# Testing A Collaborative DDoS Defense In A Red Team/Blue Team Exercise

Jelena Mirkovic *Member, IEEE,* Peter Reiher *Member, IEEE,* Christos Papadopoulos *Member, IEEE,*
Alefiya Hussain, Marla Shepard, Michael Berg, Robert Jung

***Abstract*—Testing security systems is challenging because a system's authors have to play the double role of attackers and defenders. Red Team/Blue Team exercises are an invaluable mechanism for security testing. They partition researchers into two competing teams of attackers and defenders, enabling them to create challenging and realistic test scenarios. While such exercises provide valuable insight into vulnerabilities of security systems, they are very expensive and thus rarely performed.**

**In this paper we describe a Red Team/Blue Team exercise, sponsored by DARPA's FTN program, and performed October 2002 — May 2003. The goal of the exercise was to evaluate a collaborative DDoS defense, comprised of a distributed system, COSSACK, and a stand-alone defense, D-WARD. The role of the Blue Team was played by developers of the tested systems from USC/ISI and UCLA, the Red Team included researchers from Sandia National Laboratories, and all the coordination, experiment execution, result collection and analysis was performed by the White Team from BBN Technologies. This exercise was of immense value to all involved — it uncovered significant vulnerabilities in tested systems, pointed out desirable characteristics in DDoS defense systems (e.g., avoiding reliance on timing mechanisms), and taught us many lessons about testing of DDoS defenses.**

**Index terms:** DDoS, collaborative defense, red team testing, denial of service

## I. Introduction

Testing of security systems is very challenging. The goal of the testing process is twofold: (1) it must prove that the system protects against some target threat, in various settings, with satisfactory performance, and (2) it must prove that the system cannot be circumvented by an attacker who is familiar with the design. In most cases, the testing is performed by system designers who alternate between the attacker's role to design test scenarios, and the defender's role to improve the deficiencies revealed by the testing. Besides being psychologically challenging, this testing approach results in simple and incomplete tests, because system designers are naturally biased towards proving that their system works. Therefore, they design scenarios that meet only the first goal of the testing process. Further, defense designers often make tacit assumptions about their system's behavior or working environment. These assumptions can develop into research "blind spots", where potential security flaws can lurk unseen. Because security systems in real deployment are challenged by motivated and skilled attackers, evaluating a system's robustness in realistic, complex scenarios is of paramount importance.

Red Team testing formally separates researchers into teams taking on the attacker (Red Team) and the defender (Blue Team) roles [1][2], which leads to more realistic test scenarios. Blue Team members are defense designers that provide extensive design documentation and live support to the Red Team before and during the testing process. The Red Team members study the security systems using documentation and small-scale experiments and then leverage the information gathered on the system's design to develop tests that aim to circumvent or disable its operation. Naturally, the Blue Team and the Red Team strive for a different testing outcome. Blue Team members hope that the security system will prove resilient to all attacks, while Red Team members hope to find any security holes that have been overlooked by designers. To keep the testing process objective, sometimes a White Team is formed from a neutral third party, and they are responsible for setting the rules of engagement, devising appropriate success measures, collecting statistics and evaluating results. Red Team/Blue Team exercises are invaluable to system designers — they frequently result in discovery of deficiencies that are later fixed, thus leading to more robust systems. However, they are quite costly because they require engagement and close synchronization of many people for a considerable time.

From October 2002 to May 2003 we participated in a Red Team/Blue Team exercise, sponsored by the DARPA's Fault Tolerant Networking (FTN) program. The goal of the exercise was to first combine two related security systems — COSSACK [3], developed by the Information Sciences Institute (USC/ISI) and D-WARD [4], developed by the UCLA — into a single defense and then to evaluate this defense in a variety of scenarios. The Blue Team members were researchers from USC/ISI and UCLA who have developed the COSSACK and the D-WARD systems, the Red Team consisted of researchers from the Sandia National Laboratories, and the White Team consisted of researchers from the BBN Technologies. The overall exercise lasted eight months, and yielded many interesting insights about the tested systems, leading to improvements in their design. In this paper we document our experiences from this exercise and summarize lessons learned.

This paper focuses on the Read Team/Blue Team exercise, not on the tested defenses. Specifically, we do not claim that our defenses are superior to other systems, or that their combination is superior to each defense deployed individually. Instead, we describe how the testing process was designed and customized to the defenses, how the tests were run and what we learned from them about defenses and about testing security systems on a large scale. Since Red Team testing is rare in the academic community we hope our paper will help others who engage in such endeavors to replicate our successes and learn from our mistakes. We further note that tests presented in this paper are now old, but were representative of DDoS trends during the exercise. Further, some

This research has been sponsored by DARPA grants no. N66001-01-1-8939 and N66001- 01-1-8937.

tests were designed specifically to stress-test the defense software, as is the goal during Red Team/Blue Team exercises, and do not reflect real-world attacks.

Both COSSACK and D-WARD are defenses against distributed denial-of-service (DDoS) attacks, developed in the DARPA's FTN program. During a DDoS attack, an attacker launches a coordinated, high-volume traffic from many compromised machines towards a common victim. The traffic interferes with the victim's operation and its legitimate users experience prohibitively slow replies or a complete service disruption. In addition to challenges of security systems' testing discussed above, DDoS defense testing must overcome the challenge of scale. DDoS attacks involve anywhere from a few hundred to half a million attack machines, distributed all over the Internet. To realistically test a DDoS defense we would need an Internet-scale testbed — an impossible goal. Instead, testers must select important characteristics of an attack scenario, such as traffic aggregation, attack dynamics, legitimate traffic mix, and similar, to faithfully reconstruct in a testbed. The selected features are then scaled down carefully, to fit the testbed at the available size.

In Section II we describe the testing methodology used in the exercise. We provide a brief overview of the two defense systems and their integrated version in Section III, sufficient to understand testing decisions and attack strategies. Section IV provides more details about the experimental setup; the performance metrics for the defense evaluation are defined in Section V. We provide results of the executed tests in Section VI and recount lessons learned in Section VII.

## II. Testing Methodology

DARPA regularly encourages systematic testing efforts for the research projects that it currently funds, and it has funded numerous Red Team/Blue Team exercises. To keep the cost of these exercises reasonable, defense tuning, attack development, and defense evaluation sometimes occur sequentially in separate phases. In the first phase, the Blue Team develops and tunes the tested system's code. Once developed, the code is frozen (further changes are forbidden) and handed over, with all supporting documentation, to the Red Team. In the second phase, the Red Team analyzes the system code, designs attacks to stress the system and exploit vulnerabilities and tests them in small-scale scenarios. Sometimes the final tests are automated so that they can be run by the White Team in the third phase. The White Team also organizes and coordinates the entire exercise, which includes testing infrastructure design and implementation, experiment planning, equipment setup, execution of experiments and data analysis. While there is an extensive communication between teams during the entire exercise, the separation of tasks into sequential phases reduces costs because it minimizes the need for simultaneous, live engagement of all three teams.

The COSSACK/D-WARD Red Team/Blue Team exercise involved three teams from four organizations. The Blue Team consisted of COSSACK authors Christos Papadopoulos, Bob Lindell, John Mehringer and Alefiya Hussain from USC Information Sciences Institute, and of D-WARD authors Jelena Mirkovic and Peter Reiher, from the University of California Los Angeles. The Red Team members were Tom Obenauf, Michael Berg, Robert Jung, and Dino Dai Zovi from Sandia National Laboratories. All have participated in numerous Red Team exercises prior to this one and had rich experience in system testing. The White Team members were William Nelson, Ken Theriault, Wilson Farrell, Lyle Sudin, Conor Sibley, Marla Shepard and Pam Helinek from BBN Technologies.

The Red Team/Blue Team exercise employed a single large topology, consisting of nine disjoint edge networks containing a mix of routers, clients, and servers. Some networks deployed the modified COSSACK/D-WARD system. The evaluation consisted of presenting various mixes of benign (a result of normal network operation) and malign (attack) traffic to the instrumented edge networks, and assessing the protection offered by the combined actions of COSSACK/D-WARD systems.

The central metric employed in the exercise was the availability of service to legitimate traffic: if this metric was significantly below the value observed in the absence of attacks, the attacks were considered to have succeeded. Since a complete assessment must also characterize the operational cost of employing COSSACK/D-WARD, statistics such as the rate of false positives and false negatives, the speed of response in detecting and responding to an attack, and any loss in bandwidth of a defended network were also collected.

### A. Rules of engagement

At the experiment planning phase, it was agreed that the Red Team will observe the following rules of engagement: (1) Because some security aspects of the existing COSSACK and D-WARD software, which are not relevant to their ability to detect and respond to DDoS attacks, were not yet implemented (e.g. such as protecting themselves from attack), those capabilities were assumed for the duration of the experiment. Attacks intended to directly disable the defensive technology (e.g., crash it by presenting it with malformed network traffic, kill processes on a machine, etc.) were off limits. Attacks intended to defeat the defensive technology through means other than its normal operating environment and conditions (e.g., by modifying its configuration files that contain detection parameters) were also off limits. This means that the main goal of Red Team attacks was to evade detection or response by COSSACK/D-WARD while still creating a significant denial of service on legitimate traffic, rather than to directly attack the defense system. (2) Attacks targeting routers were off limits as they were not protected by the defense. (3) Since D-WARD functionality encompasses ingress filtering [5], random spoofing from COSSACK/D-WARD deploying networks could easily be detected. For this reason, Red Team was allowed to perform only subnet spoofing from defended networks. Any spoofing from non-defended networks was permitted. (4) Red Team was not allowed to attack unprotected subnets (not containing a defensive technology) or to launch intra-subnet attacks; both of these attack categories cannot be eliminated by the tested defense because they do not traverse it. (5) Application-level attacks (aka "flash crowds"), where an attack consists of fully legitimate service requests, were forbidden as defenses were not designed to protect against them. To summarize: all attacks employed in this experiment were required to be of a

DDoS nature, be launched over network links observed by either COSSACK or D-WARD, and be subject to D-WARD responses.

