

# A Hybrid Honeypot Architecture for Scalable Network Monitoring

Michael Bailey, Evan Cooke, David Watson, Farnam Jahanian  
University of Michigan  
*{mibailey, emcooke, dwatson, farnam}@eecs.umich.edu*

Niels Provos  
Google Inc  
*niels@google.com*

October 27, 2004

## **Abstract**

To provide scalable, early warning and analysis of new Internet threats like worms or automated attacks, we propose a globally distributed, hybrid monitoring architecture that can capture and analyze new vulnerabilities and exploits as they occur. To achieve this, our architecture increases the exposure of high-interaction honeypots to these threats by employing low-interaction honeypots as frontend content filters. Host-based techniques capture relevant details such as packet payload of attacks while network monitoring provides wide coverage for quick detection and assessment. To reduce the load of the backends, we filter prevalent content at the network frontends and use a novel handoff mechanism to enable interactions between network and host components. We use measurements from live networks over five months to demonstrate the effectiveness of content prevalence as a filtering mechanism. Combining these observations with laboratory measurements, we demonstrate that our hybrid architecture is effective in preserving the detail of a specific threat while still achieving performance and scalability. We illustrate the benefits of this framework by showing how it enables earlier, higher-confidence detection, more detailed forensics, and robust signatures for mitigation of threats.

# 1 Introduction

Network assets are increasingly vulnerable to rapidly moving threats of today's Internet. Automated attacks have impact beyond the scope of the individual host and enterprise, not only infecting vulnerable hosts with malicious code but also denying service to legitimate users of the network. The prime example of such threats can be found in Internet worms. Their unique properties blur traditional community distinctions and make detection, characterization, and quarantining problematic. A range of different communities with disparate views of global events as well as historically different goals and objectives has proposed honeypot-based solutions to measure, track and understand these Internet-wide threats [5, 18, 26, 27, 35].

The host-based honeypot community has focused their attention on understanding the techniques used to attack a system and the motivation of the individuals perpetrating the attack [35]. Their tools traditionally consist of actual hosts running actual services on actual operating systems, and careful monitoring of the resultant interactions. Complex mechanisms for reducing risk, monitoring, and alerting have evolved, but the goal has remained constant. On the other hand, the network-based sensor community has primarily focused on broad characterization of threats through traffic monitoring. This is particularly effective in exposing threats that adversely impact availability such as congestion created by worm propagation and routing instability. While the motivations and techniques are of some interest (e.g. an attack creates more DDoS zombies) this community is primarily interested in attacks that are able to bring down a network or one of their customers'. This community has traditionally approached sensing by using routing infrastructure to route known bad packets (e.g. Bogons) to infrastructure that reports broad trends in traffic amounts and type.

The recent prevalence of Internet worms, however, has started to blur the distinction between the various community efforts. In particular, the properties of worms pose significant challenges to each community's effort to detect, characterize, and quarantine them using their existing tools and measurement locations. First and foremost, worms are globally scoped, respecting no geographic or topological boundaries. In addition, worms can be exceptionally virulent and can propagate to the entire population of susceptible hosts in a matter of minutes [21]. Their virulence is extremely resource taxing and creates side effects that pose problems even for those that are outside the vulnerable population. To make matters worse, worms have the potential to be zero-day threats, exploiting vulnerabilities for which no signature or patch have been developed.

As a result, current techniques do not provide sufficiently early or comprehensive intelligence about these attacks. The scope of existing host-based techniques is too small to capture useful information such as the size of the infected population or provide warning early in the growth phase. On the other hand, network sensors do not interact sufficiently with the worm and lack enough information to characterize the vulnerability exploited or the effect on local machines. Therefore, system defenders are often left scrambling to respond to an attack after significant damage has already occurred, or left trying to understand and counter an attack with only isolated bits and pieces of the attack behavior. To be effective at assuring the availability, confidentiality, and integrity of Internet resources, it is necessary to combine information from disparate network resources each with differing levels of abstraction into one unified view of a threat.

