Autonomic Response to Distributed Denial of Service Attacks*

Dan Sterne[1], Kelly Djahandari[1], Brett Wilson[1], Bill Babson[1], Dan Schnackenberg[2], Harley Holliday[2], and Travis Reid[2]

[1] NAI Labs, 3060 Washington Road, Glenwood, MD 21738
{Dan_Sterne,Kelly_Djahandari,Brett_Wilson}@nai.com, wbabson@mindspring.com
[2] Boeing Phantom Works, MS 88-12, PO Box 3999, Seattle, WA 98124-2499
{daniel.d.schnackenberg,travis.s.reid,harley.s.holiday}@boeing.com

Abstract. The Cooperative Intrusion Traceback and Response Architecture (CITRA) [1] and the Intruder Detection and Isolation Protocol (IDIP) [2] provide an infrastructure that enables intrusion detection systems, firewalls, routers, and other components to cooperatively trace and block network intrusions as close to their sources as possible. We present the results of recent testbed experiments using CITRA and IDIP to defend streaming multimedia sessions against the Stacheldraht DDoS toolkit. Experimental data suggests that these technologies represent a promising approach for autonomic DDoS defense.

1 Introduction

Distributed Denial of Service (DDoS) attacks are a critical threat to the Internet. Increasingly powerful DDoS toolkits are readily available to potential attackers and essential systems are ill prepared to defend themselves. For example, in January 2001, Microsoft's web sites hosting Hotmail, MSN, Expedia, and other major services were largely inaccessible for 22 hours because of a DDoS attack. The security community has long known that DDoS attacks are possible, but only recently have such attacks become so easy to launch and popular with hackers.

Although technological advances for DDoS defense are beginning to emerge ([3,4]), the current state of practice relies primarily on expert-labor-intensive manual procedures by network administrators. These procedures consist primarily of two activities: 1) "input debugging" [5], in which administrators at a router near the DDoS victim use network traffic probes and statistics to identify the router's physical interfaces through which the DDoS flooding traffic enters their network; and 2) mitigation of network traffic flow through those interfaces by inserting packet filtering or rate limiting rules into the associated router. Once the offending input interfaces are identified, administrators typically contact their

* This research was supported by DARPA/Rome Laboratory Contracts F30602-98-C-0012, F30602-99-C-0181, and F30602-97-C-0309. Distribution Statement "A", Approved for Public Release - Distribution Unlimited.

counterparts at upstream organizations from which offending traffic is being forwarded (e.g., an Internet service provider - ISP). The upstream organizations' administrative personnel carry out input debugging and traffic mitigation procedures on their own routers and contact their upstream counterparts. This process continues upstream as far as possible until either 1) the flood sources are identified and extinguished, or 2) no further upstream cooperation can be obtained.

These procedures have several crucial drawbacks.

- They require immediate availability of highly skilled network administrators, who are in chronic short supply.
- They are time consuming, even for network gurus. It may take hours or longer to diagnose and shutdown a DDoS attack, allowing downtime and associated costs to mount.
- They do not scale. While it is possible to manually trace and mitigate DDoS attacks from a small number of networks on an infrequent basis, these procedures are impractical for attacks involving hundreds of networks or repetitive "whack a mole" attacks in which new flooding sources pop up as current ones are stopped. Increasingly sophisticated attacks like these are likely to become common as hacker toolkits evolve.

The Cooperative Intrusion Traceback and Response Architecture (CITRA) and the Intruder Detection and Isolation Protocol (IDIP) on which it is based provide an infrastructure enabling intrusion detection systems (IDS), firewalls, routers, and other components to cooperatively trace and block network intrusions as close to their sources as possible. By automating the manual attack traceback and mitigation procedures used today, this infrastructure enables networks to respond to intrusions autonomically, and in so doing, addresses the drawbacks cited above.

CITRA and IDIP were initially developed long before the advent of DDoS toolkits [6], but were intended to help protect networks against a broad spectrum of attacks. We have recently adapted CITRA and IDIP for DDoS protection and subjected them to experimental evaluation in a testbed environment. This paper describes the results of tests using these technologies to defend streaming multimedia sessions against the Stacheldraht DDoS toolkit [7].

This paper is organized as follows. Section 2 provides background information about CITRA and IDIP. Section 3 describes our testbed, test scenario, and measurement procedures. Sections 4 and 5 present experimental results and the observations we derived from them. Section 6 discusses related work, and Section 7 provides a summary and conclusion.