There were no limits on the reconnaissance and attack tools that could be employed and the Red Team had full knowledge of the entire system. The Blue Team was not permitted to manually change the underlying software, configuration, or defensive strategy once experimentation had started, to avoid the problem of any changes nullifying previous results.

## III. Defense Systems

In this section we provide high-level details about the COSSACK and the D-WARD system, and their integration. Readers interested in learning more about COSSACK and D-WARD should refer to publications [3] and [4].

### A. COSSACK

COSSACK is a distributed DDoS detection and response system. COSSACK components, called *watchdogs* are located at the edge networks. Each watchdog monitors its own network and it shares information and coordinates actions with other watchdogs via a multicast framework *yoid* [6], to collectively combat DDoS attacks. Localized information can be gleaned using a variety of collection tools, such as SNMP statistics, Cisco NetFlow [7], and IDS tools such as Snort [8]. The tested implementation of COSSACK uses Snort with a custom-developed plug-in, which can be controlled by the watchdog dynamically during attacks.

When a DoS attack is launched against a COSSACK-protected network the following operations take place: (1) The watchdog at the victim edge network detects the attack at the ingress point; (2) The watchdog instructs the IDS (Snort) to compile source address information and attack signature data (rates, type of attack, etc.) into a message; (3) The watchdog multicasts this message to other watchdogs in the network. It also advertises an attack-specific multicast group; (4) Watchdogs representing the implicated source networks join the attack-specific multicast group and perform in-depth analysis of their outgoing flows to determine if attack machines exist within their infrastructure; (5) Source networks that identify attack machines deploy countermeasures to stop attack flows. Local responses will be dictated by the local policy and the information received from the victim network's watchdog.

COSSACK watchdogs rely on an existing intrusion detection system to detect attacks. In the tested system, a Snort [8] plug-in has been implemented for attack detection. Packets captured by Snort are guided through a series of processing steps, one of which is filtering against a rule database containing known attack patterns that are matched against the header and payload information. In the tested system this rule base was static, and was a reason behind COSSACK's failure to detect some simple attacks, such as TCP data flood with spoofed packets (see Section VI-B), whose traffic did not match the rules.

### B. D-WARD

D-WARD is a source-end defense, installed at a gateway router between the deploying network (source network) and the rest of the Internet. D-WARD's goal is to detect outgoing attacks and to rate limit attack streams. At the same time, legitimate traffic must proceed undisturbed even when its destination is the alleged victim of the attack.

D-WARD monitors all packets passing through the gateway router and gathers statistics on two-way communication between the source network and the rest of the Internet. These statistics are mainly counts of packets and bytes going in each direction, in four traffic categories: TCP, UDP, ICMP and other. Statistics are collected at the granularity of an *aggregate flow* and a *connection*, where the aggregate flow represents all traffic exchanged between a source network and a foreign host, and the connection represents all traffic between two hosts and port numbers, where one host is within a source network and the other is a foreign host. Periodically, statistics are compared to models of normal traffic that mainly test if the amount and context of the reverse traffic match the amount and context of the outgoing traffic.

D-WARD classifies TCP and ICMP flows or connections as part of an attack if the ratio of packets sent to packets received, smoothed over a suitable period, is higher than a specified threshold. Otherwise, such flows and connections are classified as "good." For UDP flows, several criteria are used to determine if the flow is an attack. These include the number of connections to a given destination, the number of packets per connection, and the maximum sending rate. The values of these are compared to thresholds to determine if the flow is an attack. At the time the testing was performed, D-WARD only classified UDP flows, not UDP connections. If attack were detected, all UDP connections were rate limited.

The rate limit is set for flows that do not match normal traffic models, and it is based on their past behavior and aggressiveness. D-WARD forwards outgoing packets to their destination if one of the following three conditions hold: (1) They do not belong to a rate-limited flow; (2) They belong to a rate-limited flow but are part of a good connection; (3) They belong to a rate-limited flow and there is sufficient available bandwidth.

### C. COSSACK/D-WARD Combined

In the combined operation, COSSACK and D-WARD both perform detection of DDoS attacks, but only D-WARD deploys responses to those attacks. Each edge network in the exercise runs both a COSSACK watchdog and a D-WARD system. When the local watchdog near the victim detects an attack it sends alerts to remote watchdogs, using COSSACK's multicast mechanisms.

The COSSACK message carries information about the type of the attack, the IP of the machine under attack, IP ranges of networks that COSSACK believes are sources of the attack, a suggested rate limit for D-WARD to impose, and a flag showing whether this is a message concerning a new, ongoing or aborted attack. If a local watchdog determines that the network it covers is on the list of suspect source networks, it will forward the message to its collocated D-WARD system.

The D-WARD's response to a COSSACK's message depends on the type of the message and any existing knowledge of the attack specified in the message. In case of newly detected UDP attacks, D-WARD regards COSSACK's detection signal as authoritative, and will impose the requested rate limit on the attack. This rate limit will be regulated only through COSSACK's messages, i.e. it will be removed instantly when COSSACK

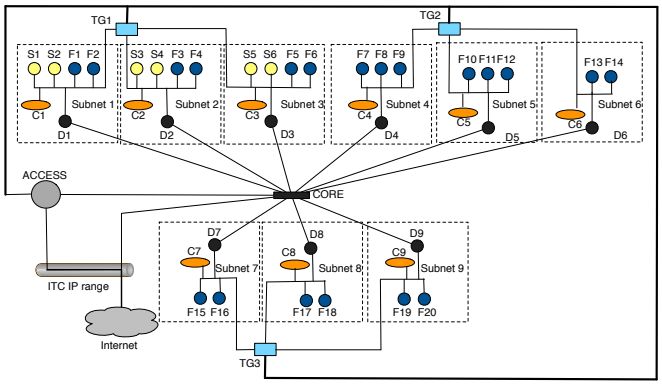



Fig. 1. Topology used in the exercise

detects the end of the UDP attack. In the case of a newly detected TCP or ICMP attack, D-WARD accepts the attack signal but invokes its own rate-limiting algorithm to calculate the appropriate rate limits. Similarly, messages about the attack's end are regarded as a signal that the rate limits should be slowly lifted. This difference in handling TCP/ICMP and UDP attacks was engineered because our preliminary tests indicated that D-WARD's detection of TCP/ICMP attacks was more accurate than COSSACK's, while COSSACK was more accurate in detecting UDP attacks. Our experimental results have shown, however, that D-WARD was overall more accurate in detecting attacks and did not benefit from being integrated with COSSACK.

## IV. Experiment Setup and Scenarios

### A. Topology

The experimental topology consisted of nine edge networks connected by a core network, which was simulated by a multi-ported Cisco router. The topology is shown in Figure 1 and hardware specifics of nodes in the topology are given in Table I. The system had three types of components: (1) clients (F 1–20) and servers (S 1–6) residing on different edge networks, (2) edge network border routers deploying COSSACK and D-WARD (C 1–9 and D 1–9), and (3) a core network consisting of a router only. Each edge network's clients and servers exchanged legitimate traffic, consisting of TCP, UDP, and ICMP packets, over the core network, with the clients and servers on the other edge networks. During attacks, some clients were subverted and played the role of attackers.

| Functionality | No. | Name | Brand | Model | CPU | Memory | HD |
|---|---|---|---|---|---|---|---|
| Core router | 1 | CORE | CISCO | 3640 | | | |
| DWARD | 9 | D 1–9 | DELL | GX240 | Pentium IV @ 1.6GHz | 512 MB | 40GB |
| COSSACK | 9 | C 1–9 | DELL | GX240 | 6 Pentium IV @ 1.6 GHz | 512 MB | 40GB |
| | | | | GX1p | 3 Pentium III @ 600 MHz | 256 MB | 20GB |
| Control host | 1 | ACCESS | DELL | GX1p | Pentium III @ 600 MHz | 256 MB | 20GB |
| End hosts | 20 | F 1–20 | DELL | GX1p | 10 Pentium III @ 600 MHz | 256 MB | 20GB |
| | | | | GX1 | 10 Pentium II @ 450 MHz | 256 MB | 20GB |
| Servers | 6 | S 1–6 | DELL | GX1p | 4 Pentium III @ 600 MHz | 256 MB | 20GB |
| | | | | GX1 | 2 Pentium II 450 MHz | 256 MB | 20GB |
| Traffic gen. | 3 | TG 1–3 | DELL | GX1p | Pentium III @ 600 MHz | 256 MB | 20GB |

TABLE I
Configurations of machines involved in the exercise

This topology consisted of more real machines than was typical in DDoS testing at the time, but still far fewer machines than are involved in real-world DDoS attacks. It is worth considering if experiment results would be any different if a larger, more realistic testbed were available. In a larger network, an attacker (or in our case the Red Team) could send at a higher rate, distributing attack so that smaller numbers of packets sent from larger numbers of sites. Rolling floods could also cycle through larger numbers of nodes. COSSACK's attack detection was rule-based and would likely be as effective in a large network as in a smaller one. COSSACK's multicast mechanism would be challenged if it supported large-scale communication. In D-WARD experiments described in [4] it was shown that the defense successfully detects and controls low-rate attacks if they inflict DoS effect at the victim. Rolling floods, however, would be successful because they exploit timing in D-WARD's design. We are not aware of any existing DDoS defenses that can handle large-scale, distributed rolling floods. Administrative challenges of running this exercise with our current topology were significant. Even if a much larger testbed were available, it would likely be hard to organize a Red Team/Blue Team exercise for such scale.

### B. Legitimate Traffic

The Skaion traffic generation system [9] was utilized in this exercise to provide legitimate traffic. The traffic was generated using semi-autonomous software agents or *bots*. Bots are deployed on a set of traffic generation machines, and they can interact with each other or with live hosts during experimentation. Bots are driven by a broad stochastic representation of a desired traffic mix they work to recreate, and use real applications for traffic generation, such as Telnet, SSH, and Web applications.

A single traffic generation machine can run multiple bots, each acting as a virtual host and having a dedicated IP address. Multiple traffic generation machines can be orchestrated to generate traffic from thousands of virtual hosts, by running a slave module on each machine and coordinating all slave activities via a single master module run on a separate control machine.

