COSSACK: Coordinated Suppression of Simultaneous Attacks

Christos Papadopoulos
Robert Lindell
John Mehringer
{christos, lindell, mehringe}@ISI.EDU

Alefiya Hussain
Ramesh Govindan
{hussain, govindan}@ISI.EDU

Abstract[1]

DDoS attacks are highly distributed, well coordinated, offensive assaults on services, hosts, and infrastructure of the Internet. Effective defensive countermeasures to DDoS attack will require equally sophisticated, well coordinated, monitoring, analysis, and response. The Cossack project is developing an architecture to explore such coordination using multicast, annotated topology information, and novel blind detection techniques.

1. Introduction

As the Internet grows in size and complexity, its increased visibility and its diversity seem to attract a variety of highly damaging attacks. Of these, the distributed denial-of-service (DDoS) attacks are proving to be the most pernicious. A DDoS attack can be characterized as a simultaneous network attack on a victim from large numbers of hosts well distributed throughout the network. The attack overwhelms the victim using large aggregated firepower and renders the target inoperative, sometimes for several days. These attacks can cause significant monetary losses for businesses, but also represent a technique for stifling or expressing political dissent. If recent DDoS attacks are a good predictor of the future, we expect these types of attack to increase both in frequency, sophistication, and sheer size.

Recent DDoS attacks have revealed a disturbing pattern. These attacks are increasing in scale. Increasingly, attacks are being carried out by sophisticated, highly automated, tools that search out hosts in the Internet, vulnerable to known intrusion techniques, and install software which provides remote access and control to the attackers. This automated searching of hosts can go on for weeks or months until a large arsenal of compromised hosts has been amassed. As these tools continue to mature, they are increasingly building a hierarchical command and control infrastructure—one that includes complex cryptographic techniques to thwart detection and dismantling—to manage the scalability issues involved in controlling a network of 1000s to 10000s of hosts. This efficient use of deep hierarchical command and control makes these systems easy to operate and hard for the victim to identify resources and launch effective countermeasures to eliminate the attack.

How are these attacks currently dealt with? For the most part, they still require a high degree of manual intervention. Individuals, highly trained in both network operations and security, pour over audit data and form convincing hypotheses consistent with the audit trails. They then contact other ISPs in the Internet to confirm suspicious traffic patterns and coordinate a collective response to the attack. Attempts are being made to develop tools to automate the analysis of audit data using Intrusion Detection Systems (IDS) that perform high-speed pattern matching against a database of known attack signatures. Studies of the effectiveness of IDS systems have so far shown that they are incapable of reasonably detecting previous unknown attacks. They only perform well when presented with attacks which are represented in their signature databases.

Ongoing research efforts have, to a perhaps unhealthy degree, been focused on traceback techniques for attribution. Many believe that if one could trace back to the origin of the attack, it will be possible to effectively counter the attack by automated means. It is unclear whether it is useful to expend vast amounts of resources to traceback and identify individual soldiers of the attack when the generals at the top of the hierarchical command and control continue to operate unnoticed and uninhibited.

Is the future really this bleak? We don't believe so. A mindset change, one that focuses less on attribution of attack origins, and more on automated mitigation of such attacks, can lead to important breakthroughs in building DDoS

1.This work is supported by DARPA Grant No. N66001-01-1-8939

defenses. Two innovative technologies exemplify this mindset:

- A departure from signature-based attack detection, blind attack detection can enable defense even against a heretofore unknown form of attack. Such detection mechanisms can have a high rate of false positives, and must therefore necessarily be augmented with distributed coordination techniques that can significantly improve accuracy. Analogies from signal processing offer insight into the exploration of localization techniques that will determine the directions of an attack from the correlation of data from multiple sensors placed throughout the network. These localization techniques will not be able to determine the identity of individual attacking hosts, but provide aggregate information necessary to install effective filters at key router locations in the network. We have developed similar distributed coordination mechanisms for network fault isolation in the context of the SCAN project at ISI.

- A second key capability is the availability of annotated network topology databases. Ongoing research to scalably map the topology of large networks could be augmented to identify vulnerable hosts and targets and selectively protect these assets. These databases can be used dynamically by the coordination mechanisms to determine if a target is susceptible to an attack, and to rapidly install system defenses.

Our proposed Cossack system leverages these innovative technologies to develop an automated system for DDoS attack mitigation. Our system requires no manual intervention, will be attack signature independent, and will be largely complementary to ongoing research in traceback. In fact, one might argue that traceback will largely be obviated by Cossack if we have a way of suppressing these attacks. Traceback will still be necessary for detecting compromised hosts, but will not be the primary defense mechanism.

Cossack works as follows. Each large organization in the network will run ***watchdog*** software at its network egress. These watchdogs perform a number of important functions and coordinate their activities using a peer-to-peer multicast communications mechanism being developed in the context of the Yoid [25] project at ISI. Watchdogs are responsible for scanning the topology of their local surroundings and identifying potentially vulnerable hosts. In addition, watchdogs contain traditional IDS systems and necessary response agents. Augmenting the IDS systems are blind detection techniques which correlate information from other closely proximate watchdogs. Our coordinated system of watchdogs will focus on automating the detection and response of DDoS attacks, including blind detection techniques, and ensuring that watchdogs will not be incapacitated by the totality of the attack. Since most of the watchdogs will, in general, be far away from the target they can focus substantial resources in the timely analysis of network activity.

The high profile nature of the DDoS attacks in the Internet has spurred a flurry of interesting related work to our proposed research approach. Work is ongoing to extend current IDS systems to be more effective against DDoS attacks. A Common Intrusion Detection Framework (CIDF) is being developed to more easily combine the analysis of different IDS systems and provide a common format for sharing in a distributed platform. As we have mentioned, traceback algorithms continue to be a popular topic. Of most relevance to our work is IDIP. The IDIP work correctly identifies the need to have a robust distributed system of detectors and response agents. Unfortunately, it's underlying multicast like backplane does not provide the robust and scalability that is addressed in the peer-to-peer Yoid research. Also, IDIP focuses on traceback methods rather than the building and maintenance of topology knowledge bases as aids in the distributed correlation and detection process.

