyzed in [15].

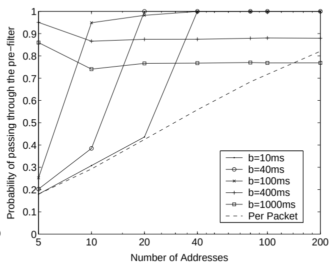


(a) Prob. of passing through

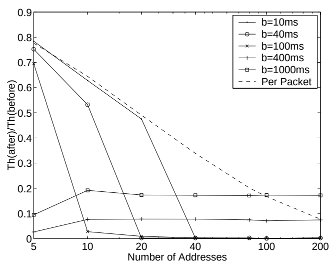


(b) TCP Throughput Change

Figure 8: Multiple Independent Hosts



(a) Prob. of passing through



(b) TCP Throughput Change

Figure 9: Random Spoofing vs. Continuous Cycle

lieving that its traffic belongs to distinct flows. As with multiple hosts, we assume that securing a large number of addresses may be difficult[10], so that an attacker has an incentive to use the smallest number of addresses to achieve a given impact.

6.1 Multiple Independent Hosts

We first consider the case where a RED-PD defense system is attacked by multiple compromised hosts. We assume that each host has been infected with a daemon program that can be commandeered by a master machine controlled by the real attacker. Except for communicating with their master, hosts do not exchange information with each other. We assume that during an attack, all compromised hosts uniformly choose one of the following three flooding strategies:

I **"Dumb"**: A "dumb" flooding strategy mimics today's brute-force attacks. For simplicity, we assume that brute-force attackers are traffic sources that constantly flood at their highest possible rates. In particular, we assume that all attacking sources are constant bit rate (CBR) sources flooding at a rate R. In our experiments, R can be either equal to $0.2C$ or equal to $1.2C$, representing two different data points for the flooding ability of an individual machine.

II **"Bandwidth Soaking"**: This strategy is akin to the "bandwidth soaking" strategy of Section 4. In this case, the attacking daemon in each host runs its own probing and rate adjustment process. The loss rate target p_{thres} is set to the same value, e.g., 4%, in all hosts. To ensure that the aggregate probing rate stays low, each daemon host probes at a fixed rate equal to $\frac{r_{pro}}{N}$, where N is the number of hosts involved in the attack. In our experiments, we set $r_{pro} = 100$Kb/sec.

III **"Periodical Blasting"**: This last strategy is the "periodical blasting" strategy of Section 5, where attackers are assumed to select their ON and OFF periods conservatively. Specifically, we chose a small ON period equal to 40ms, and a large OFF period equal to 1600ms so that consecutive ON periods were sufficiently far apart. We also assume that each attacking host is connected to the network through a high-speed access link over which it can burst at a rate of $1.2C$ when active.

[10]In the case of spoofed addresses, more and more service providers and private network domains are deploying techniques such as ingress filtering [7] to limit their use. Conversely, when valid addresses are used, securing a large enough block can also be difficult and certainly expensive.

We test the above three attacking strategies on the RED-PD system. Figure 8 reports the corresponding results for the default RED-PD parameter setting. First, Figure 8(a) plots the fraction of the attack traffic that manages to pass through the RED-PD pre-filter, as a function of the number of attackers. As shown in the figure, the RED-PD system can essentially shut-off "dumb" attackers. For example, when each host floods at 0.2C, less than 20% of its traffic passes through, and this number drops down to less than 5% when the flooding rate is 1.2C. As mentioned before, the successful regulation of "dumb" attackers by the RED-PD mechanism is because of their high, constant transmission rates, which result in losses in all scanning lists and therefore easy detection. The figure also illustrates that as the number of attackers increases, the amount of attacking traffic passing through the RED-PD system actually goes down. This is because a larger number of attackers triggers a higher link loss rate, which also reduces the traffic that each legitimate TCP user is allowed to send, i.e., $f(r, p)$.

In contrast, the added intelligence of both the "bandwidth soaking" and the "periodical blasting" attackers, allows them to readily evade detection by the RED-PD scheme. Although "bandwidth soaking" attackers are still detected, their adaptive nature ensures that unlike dumb attackers that keep blasting at their maximum rate, they stop increasing their rate once they sense that the loss rate in the RED-PD system reaches the target 4%. This ensures that the amount of pre-filtering received by "bandwidth soaking" attackers never exceeds 5%.