In this paper, we examine honeypots in network- and host-based deployments and discuss their respective tradeoffs such as scalability and fidelity. We argue that a hybrid architecture that uses low-cost, network-based honeypots as filters to reduce the load of higher cost, high-interaction honeypots is an attractive way to balance the tradeoffs and address these challenging problems.

Our architecture combines multiple tiers of sensors into a distributed indication and warning system that can provide intelligence on brewing threats in a network. In this hybrid system, honeypot sensors operating at network and host layers are combined with intelligent filtering to enable network defenders to quickly assess and react to new threats. The architecture is unique in several ways:

- **Hybrid sensing framework.** Designed to achieve the scale of low-interaction honeypots and the interactivity of high-interaction honeypots without the corresponding drawbacks, this architecture is unique in that it proposes integrating network sensors and host based honeypots.
- **Evaluation of content prevalence as a filtering mechanism.** While other means of dealing with the massive quantity of data seen in wide address measurement have been explored (e.g. source-destination filtering), we show that content filtering is an effective way of achieving accurate characterization without incurring the cost of monitoring a large number of individual hosts.

Minimum Interaction	Example Tool	Example Threat
Capture connection attempt	Caida Network Telescope[19]	Sapphire[20], Code-Red[32], Witty[31]
Response on 1 or more ports	IMS[7]	Blaster[24]
Application Response	Honeyd[26]	Sasser[8]
End host behavior	Live Hosts[36][17]	Slapper[1]

Table 1: Honeypot interactivity level required to capture remote exploit. Increasing levels of interactivity are shown along with tools which capture that level of interactivity and exploits whose relevant features appear only at that level of interaction or greater.

- **Enabling novel techniques for Detection, Forensics, and Mitigation.** We discuss the application of this hybrid framework and show how it enables several unique threat monitoring capabilities.

The remainder of this paper is organized as follows. Section 2 introduces several of the key tradeoffs of honeypot detection methods and highlights the advantages associated with each. To balance the tradeoff between interactivity and scope in particular, we introduce a hybrid architecture in Section 3. It uses distributed low cost lightweight network sensors combined with an intelligent filtering mechanism to guide the interaction with higher cost high-interaction honeypots. A central control component aggregates measurements and provides a feedback loop between the high-interaction backends and the low-interaction filters based on load and performance information. Section 4 highlights the advantages of this approach by showing how the unique characteristics of the hybrid architecture enable new capabilities. We discuss related work in Section 5 and conclude in Section 6.

## 2 Understanding Interactivity and Coverage

In this section, we discuss the tradeoffs in design between high-interaction and low-interaction honeypots. In particular, we focus on the tradeoffs associated with the behavioral fidelity of a system, and its ability to provide breadth of coverage.

### 2.1 Behavioral Fidelity

Behavioral fidelity refers to the degree to which any infrastructure is able to emulate the interactions and behavior of an end host node. Higher fidelity implies a closer approximation of end host behavior. Spitzner et al. employ a similar notion of interactivity to help identify the means and motivations of individual hackers [35, 36]. In the context of a global detection and warning system, behavioral fidelity is important as various degrees of emulation are required to capture and differentiate different types of worms.

Table 1 provides an illustrative view of the behavioral fidelity levels and their impact. The first column shows the level of interaction. The second column provides an example system which provides this level of fidelity (in general, higher fidelity provides lower levels of interaction as well). The final column shows a captured threat that required at least that level of interactivity to differentiate the threat from other threats. The first row shows threats that can be caught simply by capturing the connection attempt. These threats act on a port that has no legitimate traffic or contain headers that are uniquely identifiable. This may also include UDP and ICMP threats in which the payloads or exploits occur in one packet. The second row shows threats that require a system to respond on a port. These are TCP threats that require session construction to illicit the first message, or threats that OS fingerprint by looking for active ports. The next higher level of interactivity are those threats that require application response. These threats require specific responses due to application semantics or limited scope of vulnerability (e.g. checking for a specific version of Apache before attempting sending an exploit). Finally, we have threats that require full host behavior, such as those threats that attack previous threats or threats for which the created backdoors and behaviors are more important to detect than the threat propagation itself.