2. DDoS Background

Network security is rapidly becoming a vital issue in the Internet. Statistics collected by CERT [1] show that security incidents have been increasing at an alarming rate. In the past three years, for example, security incidents have been doubling each year. In 1988, when CERT was established, there were 6 security incidents reported; that number in 2000 stands at about 22,000. While there are several contributing reasons including social reasons and mis-configuration, many security breaches occur due to security holes in new software. For example, a report issued by the government last year listed over 1000 software vulnerabilities discovered during that year alone [6]. This is important, because software vulnerabilities allow systematic compromise of a large number of hosts.

One of most recent types of network attacks is the Distributed Denial of Service (DDoS) attack. DDoS exploits first gained the attention of computer security professionals around Fall 1999. In February 2000 came the first highly publicized DDoS attack, which crippled several prominent web sites including Yahoo, Amazon, CNN and eBay. DDoS attacks have proliferated over the past year [2] and indications are that more are in the works. Several government security agencies have identified the seriousness of the problem and have issued advisories [3] and tools to detect compromised systems [4]. At least five of the advisories issued in the year 2000 were specific to DDoS attacks [5]; no such advisories were issued in 1999.

2.1 Anatomy of a DDoS Attack

Staging a typical DDoS attack requires several steps. First, an attacker breaks into many machines, perhaps using some of the publicized software vulnerabilities. For each compromised machine, the attacker installs the attack tool, and then moves on to the next victim. The latest cracking tools automate the entire process to the degree where the attacker can literally sit back and watch the arsenal of compromised machines grow to hundreds or thousands. Once the attacker has hijacked enough machines, the attacker configures them in a hierarchical structure, as shown in Figure 1.

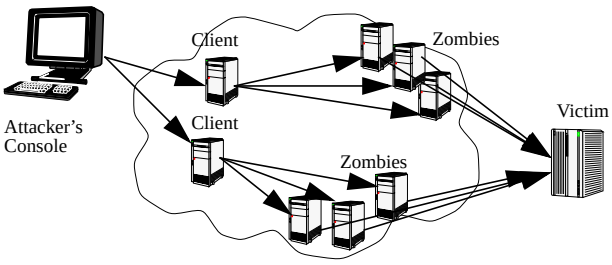


Figure 1: A DDoS attack

At the bottom level of the hierarchy are the "zombies", which are the actual machines carrying out the attack. Zombies lie dormant, passively listening for instructions that identify the target and the type of attack to be invoked. Many attack tools support several types of attack. At the intermediate level are the "clients", each one responsible for feeding instructions to a subset of the zombies. And finally, at the highest level is the attacker's console. This configuration allows the attacker to trigger a large scale attack by simply sending messages to a few clients. Depending on the number of zombies and the network capacity at the target, the results can be devastating.

2.2 What Makes DDoS Attacks Possible?

The rapid expansion of the Internet and the proliferation of low-cost PCs are two important factors that have made DDoS feasible. In addition, the following recent trends have contributed to the rise in DDoS attacks:

- The increase in the number of new software and the (inevitable) security vulnerabilities that accompany them, present many opportunities to hijack computers.

- The number of computers with broadband connections (xDSL, cable modems) has been rapidly increasing. Not only do these computers pose a danger (if hijacked) due to their high-speed connections, but their "always on" nature makes them far more susceptible to compromise.

- The lack of automated security update of software vulnerabilities means that the user is responsible for carrying out this task manually. Since many users either lack the time, knowledge or motivation to do so, many systems remain running software with known insecurities.

- The availability of attack tools (along with instructions on how to use them) on several web sites (many outside the US), drastically expands the number of potential attackers, who no longer need to understand the operation of the tools in order to use them. Termed "script kiddies", this is a new breed of attacker, one who can use attack tools without understanding them.

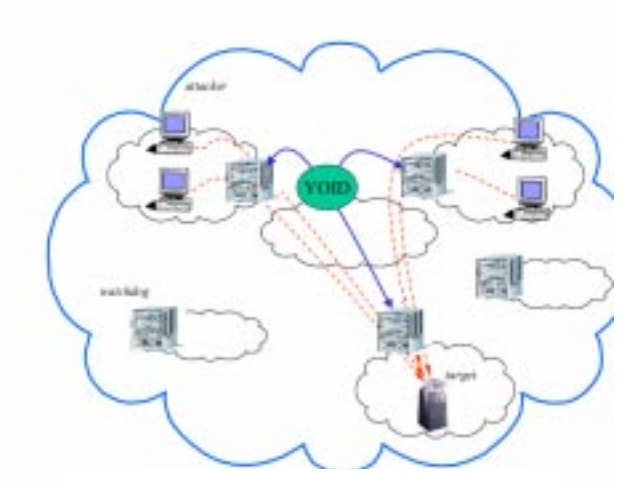
Most DoS attacks hide the true origin of the attacker by using spoofed source addresses. DDoS attacks are particularly attractive because their nature makes attribution even harder. Unlike traditional single-source attacks, DDoS attacks are virtually impossible to trace due to the numerous attack paths and the multiple levels of indirection. Moreover, attack tools are constantly evolving and some already incorporate defenses like encryption and "decoy" packets to sidetrack traceback.

In summary, efforts to improve immunity to DDoS attacks by manually securing systems or by tracing back the attack although commendable, are difficult to achieve. The lack of attribution, impossibility of securing every machine on the Internet, and difficulty of performing intrusion detection, mean that host-based or highly localized solutions to neutralize DDoS attacks will not work. What is needed is a solution that can be deployed on the entire Internet, which can leverage off topology knowledge and distributed algorithms for attack detection and coordination.