As for "periodical blasting" attackers, their highly transient nature allows them to evade RED-PD regulation most of the time. In particular, when the number of attackers is small (no more than 20), periodically blasting for a short duration of 40ms will not trigger losses in K (K=3) or more scanning lists in a scanning interval, which guarantees that no attacking host is detected. When the number of attacking hosts increases, the length of the RED-PD scanning list ($\frac{K}{M}T(r, p)$) decreases because of the higher link loss rate, which enables the defense system to occasionally identify the attackers. This notwithstanding, the amount of attacking traffic subjected to pre-filtering is still small, i.e., $\leq 12\%$. This is mainly due to the long 1600ms idle period, which prevents attacking hosts from reappearing too quickly after they are detected. As a result, attackers avoid periods of time during which their dropping probability in the pre-filter is high, and only resume transmission when this probability has dropped back down to a low enough level. As we shall see in Section 6.3, even when attackers are *always* detected, a long idle period of 1600ms is still sufficient to allow a substantial fraction of the attacking traffic to evade pre-filtering.

Figure 8(b) demonstrates the RED-PD system's ability to preserve TCP throughput, in the presence of the three different types of attackers we have just investigated. The figure plots TCP throughput as a function of the number of attackers of each type. As can be seen from the figure, the RED-PD system is somewhat effective at protecting TCP users from both "dumb" attackers and "bandwidth soaking" attackers, as the impact of both types of attackers only increases gradually with the number of attacking hosts. For example, with 10 attackers, TCP flows maintain from 60% to 80% of their original throughput, but this number drops down to 10% to 60% when the number of attackers reaches 40[11]. The underlying reasons are, however, different for the two types of attackers. The RED-PD system is efficient against "dumb" attackers because it can readily detect them and, therefore, filter a large fraction of their traffic. For "bandwidth soaking" attackers, the RED-PD system only needs to take advantage of their adaptive nature as they immediately back-off as soon as it applies a slight increase in their drop probability.

Figure 8(b) further confirms that the RED-PD system is quite vulnerable to "periodical blasting" attackers. Specifically, because the RED-PD system filters out only a fraction of the attacking traffic, a total of just 10 attackers manages to reduce the throughput of 32 TCP flows by more than 70%. This is comparable to the effect of injecting 75 additional legitimate TCP flows and is far greater than what can be achieved by a similar number of attackers with any of the other attacking strategies [12]. As the number of attacker increases, the corresponding impact increases further due to the presence of more attacking traffic. As can be shown in the figure, merely 40 "periodical blasting" attackers can almost completely "shut off" the TCP flows.

6.2 Multiple Spoofed Addresses

We now proceed to explore cases where an attacker is capable of assuming *multiple identities* by switching its source address among either a number of spoofed addresses or a block of regular IP addresses it has acquired. As mentioned before, switching source address enables an attacker to fool the RED-PD system into believing that lost packets from the attacker actually belong to different flows. This helps reduce the odds that any of the attacker's identities loses enough packets to be detected by the RED-PD system. Furthermore, even when the RED-PD system successfully identifies one identity, only that identity is affected and, therefore, only a small fraction of the attacking traffic will be dropped. Additionally, a single host assuming multiple identities can create arbitrary traffic patterns for each identity, simply by controlling how it switches among them. For example, the attacker could cycle through its available addresses on a packet-by-packet basis, or after a certain number of packets have been transmitted from a given address. This added dimension can make it even harder for defense systems to spot the multiple identities of an attacker. In this sub-section, we assess the RED-PD system's efficiency against an attacker with multiple identities, as the number of identities available to the attacker varies and for different strategies for switching among them.

We again consider an attacker flooding at a constant rate $R = 1.2C$. The attacker is assumed to have access to N distinct addresses, and uses it according to one of the two strategies below:

I **Random Spoofing**: For each transmitted packet, its source address is randomly selected (with equal probability) from the N available addresses.

