Effective Network Layer for the Internet Bandwidth Flooding Attacks

[1]R.Akshara, [2]S.Kranthi Reddy, [3]T.Ramkumar

[1, 3] CSE Dept, Vignan Institute of Technology and Aeronautical Engineering
[2]CSE Dept, Vignan Institute of Technology and Science, Hyderabad

Abstract: In a bandwidth-flooding attack, compromised sources send high-volume traffic to the target with the purpose of causing congestion in its tail circuit and disrupting its legitimate communications. In this paper, we present an Active Internet Traffic Filtering (AITF) Protocol, a network-layer defense mechanism against such attacks and aggregate similar kind of packets to fall into one category to find the malicious nodes and using active internet filtering technique we can control the flooding and bandwidth attacks. A network-layer filtering mechanism that enables a receiver to explicitly deny tail-circuit access to misbehaving sources. Our main contribution is Active Internet Traffic Filtering, a protocol that leverages recorded route information to block attack traffic. We conclude that the network-layer of the Internet can provide an effective, scalable, and incrementally deployable solution against bandwidth-flooding attacks.

Key Words: Traffic Filtering, AITF, DDoS, ISP.

1. INTRUDUCTION

The World Wide Web (WWW) has experienced remarkable growth in recent years. Businesses, individuals and governments have found that web applications can offer efficient and reliable solutions to challenges of communicating and conducting commerce in the 21th century. Various corporate bodies whose business model completely focuses on the Web like Google, Yahoo, Amazon etc. have taken web interactions to newer heights. As many enterprise applications dealing with sensitive financial and medical data turn online, the security of such web applications has come under close scrutiny. Compromise of these applications represents a serious threat to organizations that have deployed them, and also to users that trust these systems to store confidential data. The potential downtime and damages that could easily amount to millions of dollars have also prohibited many mission critical applications, which could greatly benefit users, from going online. Hence, it is crucial to protect these applications from targeted attacks.

In a distributed bandwidth-flooding attack, a large number of compromised sources send high-volume traffic to the target in order to create congestion and packet loss in its tail circuit; as a result, the target's communication to legitimate sources deteriorates. It has been shown that such attacks can exploit the behavior of legitimate TCP sources (which back off in the face of packet loss) to dramatically reduce their throughput or, in the case of long-lived flows, drive it to zero [1]. Real-life reports complement such analysis: The first well

documented incident we are aware of is the 2001 attack against the Gibson Research Corporation (GRC) web site. To block the flood, GRC analyzed the undesired traffic, determined its sources, and asked from their Internet service provider (ISP) to manually install filters that blocked traffic from these sources; in the meantime, their site was unreachable for more than 30 hours [2]. More recent attacks are less well documented (the victims are increasingly unwilling to reveal the details), but hint that botnet sizes have increased beyond thousands of sources, while undesired traffic is harder to identify—an article on a 2003 attack against an online betting site reports that the undesired traffic came from more than 20 000 sources, its rate ranged from 1.5 to 3 Gbps, and it was addressed at routers, DNS servers, mail servers, and web sites [3].

Despite the magnitude of the problem and the indications that it is getting worse, no effective solution has been deployed yet. There are two basic steps in stopping a bandwidth-flooding attack: 1) identifying undesired traffic and 2) blocking it; this paper addresses the latter. To prevent undesired traffic from causing legitimate-traffic loss, it must be blocked before entering the target's tail circuit, for example, inside the target's ISP. The first solution that comes to mind is to automate the approach followed by GRC: one can imagine an ISP service, in which a flooding target sends *filtering requests* to its ISP, and, in response, the ISP installs wire-speed filters (i.e., filters that do not affect packet-forwarding performance) in its routers to satisfy these requests; each filtering request specifies traffic from one undesired-traffic source to the target. The problem with this approach is that it requires more resources than ISPs can afford: Wire-speed filters in routers are a scarce resource, and this is not expected to change in the near future. Modern hardware routers forward packets at high rates that allow only few lookups per forwarded packet; to reduce the number of per-packet lookups, router manufacturers store filters—as well as any state that must be looked up per packet, e.g., the router's forwarding table—in TCAM (ternary content addressable memory), which allows for parallel accesses.

2. LITERATURE SURVEY

Flooding attack is an attack that attempts to cause a failure in a computer system or other data processing entity by providing more input than the entity can process properly. Flooding is a Denial of Service (DoS) attack that is designed to bring a network or service down by flooding it with large amounts of traffic. Flood attacks occur when a network or service becomes so weighed down with packets initiating incomplete connection requests that it can no longer process genuine connection requests. By flooding a server or host with connections that cannot be completed, the flood attack eventually fills the hosts memory buffer. Once this buffer is full no further connections can be made, and the result is a Denial of Service.