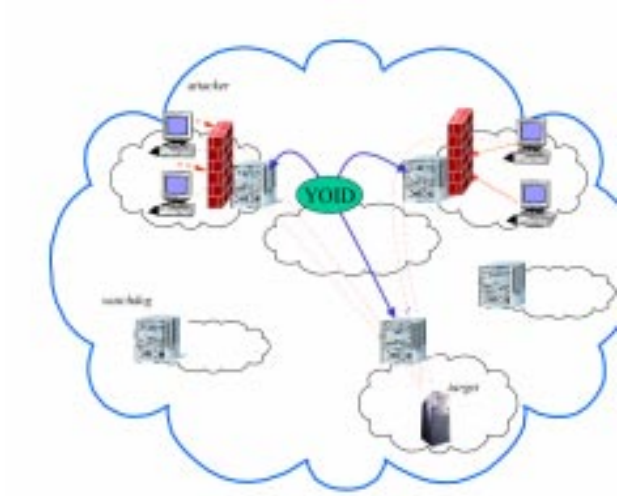
3. Cossack Architecture Overview

The Cossack architecture is shown in Figure 2. The principal element in Cossack is a watchdog, a software subsystem that resides at site egress points. Each watchdog monitors its own network and shares information with other watchdogs. Localized information can be gleamed using a variety of collection tools, such as SNMP statistics, Cisco NetFlow, and IDS tools such as Snort. A watchdog has two principal functions:

- It locally detects the onset of an attack, possibly using an existing intrusion-detection system, but possibly using other, blind techniques.



(a)



(b)

Figure 2: Cossack architecture

- Using a variety of techniques based on coordination with other watchdogs or consulting a topology database or both, increases its confidence in the local detection of the attack, and takes evasive action.

Figure 2(a) shows the onset of an attack and Figure 2(b) its suppression by Cossack watchdogs. In the described scenario, attackers have, over the course of several weeks, acquired access to several hundred or more zombies and have launched an attack by instigating several controllers across the network to instruct the zombies to simultaneously launch an attack a specified target.

When this happens, zombies start simultaneously sending traffic to the target site. The watchdog near the victim notices a preponderance of attack traffic destined towards the target. This is a signal that an attack is in progress. The following sequence of events follow:

- The watchdog instructs the IDS to compile source address information and attack signature data (rates, type of attack, etc.)

- The watchdog multicasts an attack notification to other watchdogs in the network indicating the attacking source networks. It also advertises an attack specific multicast group that will be used for subsequent coordination.

- Watchdogs representing the implicated source networks join the coordination multicast group.

- After receiving attack information hints, each source network watchdog performs in depth analysis of particular outgoing flows to determine if zombies exist within its infrastructure.

- Source networks that identify zombies deploy countermeasures to prevent a continuance of the attack. Local responses will be dictated by a combination of local response policy and the policy information received from the victim side watchdog.

Future work in Cossack will fold in blind detection techniques and vulnerability information gleamed from automated scanning and mapping tools. Armed with these more sophisticated methods, watchdogs will be able to identify and respond to attacks in a more distributed fashion:

- When the watchdog detects this attack, it first checks if the target is vulnerable to this type of attack (possibly based on an annotated topology database which contains system fingerprints of vulnerable sites in the Internet).

- If the target is deemed potentially vulnerable, then the watchdog contacts other nearby watchdogs to determine if they have seen this attack at approximately the same time.

- Consensus algorithms can increase this watchdogs confidence in its detection.

The watchdog may then take evasive action by filtering traffic to the target, thereby eliminating the attack before it has done much damage.

4. Current Approach

4.1 Source Spoofing

At this stage of our research, we do not address the issue of source spoofing. We believe that there are sufficient technical remedies to address the problem. Edge networks can employ egress filtering at their borders, and ISPs can deploy ingress filtering for their customers. Other work addresses the dissemination of network source address information at the BGP routing level [24].

These solutions do not address the issue of host address spoofing within the address space of a given edge network. In our DDoS approach, we are interested in determining the responsible networks that source a particular attack and are less concerned with identifying individual attacking hosts. Coordination with the source network will help in identifying and neutralizing offending hosts.

4.2 DDoS detection in the core of the network

Earlier in the project, we investigated placing monitoring entities inside the core of the network which observed the aggregate flows from source to destination networks. Coordination between monitoring entities was used to determine the total sum of aggregate traffic towards a given destination network. Using this architecture, we attempted to detect DDoS attacks by discovering excessive flow rates towards a given destination network.

This approach has some desirable features, but in the end, the drawbacks forced us to abandon it. Monitors in the core of the network have the ability to observe traffic traversing the network through many different paths using a smaller number of distinct monitoring points. On the other hand, the communication link bandwidths near the core of the network are very large, which hinders the ability to perform any appreciable traffic analysis at line rate. The thorniest problem, however, was how to set the aggregate rate thresholds for a given destination edge network. In the core the network, there is little to no information about the size of the links near the edges of the network. Coarse approximations of edge link size might be gleamed from statistical (average and peak, etc.) rate observations, but the burstiness of network traffic is likely to lead to highly erroneous estimates.

4.3 Edge based detection

Rather than observing traffic in the core of the network, we decided to adopt an approach that involved observing traffic at the egress/ingress point of individual edge networks. Observation of egress edge network traffic is being explored in the D-Ward project [22]. The D-Ward approach of performing localized attack detection at the source edge network shows reasonable promise. But without any coordination among instances of D-Ward agents, the detection process is likely to be error prone and penalize non-attack traffic.