Regardless of the chosen level of emulation, it is clear that there will always remain some threat which requires a closer approximation of end-host behavior. Therefore, to be comprehensive, a system must include actual hosts as part of its monitoring infrastructure.

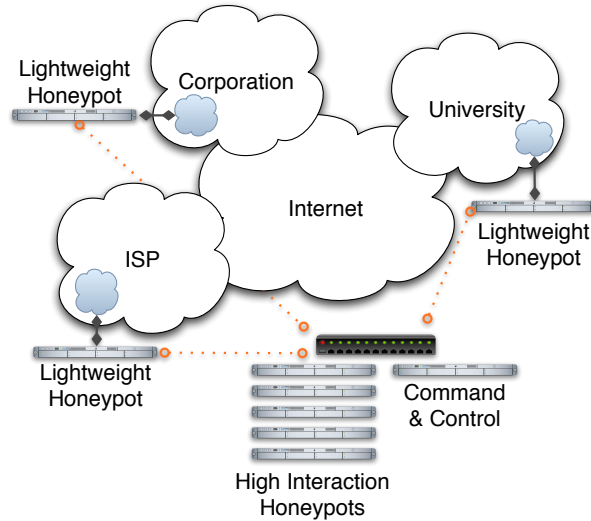


Figure 1: A hybrid architecture consisting of three components: lightweight virtual honeypot frontends, high-interaction backends and a control component. The virtual honeypot frontends are responsible for answering traffic while forwarding interesting traffic only to the backends which can be fully compromised. The control component monitors the load of the backends and can dynamically adjust the behavior of the frontends.

## 2.2 Breadth of Coverage

Coverage refers to the ability of an infrastructure to effectively capture globally scoped events. Moore has shown that increased monitor size yields increased visibility in terms of time to detection and infected population size [22]. Unfortunately, IPv4 space is limited, and there are a small number of wide address blocks available for instrumentation. Therefore, increasing coverage requires combining available smaller sized blocks. While it is certainly the case that increasing the size of monitored space in this way will increase visibility, it may also be the case that true coverage can only be achieved by examining large blocks and collections of disparate blocks. It has been shown that address blocks in different networks see different threat traffic [6]. This effect may be the result of distribution of the affected population, the target selection function, statistical variance, security policy, or the underlying physical topology.

Increased monitoring size decreases the time to detect the global onset of a threat. Different, topologically diverse blocks see different views of these threats. Because of these two factors, we believe that effective coverage is achieved through a distributed collection of sensors that monitor as much address space as possible in as many diverse locations as possible. This notion is often at odds with behavioral fidelity in that scaling high-interaction honeypots to tens of millions of hosts is problematic at best. To be effective, one must balance these two goals.

## 3 Architecture

To provide both behavioral fidelity and breadth of coverage, we propose an architecture that is highly scalable but still delivers very accurate detection. We know that lightweight virtual honeypots can instrument a large address space effectively. On the other hand, we know that they do not support full behavioral fidelity, e.g. a new threat may fail to be accurately captured by a low-interaction honeypot. This is almost the reverse for high-interaction honeypots: a new threat can successfully interact with such a machine, but the high-interaction system does not have optimal performance and would not scale to a large address space.

We would like to take advantage of the scalability provided by the low-interaction virtual honeypots while still being able to provide detailed behavioral analysis. We achieve this by combining the scalability and interactivity of different honeypot technologies through a hybrid system as shown in Figure 1. This hybrid architecture consists of three components:

1. **Lightweight Honeypots.** Scalable low-interaction honeypots are widely deployed and monitor numerous variable sized unused network blocks. These boxes filter requests to a set of centralized, high-interaction honeypots.
2. **High Interaction Honeypots.** A collection of centralized VMware [?] (virtual machine) hosts running a wide variety of host operating systems and applications. Filtered traffic from the lightweight sensors are directed here for behavioral analysis.
3. **Command and Control.** A centralized mechanism provides a unified view of the collection infrastructure including overall traffic rates, source and destination distributions, as well as global payload cache state. This mechanism coordinates connection redirection across sensors. It also monitors backend performance and provides feedback to the frontend redirection mechanisms.