2 Background

CITRA is based on IDIP[1] [1,2,6], a protocol for reporting intrusion-related events and coordinating attack traceback and automated response actions. As shown

[1] In [2] and earlier reports, "IDIP" is used to refer to both a protocol and an architecture. However, for terminological clarity, the authors recently introduced the term

in Figure 1, CITRA components are organized at two fundamental levels. First, CITRA communities are administrative domains controlled by a management component called a Discovery Coordinator (DC). Second, CITRA communities consist of neighborhoods connected to each other via boundary controllers, i.e., routers and firewalls. A CITRA neighborhood is the collection of CITRA-enabled devices (neighbors) that are adjacent in the sense that no other CITRA nodes are interposed between them. CITRA utilizes the neighborhood structure to trace and respond to intrusions; the DC, with human oversight, monitors and directs activities throughout the community. We are also developing features to support a third level of organization: cooperation among multiple CITRA communities according to business relationships [1]. Features to support multicommunity operation include policy-based restrictions on 1) exchange of traceback and blocking services, and 2) release and propagation of sensitive information about perceived attack severity, deployed sensors, and internal topology.

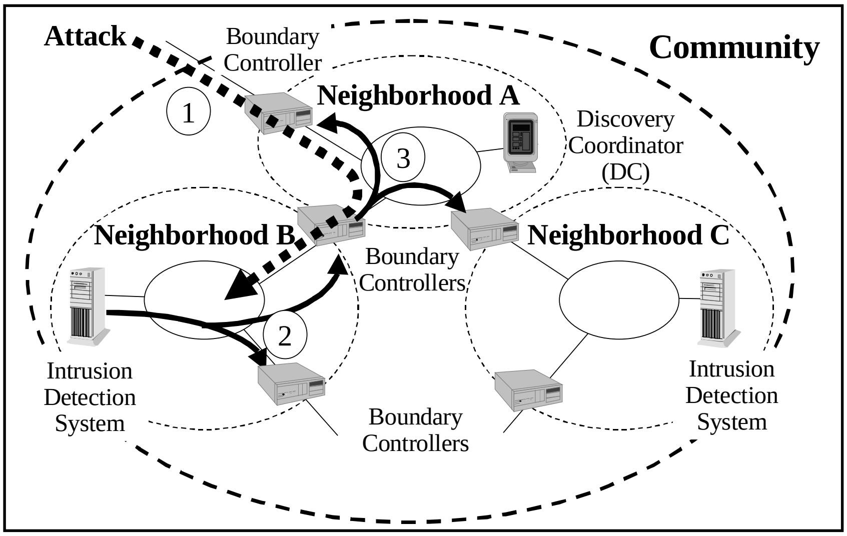

Fig. 1. Attack Traceback and Mitigation across Neighborhoods within a CITRA Community

CITRA-enabled devices collect network audit data used in traceback, illustrated in Figure 1. If a CITRA-enabled detector detects an attack (step 1), the detector sends a traceback request to each CITRA neighbor (step 2). Each boundary controller and host along the potential path of an attack uses its network audit trail to determine if the packets associated with the attack passed through it. If so, the device sends a traceback request to its neighbors (step 3).

"CITRA" [1] to refer to the architecture, while retaining the use of "IDIP" to refer to the protocol. We continue that usage here.

This continues until either the attack is traced back to the source of the attack or to the edge of the CITRA system. This technique is immune to address spoofing because it relies on empirical (audit) data rather than the contents of IP source address fields to determine whether a boundary controller is on the attack path.

At each CITRA component along the attack path, responses are taken in accordance with CITRA policy mechanisms [1]. For example, at a CITRA-enabled firewall or router, the policy may specify that the service (port) being attacked at a host should be blocked for all requests originating from the attacker's address or network for a specified time. For CITRA-enabled hosts, the policy may specify that the offending process should be killed or the offending user's account disabled. The current implementation attempts to use the "narrowest" network response that stops the current attack, minimizing the negative impact the response might have on legitimate system users. A key premise underlying CITRA, IDIP, and corresponding manual procedures used by network administrators is that *moving attack mitigation actions upstream* increases their effectiveness and minimizes the collateral impact on other traffic.

Autonomous responses by individual components along the attack path are temporary, e.g., for two minutes; they stop damage immediately and buy time for the DC to formulate a more reasoned response. As detectors and CITRA devices respond to traceback requests, each device sends a report to the DC describing the responses it has taken. This enables the DC to gain a global view of how the attack and autonomic responses moved through the community. By combining this view with system topology information, in principle, the DC can determine an optimal community response, which it orchestrates by sending directives to each relevant CITRA platform in the community. For example, the DC can remove redundant responses along the attack path to minimize the potential negative impact or increase the duration of a response so that it becomes relatively permanent.

In addition, by collecting intrusion detection and response information at the DC, CITRA supports community-wide and cross-community aggregation and correlation of attacks.

CITRA and IDIP are supported by software libraries that facilitate the integration of existing components. Over the past several years, a variety of IDSs, boundary controllers, host-based responders, and other components have been integrated together, including commercial products and research prototypes [2].

3 Experiment: Autonomic Response to DDoS

Given the emergence of DDoS attacks as a critical risk, the authors sought to 1) investigate the capability of a CITRA-enabled network to defend itself against DDoS attacks and 2) further explore the premise that these technologies are applicable to a wide range of network intrusions.