Objective of the paper is the current intrusion detection and prevention systems seek to detect a wide class of network intrusions (e.g., DoS attacks, worms, port scans) at network vantage points. Unfortunately, even today, many IDS systems we know of keep per-connection or per-flow state to detect malicious TCP flows. Firewall, IPS and IDS devices are not suitable for controlling the flooding and bandwidth attacks.

In the existing system, the Net screen, Frontier, etc are the well known Denial of service attack detection systems. These

systems use hard-coded programs for detecting the each DoS attack. All the programs should be executed to detect the attacks at the same time, consuming more CPU cost. Improving and utilizing the bandwidth is a challenging thing. Because of all these Intrusion Detection Systems (IDS) uses hard coded programs, and are placed in the embedded device also are in the binary format, users cannot modify the logics of the existing programs, once the product has been released. All these IDS systems use a list of Botnet IP Addresses to stop the malicious traffic flows.

In this paper we have applied the features of Active Internet Traffic Filtering Protocol for effective network layer of the Internet band width flooding attacks to improve the detection performance. It is a simple logic, can detect most of the TCP based attacks. In the traditional approach, we can't modify the logics of the detection system. Using AITF we can improve the detection performance than of the traditional approach.

3. ACTIVE INTERNET TRAFFIC FILTERING (AITF)

In this paper, we present Active Internet Traffic Filtering (AITF), a network-layer filtering mechanism that enables a receiver to explicitly deny tail-circuit access to misbehaving sources, while addressing these challenges. We show that:

• AITF enables a receiver to preserve on average more than 80% of its tail circuit in the face of a SYN-flooding attack that exceeds the target's tail-circuit capacity by a factor of 10;
• AITF requires an amount of per-client resources affordable for today's ISPs; the cost of these resources is not expected to increase with time, as long as botnet-size growth does not outpace Moore's law;
• AITF does not require any pre-configured inter-ISP relationships or any public-key infrastructure; it is incrementally deployable, in the sense that even the first two ISPs that deploy it can maintain their connectivity to each other in the face of bandwidth flooding.
We conclude that the network layer of the Internet can provide an effective, scalable, and incrementally deployable solution against bandwidth-flooding attacks.

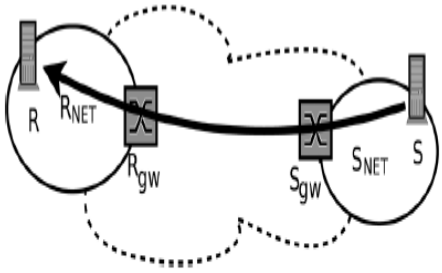


Fig. 1. Source S sends undesired traffic to receiver R through routers S_{gw} (in S's domain) and R_{gw} (in R's domain). S_{NET} and R_{NET} have deployed AITF (and the underlying path-identification mechanism). R identifies $\{S\ S_{gw}\ R_{gw}\ R\}$ as an undesired flow.

4. THE BASIC AITF PROTOCOL

We now describe the basic elements of the Active Internet Traffic Filtering (AITF) protocol. For simplicity, we initially assume all domains that deploy AITF to be honest and well behaved.

A. Players

AITF involves four players per undesired flow, illustrated in Fig. 1:
• The *receiver* R is the target of the undesired flow.
• The *source S* is the node generating the undesired flow.
• The *receiver's gateway* R_{gw} is a border router located in R's ISP, on the path from S to ,R before R 's tail circuit.

Note that R_{gw} is not significantly affected by the attack; if it were (i.e., if its own tail circuit were congested),R_{gw} itself would be the "receiver," while the role of the receiver's gateway would be played by another router upstream. The *source gateway* S_{gw} is a border router located in S's ISP, on the path from S to R . These four players communicate through AITF messages, which include one or more *filtering requests*. Each filtering request includes the specification of an undesired flow F and the amount of time W_f (called the *filtering window*) for which the requester does not want to receive F .

1) The source gateway S_{gw} cooperates with the receiver's Gateway R_{gw} to help the receiver.

2) Filtering requests are not malicious, i.e., they indeed originate from the specified undesired-traffic receiver R and correspond to traffic indeed sent from the specified source S.

3) The receiver can trust the path specified inside each received packet, i.e., it knows the true source S and the true source gateway S_{gw} for each undesired flow.