All of our address space is instrumented by low-interaction honeypots. The role of these honeypots is to filter the uninteresting traffic by answering it locally. Interesting traffic is forwarded to a set of backends that constitutes our high-interaction honeypots. We can classify uninteresting traffic roughly as all packets that do not belong to an existing connection, SYN packets that do not lead to an established three-way handshake, payloads that we have seen many times before. In Section 3.1, we show that there is significant repetition in the first payload carrying packet of many remote exploits because these exploits are repeatedly targeting particular vulnerabilities or services. While this mechanism can not distinguish between all threats, it serves as an effective mechanism for alerting on new threats. To monitor for the occurrence of new threats that have the same first payload, we support sampled connection handoff, periodically sending known signatures to the backend. This sampling is dynamic relying on a feedback mechanism from the profiled backend honeypots.

Our backends run VMware to provide several high-interaction systems per physical machine. Their network setup prevents them from contacting the outside world. However, instead of blocking outgoing connections, we redirect them back to the heavyweight honeypots. If a worm was to infect one of the machines, it would successfully infect more susceptible backends as long as uninfected backends are available. This mechanism is unique in that it allows us to watch actual worm propagation, including exploit as well as payload delivery in a controlled environment. Once this type of propagation is detected, the checkpointing features of VMware will not only allow us to save these infected states and corresponding forensic information, but also return the machines to a known good state for further use in analysis. To recognize when a backend has reached an abnormal state, we monitor their network connections and may also analyze changes to the file system.

The final piece of the architecture is a control component. The control component aggregates traffic statistics from the lower-interaction frontends and monitors the load and other data from the backends. The control component is responsible for analyzing all received data for abnormal behavior such as a new worm propagating. We achieve this by two separate analysis phases that we combine at the end. One phase analyzes the network data of the frontends for statistics that relate to worm propagation: e.g. increases in source IP addresses for a certain port, increases in scanned destination IP addresses for a certain port, and all of this correlated with payload data. The other phase keeps track of the behavior of the backends. If we notice that the backends initiate unusual network connections that is normally a sign of worm infection. These unusual network connections and other forensic information can be stored when abnormal behavior has been detected.

### 3.1 Integration of Sensors

To achieve scale for the backend heavyweight honeypots, we deploy a distributed collection of lightweight sensors which filter connections to the backend. There are two key components of this distributed lightweight filtering mechanism. A decision regarding when to handoff a connection, and a mechanism for performing that handoff.

The key notion of filtering is to capture the relevant details of every threat. An idealized mechanism achieves scale by only forwarding one instance of each unique threat to the backend honeypot. While any identification of unique threats is imperfect, several techniques from worm detection are applicable. Simple traffic methods use models of normal network behavior to determine when traffic is different from normal. Content prevalence approaches analyze the distribution of content sequences in payloads to create distributions of content, alerting when a specific piece of content becomes widely prevalent. All of these mechanisms are useful in identifying interesting traffic to redirect to the backend. In this paper, we examine lightweight content prevalence to decide when to handoff a connection.

A potential drawback of content prevalence as a mechanism for filtering interactions to the heavyweight honeypot is that we will be unable to determine if a specific connection should be redirected to the backend honeypots until