Although some DDoS traffic can be easily distinguished from legitimate traffic, this is not true in the general case. More sophisticated DDoS toolkits generate traffic that "blends in" with legitimate traffic and therefore cannot be blocked by router packet filters without simultaneously blocking legitimate traffic. For such attacks, traffic rate limiting may be more useful than packet filtering. Rate limiting is an approximate mitigation strategy because it allows some DDoS traffic through and may inadvertently discard some legitimate traffic[2]. Nevertheless, if rate limiting parameters are chosen appropriately, rate limiting can often ensure that enough useful bandwidth is available for legitimate traffic to allow continuation of critical business activities, albeit at reduced speeds.

With this in mind, the authors integrated a simple rate limiter function on CITRA-enabled Linux routers so that it could be used as an alternative intrusion response. The rate limiter utilized token bucket rate limiting services provided by the netfilter subsystem included in a recent release of the Linux kernel (2.4.0-test2).

3.1 Objective

The overall objective of the experiment was to provide evidence to support or refute the premise that CITRA and IDIP, as manifested in recent prototypes, can defend against DDoS attacks. In particular, the authors sought to determine whether CITRA's autonomic activation and upstream propagation of rate limiting could provide sufficient protection during a Stacheldraht v4 [7] attack to allow resumption of a representative, bandwidth-intensive network application. We chose Stacheldraht because it is representative of the DDoS hacker toolkits that emerged in early 2000. Stacheldraht can generate ICMP and UDP floods, TCP Syn floods, and Smurf attacks. It provides one or more master servers that issue commands to multiple distributed agents that serve as flooding sources. Master servers can target floods at arbitrary machines and ports. This allowed us to configure a DDoS attack that is highly disruptive of streaming media sessions.

3.2 Test Application

We chose streaming audio/video as our representative application in the form of the RealNetworks' RealSystem© server and RealPlayer© client[3]. This application is one of the most widely used streaming media applications on the Internet. It provides a number of configuration parameters useful for our experiment including choice of transport protocol (TCP, HTTP, or UDP) and audio/video encoding rates; the latter can be selected according to the bandwidth available between each client and the server. The RealPlayer client also provides real-time read-outs of targeted versus actual bandwidth in the form of line graphs or histograms. We used these to monitor the impact of DDoS attacks.

[2] Reliable delivery mechanisms that support legitimate traffic will detect lost packets and retransmit them.

[3] RealSystem and RealPlayer are registered trademarks of RealNetworks, Inc.

3.3 Experiment Topology and Scenario

The experiment topology is shown in Figure 2. The 100Mbps hub in the center is intended to loosely represent the Internet or the backbone of some other large network. It provides connectivity among five other subnets, each of which represents a different organization's network. Each of these is represented by an ethernet switch or hub with attached nodes and is connected to the "Internet" via a CITRA-enabled Linux router. The network at the bottom center of the figure represents a server farm and network operations center. The RealSystem server is attached to this network. The server is used as the DDoS attack's target. This configuration models highly-publicized recent DDoS attacks in which internet-accessible web servers and their surrounding infrastructure were targeted. Figure 2 also shows the primary flow of traffic from the server to each client. Not shown are control channels that flow in the opposite direction, i.e., from each client back to the server. A CITRA-enabled network IDS also resides on this network. The detection system is a simple flood threshold detector based on the public domain utility *iplog*. Also connected to this network is the server community's DC.