II **Continuous Cycle**: In this scenario, the attacker deterministically cycles through the N addresses, and uses each address for a fixed duration b before moving on to the next address. The attack cycle is assumed to repeat continuously [13]. Specifically, as soon as the attacker completes one cycle, which has duration $L_C = Nb$, it immediately starts a new one. Such a strategy essentially generates a number of staggered ON-OFF sources that each have an ON period of duration b and an OFF period of duration $(N-1)b$. Based on our earlier discussions, we expect that as long as the ON period b is short *and* the number of available addresses N is large enough such that the duration of the OFF period is "long" enough, such a strategy will allow the attacker to avoid detection altogether.

We compare the impact of the continuous cycle and random spoofing strategies in Figure 9. The dashed line curve corresponds to the random spoofing strategy, while the solid line curves represent the continuous cycle strategy for different values of b varying from 10 msec to 1000 msec. In Figure 9(a), we first plot as a function of the number of available addresses the percentage of the attack traffic that is unfiltered. The impact of this remaining traffic on TCP throughput is then shown in Figure 9(b).

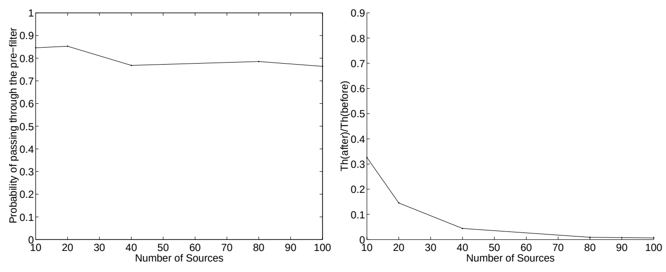
From the figure, we see that when source addresses are selected randomly, the fraction of the attacking traffic surviving pre-filtering grows essentially linearly with the number of available addresses. This is because each address is seen as sending at a roughly constant rate of R/N, out of which the RED-PD scheme only lets through an amount consistent with what it considers to be TCP-friendly. Each time a new address is added, the attacker is able to get the equivalence of one more TCP flow worth of traffic through the RED-PD system. Consequently, its impact on the throughput of TCP flows also grows linearly with N.

In contrast, the RED-PD system is more vulnerable to the ON-OFF behavior of the continuous cycle strategy, as it succeeds in consistently getting more traffic to pass through the RED-PD system, and therefore achieving a greater impact on the throughput

[11] Note that "dumb" attacker are seen as more effective than "bandwidth soaking" attackers because of their much higher transmission rate. Consequently, even though they are subjected to much steeper penalties, e.g., a pre-filtering rate that often exceeds 90%, their residual traffic remains larger than that of "bandwidth soaking" attackers and, therefore, has a bigger impact on the throughput of TCP flows. However, it should be pointed out that because "dumb" attackers are easily identifiable (they lose huge numbers of packets), one could opt to shut them off entirely instead of only filtering a fraction of their traffic, thereby completely eliminating their impact on TCP flows.

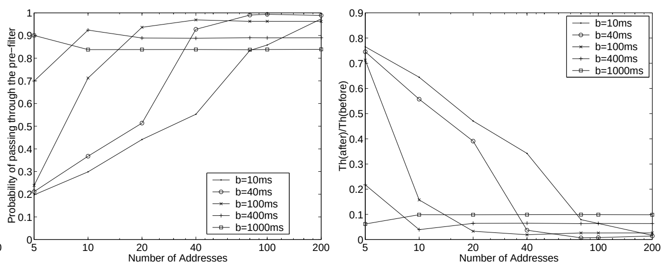
[12] It is worth noting that a higher flooding rate R, translates into an even bigger impact. In particular, recall that in Section 5.3, a single "periodical blasting" attacker was able to decrease TCP throughput by more than 50%, which is not far from the 70% reduction obtained in this experiment. This is mainly because of the higher peak rate used in that earlier experiment, i.e., $R = 4.5C$, instead of $R = 1.2C$ assumed here. This allowed attackers to transmit more packets faster in each of their ON periods. It is expected that increasing the flooding rate to this larger value would allow 10 attackers to trigger an even larger reduction in the throughput of TCP flows.