In our exercise, there were three traffic generators (TG 1–3 in Figure 1), each running multiple bots. These hosts were connected together via the out-of-band network (denoted with bold lines in the Figure 1) that was off-limits to both legitimate message traffic and attack by the Red Team. All network links had 10 Mbps bandwidth, with the exception of the traffic generation network, which was configured to run at 100 Mbps speed.

### C. Attacks

The task of the Red Team was to design attacks that measurably *degrade* or *deny* availability of network resources on one or more edge-networks. *Degradation* implies decrease in the throughput of traffic to and from an edge-network, relative to the throughput observed in the absence of an attack. *Denial* implies that the services of an edge-network are no longer available despite the actions of COSSACK/D-WARD. Red Team focused on attacks that stress or bypass defenses, thus tests did not include partial deployment scenarios or scenarios that test scalability of defenses. Specifically for partial deployment scenarios, both COSSACK and D-WARD deploy a response close to the sources of attack. If some attackers reside in networks that do not run D-WARD or

COSSACK, their traffic will reach the victim unhindered and, if its volume suffices, it will create denial of service. Since all links in our experiment were 10 Mbps they could be flooded by a single attack machine generating large packets (1,200 byte packets map into $\approx$ 1,042 packets per second that can easily be generated by a single machine). In the real Internet, larger link capacity in networks hosting servers (frequenly $>$ 1 Gbps), and a smaller link capacity of bots (according to publication [10] an average bot can generate at most 100 Kbps) map into the requirement of at least 10,000 bots that need to be located in legacy networks for a successful flooding attack. Additionally, a host under TCP SYN flood attack that does not deploy syncookies can be brought down by roughly 100 packets per second, which can be generated by a single attack machine.

Special care was taken to ensure that attack code cannot "escape" the experimental environment, either accidentally or by real attackers compromising experimental hosts and stealing the code to reuse it later in real attacks. All the executable code for attacks was uploaded to the experimental network via SSH over VPN. The attack code had no inherent propagation capabilities of its own, and thus could not self-propagate if it were accidentally released. Further, all attack code that generates IP packets contained algorithms to filter all outgoing packets with addresses other than those used in the exercise. All hosts in the exercise were assigned globally non-routable addresses.

Attack Strategy Development: Chronologically, the attack strategy was developed in the following manner: (1) Attacks were postulated and categorized into general classes (2) Attacks were either discarded because the teams agreed that they were off limits or chosen for full development and testing on standalone systems and lightweight networks. (3) Reconnaissance was performed on a live network. This reconnaissance did not use the final network that was utilized for the experiment, but instead a smaller network located at the Red Team's institution. The consequences of different implementation platforms later created much difficulty because some developed attack strategies were not as effective in the real experiment as they were in the reconnaissance phase. (4) Attacks were tested and refined on the reconnaissance network. (5) Attacks were turned over to the White Team for execution, data collection and analysis.

Classes of Postulated Attacks: (1) *Timing attacks*, which capitalized on temporal interdependencies between software components, network timing characteristics, etc. Since the nature of the defensive technology — and the underlying DDoS problem itself —– is highly temporal in nature, the majority of attack strategies had either a basis in, or at least an important component of, timing strategy. (2) *Communicating attack threads (Smart Mobs)* — the Red Team conjectured that attacks, which implemented an intelligent communication strategy to coordinate their activities, would be more successful than brute-force attacks. (3) *Synergies achieved through simultaneous attacks* — attacks were hypothesized in which the combined effects, and hence likelihood of success, of multiple attacks would be greater than the effects of any of the individual attacks. Since the defensive technologies were assumed to detect and respond to brute-force attacks, the synergies sought resulted in subtler, stealthier attack behavior.

Discarded Attacks: A variety of existing attacks and "out-of-the-box" tools were considered. Ultimately, the Red Team developed its customized attack tools that combined useful attack strategies from out-of-the-box tools but were more easily manipulated and customized during exercise.

Attack Packet "Labeling": To enable analysis of the network traffic captured during experiment runs, the Red Team agreed to "label" packets that its tools were generating. This packet labeling allowed categorization of packets as either normal or attack traffic by the White Team instrumentation that collected experiment's statistics. The marks were used for statistics collection purposes only. Both COSSACK and D-WARD ignored the packet labels.

Several techniques were explored to achieve packet labeling. Eventually a technique was selected by which the IP timestamp option bit was added to the attack packet IP headers and all attacks were developed using this labeling. Later, when attack scripts were ported from the Red Team's network to the experimental network which used a Cisco router, it was discovered that the Cisco 3600-family routers exercised different processing paths for normal packets (which did not contain the IP timestamp option) than for attack packets. These different processing paths affected both the priority of packets passing through the routers and the speed with which packets were processed. To address this problem the Red Team attack tools were eventually modified to use the Type of Service (TOS) field in the IP headers for labeling. This technique allowed positive identification of attack traffic packets.

### D. Experiment Control

Experiment control and data collection was coordinated from the experiment control host (labeled *Access* in Figure 1). This host executed the BBN experimentation framework.

## V. Performance Metrics

This experiment employed two principal performance metrics: (1) Fraction of legitimate traffic throughput maintained, measured as percentages of different kinds of legitimate traffic maintained at the pre-attack levels by the defensive system. (2) Whether the DDoS attack was detected and responded to. Several additional sub-metrics were useful for characterizing various aspects of the defenses: (1) Rate of false positive and false negative detections. (2) Time to detection, measured as the amount of time from the attack launch until the system decides it is indeed an attack. (3) Time to respond, measured as the amount of time from attack detection until the legitimate traffic returns to a baseline level. (4) Severity of the attack — the damage inflicted on legitimate traffic in an undefended system.

These performance metrics were chosen at the time and agreed upon by all team members. Several years later, team members Reiher, Hussain and Mirkovic worked with collaborators from SPARTA and Purdue University to develop novel, more accurate metrics for measuring denial of service in experiments. That work is described in publication [11], but since detailed packet traces from our Red Team/Blue Team exercise were not preserved we cannot apply new metrics to experiments described in this paper.

Further, the experiment methodology did not permit us to analyze higher level effects of attacks. Thus, we do not know

and cannot determine if defense actions tended to cause some connections be dropped completely, or all connections to receive degraded service. Based on our understanding of D-WARD's functionality we believe the latter was the case.

### A. Data Validation

Data validation was performed by repeatedly executing runs to ensure that the results obtained were consistent. Each experiment was run 5 times. The first 4 iterations used 4 different seed values for random number generation needed for traffic generators, and the fifth iteration repeated the initial seed value. The different seed values provided for statistical variation of the data. Unfortunately only results from single test runs were kept after the exercise and used for the final report and for this paper. Thus our graphs reflect results from single runs, and lack confidence intervals.

## VI. Results

In this section we show selected experimental results from all tested attacks and discuss defense performance in detail. All attacks generate packets with random contents except for HTTP flood attacks that generate valid HTTP requests.

### A. Baseline — No Attack

The baseline runs have no attacks present and consist of only legitimate traffic. There were three different legitimate traffic levels that were used to test false positives introduced by the defense systems — low, high and mixed. The high traffic pattern used approximately 10% of the available bandwidth; the low traffic pattern used approximately 2% of the available bandwidth; and the mixed traffic pattern started at the low level and after 10 minutes increased to the high level to simulate a sudden change in traffic behavior. Each baseline was run for 20 minutes.

Each traffic mix was run without any defense, with D-WARD only, COSSACK only and with integrated COSSACK/D-WARD system. The behavior of the system with only COSSACK running is equivalent to the behavior of the system with neither defensive system running as only D-WARD can implement responses.

Table III summarizes the average baseline behavior in each configuration. In each case 99.8 to 100 percent of the traffic sent was received. In the low traffic pattern runs, there was essentially no difference in the average number of packets received with or without defenses. Under the mixed and the high traffic pattern, slightly lower average number of packets were received with defenses, indicating packet loss due to false detections.

| Traffic pattern | Defense | % Received |
|---|---|---|
| High | Cossack | 99.9 |
| | D-WARD | 99.8 |
| | Both | 99.9 |
| Low | COSSACK | 100.0 |
| | D-WARD | 100.0 |
| | Both | 100.0 |
| Mixed | COSSACK | 99.8 |
| | D-WARD | 99.9 |
| | Both | 99.9 |

TABLE III
AVERAGE BASELINE THROUGHPUT

Figures 2 (a)—(c) show packet counts received at subnet 1 over the duration of the experimental run, in 30 second intervals. The time is shown on the $x$-axis, relative to the start of the run. The majority of the traffic is TCP, and the number of UDP packets is generally fairly consistent for the duration of the run. The low traffic pattern sends about 15%, and the mixed traffic pattern sends about 45% of the traffic sent with the high traffic pattern.