For this and other reasons, we decided to explore the idea of performing the attack detection at the border routers of edge networks. This approach seems promising for a number of reasons. First, the destination network has the most information about the size of the link entering the network and what are reasonable traffic rates based on fixed or historical knowledge. Second, the signal to noise ratio of the DDoS traffic is highest at the destination edge network where all of the traffic has now been aggregated. Third, the destination network can make a more reasonable decision on responses to ongoing attacks by factoring in local policy information.

Once an attack on an edge network has been detected, the next step is to coordinate with other portions of the network to fully determine what type of attack is underway and how to take an appropriate response. A technique that has been explored by the research community involves the idea of performing pushback [21], or following the attack back from the victim's network into the victims providers with the goal of blocking the traffic further upstream of the victim to alleviate the congestion at edge network's link(s). This model has technical merit but suffers from the lack of a reasonable economic incentives for deployment by providers. Providers must perform extra work to trace and block attack flows traversing their infrastructure. In addition, some providers make money off of attacks, since their customers pay based on traffic volume, rather than a fixed price for a given link capacity. Customers are generally unwilling to pay extra to their provider to block attack traffic and the provider sees no incentive to offer the service for free.

4.4 Design Assumptions

A premise of our design is that all or most edge networks have incentives to manage their networks in a responsible manner. Allowing the use of their resources to attack other networks is costly both in terms of resources consumed and in terms of the negative reputation that will be disseminated to the larger network community. Other networks, with offered services, may refuse to service edge networks that host DDoS attacks and show a lack of responsiveness when attacks are reported.

Analogies exist for other network based services. For example, edge networks known to offer open mail relays or host spamming activity are blacklisted by other edge net-

works. Networks check the blacklist and refuse to accept mail from offending edge networks. Similar techniques are used to ostracize poorly or unmanaged networks from participating in net news distribution.

In Cossack, the victim's edge network coordinates with the source networks hosting an attack to provide the information necessary to detect and respond to the attack. We view this exchange of information as merely a hint to a source network that something suspicious is emanating from their network. It is then the responsibility of each edge network to analyze their outgoing traffic and determine when something erroneous is occurring and what response they should take. This is consistent with the distributed management model in the Internet today.

Source edge networks are in the best position to determine what an acceptable policy should be for a host that is engaged in a DDoS. For example, at the USC campus, the network operations group will completely disconnect network service for a host that is determined to be participating in an outgoing attack. Other commercial ISPs may wish to merely block certain services or flows from a given host until the problem is addressed, rather than completely discontinuing service.

4.5 Securing the Watchdogs

The network of watchdogs must be protected from attack itself. There are a number of steps that can be taken to minimize the vulnerabilities of the watchdogs. First, the IDSs and corresponding watchdog software should be deployed on a separate host that will configured with most network services disabled. This will minimize the possibility of external attack of the host by known implementation exploits of common services. Second, the communications between watchdogs should be protected. Ideally, in the large Internet setting, watchdogs should digitally sign their messages sent to other watchdogs in a manner that allows other watchdogs to validate the authenticity of the sending watchdog. Current public CA infrastructures that are deployed in the Internet should suffice for this purpose. Watchdogs for a given edge network, would use that network's registered key pair for signing outgoing messages to other watchdogs. Watchdogs that receive messages from other watchdogs would check that the signature matches the registered public key. In our current prototype of Cossack we will use a shared secret HMAC as a placeholder for the more general public/private key pair solution.

Another issue that must be addressed is how to protect the communications of the watchdogs when the links in and out of an edge network are completely saturated during a DDoS attack. We believe this is where a multicast based communications strategy can really make a difference. Rather than contacting all of the edge networks individually, information is multicasted out to all necessary recipients. The bandwidth required for this operation is minimal and a network could use a low bandwidth connection such as a dialup PSTN link for such a purpose. Alternatively, edge networks could prearrange to provide command and control services for other networks either in a paid or reciprocal fashion. Attack information would be relayed over a low bandwidth connection another edge network that was not under attack. That network would then to responsible for coordination with the necessary source networks to help them pinpoint and stop the attack.

5. Watchdog Architecture

The watchdog is the main analysis, decision, and coordination element in the Cossack architecture. It accepts input from one or more data sensors, analyzes the data, shares information with other watchdogs and makes decisions on how to respond to attacks. The components of the watchdog are shown in Figure 3.

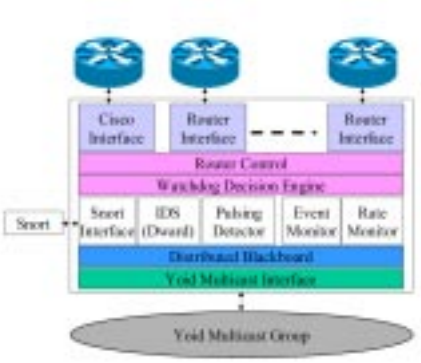


Figure 3: Watchdog architecture

The core of the watchdog is implemented in the Java programming language, which allows for portability to many operating systems. We have experienced no performance problems with the watchdogs, as they deal mostly with high-level events rather than per-packet events, which are handled by the sensors.

In the current implementation watchdogs accept input from one or more snort plugins (described next). Upon startup, each plugin creates a TCP connection with its associated watchdog and begins sending flow statistics. The watchdog

may control the flow of this information by supplying filtering rules to the snort plugin.

The watchdogs currently have an interface to control Cisco routers. Through this interface, the watchdogs may set filtering or blocking rules in response to attacks.

5.1 Snort Plugin

As mentioned earlier, Cossack watchdogs rely on existing IDS to detect attacks. We have selected to use snort [13] for our experiments. Snort has a number of desirable characteristics. It is open source, it has an established user community, and it is actively supported. In addition, there has been significant effort to optimize snort to support traffic capture and analysis at fairly high data rates.

The internal architecture of snort is very modular, and easily accommodates the Cossack extensions. 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. Currently, the rule base is static during a snort execution. Cossack uses the rule database in two ways. First, as a fast prefilter to pass only packets belonging to flows of interest to the Cossack. Here we use snort to break up packets into different protocol groupings to keep statistics on each individual grouping. This information helps the watchdog diagnose an attack, but also allows the Watchdog to define a more specific filter to install in a router to stop the attack.