[13] In [28], we also evaluate a slightly different "padded cycle" strategy, which fills idle periods between two consecutive addresses in order to ensure that every attack cycle is long enough. It can be shown that by guaranteeing a minimum spacing between each attack cycle, the attacker can achieve a bigger impact than the continuous cycle strategy when the number of available addresses is small. However, due to the added idle periods, this strategy also needs a larger address pool to achieve the maximal impact. Hence, overall speaking it performs similarly as its continuous version. For simplicity, we omit reporting the details here.



(a) Prob. of passing through (b) TCP Throughput Change

Figure 10: Periodical Blasting Attackers



(a) Prob. of passing through (b) TCP Throughput Change

Figure 11: Cycling Through Addresses

of TCP flows. The figure actually shows that this can happen in two different scenarios. Specifically, when the attacker selects relatively small values for b, e.g., from 10 to 100 msec, it is typically sure to avoid losing packets in K scanning lists or more during a given ON period. However, when N is small, the corresponding cycle duration Nb will usually be too small, so that the next ON period of a particular address ends up falling within the same scanning interval. That address will then have losses in enough scanning lists and be detected and filtered by the RED-PD defense. When the number of available addresses increases, the corresponding cycle length eventually becomes large enough to ensure that successive ON periods fall in different scanning intervals. This then results in the attacker's traffic avoiding detection altogether. This behavior is illustrated in Figure 9(a) by the sharp increase in traffic passing through for curves associated with small values of b when N grows large, and the corresponding impact on the throughput of TCP flows shown in Figure 9(b). As expected, the smaller the value of b, the larger N is needed to ensure that all the attacker's traffic gets through. A very different scenario arises when the attacker chooses a large value for b. When b is large, each individual address will nearly always contribute losses in K or more scanning lists, and therefore be identified by the RED-PD system. However, because b is large, the cycle duration Nb will also be large even for relatively small values of N. As a result, this usually ensures that by the time the address is reused again, it has already been "forgotten" by the RED-PD system. However, when the loss rate p is high and the duration of a scanning interval is short, a large enough b can often by itself occupy multiple scanning intervals. This therefore results in some fraction of the attacking traffic being always dropped, no matter how many addresses are available. This explains why the curves associated with $b = 400$ msec and $b = 1000$ msec in Figure 9(a) cannot reach a probability of passing through equal to 1 even when N is large.

6.3 Single List RED-PD Confi guration

So far, we have demonstrated that the *default* RED-PD setting is particularly vulnerable to a small number of ON-OFF traffic sources (identities). In the last set of experiments, we investigate whether this weakness can be eliminated via simple modifications to the default RED-PD configuration.

As pointed out earlier, the reason behind the RED-PD system's susceptibility to bursty ON-OFF traffic is in part due to the design of its detection mechanism, which overlooks lossy periods that fall within $(K-1)$ scanning lists. From Equation (3), it is easy to see such situations will occur less frequently when M grows larger. In particular, if M is large enough, each scanning list will become so short that any significant traffic burst will span over more than K lists and thus get detected. However, maintaining a large number of scanning lists raises management overhead and storage requirement. In fact, the RED-PD system can be modified to provide a similar level of detection using only a single scanning list. Specifically, the system can treat an entire scanning interval as a single scanning list and then directly count the actual number of losses associated with each flow in that interval. If a flow contributes more than K lost packets in a scanning interval, it will be marked as malicious. Intuitively, directly counting losses achieves the same effect as using an infinite number of scanning lists. This is because when $M \rightarrow \infty$, each loss typically falls in a separate scanning list, so that the number of losses in an interval is equal to the number of lists containing lost packets. Although using a single list can increase false positives in certain cases [18], it can dramatically improve detection capabilities with only a relatively small additional cost. Therefore, we focus on this approach in the next set of experiments. It should be clear that an approach based on counting losses can always identify high-rate ON-OFF sources, as each ON-period will usually contribute many lost packets. However, it is unclear if this will translate into enough regulation of the attacking traffic to eliminate or even minimize its impact. We test a single list RED-PD system with a configuration of $K = 3$ and $r = 40$ms. We consider the cases of either multiple attackers using the "periodical blasting" strategy, and of one attacker that employs the "continuous cycle" strategy to switch beetween multiple addresses.