Table IV summarizes the average number of false positive detections, actions and classifications from the attack-free baseline runs when both defensive configurations were running. COSSACK did not have any false positive detections during the baseline runs, while D-WARD had some false positives. Therefore runs with only COSSACK deployed are false-positive free while runs when only D-WARD is deployed have identical results with runs when both systems are active. The "detections" field in the table shows the number of false attack detections per run, and the "actions" field shows the duration of rate-limiting, calculated from the moment when a rate limit is imposed on the flow to the moment it is removed. Since the rate limit is changing dynamically, a flow may not experience drops even if it is rate-limited. The fields "TCP/UDP misclassified" show the average percentage of TCP and UDP connections that were not classified as "good" and therefore could experience packet loss due to defensive actions if any actions are triggered. In case when no false positive detections occur no packets will be dropped from these connections.

| Traffic | Detections | Action | TCP misclassified | UDP misclassified |
|---|---|---|---|---|
| High | 4 | 36 s | 0.56 % | 0.57% |
| Low | 0 | 0 s | 0.47 % | 0.79% |
| Mixed | 2 | 27.5 s | 0.57 % | 0.63% |

TABLE IV
FALSE DETECTIONS, ACTIONS AND MISCLASSIFICATIONS PER RUN

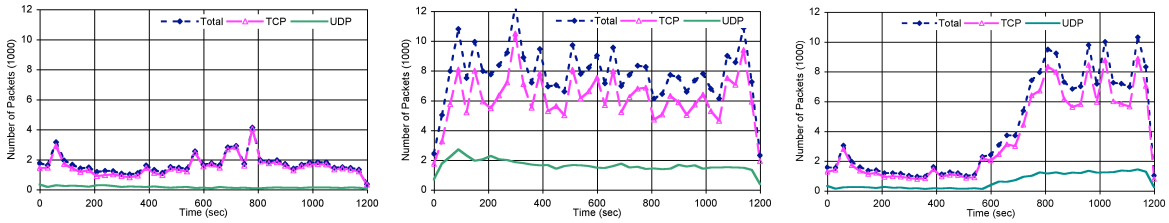
The overall frequency of false positive detections was very low: we have at most 4 false attack detections during the experiment. Also we observed that higher traffic loads caused the false detection rate to increase significantly (0 with low traffic load, 2 with mixed and 4 with high traffic load). Falsely triggered defensive actions lasted for about half a minute. Misclassifications fluctuated in the range 0.47% to 0.79% and there was no clear correlation between the number of misclassifications and the traffic rate. Since overall less than 0.2% of traffic was dropped by the defenses during baseline tests, we conclude that the defense actions did not have a large negative impact on the performance and that they were able to effectively handle changes in the average network load. In the rest of the paper we show the results of experiments performed with the combined COSSACK/D-WARD defense and the high traffic baseline mix.

### B. Spoofing Floods

In spoofing floods attack machines spoof source addresses from the /24 subnet where the machine is located. In each of the spoofing flood attacks, the attackers were on subnets 4, 5, 6, and 7, and one server host on subnet 1 was being attacked. There were four tested attacks: *Spoofing Flood TCP*, *Spoofing Flood UDP*, *Spoofing Flood ICMP* and *Spoofing Flood All*. The first

TABLE II
EXPERIMENTAL RESULTS SUMMARY

| Name | Attack? | Defense? | Source rate | Transport rate | Victim rate | D-WARD detected? | COSSACK detected? | Response time |
|---|---|---|---|---|---|---|---|---|
| **Baseline** | | | | | | | | |
| Baseline | No | No | 300 ± 50 pkts/sec | 100 % | 300 ± 50 pkts/sec | No attack | No defense | No response |
| **Spoofing floods** | | | | | | | | |
| Spoof—TCP | Yes | No | 11 % | 38 % | 4 % | | | |
| | Yes | Yes | 91 % | 96 % | 88 % | Yes | No | 11 sec. |
| Spoof—UDP | Yes | No | 13 % | 38 % | 5 % | | | |
| | Yes | Yes | 90 % | 97 % | 87 % | Yes | No | 10 sec. |
| Spoof—ICMP | Yes | No | 12 % | 34 % | 4 % | | | |
| | Yes | Yes | 86 % | 97 % | 83 % | Yes | Yes | 8 sec. |
| Spoof—All | Yes | No | 11 % | 37 % | 4 % | | | |
| | Yes | Yes | 89 % | 97 % | 86 % | Yes | No | 9 sec. |
| **Rolling floods** | | | | | | | | |
| Roll—1 | Yes | No | 56 % | 91% | 51 % | | | |
| | Yes | Yes | 91 % | 96 % | 87 % | Yes | Yes | 145 sec. |
| Roll—1—Large | Yes | No | 50 % | 90 % | 45 % | | | |
| | Yes | Yes | 91% | 94 % | 85 % | Yes | No | 223 sec. |
| Roll—2 | Yes | No | 35 % | 79 % | 28 % | | | |
| | Yes | Yes | 92 % | 93 % | 85 % | Yes | No | 65 sec. |
| Roll—2—Large | Yes | No | 33 % | 76 % | 25 % | | | |
| | Yes | Yes | 93 % | 93 % | 86 % | Yes | No | 96 sec. |
| Roll—3-Large-Subnet-1 | Yes | No | 41 % | 83 % | 34 % | | | |
| | Yes | Yes | 94 % | 96 % | 90 % | Yes | Only on subnet 9 | 49 sec. |
| Roll—3-Large-Subnet-2 | Yes | No | 33 % | 87 % | 29 % | | | |
| | Yes | Yes | 89 % | 95 % | 85 % | Yes | Only on subnet 9 | 50 sec. |
| **Combination floods** | | | | | | | | |
| HTTP—Roll | Yes | No | 56 % | 89% | 50 % | | | |
| | Yes | Yes | 95 % | 96 % | 91 % | Yes | Yes | 246 sec. |
| Conn—Roll-1 | Yes | No | 40 % | 91% | 36 % | | | |
| | Yes | Yes | 57 % | 92 % | 52 % | Yes | No | 365 sec. |
| Conn—Roll-2 | Yes | No | 66 % | 99% | 66 % | | | |
| | Yes | Yes | 63 % | 98 % | 62 % | Only on subnet 7 | No | Only on subnet 7 |
| **ACK proxy attacks** | | | | | | | | |
| ACK—Proxy-1 | Yes | No | 56 % | 96% | 54 % | | | |
| | Yes | Yes | 56 % | 97 % | 54 % | Yes | Yes | No response |
| ACK—Proxy-2 | Yes | No | 16 % | 54% | 9 % | | | |
| | Yes | Yes | 17 % | 57 % | 10 %/ | Yes | Yes | No response |
| ACK—Proxy-3 | Yes | No | 15 % | 36% | 5 % | | | |
| | Yes | Yes | 14 % | 46 % | 7 % | Yes | Yes | No response |




(a) Low traffic pattern (b) High traffic pattern (c) Mixed traffic pattern

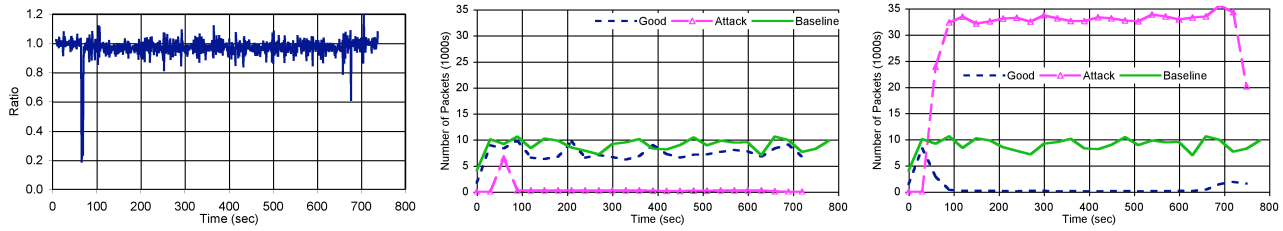
Fig. 2. Legitimate Traffic

three attacks generate TCP, UDP and ICMP flood, respectively. The fourth attack generates a random mix of traffic using these three protocols. The goal of these attacks is to consume network bandwidth at the victim, so they generate 1,200 byte packets to target the bandwidth with a few packets per second. This packet size was chosen arbitrarily. It is close to the maximum packet size of 1,500 bytes for the Ethernet medium used in the experiment. The TCP flood uses TCP data packets. We test a TCP SYN flood in Section VI-F. Each attack machine sends packets as quickly as its Ethernet device allows. In each attack scenario, four attack machines flood one server machine.

Expected outcome: From the defense system specifications, the Red Team expected that these attacks will be effectively detected and controlled. They were included in the tests for completeness and to demonstrate the system's operation under simple floods.

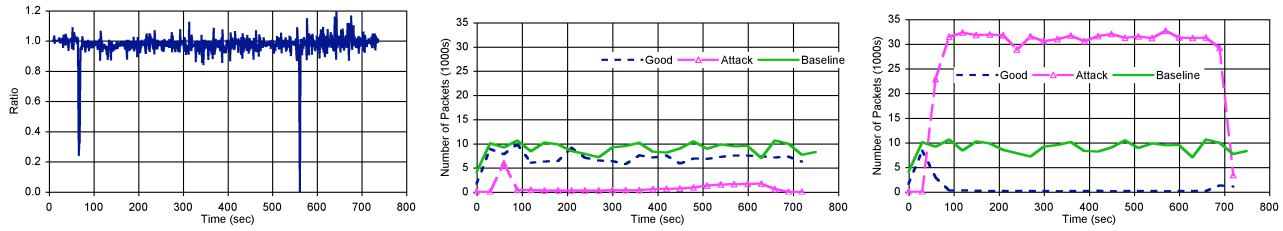
Summary of results: Table II summarizes the results of test runs with spoofing attacks, showing the effect on legitimate traffic as a percentage of the baseline case. In the table, the reduction in rate at which the victim receives legitimate traffic in the presence of an attack is subdivided into two categories: (1) The reduction in *network transport rate* refers to the amount of good traffic that was erroneously dropped by the defenses and not received by the victim subnet. (2) The reduction in *source generation rate* refers to the drop in the quantity of good traffic the traffic generator was able to send, as compared to the baseline case, due to network congestion slowing down good TCP traffic generation. The rate of legitimate traffic at the victim is the product of these two rates.

Figure 3(a) compares the amount of legitimate traffic sent to victim subnet 1 against the amount of legitimate traffic received at subnet 1 for a *Spoofing Flood UDP* attack with a defended system. After the attack starts, this percentage drops sharply for several seconds, while the attack is being detected and the defense system is calibrating a response. After this initial disruption, legitimate traffic is being well protected by the defense and the percentage returns to values close to 100%. As Table II shows, approximately 97% of the legitimate traffic sent during the attack was received at the victim. The traffic generators sent 90% of the baseline traffic level. Thus, in this run the amount of traffic received by subnet 1 was 87% of what it received in the baseline case yielding a 13% net loss. The defended system showed a significant improvement over the undefended system, where the transport rate was 38%, the source rate was 13% of the baseline,




(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 3. Results for Spoofing Flood UDP Attack




(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 4. Results for Spoofing Flood ICMP Attack

and the victim network only received 5% of the baseline traffic as a result of the congestion in the network, yielding a 95% net loss. D-WARD detected and responded to this attack, since the attack created sufficient disturbance in sent-to-received packet ratio needed for D-WARD's attack detection and separation of legitimate from attack connections. COSSACK did not detect this attack because static Snort rules did not match the attack traffic. The defenses were able to throttle the attack within 10 seconds.