The second use of the snort database is to define specific patterns that identify individual, possibly malformed packets that attempt to exploit known flaws in software implementations. These packets are directly reported to the watchdog without being aggregated.

The current implementation of the snort plugin is shown in Figure 4. During normal operation, the plugin keeps packet rate statistics for different flows grouped by address prefix. The plugin constructs a prefix tree data structure, which allows for quick aggregation of prefix information. To keep the tree from growing unbounded, the plugin performs periodic garbage collection after state expires or the tree grows beyond a certain size. The plugin supports hundreds of simultaneous packet flows by dynamically building an aggregation tree based on packet rate.

In addition to the destination network prefix tree, the plugin is capable of maintaining a source prefix tree. The source tree is constructed on demand, when instructions are received from the watchdog. The Watchdog issues such commands if after monitoring packet rate reports from Snort it determines that a host may be under attack. The watchdog then asks the plugin to construct the source prefix tree for that destination. The tree is finally reported back to the watchdog, which then contacts the Watchdogs monitoring the source network(s).

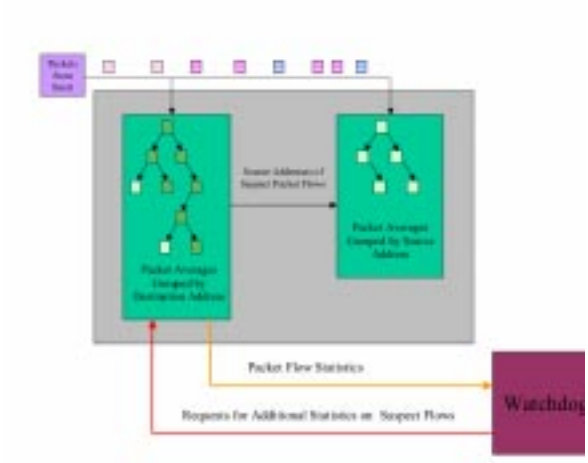


Figure 4: The Cossack snort plugin

6. Understanding Attacks: Detection and Classification

One of the goals of Cossack is rapid detection and classification of attacks. To this goal, we are investigating the existence of invariants during the first few seconds of the attack. Currently, our efforts are focused on the ramp-up behavior and the spectral content of an attack stream.

Determining quickly whether an attack is centralized or distributed is important in planning the defense strategy for such attack. If an attack is centralized, then a single, well-placed filter should suffice to neutralize the attack. If the attack is distributed, then more effort is needed to determine the egress points of the attack and define the proper filters.

In order to analyze and understand attacks better, we have deployed a packet trace machine, which continuously captures packet traces. We then, sift through the traces using a combination of automated scripts and manual examination looking for attacks. We describe this process next.

6.1 Trace Infrastructure

We monitor peering links to Verio and Cogent at Los Nettos, a regional area network in Los Angeles. Los Nettos also

has peering relationships with Genuity and the LA-Metropolitan Area Exchange as shown in the Figure 5. Los Nettos

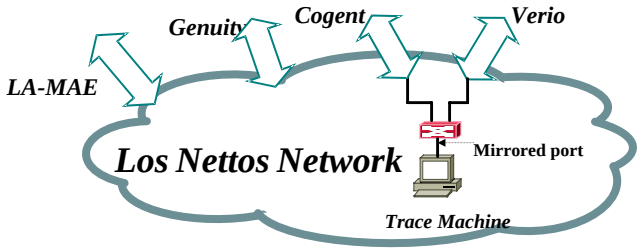


Figure 5: Packet tracing in the Los Nettos network

has a diverse clientele including academic institutions and corporations around the Los Angeles area. The average day time load on the trace machine is 110Mbps at about 38,000 packets per second (pps). The average kernel packet drop rate is below 0.04% during normal operation. During an attack, if packet rates exceed 100,000 pps the drop rate increases to 0.6%. We also observed 11% packet loss at the switch used to mirror traffic to the trace machine. These drops could result in a reduction of the observed attack intensity.

Attack detection consists of continuously capturing packet headers every two minutes using tcpdump and subsequent off-line analysis to determine if an attack is in progress. The detection script flags packets as attack packets if a large number source IP addresses talk to the same destination IP within one second; otherwise the packet headers are deleted. Manual verification is then used to confirm the presence of an attack. We experience a false positives rate of 25--35%; in other words, these packets have been flagged by the detection script but do not contain an attack on manual examination. A large number of false positive are generated due to database updates between servers and network/port scans that result in TCP resets targeted back to the scanner.

6.2 Ramp-up Behavior

In a distributed denial of service attack, the master triggers the attack by sending commands to multiple zombies, which in turn generate the attack traffic. Commands can either be sent *apriori*, instructing the zombies to start the attack at a particular time, or act as a trigger to start the attack immediately. Regardless of the mechanism used, if many zombies are involved there is always some synchronization skew that manifests itself as a slope during the attack ramp-up. The ramp-up behavior is due to the gradual addition of new zombies to the attack tree resulting in an overall increase in the aggregate attack rate seen close to the victim. The duration of the attack ramp up is usually limited by the time difference between the start of the first and the last zombie participating in an attack.