Figure 10 shows results for an increasing number of independent "periodical blasting" attacking hosts. The traffic profile of each attacking host is kept identical to that of the earlier "multi-host" scenario, i.e., $R = 1.2C$, $b = 40$ms and $I = 1600$ms. As can be observed from Figure 10(a), the single list design is indeed capable of detecting the attackers and dropping part of their traffic. In comparison to the multi-list scheme of Figure 8, the amount of filtered attacking traffic is seemingly larger, as all attackers are now correctly identified by the RED-PD defense. However, even this increase is not sufficient, as the "proportion" of traffic actually filtered remains small, i.e., no more than 25%. This is again because of the latency between the time when an attacker is first *detected* and when the system starts actually *throttling* its traffic. As pointed out in Section 5.2.2, the RED-PD system only starts penalizing flows classified as non-conformant in the scanning interval following their detection. Moreover, even after the filtering process starts, not every packet of the malicious flow is dropped. Instead, the dropping probability grows gradually, which further limits the responsiveness of the RED-PD regulation scheme. As a result, a highly transient attacking source with a small ON period and large

OFF period is likely to disappear before the drop probability becomes large, an will remain silent until it is almost "forgotten" by the defense system. Because the single list configuration of the RED-PD system does not result in substantially more stringent regulation, the impact of the attackers is essentially the same as in the multi-list scenarios. As reported in Figure 10(b), given that most of their packets are still passing through the pre-filter, 10 attacking sources are again sufficient to significantly reduce TCP throughput. Increasing this number to 40 results in TCP flows losing more than 95% of their original throughput.

The case of an attacker that spoofs its source address based on the "continuous cycle" strategy is considered in Figure 11. We assume that the ON-period for each identity varies from 10ms to 1000msec. From the figure, we see that using the single list configuration helps improve the system's efficiency for scenarios where $b \leq 100$ms. In these cases, the number of distinct addresses the attacker needs to avoid filtering rises by a non-negligible amount. Specifically, for a value of $b = 10$ms, it takes more than 100 addresses to completely avoid detection. When $b = 40$ms, the attacker needs at least 40 addresses to make 90% of its traffic pass through the pre-filter, and this number drops down to 20 for $b = 100$ms. For small values of b, the attack cycle is short so that since the RED-PD now detects all attackers, it still remembers them when the cycle repeats. A larger b increases the attack cycle, and therefore calls for fewer addresses to ensure that previously detected identities have been forgotten by the RED-PD system. When b becomes very large ($b \geq 400$ms), the use of a single list configuration does not yield significant improvements, because long ON periods are detected by both the single list and the multi-list systems. Note that in spite of the improved efficiency of the single list system, it is again still not able to filter enough of the attacking traffic to meaningfully protect TCP flows against it. For example, as is shown in Figure 11(b), as long as $b \geq 40$ms, the attacker needs only 40 addresses to reduce TCP throughput by more than 90%. This is again because of the latency with which the RED-PD regulation mechanism responds once an attacker has been identified.

7. RELATED STUDIES ON UNORTHODOX ATTACKS

There have been several recent studies aimed at exploring the vulnerabilities of existing network infrastructure to emerging DoS attacks. These investigation were mostly carried out from the standpoint of the attacking schemes, namely, to examine the extent to which attack schemes other than brute-force flooding might be able to disrupt various network services. The two most relevant ones are "Shrew" attacks [15] that exploit a specific vulnerability in the design of the TCP protocol, and Reduction of Quality (RoQ) attacks [11] that take advantage of dynamic system behaviors to degrade the performance of TCP users. Our work differs from those studies not only in that it covers a broader range of approaches, but also because its main goal is to expose the vulnerabilities of existing *defense mechanisms*, in order to understand how to build better ones. In particular, we explicitly considered both aggregate level and flow level defense systems, for which we explored different attacking schemes whose design was directly motivated by the weaknesses identified in those systems. In the following, we expand briefly on the similarities and differences between the attack strategies considered in this paper and those of "Shrew" and RoQ.