B. Algorithm Overview

Once a receiver R identifies an undesired flow F , it contacts the corresponding source S and asks it to stop sending F for an amount of time W_f. R's request is propagated through R_{gw} and S_{gw} , which temporarily block F to immediately protect R until S complies. The parameters of the protocol are defined. More specifically, R sends a filtering request to its gateway R_{gw} to block F for W_f . In response R_{gw}, installs a temporary filter that blocks F for time $T_{dr}<<W_f$ and forwards the request to the source gateway S_{gw} ; once S_{gw} satisfies the request, R_{gw} removes its temporary filter. Similarly,S_{gw} installs a temporary filter that blocks F for time $T_{ds}<<W_f$, logs the request for W_f, and forwards the request to S ; once S satisfies the request, S_{gw} removes its temporary filter. If S does not cooperate (i.e., continues to send F),S_{gw} classifies S as *non-cooperating* and blocks all S -originated traffic. If S "pretends" to cooperate (i.e., stops sending F, but resumes
Before W_f has elapsed), the following takes place:R sends a second filtering request against S ; upon receiving this second request,S_{gw} checks its log, detects that S has already been told to stop sending F , classifies S as non-cooperating, and blocks all S-originated traffic.

5. SIMULATION RESULTS

We use real Internet routing table data to build a realistic simulation topology. We simulate different attack scenarios, where multiple attack sources (up to a million) attack a single victim, connected through a 100 Mbps link; the victim's gateway uses up to 10,000 filters to protect the victim. For each scenario, we plot the bandwidth of the attack traffic that reaches the victim as well as the victim's good put as a function of time, i.e., we show how fast attack traffic is blocked and how much of the victim's good put is restored.

Framework

We built our simulator within the Dartmouth Scalable Simulation Framework (DaSSF). To create our topology, we downloaded Internet routing table data from the Route Views project site. We map each AS and each edge network to a separate AITF domain -- we derive AS topology and peering relationships by applying Gao's algorithm for inferring inter-AS relationships to the Route Views data; we derive edge network topology by roughly creating one edge network per advertised class A and class B prefix. Each AITF domain is represented by one AITF router. AITF routers are interconnected through

OC-192 (9.953 Gbps) and OC-48 (2.488 Gbps) full-duplex links. End-hosts are connected to their routers through Fast (100 Mbps) and Thin (10 Mbps) Ethernet full-duplex links. Internet round-trip times average 200 msec. Host-to-router round-trip times average 20 msec. In all scenarios, T_{tmp} = 1 sec and T_{long} = 2 min.

Filtering Response Time

Our first experiment demonstrates that AITF achieves a filtering response time equal to the one-way delay from the victim to its gateway, i.e., on the order of milliseconds. It also demonstrates that "lying" gateways are quickly detected and blocked; the worst each lying gateway can do is cause up to two "spikes" spaced out by at least T_{tmp} seconds.

Scenario 1: The victim receives a flooding attack by 10,000 attack sources, each behind its own attack gateway. The bandwidth of the attack (before defense) is 1 Gbps. All attack gateways are lying, i.e., they agree to block their undesired flows and then break the agreement.

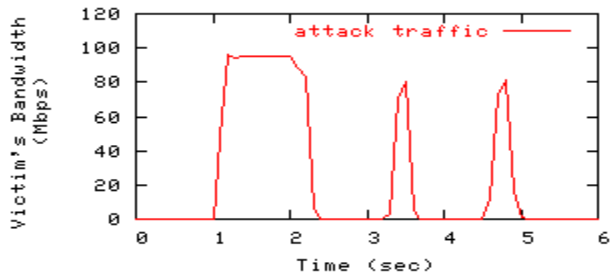


Figure 2: t=1 sec: attack starts; t=2-3 sec: *V* detects the attack and sends 10,000 filtering requests to its gateway; t=3-4 sec: flows recur for the first time; t=4-5 sec: flows recur for the second time, and V_{gw}blocks all traffic from the gateways.

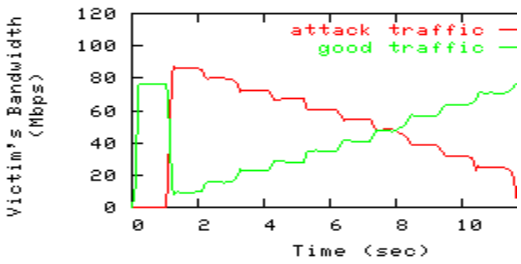
Figure .2 illustrates that V_{gw} blocks attack traffic within milliseconds from the moment the attack is detected; attack traffic recurs twice and is completely blocked within 2 seconds. V_{gw} gives two chances to each attack gateway to honor its filtering agreement; when the attack gateways break their agreements twice, all their traffic to *V*is blocked. We run the experiment only for 10,000 undesired flows (which allows the victim to have all of them blocked within 1 sec), so that the "spike" effect due to the recurring flows is visible.