We are investigating this behavior in the attacks we captured in our traces. We observe various types of ramp up behavior, ranging from 200 ms to 14 seconds. Figure 6 shows

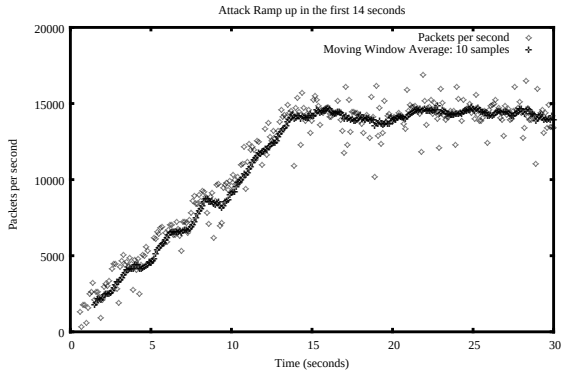


Figure 6: Attack ramp-up

an attack, which went through a 14 second ramp up. An attack such as this one, is most likely a distributed attack, one where perhaps the attack streams are initiated based on local time. Alternatively, the slow ramp-up may be the result of an attack tool attempting to fool an IDS that is looking for sharp increase in traffic. Other attacks exhibit a much quicker ramp-up, sometimes as low as 200 ms. We believe that such attacks were generated by a single machine.

Looking at the attack ramp-up alone is not sufficient to distinguish between centralized and distributed attacks. For this reason, we are currently looking at the spectral characteristics of attack streams, which we describe next.

6.3 Spectral Analysis

We use the frequency information encoded in the attack stream to gain insight into the attack spectral signature and identify the number of attackers. We analyze the frequency spectrum of the attacks and observe distinct differences between what we believe to be centralized and distributed attacks[1].

1.This classification is done manually for now, by looking at header information. We expect this to be done automatically once our algorithms are tuned correctly.

For each attack, we calculate the normalized spectral distribution [20, 23]. The spectrum $S(n)$ of attack obtained by the discrete-time Fourier transform of the autocovariance sequence $r(k)$ and is given by:

$$S(n) = \sum_{k=-\infty}^{\infty} r(k) \cdot e^{(-ink)}$$

The autocovariance of an attack stream is a measure of how similar the attack is to itself at various time lags k. When $k = 0$ we compare the attack signal to itself and the autocovariance is maximum and equal to the variance of the stream. When $k > 0$ we compare the attack stream with a shifted version of itself. The autocovariance at lag k is calculated using:

$$r(k)) = \sum_{t=0}^{\infty} (x(t) - \bar{x}) \cdot (x(t+k) - \bar{x})$$

where $\bar{x}$ is the expected value of the attack stream $x(t)$.

The normalized spectral distribution $F(n)$ is calculated by integrating the spectrum and normalizing it as given by the following equations:.

$$f(n) = \sum_{i=-\infty}^{\infty} (S(i+1) - S(i))/2$$

$$F(n) = (f(n))/(max(f(n)))$$

The normalized spectral distribution function gives the distribution of the amplitude at different frequencies in the spectrum allowing us to compare the frequency spectrum of different attacks. The attack stream $x(t)$, $0 \le t \le 30$ seconds is an aggregate of the attack packets observed in 1ms. Hence highest frequency observable in the spectrum is 1000Hz. The Fourier transform is symmetric about 500Hz, hence we plot only the first half of the spectrum.

In all centralized attacks, we observe the normalized cumulative spectrum to be linear as shown in Figure 7. The spectrum indicates that there are no frequencies that dominate and the amplitude is evenly distributed over all the frequencies. We found that 60% of the amplitude energy is located above 320Hz and lacks discontinuity. We believe that the even distribution of the amplitude across all frequencies is due to the interaction of the attack tool with the resources on the host machine. The attack tool generates packets constantly and is limited only by the available computing power or connection bandwidth. The high frequencies are the result of rapid packet generation while lower frequencies are due sharing of resources and scheduling on the host.

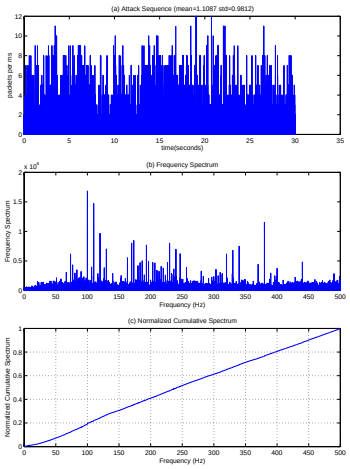


Figure 7: Frequency spectrum of a centralized attack

Figure 8 is an example of a reflected attack spectrum. In reflected attacks, the lower frequencies dominate the spectrum and account for most of the amplitude energy in the normalized cumulative spectrum. In distributed attacks, 60% of the energy and discontinuity in the spectrum are located below 200Hz. We believe the lower frequencies are generated due to the zombie cycling through a battery of reflectors in a round-robin fashion resulting in the amplitude energy concentrated at the lower frequencies or the use of rate lim-

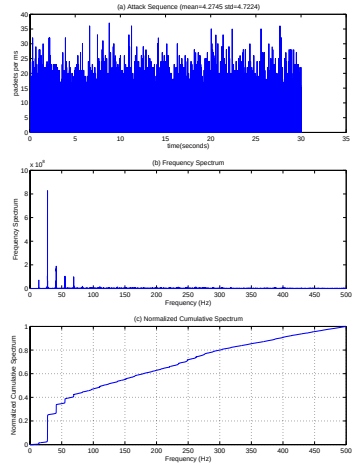


Figure 8: Frequency spectrum of a distributed attack

iting mechanism, such as suspending packet generation for a few milliseconds, used by some attack tools.

7. Related Work

The D-Ward system [22] monitors outgoing traffic from a given source network and attempts to determine outgoing attack traffic. Attack traffic is identified by comparing the traffic patterns against models of reasonable congestion control behavior. For example, TCP traffic is monitored and compared to an equational approximation of the TCP congestion control model. TCP streams that are observed violating the behavior of the model is marked as an attack and is subsequently throttled back by the edge network's egress router. The amount of throttling is proportional to the flows deviation from it's expected behavior. In a similar fashion, the same approach can be applied to other transport protocols as long as some measurement information is available to take the place of the TCP ACK traffic that is observed at the source network's border router. The health of destination hosts can be gleamed using ICMP echo/reply probes or other techniques that generate the necessary 2-way traffic needed to analysis the compliance of a given flow to reasonable congestion control behavior.