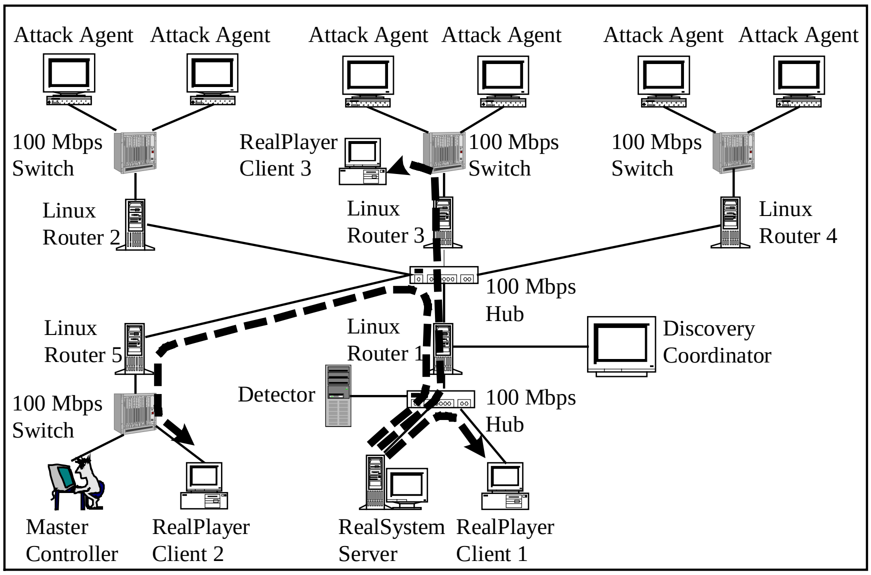

Fig. 2. Topology and Streaming Media Sessions

Three RealPlayer clients have been placed on three different networks so that the effects of the DDoS attack and CITRA's autonomic response at multiple locations throughout the experimental topology can be observed and monitored. RealPlayer Client 1 resides on the same network as the server and is shown adjacent to it. RealPlayer Client 2 resides on the network in the lower left corner of the figure. The RealPlayer Client 3 resides on the network in the upper center of the figure, which, unlike Client 2's network, also includes Stacheldraht flooding agents. Consequently, Client 3's path to the server is identical to one of the flooding paths to the server. As test data, we used an 8-minute, 11-second continuous motion video showing how Boeing 737-300 aircraft are assembled. This audio/video file was encoded at 200.1 Kbps. We configured RealPlayer to use the "best quality" playback setting with available bandwidth set to 10 Mbps, the maximum allowable value, and 5 seconds of data buffering, the minimum effective value. Using RealPlayer readouts, we observed an average data rate of 316.7 Kbps over the 310 second period during which all of the data was transmitted to the client. The data rate for a single client varied during this period from 184.9 Kbps to 668.5 Kbps, with bursts apparently associated with periodic buffer refill requests. We configured RealPlayer/RealMedia to use UDP as the transport protocol.

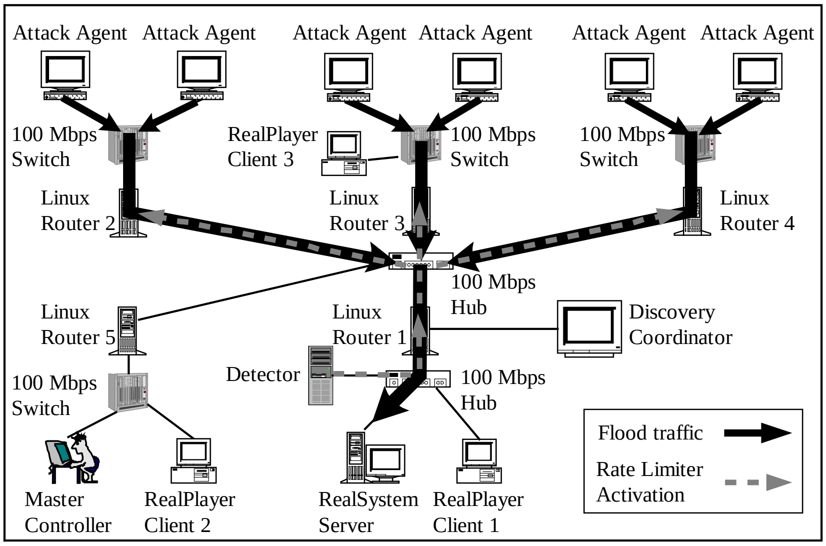

Fig. 3. Stacheldraht Flooding and Autonomic Rate Limiting

As shown in Figure 3, Stacheldraht agents reside on the six workstations on the top row of the figure; one pair of agents resides on each of the three upper networks. The Stacheldraht master controller resides on the network in the lower left of the figure. Since the master does not need to communicate with its agents during the attack, its location during the experiment is not significant. We selected Stacheldraht's UDP flooding attack and chose the RealSystem server as the attack victim. This sends UDP packets to the server having IP headers that are virtually indistinguishable from the control flow packets sent to the server by RealPlayer clients. When a flooding attack occurs, it causes sufficient congestion to prevent these control packets from reaching the server and data packets from reaching the client. This in turn causes the video sessions to freeze after previously buffered data has been exhausted.

Figure 3 also shows the positioning and interaction of the CITRA-enabled components that participate in the autonomic response. When the detector near the RealSystem server detects the flood, it sends a traceback and mitigation request to its CITRA neighbors. Its only neighbor is Linux Router 1, which determines that it is on the flood path. Consequently, it activates a simple rate limiter that is applied to all subsequent UDP packets addressed to the server. For purposes of this experiment, rate limiting parameters were fixed at a maximum average rate of four packets per second with a burst rate of 10 packets per second. Router 1 then propagates the traceback and mitigation request to each of its neighbors, i.e., Linux Routers 2, 3, 4, and 5. Routers 1 through 4 each determine that they are on the attack path and activate rate limiting using the same parameters. Router 5, however, determines that it is not on the attack path and does not activate rate limiting.

3.4 Metrics and Instrumentation

Our objective was to measure the extent to which automated response could enable resumption and continuation of RealPlayer sessions during a DDoS attack. This was measured subjectively by observing the video displayed at each client, and objectively by measuring the number of packets received at each client. To obtain this measurement, we ran *tcpdump* on each client and the server, configured to capture all traffic between the client and server. We processed the tcpdump data with a simple script that counted the number of packets in each 5 second interval for import into a spreadsheet.

Beyond this instrumentation, we used a packet capture tool to monitor the server's LAN segment. This enabled us to monitor the progress of the DDoS attack and response and ensure that the experiment was working according to plan.

We performed five test runs each of normal RealPlayer use, RealPlayer under attack with no response, and RealPlayer under attack with IDIP response. In each case, we collected the tcpdump data for later analysis.