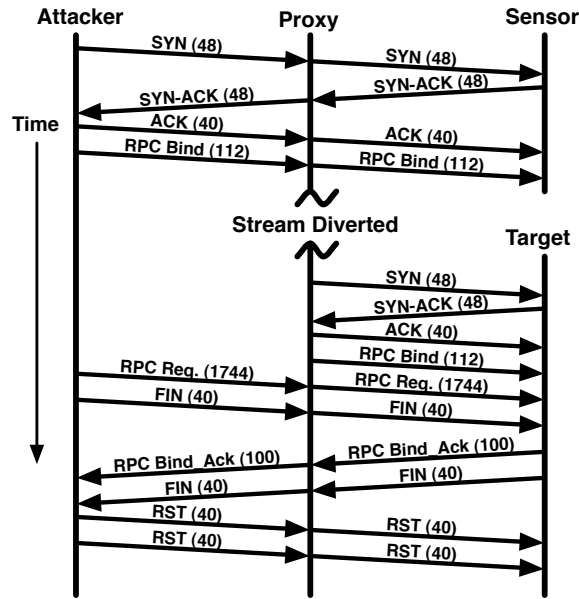


Figure 2: The graph shows integration of the lightweight sensor and the connection handoff mechanism as seen from during the Blaster worm.

after a session has been established and content or payload has been received. One solution to this problem is to store sufficient incoming connection information in order to replay the session after a decision has been made. While this is tractable for small address spaces, a cheaper solution is to only store state for handed off connections. In order to avoid saving state for every incoming connection, we will make the content prevalence decision based off the first payload packet of any conversation. This requires us to wait until the first payload packet (typically an ACK-PUSH) and use the data in that packet to make the handoff decision. While we must now store state for each connection that is handed off (because we need to rewrite sequence numbers as in [2]), this will cut down on the pace needed.

Figure 2 provides an example of this handoff mechanism as seen in the case of a Blaster worm infection attempt. Four components play a role in this handoff. The attacker who is attempting to exploit a vulnerability, a proxy which decides when to handoff the connection and manages the storing of any state or rewriting of packets, a sensor which characterizes connection attempts and stores payload signatures, and finally the high-interaction honeypot, labeled “target.” In the example, the attacker begins by initiating a TCP connection to port 135. The proxy allows these connection attempts to pass through to the sensor which ACKs these requests to elicit the payload. The attacker then ACKs the SYN-ACK and immediately follows with a packet containing the application level RPC Bind packet. This first packet with a payload is intercepted by the proxy, and the payload is checksummed. If the payload is a new payload, a decision to divert this connection to the backend is made. This decision may be made locally, or through coordination with the centralized mechanism which has a global view of the signature cache. A forwarding mechanism is installed and all other packets from that source are diverted to the target. This is important not only for completing the remainder of the existing session, but also to proxy all other connections from the same source. This allows for threats that utilize multiple connections to complete the infection such as the Blaster worm. Note that because we wish to have consistent data, the lightweight sensor also receives a copy of the RPC Bind payload. Once the decision to divert the stream is made, the proxy must establish a connection with the backend using the sequence number information from the RPC Bind. It establishes a connection to the backend host and replays the RPC Bind. Note that when the proxy sends the SYN, the target has a sequence number that is unknown to the attacker, and thus the proxy must store state for this proxied connection and must rewrite packets to reflect these new sequence numbers. With this connection established and the proxy rewriting packets, the connection proceeds as normal. The forwarding mechanism divert rule remains in place for that source address for a timeout period so that, as is the case with Blaster, a follow up connection to another port is possible.

In this section, we discussed several mechanisms for deciding when to perform a connection handoff and dis-

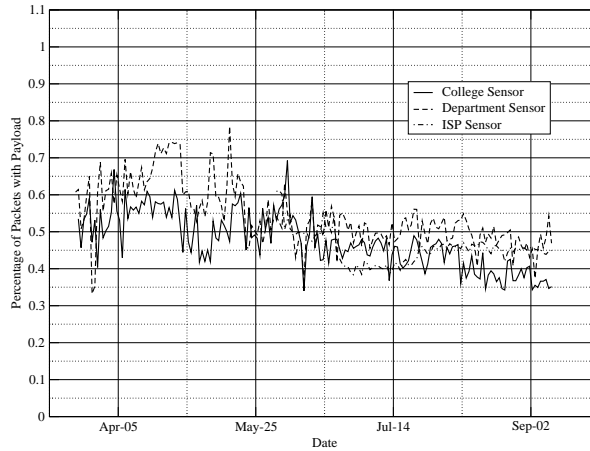


Figure 3: The percentage of packets with payloads as observed by three sensors over a period of six months. Roughly 50 percent of the traffic does not contain a payload.