Another coordination approach that has been explored is traceback [19]. In SPIE [26], state is stored in the network for a short period of time that enables edge networks to trace back the origin of a given packet. A query mechanism traces back the into the network looking for evidence of a packet traversing particular routers. A probabilistic match algorithm follows back a small number of possible paths until the correct path is determined.

Recent efforts on neutralizing DDoS attacks have focused on attribution via IP traceback [7, 9, 10, 16]. The immediate goal is to locate the hosts the attack originates. Traceback also offers the hope of locating the attacker through the instruments of the attack. Traceback schemes can be divided in two categories: (a) *probabilistic packet marking (PPM)*, and (b) *tunnelling* techniques. While PPM techniques work well for single-source attacks, they are woefully inadequate for large DDoS attacks. The main reason, as argued in [8], is that there exists a trade-off between localization and marking probability, path length and traffic volume. A similar view is presented in [11]. Tunneling [17] techniques reduce the number of hops for selected packet streams and thus make traceback easier. Tunneling techniques require the ability to dynamically set up tunnels between any access points and thus require substantial support from the network. They also suffer from the same limitations as PPM techniques.

Intrusion Detection Systems (IDS) and Anomaly Detection Systems (ADS) act as tripwires during an attack. Current IDS [13,14,15,12] rely on fast pattern matching to detect a signature of an attack. However, this implies that the signature must be known *a priori* meaning that IDS is powerless against *new* or even slightly *mutated* attacks. ADS (see [18] and its references) monitor networks and learn what constitutes "normal" network traffic by developing models which are updated over time. These models are applied against new traffic and if a mismatch occurs, the new traffic is flagged as "suspicious". While conceptually attractive, ADS systems require substantial training, and an intruder can still defeat them by introducing attack traffic gradually. Moreover, modeling normal traffic has proven very difficult in practice, as networks get large and the application mix becomes complex.

8. Cossack Demonstration

We demonstrated a prototype of Cossack at the DARPA Fault Tolerant Networks (FTN) PI meeting in July 2002. In this demonstration, we a showed how the coordination in the Cossack architecture could be harnessed to identify and respond to a low level pulsing attack being generated by 100 zombies. Each zombie had a sending rate of 100 packets per second. It transmitted for approximately 5 seconds and then went silent for 25 seconds, maintaining a duty cycle of about 17%.

Given the low rate of each zombie, it would be difficult. even for a diligent operator, to detect this pulsing behavior for a single flow originating at a particular network. At the target site, however, after all flows have aggregated, detection becomes relatively easy. Once the target watchdog detects the aggregated attack, it notifies the watchdogs at the source networks, which take action.

We prototyped a pulse attack detector by analyzing the average rate of a given flow over a series of time scales ranging from 5 to 30 seconds. If the ratio of the largest to smallest average exceeded a preset threshold, the detector classified the flow as a pulsing attack. This simple technique will eventually be replaced by a more robust FFT analysis of the suspect flow.

The watchdogs looking at the outgoing traffic from the attack machines do not initially see anything out of the ordinary. However, the victim quickly sees the aggregation of all attack traffic and identifies the attack.

To simplify the size of the testbed required for this demonstration, we aggregated 98 of the attack zombies onto one host. This allowed us to emulate the attack of 100 zombies

by using just three hosts. The test bed consisted of 7 PCs running GNU/Linux and 4 Cisco routers. The network was setup in a tree topology with a victim PC at the root, the Cisco routers in the center and 3 attack machines at the leaves. Every machine was on its own separate subnet and a watchdog PC was configured to monitor each subnet.

The demonstration scenario is shown in Figure 9. Two

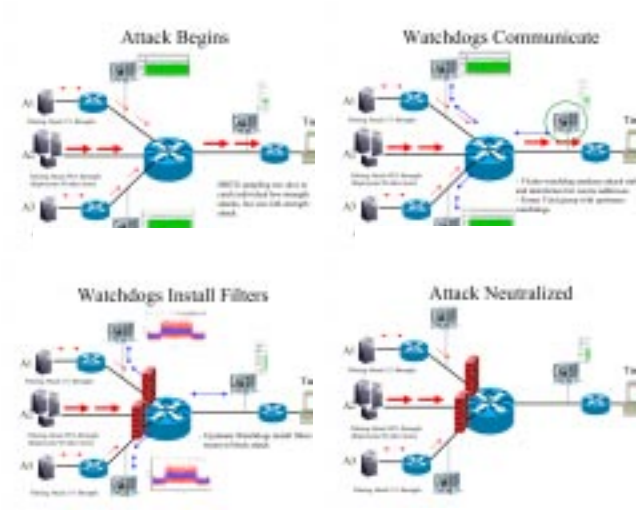


Figure 9: Cossack demonstration

attack machines were configured to send low rate pulsing attacks toward the victim. Each of these machines represents a typical DDOS zombie and outputs a small percentage of the total attack traffic. The third attack machine was used as a trigger mechanism for the victim watchdog. This machine represented the 98 out of 100 other hosts that are also sending low level attack traffic. Only the victim machine and victim watchdog actually sees the trigger attack traffic. All the watchdogs are constantly analyzing the network traffic going in and out of their subnets.

Once the attack traffic trigger starts, the victim watchdog looks at the high rate of traffic going towards the victim and asks snort to builds a source address tree of packets destined for the victim. After receiving this tree from snort the watchdog makes an announcement to other watchdogs multicasting the networks that may have machines participating in the attack and invites them to join a multicast group designated for that attack.