4 Experimental Results and Interpretation

During normal operation (i.e., no flooding present), RealPlayer clients were able to complete video data transmission and display with no visible problems. However, when a Stacheldraht attack was initiated, flooding of the networks and routers was sufficient to immediately prevent RealPlayer clients from communicating with the server, freezing their video images shortly thereafter. With autonomic response enabled, the attack had no perceptible effect on clients, which continued playback without interruption. Traceback and mitigation request messages were able to move upstream against the flood, causing CITRA-enabled routers in the flood path to activate rate limiting, reducing downstream flooding and enabling resumption of RealPlayer sessions.

Figure 4 shows results from four representative experiment runs. The x axis represents time in 5 second intervals; the y axis represents the number of packets received per second at Client 3. Packet rates were nearly identical for all three clients for each run.

The "Normal" run (Figure 4a) represents the number of packets per second captured at Client 3 when no flooding is present. During the initial 30 seconds, RealPlayer clients received about 140 packets per second, to fill playback buffers. This would taper off to about 50 packets per second after the first 30 seconds, with occasional bursts of about 140 packets per second, presumably to refill buffers. This was consistent with the data rates shown on the client statistic displays. For the three clients together, this would result in about 420 packets per second initially, with the network load reduced to about 150 packets per second. Note that although the video was over 8 minutes long, packets arrive at the client for only a little over 5 minutes. This is because the client caches the data to ensure smooth operation during short periods of congestion. We attempted to disable this feature by setting the cache length to 0 seconds, but this showed no noticeable difference from when the value was set to 5 seconds, which was the value used in the experiment.

Figure 4b shows that during the "Flood" run, the packet reception rates for clients drop to zero very quickly after the flood starts (approximately 40 seconds into the run) and stay at zero for the duration of the run. These measurements are consistent with the RealPlayer statistics graphs displayed on the clients. The extent of flooding was confirmed by the packet capture tool on the server's LAN segment, which showed data rates on that segment of about 75 Mbps, i.e., at least 75% loading of the LAN's 100 Mbps capacity. We also observed congestion indications on the "Internet" hub connecting the five routers. The packet collision indicator light remained lit continuously, reflecting frequent unsuccessful transmission attempts. We suspect that the load on the server LAN segment was artificially constrained by throughput limitations of Router 1, a 200 MHz Pentium Pro that connected the server LAN to the rest of the network.

When we ran flood attacks with autonomic responses enabled, we observed two different behavior patterns: (1) full recovery (Figure 4c) and (2) degraded recovery (Figure 4d). The "Full Recovery" histogram shows a dramatic drop in packets received at the client for approximately 10-12 seconds beginning around 40 seconds into the run, followed by a packet reception pattern similar to normal operation. This 10-12 second gap represents the time between the onset of the attack and the autonomic activation of rate limiting throughout the network. For full recovery, the client received the same number of total packets as during non-flood runs and the client displays showed no evidence of an attack, suggesting that cached data was sufficient to cover the short, flood-induced outage.

The "Degraded Recovery" histogram shows a longer transmission reception time with lower packet rate and fewer total packets received at the client. The total number of video packets received at the client decreased dramatically (about