7.1 "Shrew" Attacks

The "Shrew" attack described in [15] exploits the fact that TCP retransmission time-out settings are globally uniform across many TCP versions. By periodically sending out a burst of packets with a period that matches the minimum TCP time-out duration, "Shrew" attackers can prevent TCP flows from receiving any retransmission acknowledgments, thus causing them to repeatedly time-out. Due to its highly transient behavior, "Shrew" attackers can circumvent the RED-PD defense in typical configurations. When studying the RED-PD system, we also explore the threat posed by properly tuned traffic bursts, i.e., our "Periodical Blasting" scheme. However, our primary focus is on understanding whether and how such ON-OFF traffic patterns can be used to avoid detection by the RED-PD system, and only when those conditions are met are we interested in how they impact TCP performance. This is in contrast to the "Shrew" design, which exclusively focuses on maximizing the impact on TCP. Specifically, the "Periodical Blasting" scheme makes no effort to explicitly synchronize its attack cycle with TCP. Instead, our main purpose is to demonstrate that by properly crafting its ON-OFF behavior, such an attacker can exploit inherent weaknesses of the RED-PD design to avoid regulation, so that it can significantly degrade the performance of TCP flows.

7.2 RoQ Attacks

In [11], the authors investigate a form of attacks that is capable of degrading performance at a target network element, e.g., a network queue, by forcing it to operate in a region with a high degree of transient variations. The RoQ attacks rely on bursty ON-OFF traffic sources with fixed periods, and impact performance by creating intense queue fluctuations at the targeted link. The generic notion of RoQ attacks is essentially similar to our "Periodical Blasting" strategy. We acknowledge the influence that the RoQ attacks had on enhancing our understanding of why such sources could effectively disrupt TCP traffic. However, the study in [11] is primarily concerned with traffic sources that do not violate the TCP-friendly criterion. Although such sources will normally sail through the RED-PD defense, their impact typically amounts to limited service degradations. In contrast, we investigated a much broader range of ON-OFF attacks to better understand when they could foil the RED-PD defense, and showed that more aggressive "Periodical Blasting" attackers can reduce the throughput of TCP flows substantially.

8. CONCLUSIONS AND FUTURE WORK

This paper is aimed at understanding the extent to which existing router-based defense mechanisms can be defeated by attackers employing traffic flooding strategies more complex than today's "brute-force" attacks. This was accomplished by focusing on two representative systems for which we developed attacking strategies that highlighted some of their intrinsic weaknesses. Specifically, we studied both aggregate defense systems that rely on the definition of a malicious "aggregate" to regulate traffic, and flow level defenses that continuously monitor individual flows for conformance with a pre-defined "normal" user behavior. We showed that defending against DoS attacks at the aggregate level cannot eliminate malicious behaviors at the flow level. A single malicious "bandwidth soaking" attacker can gradually nudge out legitimate TCP flows and grab a major fraction of the link capacity without causing anomalies at the level of the entire system. Based on this, we expect that some level of "flow-level information" is needed in order to built effective aggregate schemes, e.g., by combining information about flows that belong to a common aggregate. In particular, flow level defenses such as RED-PD can efficiently contain this type of attacker. However, RED-PD remains vulnerable to highly transient traffic patterns such as those used by the "periodical blasting" attackers. Specifically, as these attackers blast traffic for only a short amount of time and then stay idle for a long period, they are often overlooked by the defense system if it only focuses on the long-term characteristics of a flow. Even when the defense system is able to identify the attacking traffic, this is often not enough because of the latency with which the defense system responds. When combined with the fact that the system has finite memory, and therefore forgets attackers after a certain period of time, this allows even a small number of such attackers to still inflict significant damage on the performance of TCP traffic.

This paper is a first step in an effort towards designing more robust defense mechanisms. The understanding derived from exploring the weaknesses of the ACC-Pushback and the RED-PD defenses is currently being leveraged [27] to devise more effective schemes. As indicated by our findings, building effective defenses calls for both accurate and rapid identification of "abnormal" traffic at the flow level, followed by prompt regulation of that traffic. In [27], we are using RED-PD as our starting point, and improving it in a number of directions. A design similar to the "single list" RED-PD system that counts the number of lost packets, provides a reasonably effective solution for the detection of malicious traffic. However, there is an inherent trade-off in that ensuring the fast detection of attackers also increases the likelihood of false positives. This effect is then compounded depending on the type of regulation being used. Ideally, one would like to apply the most stringent filter, i.e., dropping all packets belonging to flows identified as abnormal, but this will also have the effect of increasing sensitivity to occurrences of false positives. A more progressive filter, as used in RED-PD, minimizes this problem but at the cost of increased exposure to very bursty attackers, as we have shown. A possible approach for accommodating this dilemma is to rely on a combination of mechanisms operating at different time-scales. We are currently prototyping such a double time scale design. At the short time scale, we rely on a stringent regulation mechanism, but with a detection component that is based on a looser definition of traffic abnormality than strict TCP conformance, i.e., by allowing a higher rate than that of a regular TCP flow. At the longer time scale, we rely on a progressive preferential dropping mechanism combined with a detection component that enforces rigorous TCP conformance. We hope to report on the performance and complexity of this design shortly.