Figure 4 shows results for the *Spoofing Flood ICMP* attack. In addition to the initial service disruption, while the attack was being detected, there was another period of poor service at about 550 seconds. This occurred because the defense erroneously detected the end of the attack and initiated the rate limit increase, allowing the attack to disrupt legitimate traffic's service when the rate limit became sufficiently high. Afterwards, the defense detected the attack for the second time and reestablished the rate limit, restoring good service to legitimate traffic.

For space reasons we do not show Figures for *Spoofing Flood TCP* and *Spoofing Flood All* attacks. The defense responded to *Spoofing Flood TCP* in a way similar to its response to *Spoofing Flood UDP*. The response to *Spoofing Flood All* was very similar to the response to *Spoofing Flood ICMP*.

Red Team's Commentary of the Results: The defenses were very effective against the baseline spoofing attacks, as expected. In all four attacks, the baseline traffic was reduced to about 5% of its original volume in the presence of an attack without the defensive software, but to about 80% with the defensive software operational. Over the duration of the experiment runs, these percentages remained fairly consistent, indicating the network as a whole reached a steady state reasonably quickly.

### C. Rolling Floods

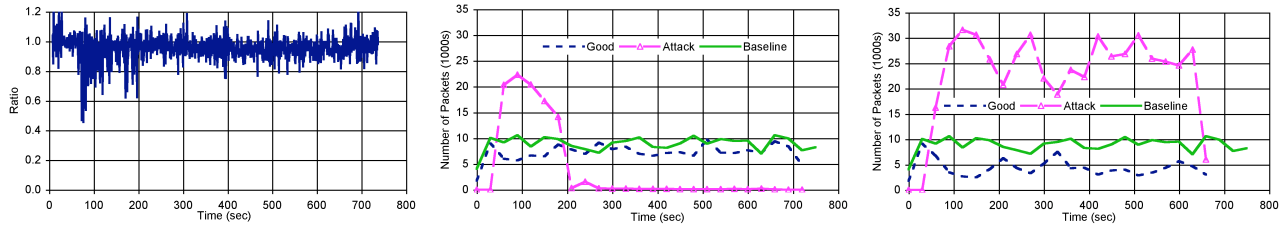
These attacks take advantage of the timing involved in the COSSACK/D-WARD system. Red Team experiments indicated that it took approximately 5 seconds for attack traffic to be completely throttled by D-WARD, and it took approximately 60 seconds without attack for COSSACK to clear a machine's "bad" reputation. Rolling flood attacks coordinate several flooding agents to take advantage of these timing characteristics. Each agent cycles through its targets, flooding them for 5—6 seconds and then lies dormant for a 60 second period. These attacks are similar to pulsing floods [12].

Rolling flood attack was split into several attacks to test the following hypotheses separately: (1) A target can be flooded continuously throughout the entire time window. (2) A target's services can be effectively degraded for most of the time window. The rolling attacks each generate TCP, UDP or ICMP packets in a random mix, and use subnet spoofing. Each attack machine floods as quickly as its Ethernet device allows. In general there were 7—8 attackers targeting 1—2 servers throughout the attack.

Expected outcome: Since these attacks rely on timing to trick defense systems, spoofing and shifting between subnets, it was expected that they should be successful most of the time.

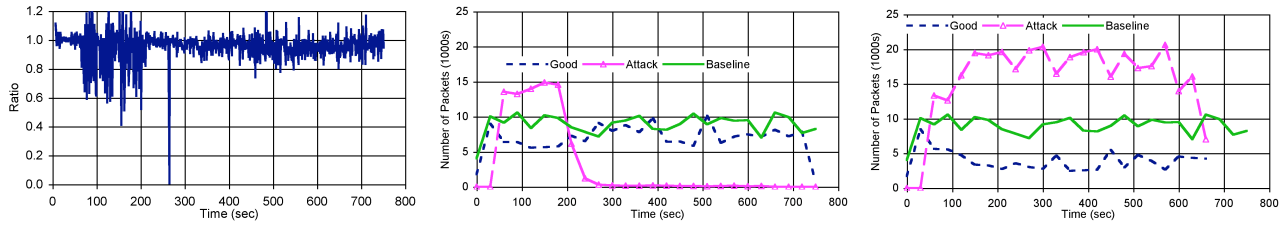
Summary of results: Table II summarizes the results of the rolling attacks. The attacks have a severity rating of medium because they did not have as large an impact on the undefended system as did the spoofing flood attacks.

In the *Roll-1* attack there was one attacker in each subnet 3 through 9, each attacking in sequence, one at a time, for 5 seconds. Attack packets had a random payload size between 500 and 1,000 bytes, not including protocol headers. Figure 5(a) compares the amount of legitimate traffic sent to subnet 1 against the amount of legitimate traffic received at subnet 1 for a defended system. As with the spoofing flood attacks, this percentage is close to 1 most of the time, but the variance is larger indicating that the protection is sporadic rather than stable. According to Table II, the source generated 91% of the baseline traffic level, and the network transport rate was 96% during the attack, resulting in



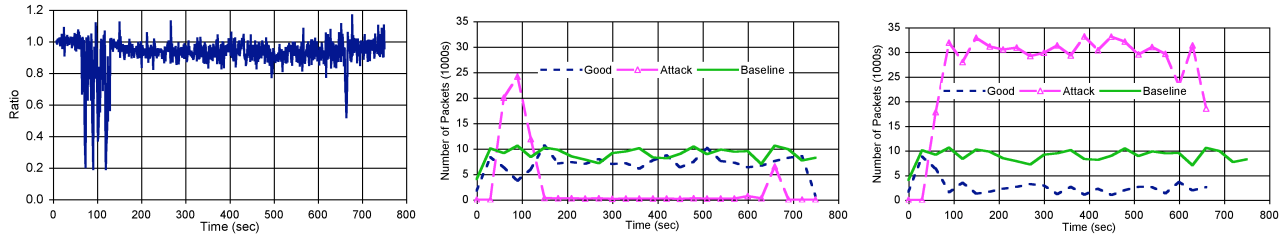

(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 5. Results for Roll-1 Attack



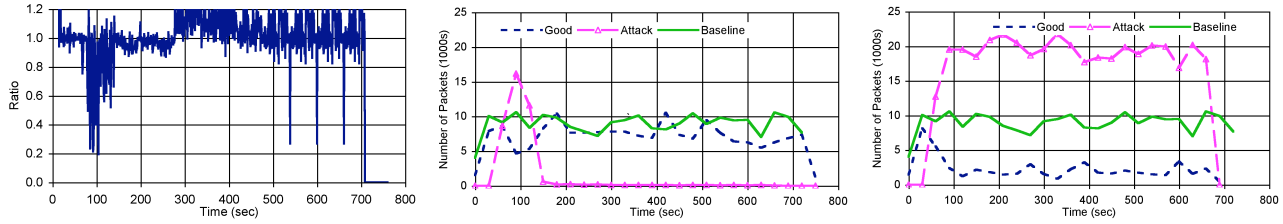

(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 6. Results for Roll-1-Large Attack




(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 7. Results for Roll-2 Attack

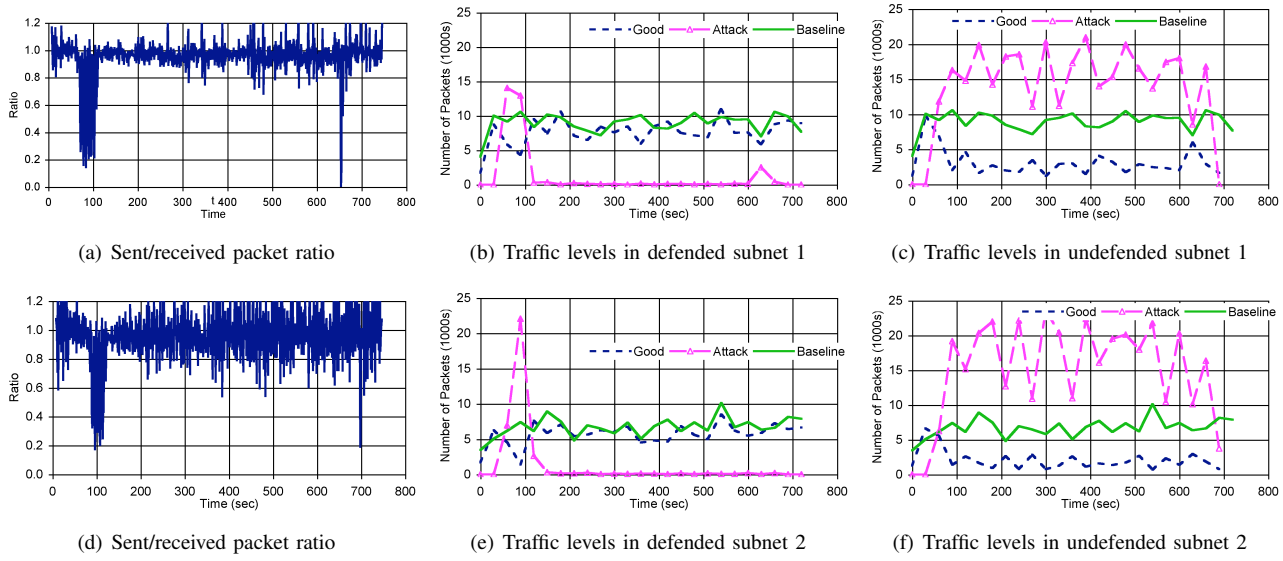



(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 8. Results for Roll-2-Large Attack

the victim receiving 87% as much legitimate traffic as in the baseline case (for a loss of 13%). In the undefended system, the source generated 56% of the baseline traffic level, and the network transport rate was 91% during the attack, resulting in the victim receiving 51% as much legitimate traffic as in the baseline case (for a loss of 49%). This attack had a smaller impact against the undefended system than the spoofing flood attacks because it did not send as much traffic to the victim. The defense needed 145 seconds to fully throttle the attack. Both D-WARD and COSSACK detected this attack, since both sent-to-received packet ratios were abnormal and Snort rules were matched.