The low level packet rate attack networks' watchdogs joined this group and exchanged packet rate information with each other and with the victim watchdog. Each watchdog with an attack network then analyzed its outgoing traffic to find the offending machine and installed a packet filter in its network's Cisco router to block the attack traffic from reaching the victim. With the filters in place the victim watchdog doesn't see any attack traffic so it closes down the multicast communication group. Although the communication channel was closed, each of the attack network watchdogs independently continued to monitor its network traffic for attack traffic and once that traffic stopped they logged into their router and removed the packet filter.

9. Release

Public releases of our software is available at http://www.isi.edu/cossack/. Source is available for both the watchdog and the Snort plugin module. The programmable flow analysis module for Snort can be used independently of the watchdog and may be of interest to researchers and network operators for other purposes.

10. Future Work

The weakest part of the current Cossack prototype is the source edge network attack detection methods. When a source network receives information about a potential attack, it needs to monitor all outgoing traffic towards that host to determine if any attack flows are emanating from the source network. Currently we have fairly crude methods to analyze outgoing flows. We have a simple detector for pulsing attacks and full rate "blasting" attacks.

We would like to incorporate source end sensors that are not signature specific. We are investigating techniques that perform a spectral analysis of the outgoing flow in hopes of determining distinguishing characteristics. The D-Ward project has developed a detector for determining whether or not TCP flows are behaving in a manner consistent with equational models of congestion control.

We need to have better filtering methods at the victim end to limit the number of source networks that need to contacted about an attack. For certain network servers, the number of source networks that are requesting services at any given time may be very large. For example, a web server may be serving 10,000 edge networks while being attacked by only 100 zombies. Once we identify the source networks at the victim's end, we would ideally like to filter the list down to contain only those source networks that are involved in the attack. In practice, it would be sufficient to be able to filter out most of networks not involved in the attack. Since the information sent towards a source network is only a hint, we can accept some amount of false positive results.

11. Conclusions

DDoS attacks remain an elusive threat to the Internet. Attacks are increasing both in size and frequency of occurrence. We believe that without coordination, networks are going to remain ineffective at combating sophisticated DDoS attacks.

The Cossack architecture addresses the DDoS detection and response problems using a highly distributed architecture. This architecture combines multicast communications, traditional IDS systems, network topology, vulnerability information, and novel blind detection techniques into a powerful combination that should prove to be effective against a wide variety of DDoS attacks.

Our recent demonstration of Cossack, at the previous DARPA FTN PI meeting, showed how the concepts embodied in our approach, were able to detect and block a low rate pulsing attack emanating from many zombies. This type of attack generally eludes strictly localized detection methods. We have a long way to go until we have a complete solution to the DDoS problem, but we feel that the Cossack approach is heading in the right direction.

12. References

[1] http://www.cert.org/stats/cert_stats.html
[2] http://staff.washington.edu/dittrich/misc/ddos/timeline.html
[3] http://www.nipc.gov/warnings/advisories/2000/00-063.htm
[4] http://www.nipc.gov/warnings/advisories/2000/00-044.htm
[5] http://www.nipc.gov/warnings/advisories/2000.htm
[6] http://www.nipc.gov/cybernotes/2000/cyberissue2000-26.pdf
[7] Bellovin, "ICMP Traceback Messages", draft-belovin-itrace-00.txt, work in progress (expired Sept. 2000).
[8] Park, K., Lee, H., "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack", Technical Report CSD-TR 00-013, Purdue University, June 2000.
[9] Savage, S., Weatherall, D., Karlin, A., Anderson, T., "Practical Network Support for IP Traceback", Proceedings of Sigcomm 2000.
[10] Song, D., Adrian, P., "Advanced and Authenticated Marking Schemes for IP Traceback", Technical Report No. UCB/CSD-00-1107, University of California at Berkeley, June 2000.
[11] http://www.nanog.org/mtg-0006/savage.html.
[12] http://packetstorm.securify.com/UNIX/IDS/
[13] http://www.snort.org/
[14] http://www.nfr.com/
[15] Cisco Secure IDS: http://www.cisco.com/univercd/cc/td/doc/pcat/nerg.htm
[16] http://www.darpa.mil/ito/psum2000/J910-0.html
[17] Stone, R., "CenterTrack: An IP Overlay Network for Tracking DoS Floods." Presentation at NANOG17, October 1999. http://www.nanog.org/mtg-9910/robert.html.
[18] Ghosh, A. Schwartzbard, A., "A Study in Using Neural Networks for Anomaly and Misuse Detection," in Proc. of Usenix99, Washington DC. August 1999. http://www.usenix.org/publications/library/proceedings/sec99/full_papers/ghosh/ghosh_html/.
[19] Steven Bellovin. ICMP traceback messages. Internet Drafts: draft-bellovinitrace-00.txt.
[20] Chris Chatfield. The Analysis of Time Series: An Introduction. Chapman and Hall texts in Statistical science series. Chapman & Hall, University of Bath, UK, 1996.
[21] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In Proceedings of Network and Distributed System Security Symposium, San Diego, CA, February 2002. The Internet Society.
[22] Peter Reiher Jelena Mirkovic, Greg Prier. Attacking DDoS at the source. In 10th IEEE International Conference on Network Protocols, Paris, France, November 2002.
[23] Petre Stoica and Randolph Moses. Introduction to Spectral Analysis. Prentice-Hall, Upper Saddle River, NJ, 1997.
[24] Park, K., Lee, H., On the Effectiveness of Route-based Packet Filtering for Distributed DoS Attack Prevention in Power-law Internets, in Proceeding of the ACM SIGCOMM'01, pages 15--26, 2001.
[25] Francis, P., Pryadkin, Y., Radoslavov, P., Govindan, R., Lindell, B., Yoid: Your Own Internet Distribution. Work in Progress, http://www.isi.edu/div7/yoid/docs/index.html.
[26] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, W. Timothy Strayer. Hash-Based IP Traceback. Proc of ACM Sigcomm 2001. San Diego, CA 2001.