cussed a specific mechanism, namely content prevalence. We showed how the need to wait for payload packets complicates connection handoff and demonstrated a specific mechanism for accomplishing that task. In the next section, we show the effectiveness of content prevalence in filtering interactions between the lightweight and heavyweight honeypots.

### 3.2 Evaluating the Effectiveness of Content Prevalence

A key to understanding the effectiveness of content prevalence as a filtering mechanism is the observation that many packets observed by a sensor do not contain a payload. These packets may be the result of a scan, backscatter, or part of the three way handshake. Figure 3 shows the percentage of packets with payloads as observed by three sensors over a period of six months. Roughly 50% of the traffic does not contain a payload. Each of the three sensors observes approximately this same percentage of traffic.

One interesting observation made in earlier work is that the contents of the first payload packets are seen many times across blackhole sensors [6]. In fact, roughly 95% of these payloads have been seen before at the same sensor. Figure 4 shows the percentage of payloads seen before at the same sensor during the same day as observed by three sensors over a period of six months. The small number of drops in the rate down to 70% are the result of dictionary attacks where each payload contains a different password from a large dictionary.

A question arises as to whether or not the payloads seen at one sensor are local to that sensor, or are seen globally. Figure 5 shows the percentage of payloads that are seen at 0, 1, 2, or 3 sensors over a 5 month period. In the first month and a half, only two sensors are active. In the two sensor case, less than five percent of the packets are cache misses, 35% are cache hits, but only seen at one sensor, and 60% of all payloads are seen at both sensors. When a third sensor is brought online we see several months of packets in which over 90% of the packets are seen by all three sensors. This discrepancy is most likely the result of the large packet rate seen by the third sensor whose distribution of payloads is mostly global in nature.

In this section, we have shown the effectiveness of content prevalence in reducing the candidate set of packets to be considered for backend handoff. By observing content as seen at three sensors over a five month period we have shown both the percentage of packets that contain no payload, as well as the local and global cache payload rates combined to achieve a significant reduction in the number of connections required by the backend. Experimental numbers show that only 0.25% of packets need to be sent to the backend.

### 3.3 Estimating System Performance

In order to understand the impact of content prevalence as a filtering mechanism we looked at the workload for each sensor. Figure 6 shows the number of packets seen each day across three sensors for a period of 5 months. It is

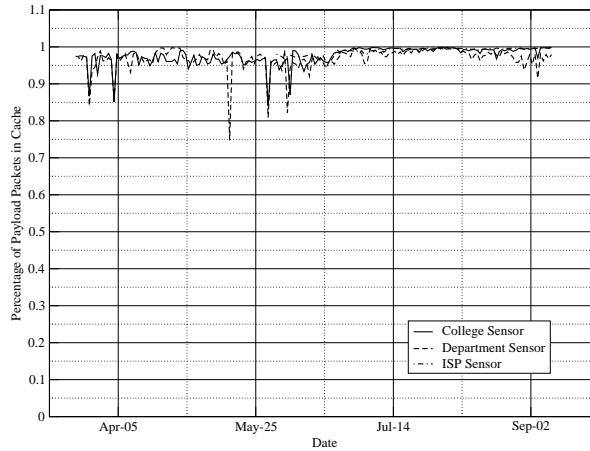


Figure 4: The percentage of payload cache hits as observed by 3 sensors over a period of six months. Roughly ninety five percent of the packets with payloads have been seen before.

interesting to note the variations across sensors, both in absolute magnitude and in representation of specific events. The collection of three sensors observes on average  $7.5e+06$  packets per day with spikes up to  $2.5e+07$  packets. This equates to roughly 87 pps across the three /24 sensors. Applying the observations from the previous section we can calculate the the load generated on the backend after filtering based on content prevalence.

$$\begin{aligned}
 & \text{Connection rate} \times \% \text{ packets w/payload} \\
 & \quad \times \% \text{ payloads new} \times \% \text{ globally new} = \\
 & 87 \times .5 \times .05 \times .10 = \\
 & .217 \text{ CPS handed off to the back end}
 \end{aligned}$$

This is a drastic reduction in the overall number of connections the backend must see. As we will show, easily within the capabilities of both the lightweight and heavyweight honeypots.