The *Roll-1-Large* attack is a modified *Roll-1* attack with an increased payload size of 1,200 bytes. Figure 6(a) compares the amount of legitimate traffic sent to victim network 1 against the amount of legitimate traffic received at network 1 for a defended system. Some legitimate traffic was dropped during the defense stabilization period and at around 270 seconds. According to the Table II the source generated 91% of the baseline traffic level, and the network transport rate was 94% during the attack, resulting in the victim receiving 85% as much legitimate traffic as in the baseline case (for a loss of 15%). In the undefended system, the source generated 50% of the baseline traffic level, and the network




(a) Sent/received packet ratio (b) Traffic levels in defended subnet 1 (c) Traffic levels in undefended subnet 1

(d) Sent/received packet ratio (e) Traffic levels in defended subnet 2 (f) Traffic levels in undefended subnet 2

Fig. 9. Results for Roll-3-Large Attack

transport rate was 90% during the attack, resulting in the victim receiving 45% as much legitimate traffic as in the baseline case (for a loss of 55%). The impact of this attack was somewhat larger than the impact of *Roll-1* attack because of a larger packet size. The defense needed 223 seconds to fully throttle the attack. D-WARD detected this attack, while COSSACK did not. Again, D-WARD's success can be attributed to abnormal sent-to-received packet ratios created by this attack, while COSSACK failed to detect the attack because it did not match its static Snort rules.

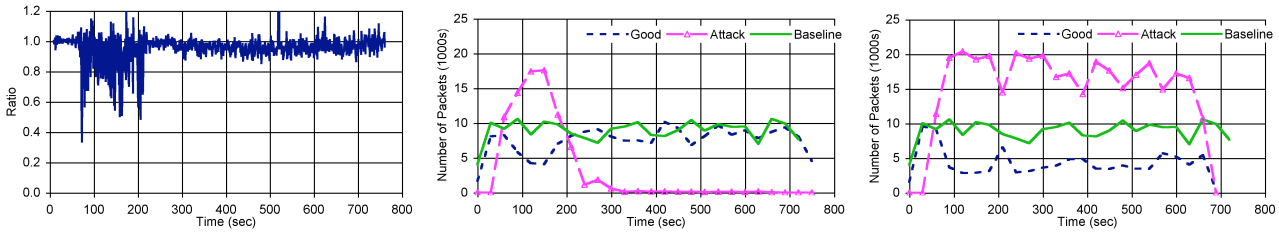
Figures 5(b), 5(c), 6(b) and 6(c) compare the levels of legitimate and attack traffic received at the victim network in the defended and the undefended configurations for *Roll-1* and *Roll-1-Large* attacks. In the defended network, the legitimate traffic level was comparable to the baseline case, while it was about 50% lower in the undefended network.

In the *Roll-2* attack, there was one attacker in each subnet 3 through 9 with two attackers always simultaneously attacking for 5 consecutive seconds. This attack is thus modified *Roll-1* attack with a doubled rate. Figure 7(a) shows the percentage of the legitimate traffic received at subnet 1 for a defended system. The defense stabilization period, at the onset of the attack, lasted for about 65 seconds, with heavy drops inflicted on legitimate traffic. Afterwards, the defense delivered high percentage of legitimate traffic to the victim. The source generated 92% of the baseline traffic level, and the network transport rate was 93% during the attack, resulting in the victim receiving 85% as much legitimate traffic as in the baseline case (for a loss of 15%). In the undefended system, the source generated 35% of the baseline traffic level, and the network transport rate was 79% during the attack, resulting in the victim receiving 28% as much legitimate traffic as in the baseline case (for a loss of 72%). The *Roll-2* had a greater impact against an undefended system than *Roll-1* attack because it sent more traffic to the victim. The defenses needed 65 seconds to throttle the attack. D-WARD detected this attack, while COSSACK did not.

The *Roll-2-Large* attack modifies the *Roll-2* attack by increasing the payload size to 1,200 bytes, so this is a *Roll-1-Large* attack with a doubled rate. Figure 8(a) shows the percentage of the legitimate traffic received at subnet 1 for a defended system. This percentage varied not only during the defense stabilization period (96 seconds) but also from time 300 seconds continuously until the end of the attack. These variations indicate that a defense was constantly readjusting to the changing attack traffic, and was inflicting periodic legitimate drops. The source generated 93% of the baseline, and the network transport rate was 93% during the attack, resulting in the victim receiving 86% as much legitimate traffic as in the baseline case. In the undefended system, the source generated 33% of the baseline traffic level, and the network transport rate was 76% during the attack, resulting in the victim receiving 25% as much legitimate traffic as in the baseline case. Of the rolling attacks, this attack had the greatest impact on the undefended system. The defense needed 96 seconds to throttle the attack. D-WARD detected this attack, but COSSACK did not. The explanation for defense performance in case of *Roll-2* and *Roll-2-Large* is the same as in the case of *Roll-1-Large* attack.

Figures 7(b), 7(c), 8(b) and 8(c) compare legitimate and attack traffic received at the victim network in the defended and the undefended configurations for *Roll-2* and *Roll-2-Large*. The legitimate traffic level in the defended system was comparable to the baseline, after the initial stabilization period. In undefended system, the service was significantly impaired by the attack.

The *Roll-3-Large* attack modified the *Roll-2-Large* attack by also attacking subnet 2 in addition to subnet 1. Figures 9(a) and 9(d) compare the percentage of legitimate traffic received at subnet 1 and at subnet 2 for a defended system. After the initial stabilization period of about 50 seconds, the defenses managed to keep this percentage high in both networks, but with a large variance (especially in the case of subnet 2) which indicates sporadic collateral damage. For the defended subnet 1, the source generated 94% of the baseline traffic level, and the network transport rate was 96% during the attack, resulting in the victim receiving 90% as much legitimate traffic as in the baseline case




(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 10. Results for HTTP-Roll Attack

(for a loss of 10%). In the undefended configuration in subnet 1, the source generated 41% of the baseline traffic level, and the network transport rate was 83% during the attack, resulting in the victim receiving 34% as much legitimate traffic as in the baseline case (for a loss of 66%). In the defended configuration in subnet 2, the source generated 89% of the baseline traffic level, and the network transport rate was 95% of the base-line traffic level, resulting in the victim receiving 85% as much legitimate traffic as in the base-line case (for a loss of 15%). In the undefended system in subnet 2, the source generated 33% of the baseline traffic level, and the network transport rate was 87% of the baseline traffic level, resulting in the victim receiving 29% as much legitimate traffic as in the baseline case (for a loss of 71%).

Figures 9(b), 9(c), 9(e) and 9(f) compare the levels of legitimate and attack traffic received at the victim network 1 and network 2 in the defended and the undefended configurations. In the defended system, legitimate traffic level was comparable to the baseline, while it was much lower in the undefended system. COSSACK only detected this attack as originating from subnet 9 because this subnet sent more attack traffic than the others which triggered COSSACK's alarms. D-WARD detected the attack in all involved subnets because of the consequent increase in sent-to-received packet ratio from those subnets to the victim. The defense took 49 seconds to throttle the attack on subnet 1 and 50 seconds to throttle the attack on subnet 2.

Red Team's Commentary of the Results: The percentage of legitimate traffic reaching its destination in the presence of an attack, but without defensive software, was far higher than either anticipated or observed in the previous spoofing attacks. In all variations of the rolling attacks this number was no less than 25% of the original figure (as opposed to only 5% as experienced during the spoofing attacks). A larger attack rate should have been generated to achieve higher service denial. With the defensive technology operational, legitimate traffic loads returned to the 80%-90% levels as seen with spoofing attacks, indicating the technology was not "fooled" by this attack strategy, or the specific timings used for these particular experiments. This suggests that the attacks were not optimally orchestrated for the final experiment configuration, because the reconnaissance network was different from the one used in the testing phase. The time needed by defenses to detect and respond to these attacks lead the Red Team to speculate that a more careful orchestration of rolling attacks could result in sustained service denial.

### D. HTTP Flood Attacks

These attacks generated a long stream of bogus HTTP requests that aimed to use up a target's resources in generating responses. The Red Team observed that two attacking machines were sufficient to degrade service from a single server machine. No traffic was spoofed and these attacks were not run separately, but in conjunction with other packet-flooding attacks.

Expected outcome: At a high rate, this attack should degrade the target's HTTP service. At a moderate rate, this attack should provide seemingly legitimate decoy traffic that should slow down detection or response of a defense system to another, more potent attack interleaved with HTTP flood.

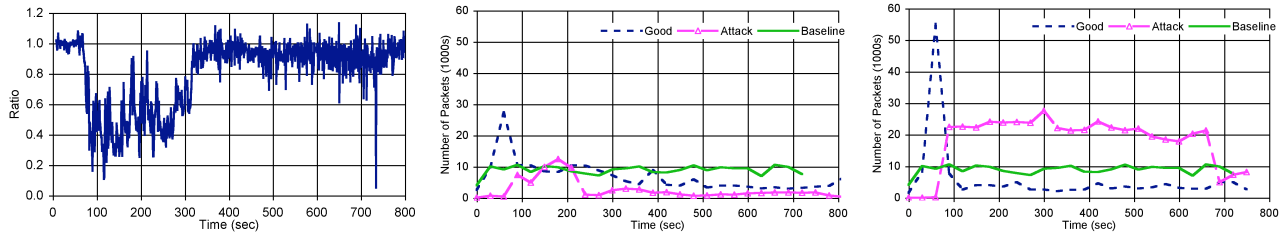
### E. HTTP-Rolling Attacks

This attack is a combination of the rolling flood *Roll-1-Large* and the HTTP flood. The rolling flood provides the majority of the attack traffic while the HTTP flood (run at a slower speed) attempts to provide "legitimate" traffic from the attacking network to redeem it at the defense system. The attackers are on the same subnets as in the *Roll-1-Large* attack.