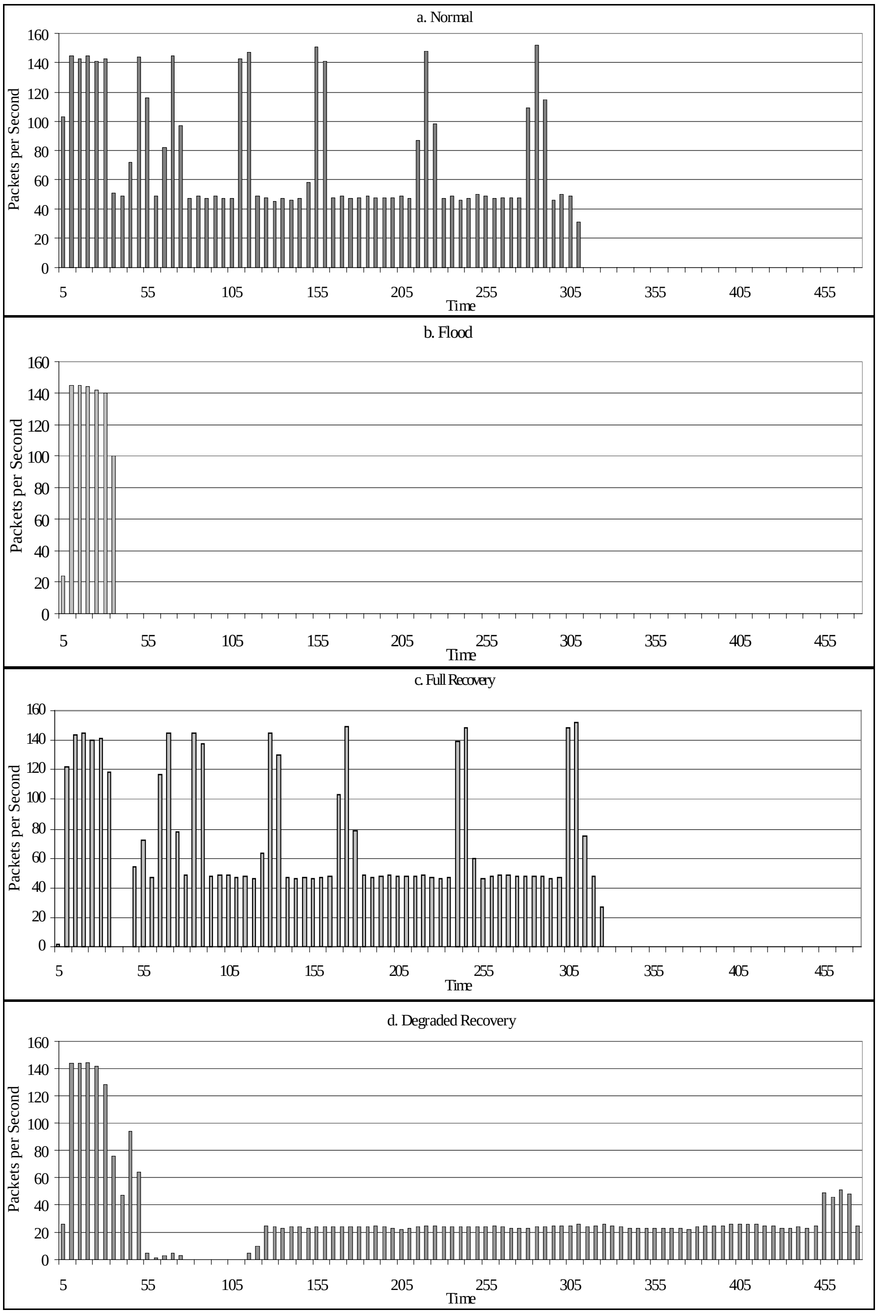

Fig. 4. Rate of Packets Received at Client 3 under Different Test Conditions

1/2) from the full recovery runs, and the duration of transmission spread out over the full video length (i.e., it appeared as if client caching had been disabled). The RealPlayer client video displays for degraded recovery runs were visibly choppier. Table 1 shows the differences in time required for video data transmission for the three different cases where the video ran to completion.

Table 1. Duration of Data Transmission

Mode	Seconds to Complete Video Data Transmission
Normal Operation (No Attack)	310
Autonomic Response with Full Recovery	325
Autonomic Response with Degraded Recovery	475

Examination of the RealSystem server's log showed that the server had detected congestion and adjusted by reducing video quality thereby lowering its demand for bandwidth. Thus, if CITRA's autonomic response did not take effect sufficiently quickly, RealServer would detect congestion and attempt to make its own adjustments.

The speed of the detector (a 366 MHz Pentium II in our case) seemed to be a significant contributor to this mode change. When the detector acted quickly, full recovery was observed. Because the detector monitors the server's LAN segment in promiscuous mode, it may have become overloaded with monitoring when the flood started. We conjectured that a more efficient detection algorithm or a faster detector platform would likely produce more consistent results, with full recovery occurring on every run.

In an attempt to validate these results independently, a research group in another organization reran the experiment in a different testbed [8]. They used the same software configuration, however, the hardware platforms differed in their configuration. In each case, higher performance components were used. The normal and flood runs produced similar results, however, in each run with autonomic response enabled, full recovery was observed, supporting our conjecture that detector speed significantly affects the recovery of RealPlayer clients. With higher speed routers, this testbed was also able to achieve 95% loading of the server's LAN segment. In the rerun, CITRA's autonomic response became effective less than two seconds after the start of the flood, as compared with the 10-12 seconds observed in our slower testbed, a significant improvement.

5 Observations

As indicated by the data above, CITRA-enabled routers in the experimental testbed traced and successfully mitigated the Stacheldraht attack much faster than possible via manual methods. Even though Stacheldraht agents continued to generate flood packets, the video sessions continued unimpeded.

These results, while encouraging, are preliminary. They constitute a single data point in a large, multidimensional space of experimental variables. For example, if more flooding agents were used, creating greater congestion, IDIP traceback and mitigation requests would encounter more collisions, delaying the requests' propagation upstream. It is for this reason that the IDIP uses UDP as a transport protocol rather than TCP. Using UDP (augmented with acknowledgements and retransmission) allows a traceback and mitigation request to be transmitted via a single IP packet. TCP, however, requires the completion of a three-way handshake first. DDoS congestion might cause the handshake to fail repeatedly, adding significant delays to the upstream propagation. IDIP's timeout and retransmit algorithm is derived from the cooperative "back off" algorithm used in TCP. An interesting question for further study is whether IDIP's ability to push traceback requests through DDoS attacks could be further improved by using a more aggressive retransmision algorithm.