9. REFERENCES

[1] Arbor networks. http://www.arbornetworks.com/resources_overview.php.

[2] Mazu profiler. http://www.mazunetworks.com/white_papers/.

[3] The network simulator – ns-2. http://www.isi.edu/nsnam/ns/.

[4] Riverhead networks. http://www.riverhead.com/index2.html.

[5] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proc. IEEE INFOCOM'01*, pages 631–640, Anchorage, AK, April 2001.

[6] S. M. Bellovin. ICMP traceback messages. Internet draft (work in progress), IETF, 2000. http://www1.cs.columbia.edu/ smb/papers/draft-bellovin-itrace-00.txt.

[7] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. Request For Comments (Proposed Standard) RFC 2267, IETF, January 1998.

[8] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. Netw.*, 7(4):458–472, 1999.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[10] T. M. Gil and M. Poletto. MULTOPS: A data-structure for bandwidth attack detection. In *Proc. USENIX Security Symposium*, pages 23–28, Washington, DC, July 2001.

[11] M. Guirguis, A. Bestavros, and I. Matta. Exploiting the transients of adaptation for RoQ attacks on internet resources. In *Proc. IEEE ICNP'04*, pages 184–195, 2004.

[12] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proc. NDSS'02*, February 2002.

[13] H. Jiang and C. Dovrolis. Guardian: A router mechanism for extreme overload prevention. In *Proc. ITCOM'02*, August 2002.

[14] R. R. Kompella, S. Singh, and G. Varghese. On scalable attack detection in the network. In *Proc. IMC' 2004*, pages 187–200, Taormina, Sicily, Italy, October 2004.

[15] A. Kuzmanovic and E. W. Knightly. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proc. ACM SIGCOMM'03*, pages 75–86, 2003.

[16] D. Lin and R. Morris. Dynamics of random early detection. In *Proc. ACM SIGCOMM'97*, pages 127–137, 1997.

[17] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3):62–73, 2002.

[18] R. Mahajan and S. Floyd. Controlling high-bandwidth flows at the congested router. In *Proc. IEEE ICNP'01*, pages 192–201, 2001.

[19] J. Mirkovic, G. Prier, and P. Reiher. Attacking DDoS at the source. In *Proc. IEEE ICNP'02*, Noverber 2002.

[20] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004.

[21] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker. Approximate fairness through differential dropping. *SIGCOMM Comput. Commun. Rev.*, 33(2):23–39, 2003.

[22] R. Pan, B. Prabhakar, and K. Psounis. CHOKe, a stateless active queue management scheme for approximating fair bandwidth allocation. In *Proc. IEEE INFOCOM'00*, pages 942–951, Tel-Aviv, Israel, March 2000.

[23] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proc. ACM SIGCOMM'00*, pages 295–306, 2000.

[24] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks. In *Proc. ACM SIGCOMM'98*, pages 118–130, 1998.

[25] H. Wang, D. Zhang, and K. G. Shin. Detecting SYN flooding attacks. In *Proc. IEEE INFOCOM'2002*, pages 1530–1539, New York, NY, June 2002.

[26] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: Behavior models and applications. In *Proc. ACM SIGCOMM'05*, Philadelphia, PA, August 2005.

[27] Y. Xu and R. Guerin. A double horizon defense scheme for robust regulation of malicious traffic. Technical report in preparation, University of Pennsylvania, 2005.

[28] Y. Xu and R. Guerin. On the robustness of router-based denial-of-service (DoS) defense systems. Technical report, University of Pennsylvania, March 2005. Available at: http://einstein.seas.upenn.edu/mnlab/publications.html.