Filtering Gain

The next two experiments demonstrate that the victim's gateway can achieve filtering gain on the order of hundreds, i.e., the victim's gateway blocks two orders of magnitude more flows than it uses filters.

Scenario 2: The victim receives a flooding attack by 100,000 attack sources. The bandwidth of the attack (before defense) is 1 Gbps. The victim's goodput (before the attack) is approximately 80 Mbps. All attack gateways cooperate.

Scenario 3: Similar to scenario 2, but the victim receives a flooding attack by 1,000,000 attack sources.



Figures 2.1 and 2.2 show that, using 10,000 filters, V_{gw} blocks 100,000 flows in 10 seconds and 1,000,000 flows in 100 seconds. Without AITF, a router needs a million filters to block a million flows; these experiments demonstrate that, with AITF, V_{gw} needs only ten thousand filters to block a million flows. Hence, AITF reduces the number of filters required to block a

certain number of flows by two orders of magnitude -- a critical improvement, since routers typically accommodate tens of thousands of filters, whereas DDoS attacks can easily consist of millions of flows. Figure 7 also reveals what happens after T_{long} = 2 minutes. We assume that attack sources are "smart", i.e., they pause sending an undesired flow when so requested (to avoid disconnection) and they restart after T_{long} = 2 minutes.

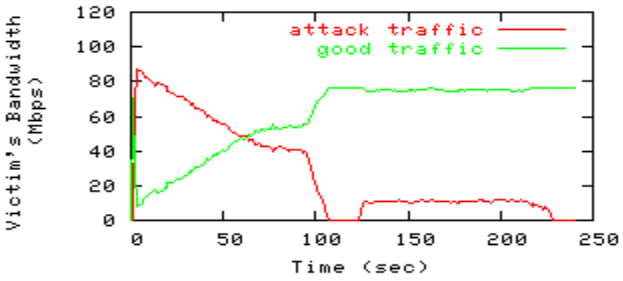


Figure 2.2: t=0-1 sec: *V* is receiving roughly 80 Mbps of goodput; t=1-2 sec: attack drops *V*'s goodput to 12% of original; t=2 sec: *V* starts sending 10,000 filtering requests/sec to its gateway; t=104 sec: *V*'s goodput is restored to 100% of original; t=122 sec: filtering requests start expiring, undesired flows are released and re-blocked, 10,000 at a time.

6. CONCLUSION

We have presented Active Internet Traffic Filtering, a network- layer filtering mechanism that preserves a significant fraction of a receiver's tail circuit in the face of bandwidth flooding, while requiring a reasonable amount of resources from participating ISPs. We have showed that: (1) AITF allows a receiver to preserve on average 80% of its tail circuit in the face of a SYN-flooding attack that has ten times the rate of its capacity. (2) Each participating ISP needs a few thousand filters and a few megabytes of DRAM per client; the per-client cost is not expected to increase, unless botnet-size growth outpaces Moore's law. (3) The first two AITF-enabled networks can maintain their communication in the face of flooding attacks, as long as the path between them is not compromised. The feasibility of AITF shows that the network-layer of the Internet can provide an effective, scalable, and incrementally deployable solution to bandwidth-flooding attacks.

7. REFERENCES

[1]. Katerina Argyraki and David R. Cheriton, "Scalable Network-Layer Defense Against Intern et Bandwidth-Flooding Attacks," IEEE/ACM transactions on networking, vol. 17, no. 4, august 2009.

[2] A. Kuzmanovic and E. K. denial-ofservice attacks (The shrew vs. the mice and elephants)," in *Proc. ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[3] S. Gibson, The Strange Tale of the Denial of Service Against GRC. com. [Online].Available: http://www.grc.com/dos/grcdos.htm

[4] S. Berinato, Online Extortion. [Online]. Available: http://www.csoonline com/read/050105/extortion.html

[5] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," in *Proc. IEEE Int. Symp. Performance Analysis of Systems and Software (ISPASS)*, Austin, TX, Mar. 2006.

[6] R. Beverly and S. Bauer, "The spoofer project: Inferring the extent of source address filtering on the internet," in Proc. USENIX Workshop on Steps Towards Reducing Unwanted Traffic in the Internet (SRUTI), Cambridge, MA, Jul. 2005.

[7] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz4Sale: Surviving organized DDoS attacks that mimic flash crowds," in Proc.

USENIX Conf.Networked Systems Design and Implementation (NSDI), Boston, MA, May 2005.