Another area for further research concerns the scalability of this technology to larger and higher speed networks. The CITRA-enabled Linux routers in our testbed perform traceback by creating audit records for network flows on an ongoing basis and examining them for attack path evidence when presented with a traceback request. This technique enables routers to trace intrusions that consist of a single IP packet. For edge routers, such flow auditing may be practical, but for core routers, it might require maintaining an unacceptable amount of state or slowing speed-critical packet forwarding operations. (See Sanchez et al [9] for a differing view.) One approach to improving the scalability and speed of flow auditing is to enable it selectively, i.e., only for flows currently under suspicion. While this might result in missing isolated instances of single-packet attacks, it might still be useful for tracing 1) single packet attacks that are repeated on a massive scale via hacker scripts, and 2) DDoS floods which, by their very nature, continue for a period of time. Alternatively, if traceback is sought only for DDoS attacks, auditing individual flows is probably unnecessary. DDoS attacks can be traced to neighboring upstream routers simply by sampling input packets for a few seconds and identifying the input interface from which flood packets entered and the link layer addresses from which they were sent.

An additional topic for further research is the choice of appropriate rate limiting parameters. Ideally, these would be chosen in a way that minimizes negative impact on benign traffic while mitigating a flood sufficiently to permit resumption of normal or near-normal operation. These parameters should be recomputed dynamically as a function of flood and normal traffic rates, topology, effect of upstream mitigation, and other factors. Optimizing the effects over a significant number of routers and network links while ensuring stability in the presence of dynamic local adjustments will probably require the application of control theory.

6 Related Work

The research most closely related to ours is recent work at AT&T Center for Internet Research at ICSI (ACIRI) called Aggregate-Based Congestion Control (ACC) and Pushback [10]. This work grew out of ACIRI's development of congestion control and congestion management techniques for routers. Traditionally, such techniques are meant to stimulate improved behavior of TCP back off mechanisms in conformant (cooperative) end systems. This is accomplished by discarding packets en route or sending signals such as Explicit Congestion Notifications to end points. In [11], ACIRI proposes using similar mechanisms including Random Early Detection with Preferential Dropping to limit the bandwidth utilized by non-cooperating flows such as the high volume aggregate flows associated with DDoS attacks. Recognizing the limitations of local, self-adjusting mechanisms in autonomous routers and the need for explicit coordination across routers, [12] proposes adding an inter-router signaling protocol similar to IDIP. This protocol allows a router near a DDoS victim to request that upstream routers apply rate limiting to specified excessive flows. Like IDIP, these requests propagate upstream as far as possible towards the flooding sources, "pushing back" the flood.

In effect, the research group at ACIRI has arrived at an approach very similar to ours but by an entirely different route. They began with packet discard mechanisms for congestion management and recently added an upstream signaling protocol. We began with an upstream signaling protocol for attack traceback and mitigation and recently added packet discard as a form of attack mitigation specialized for DDoS.

Primary differences between ACIRI's work and ours are as follows:

- ACIRI's signaling protocol includes periodic upstream refresh messages to request continued suppression of DDoS traffic and downstream status reports to determine whether suppression should be terminated. In CITRA, while the initial response to a DDoS attack is undertaken automatically by routers, the initial response is only temporary, e.g., for a few minutes. It is the responsibility of the DC in each administrative domain, potentially with human administrator oversight, to direct routers within that domain to continue the response, optimize it[4], and terminate it when no longer appropriate.
- Although [12] mentions that a server victimized by a DDoS attack could request initiation of pushback, [10], [12], and [11] focus on techniques that would allow congested routers to detect and identify DDoS attacks by analyzing their own packet drop histories. Our approach has been to develop a generalized intrusion response infrastructure into which a variety of IDSs (including ones specialized for DDoS) could be inserted to detect attacks and initiate traceback and mitigation.
- The results published by ACIRI to date are supported by simulations that describe the potential behaviors of pushback under a variety of conditions.

[4] This functionality has not been implemented.

The results presented here are based on empirical data gathered from a specific testbed and conditions that include a real DDoS toolkit (Stacheldraht) and a widely used Internet application (RealPlayer).

Arbor Networks [3] offers a managed service based on a technology that is purported to monitor router traffic for DDoS-specific anomalies, trace attacks through neighboring routers, and recommend packet filtering or rate limiting rules to the router administrator to mitigate attacks. Because Arbor regards its technology as proprietary, they have published few details. It appears however, that the technology is intended for use within individual ISPs and end user enterprises and does not attempt to propagate DDoS defense requests across administrative domain boundaries.

Recourse Technologies' ManHunt [4] is purported to "recognize and respond to DOS attacks in real time by automatically tracking the attack through the chain of ISPs so that the attack can be cut off at the source." If an upstream ISP uses ManHunt, it will be sent a machine readable trace back request which it will propagate further upstream. If the ISP does not use ManHunt, it will be sent a human readable advisory.

A number of techniques have been proposed recently for traceback of DDoS attacks including itrace [13], packet marking [14,15], use of IP Security tunnels [16], and packet flow logging [17]. Unlike the work described here, these proposals do not provide automated DDoS attack mitigation.