Expected outcome: This attack should degrade service in a manner similar to the rolling flood, but the use of HTTP "redeeming" traffic should cause a slowdown in the defense response.

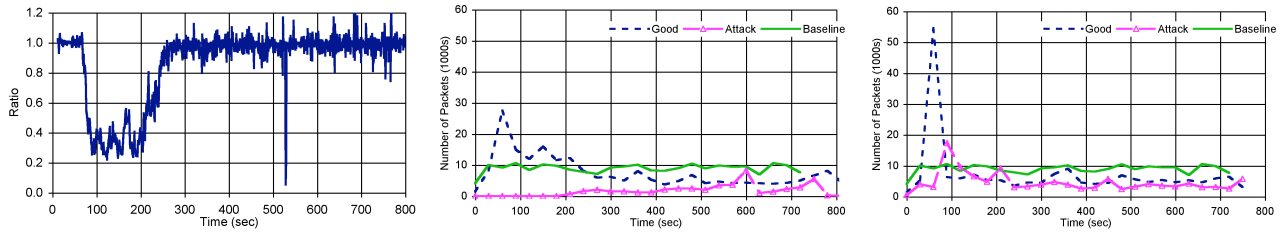
Summary of results: Table II summarizes the results of the *HTTP-Roll* attacks. The *HTTP-Roll* attack has a severity rating of medium because it did not severely degrade service to legitimate clients. In the defended configuration, the source generated 95% of the baseline traffic level, and the network transport rate was 96% during the attack, resulting in the victim receiving 91% as much legitimate traffic as in the baseline case (for a loss of 9%). In the undefended system, the source generated 56% of the baseline traffic level, and the network transport rate was 89% during the attack, resulting in the victim receiving 50% as much legitimate traffic as in the baseline case (for a loss of 50%).

Figure 10(a) shows the percentage of legitimate traffic received at network 1 for a defended system. The stabilization period was longer than for spoofing and rolling attacks — around 250 seconds. Afterwards, the defense offered stable and good protection to legitimate traffic. Figures 10(b) and 10(c) compare the levels of legitimate and attack traffic received at the victim network 1 in the defended and the undefended configurations. The legitimate traffic was lower than the baseline during the defense stabilization period, but it matched the baseline level afterwards. The undefended network had significantly lower legitimate traffic




(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 11. Results for Conn-Roll-1 Attack




(a) Sent/received packet ratio (b) Traffic levels in defended network (c) Traffic levels in undefended network

Fig. 12. Results for Conn-Roll-2 Attack

level. D-WARD detected this attack everywhere, while COSSACK did not. D-WARD's success can be attributed to the fact that excessive attack traffic created increase of sent-to-received packet ratio, which triggered D-WARD's action. COSSACK's failure to detect can be attributed to failure of the attack traffic to match static Snort rules. The effective respose took around 25 times longer than for the spoofing flood attacks.

Red Team's Commentary of the Results: This attack was less effective than Red Team predicted. The defense mechanisms successfully detected and contained the DDoS traffic. Red Team links the partial failure of the *HTTP-Roll* attack with the reduced effectiveness of rolling attacks.

### F. Connection Attacks

Each operating system has a hard limit on the number of TCP connections that can be simultaneously active, because the state table used by the OS to store connection data is of limited size. Connection attacks aim to consume the space in this table by creating *half-open* TCP connections. The attack machine sends spoofed TCP SYN packets, that lead to connection record allocation at the target, who replies with a TCP SYNACK. The attacker ignores these replies (hence the term "half-open" connection) and the allocated connection records eventually time out in absence of any activity. This attack is known in the literature as TCP SYN flood attack [13].

The overall effect of the connection attack is that the target cannot serve new users of the application whose port is under the attack. Although the attack's effect is limited to new TCP traffic, this does not discount its significance, as the majority of network traffic is TCP. The connection attack in this exercise flooded each port on the target machine. The attack was run in conjunction with other packet-flooding attacks.

Expected outcome: This attack should deny service to new TCP connections and should not be detected by defenses.

### G. Connection-Rolling Attacks

These attacks are a combination of the rolling flood and the connection flood. It was hypothesized that the effectiveness of the connection flood was limited because only service to new TCP connections is denied while existing connections and non-TCP traffic proceed unharmed. The rolling flood attack was meant to affect these other kinds of legitimate traffic to result in a more effective denial of service. The connection attack was combined with two rolling flood attacks, *Roll-1-Large* and *Roll-2-Large*, creating two test scenarios: *Conn-Roll-1* and *Conn-Roll-2*. The attack was launched from subnets 2—9 with a total of 11 attacking hosts, targeting a server in subnet 1.

Expected outcome: This attack should degrade service in a similar manner to the rolling flood, but with a higher level of impact on legitimate TCP traffic.

Summary of results: Table II summarizes the results of the *Conn-Roll* attacks The *Conn-Roll-1* attack has a severity rating of medium, and *Conn-Roll-2* attack has a severity rating of mild. Both attacks impacted the service moderately.

Figure 11(a) shows the percentage of legitimate traffic received at network 1 for a defended system and *Conn-Roll-1* attack. Figures 11(b) and 11(c) compare the levels of legitimate and attack traffic received at the victim network 1 in the defended and the undefended configurations. The Red Team used the nmap tool [14] to determine unused port numbers (so that the attack could skip these ports), and did not label the packets sent out by this tool. This accounts for higher-than-baseline legitimate traffic levels in the first 100 seconds in figures 11(b) and 11(c). Much of this unlabeled traffic was dropped by the defenses, and there was a noticeable drop in the amount of legitimate traffic

sent and received. The defense stabilization period lasted for 300 seconds. Afterwards, the defense protected ongoing TCP connections well but new connections were still impaired by the connection flood attack. Approximately 82% of the legitimate traffic sent was received at the victim when the defenses were active. Including the source generation rate slowdown, the amount of traffic received by subnet 1 was 80% of what it received in the baseline case. However, these values are skewed as they assume that the nmap packets are good. Excluding the data from the first five minutes of the run, the source generated 57% of the baseline traffic level, and the network transport rate was 92% during the attack, resulting in the victim receiving 52% as much legitimate traffic as in the baseline case. Breaking this down by protocol, 94% of the TCP traffic was received and 80% of the UDP traffic was received during this same time period. Thus, subnet 1 did exhibit a significant degradation of service as a result of this attack. In undefended network, the source generated 40% of the baseline traffic level, and the network transport rate was 91% during the attack, resulting in the victim receiving 36% as much legitimate traffic as in the baseline case.

Although the throughput of the legitimate traffic was similar in the defended and undefended cases, the defenses prevented a majority of the attack traffic from reaching subnet 1 in the defended case. The defense needed 365 seconds to throttle the attack. D-WARD detected this attack, while COSSACK did not.

The *Conn-Roll-2* attack was detected only by D-WARD at subnet 7 because that subnet had slightly increased sent-to-received packet ratio that triggered D-WARD's alarm. Consequently, defenses took very few responses. The legitimate traffic to subnet 1 was degraded, and there was little difference between the defended and undefended cases (Figures 12(b) and 12(c)). Percentage of legitimate traffic received at network 1 for a defended system and *Conn-Roll-2* attack is shown in Figure 12(a).

Red Team's Commentary of the Results: Connection floods in combination with the rolling attacks, especially once they were well underway (beyond 400 seconds into the experiment), appeared to be quite effective in blocking normal traffic despite the defensive technologies. In particular, once the experiment reached steady state, there was little difference in the throughput of normal traffic with or without the defense mechanisms running.

### *H. ACK Proxy Attacks*

The Red Team decided early on that an interesting test scenario would include helper machines outside the attack or the victim networks, who act in concert with attack machines to trick the defense. The idea was to spoof a TCP ACK for every TCP SYN packet sent to flood a target (this can easily be generalized to work for TCP data packets also). Each time an attacking machine sends out a TCP SYN packet to the target, it makes a request to a helper, who creates the ACK reply as if it were coming from the target of the TCP SYN flood. The spoofed ACK packets are meant to fool D-WARD's traffic classification mechanism, and lead to classification of attack traffic as legitimate. This attack makes assumptions that there is no ingress/egress filtering or COSSACK/D-WARD on helper subnets.

For each ACK Proxy attack, four attack machines flood a single server. Each attack machine has a dedicated ACK server. Three ACK Proxy attacks were generated. In *ACK-Proxy-1* attack, for every malicious connection request, 2 SYN packets were sent from the flooding agent to the target and 2 spoofed ACK packets were sent from the ACK Proxy to the flooding agent. In *ACK-Proxy-2*, the similar dynamics as in *ACK-Proxy-1* were deployed, but the attack and ACK traffic rate were doubled. In *ACK-Proxy-3* attack the rate was quadrupled, compared to *ACK-Proxy-1*. In all three ACK Proxy attacks, the helper machines were located in subnets 3, 7, 8, and 9, that did not run defenses.

Expected outcome: This attack will not be detected and will generate enough traffic to degrade service.

Summary of results: Table II summarizes the results of the ACK Proxy attacks. The *ACK-Proxy-1* attack has a severity rating of medium because the loss was 46% percent. The *ACK-Proxy-2* and *ACK-Proxy-3* attacks are rated as high severity because the losses were 90% and 95%, respectively.

For space reasons we omitted figures for *ACK-Proxy* attacks. There was essentially no difference in the system behavior in the two cases: both in the amount of attack traffic; and the amount of legitimate traffic received. In the defended configuration in the *ACK-Proxy-1* attack, the source generated 56% of the baseline traffic level, and the network transport rate was 97% during the attack, resulting in the victim receiving 54% as much legitimate traffic as in the baseline case (for a loss of 46%). In the undefended system, the source generated 56% of the baseline traffic level, and the network transport rate was 96% during the attack, resulting in the victim receiving 54% as much legitimate traffic as in the baseline case (for a loss of 46%). COSSACK did detect this attack because it matched Snort rules; however, the combined system did not take any responses because D-WARD was fooled by the fake ACK replies and classified all attack traffic as good. Results for *ACK-Proxy-2* and *ACK-Proxy-3* look very similar to the results of *ACK-Proxy-1* — the defense detected the attack but did not take any responses. For space reasons, we omit the corresponding graphs but we show the summary of results in Table II.