7 Summary and Conclusion

DDoS attacks are an increasingly critical threat to the Internet. Yet the current state of practice for DDoS defense relies on the instant availability of expert administrators and time-consuming manual procedures that cannot scale up. CITRA and IDIP were designed as a general infrastructure for traceback and autonomic response to network intrusions. We have recently adapted our existing CITRA prototype for DDoS by integrating a rate limiting function as an additional response option for CITRA-enabled Linux routers. The resulting system has been subjected to informal tests in a laboratory testbed. The results, which have been replicated in a second testbed by another research organization, show that under these conditions, the defense provided by CITRA-enabled components allowed RealPlayer streaming media sessions to continue operation despite a network saturation attack launched from the Stacheldraht toolkit. We are currently developing mechanisms to enforce policy-based restrictions on exchange of traceback and response services across administrative domain boundaries such as those that exist between ISPs. Potential future work includes additional effectiveness testing and analysis and potential improvements in traceback speed and scalability, ability to reliably deliver IDIP traffic over DDoS-congested network links, and ability to compute optimal rate limiting parameters dynamically.

References

1. D. Schnackenberg, H. Holliday, R. Smith, K. Djahandari, and D. Sterne, "Cooperative Intrusion Traceback and Response Architecture (CITRA)," Proceedings of the Second DARPA Information Survivability Conference and Exposition (DISCEX II), Anaheim, CA, June 2001.
2. D. Schnackenberg, K. Djahandari, and D. Sterne, "Infrastructure for Intrusion Detection and Response," Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, SC, January 2000.
3. Arbor Networks - `http://www.arbornetworks.com`.
4. Recourse Technologies ManHunt product description - `http://www.recourse.com/products/manhunt/features.html`.
5. R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," Proceedings of the 9th USENIX Security Symposium, Denver, CO, August 14-17, 2000.
6. "Protocol Definition - Intruder Detection and Isolation Protocol Concept, Dynamic Cooperating Boundary Controllers Interim Technical Report," Boeing Document Number D658-10732-1, Boeing Defense & Space Group, Seattle, WA, January 1997 (`ftp://ftp.tislabs.com/pub/IDIP/DCBC_Interim_Report.pdf`).
7. CERT©Advisory CA-2000-01 Denial-of-Service Developments, `http://www.cert.org/advisories/CA-2000-01.html`.
8. S. Ying, "IA0126 DDoS Automated Response Re-Run," presentation given at DARPA Information Assurance Program Biweekly Meeting, September 29, 2000 (`https://ests.bbn.com/dscgi/ds.py/Get/File-2392/ia0126_Brief.ppt` or `ftp://ftp.tislabs.com/pub/IDIP/Ying_briefing.ppt`).
9. L. Sanchez, W. Milliken, A. Snoeren, F. Tchakountio, C. Jones, S. Kent, C. Partridge, and W. Strayer, "Hardware Support for a Hash-Based IP Traceback," Proceedings of the Second DARPA Information Survivability Conference and Exposition (DISCEX II), Anaheim, CA, June 2001.
10. S. Floyd, S. Bellovin, J. Ioannidis, R. Mahajan, V. Paxson, and S. Shenker, "Aggregate-Based Congestion Control and Pushback," ACIRI Annual Review, December 5, 2000 (`http://www.aciri.org/floyd/talks/ACIRI-Dec00.pdf`).
11. R. Mahajan and S. Floyd, "Controlling High-Bandwidth Flows at the Congested Router," AT&T Center for Internet Research at ICSI (ACIRI), Preliminary Draft, November 20, 2000 (`http://www.aciri.org/floyd/papers/red-pd.pdf`).
12. R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network," AT&T Center for Internet Research at ICSI (ACIRI), DRAFT, February 5, 2001 (`http://www.research.att.com/~smb/papers/DDOS-lacc.pdf`).
13. Steven M. Bellovin, Editor, "ICMP Traceback Messages," Internet Draft: `draft-bellovin-itrace-00.txt`, Mar. 2000.
14. Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson, "Practical Network Support for IP Traceback," Proceedings of the 2000 ACM SIGCOMM Conference, August 2000.
15. Dawn X. Song and Adrian Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," Report No. UCB/CSD-00-1107, Computer Science Division (EECS) University of California, Berkeley, California, June 2000.

16. H. Y. Chang, P. Chen, A. Hayatnagarkar, R. Narayan, P. Sheth, N. Vo, C. L. Wu, S. F. Wu, L. Zhang, X. Zhang, F. Gong, F. Jou, C. Sargor, and X. Wu, "Design and Implementation of A Real-Time Decentralized Source Identification System for Untrusted IP Packets," Proceedings of the DARPA Information Survivability Conference & Exposition, January 2000.
17. Glenn Sager, "Security Fun with OCxmon and cflowd," Presentation at the Internet-2 Measurement Working Group, November 1998 (`http://www.caida.org/projects/ngi/content/security/1198/mt0009.htm`).