Red Team's Commentary of the Results: This set of attacks was very effective against the defense technologies and behaved as the Red Team expected.

### *I. Experiment Conclusions*

The Red Team's attacks were carefully designed to stress-test COSSACK/D-WARD defense at its performance boundaries, progressively going from simple to more sophisticated attacks. The performance of defense decreased with attack sophistication. The defense was successful against simple *Spoofing Flood* attacks and even against more sophisticated *Rolling* attacks. The performance of the defense degraded somewhat when *Rolling* attacks were combined with connection and HTTP traffic in *HTTP-Roll* and *Conn-Roll* attacks. On the other hand, these attacks did not severely degrade a victim's service in the undefended system because of their small aggregate rate. This small rate was dictated by the necessity to "roll" the attack and by the small number of edge hosts (20) in the experimental network — thus a few hosts could be active attackers at any given time. The defense failed completely to respond to *ACK Proxy* attacks, although they were detected by the COSSACK system. This failure is linked to a

D-WARD's feature that classifies TCP connections as "good" if they receive sufficient acknowledgments for their outgoing TCP packets. This feature is tricked by fake ACK packets sent by helpers and thus labels all attack traffic as good, failing to defend. Overall, the advantages of combining COSSACK and D-WARD systems were not as great as we expected, since D-WARD detected all but *ACK Proxy* attacks, COSSACK missed a lot of attacks, and the joint system was driven by D-WARD's actions.

## VII. Lessons Learned

This Red Team/Blue Team exercise confirmed that the defense systems offered effective protection against attacks they were designed to handle, but it revealed significant vulnerabilities when sophisticated attacks were deployed. The Blue Team benefited greatly from these results, which lead to improvement of the defense design in the months following the exercise. All teams felt that the repetition of the exercise with improved defenses would be very valuable, but the time and cost did not permit this.

All three experimental teams also made the following observations with regard to lessons learned: (1) Red Team reconnaissance and attack testing should occur on the network ultimately used for defense testing, as any hardware changes compromise fine tuning of the attacks. The dependency of sophisticated DDoS attacks that rely on timing on particular hardware and network configurations that was demonstrated in this experiment suggests that running such attacks in the real Internet might be harder than we expect. (2) Experimental infrastructure must remain intact after execution; this is needed for answering questions raised in the data analysis by running additional experiments. All measurements taken during the exercise, and detailed packet traces, should be saved. (3) Attacks should be launched on an isolated network because it was observed that it was difficult to shut attacks off at the end of the runs. The Red Team had to insert a kill timer that would simply kill attack processes after a desired duration. (4) Scripted attacks are easier for logging and repeatability than live Red Team actions. But, the Red Team could likely have launched more effective attacks if it had live control and the ability to tune the attacks. Such advantage would be enjoyed by real attackers. (5) Last-minute surprises should be budgeted for, both in time and money. Preparation of this exercise required extensive weekly teleconferences and several face-to-face meetings between all teams. Each phase of the exercise was longer (and thus cost more) than originally expected because the planned tasks proved more complex than anticipated. (6) Defense code must be frozen before reconnaissance, the reconnaissance must be repeated after each change and this creates the "moving target" problem for Red Team.

Additionally, the Red Team observed that any DDoS defensive strategy that relies upon timing mechanisms or timing considerations is susceptible to attacks that are themselves based in timing. Adjusting the defensive timing values to thwart the specifics of one such attack is not a generalized response, as the attackers are free to observe new defensive timing parameters and adjust their attacks accordingly. Similarly, tuning defensive parameters for responsiveness under one set of network traffic conditions does not provide a generalized defense against unpredictable network conditions. The Red Team believes that developers of defensive software technologies should work to remove such tuning requirements from their systems.

All teams agreed that attempting to recreate any sort of realistic DDoS test environment using only a very limited number of machines results in a highly artificial test environment, because the limited infrastructure simply does not have enough nodes to effectively simulate the dynamics of real-world DDoS attacks. A standalone DDoS testbed, with several hundreds or thousands of nodes and a core network mirroring today's Internet architecture of high speed backbones bordered by autonomous systems, was deemed as the only realistic test infrastructure for DDoS attacks. Such testbed has since been jointly funded by DHS and NSF through the DETER project [15].

Finally, all teams felt that if a tighter interaction between participants were possible, and if all teams were involved in all phases of the exercise, the results could have been much more rewarding. If the Red Team were allowed to perform live attacks, they would have likely been much more effective. If the Blue Team were allowed to retune their software after a successful attack, there would have been a rapid improvement in the defense quality. This refined defense could then be re-tested to validate that the detected vulnerabilities were successfully handled, and to look for new problems. Unfortunately, such an interactive exercise would have required a tremendous expenditure in time and funding, and is an unrealistic goal. All teams felt that they gained a valuable knowledge from this Red Team/Blue Team exercise that they would not have been able to obtain otherwise, and they are grateful for DARPA's support of this endeavor.

## References


[1] W. W. Gibbs, "Red Team versus the Agents," *ScientificAmerican.com*, December 2000.

[2] D. Levin, "Lessons Learned using Live Red Teams in IA Experiments," in *Proceedings of DISCEX III*, 2003.

[3] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan, "COSSACK: Coordinated Suppression of Simultaneous Attacks," in *Proceedings of DISCEX III*, 2003, pp. 2–13.

[4] J. Mirkovic and P. Reiher, "D-WARD: A Source-End Defense Against Flooding Denial-of-Service Attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 3, pp. 216–232, 2005.

[5] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," IETF RFC 2267.

[6] *YOID Homepage*, Information Sciences Institute, http://www.isi.edu/div7/yoid.

[7] *CISCO IOS Netflow*, CISCO, http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[8] Snort.org, "Snort," http://www.snort.org/.

[9] S. Corporation, "Skaion Traffic Generation System," http://www.skaion.com/products/index.html.

[10] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS Defense by Offense," in *ACM SIGCOMM 2006*, September 2006.

[11] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W. Yao, and S. Schwab, "Towards User-Centric Metrics for Denial-Of-Service Measurement," in *Proceedings of the Workshop on Experimental Computer Science*, June 2007.

[12] A. Kuzmanovic and E. W. Knightly, "Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants)," in *ACM SIGCOMM 2003*, August 2003.

[13] C. Schuba, I. Krsul, M. Kuhn, G. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997.

[14] *nmap Security Scanner*, InSecure.org, http://www.insecure.org/.

[15] T. Benzel, R. Braden, D. Kim, C.Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Experiences With DETER: A Testbed for Security Research," in *Proceedings of TridentCom 2006*, March 2006.

**Jelena Mirkovic is a computer scientist at the USC Information Sciences Institute, which she joined in 2007. Prior to this she was an assistant professor at the University of Delaware, 2003-2007. She received her M.S. and Ph.D. from UCLA, and her B.S. in Computer Science and Engineering from the School of Electrical Engineering, University of Belgrade, Serbia. Her current research is focused on computer worms, denial-of-service attacks, and IP spoofing, and is funded by the National Science Foundation and the Department of Homeland Security.**

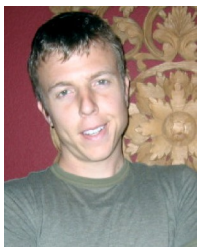
**Peter Reiher received his B.S. in Electrical Engineering and Computer Science from the University of Notre Dame in 1979. He received his M.S. and Ph.D. in Computer Science from UCLA in 1984 and 1987, respectively. He has done research in the fields of distributed operating systems, security for networks and distributed computing, file systems, optimistic parallel discrete event simulation, ubiquitous computing, naming issues in distributed systems, active networks, and systems software for mobile computing. Dr. Reiher is an Adjunct Associate Professor in the Computer Science Department at UCLA.**

**Christos Papadopoulos is currently an associate professor at Colorado State University. He received his Ph.D. in Computer Science in 1999 from Washington University in St. Louis, MO. His interests include network security, router services, multimedia protocols and reliable multicast. In 2002, he received an NSF CAREER award to explore router services as a component of the next generation Internet architecture.**

**Alefiya Hussain is a senior principal scientist at Sparta Inc. Her research interests include statistical signal processing, protocol design, security, and network measurements. She received a Bachelor of Engineering degree from Pune Institute of Computer Technology, and a M.S. and Ph.D. in Computer Science from the University of Southern California. She is a member of ACM and Upsilon Pi Epsilon. The work described in this paper was done while Alefiya Hussain was a research assistant at USC/ISI.**

**Marla Shepard received a Master of Science in Electrical Engineering from the University of Rhode Island and a Bachelor of Science in Computer Science and Engineering from the University of Connecticut. For the past 10 years, Marla has been a scientist for BBN Technololgies doing research in the Information Security group. Marla has worked on projects in a variety of areas including Red and Blue team experimentation and Public Key Infrastructure (PKI) technologies. Marla was the data analaysis lead of the White Team in the D-WARD and COSSACK experimentation.**

**Michael Berg is a Senior Member of Technical Staff at Sandia National Laboratories with five years of experience in the design and security evaluation of networked information systems. His research interests include new ways that both hardware and software can be used to improve the security of information systems. Michael has a B.Sc. in Computer Science (2002) and a B.Sc. in Electrical Engineering (2002) from the New Mexico Institute of Mining and Technology.**



**Robert Jung is a Senor Member of Technical Staff at Sandia National Laboratories with several years of experience as an active participant in Red Team projects and tool development efforts. In addition to his background and interest in Computer Security, Robert's research interests also include Supervised Machine Learning. Robert has a B.Sc. in Computer Sciences from the University of New Mexico (2002), a Masters of Business Administration from the University of New Mexico (2004), and an M.Eng. in Computer Science from Cornell University (2006).**