Each of the various components of the system must also be able to support the corresponding rates at each level in the filter hierarchy. In order to validate their ability to scale, we ran several laboratory experiments. Figure 7 shows the connection per second rate that a typical lightweight honeypot can support. In this case, Honeyd [26] is used as the lightweight mechanism. The system, which is emulating a simple TCP echo response, can scale to over 2000 Connections per Second (CPS), roughly 10 times the Packet per Second (PPS) rate seen by the collection of three /24 sensors. While each connection contains more than one packet, this is easily within the capabilities of the lightweight mechanism. The backend servers were similarly evaluated in the laboratory environment. We selected a Win2k server for our performance measurement and evaluated the CPS performance of the Windows RPC mechanism (TCP/135) and the IIS Webserver performance as seen in Figure 8. The RPC service starts to show a performance degradation around 300-400 connections per second. The actual increase in performance over time is the result of a Windows SYN flood mechanism that fast paths the TCP 3-way handshake, not passing the connection to userland until the connection is complete. The IIS Webserver show much greater performance, with only a very slight performance degradation until it reaches a 1000 CPS limit imposed by the client version of Win2k. In any case, a single server is more than able to handle the load placed on it by the three backend servers.

In this section we combine the filtering benefits of content prevalence with the workload seen at several sensors to show the number of connections per second required by the backend framework. We used laboratory experiments to show the scalability of the backend honeypots as well as the lightweight sensors. Note that, the demands placed by the three /24 sensors are modest to the point that it is legitimate to suggest that we could replace them with three heavyweight honeypot deployments. However, the intention of the architecture is to scale the monitored address space well beyond three /24 networks, to include millions of monitored addresses. Centralizing the heavyweight honeypots yields more effective utilization of those resources, as well as centralizing those components most at risk of threat propagation.

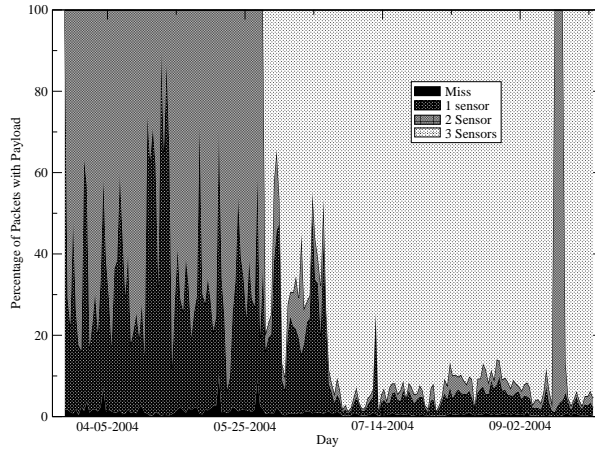


Figure 5: The global cache behavior seen at three sensors over a several month period. The y axis distinguishes payloads seen at individual sensors versus those seen at two or more. Initially, there are two sensors with the third sensor added in the beginning of June.

### 3.4 Coordination of sensors

Effective coordination between components of the hybrid architecture is necessary to achieve the best possible balance between accuracy and scalability. We employ a central controller that is responsible for several critical coordination activities: aggregating network and host based statistics from the lightweight sensors and backend honeypots, managing the VM machine state, and coordinating connection handoff to provide load balancing for the backends.

Aggregation and analysis of statistics from all sensors is the central part of our framework. The lightweight honeypots provide network statistics such as connection information and payload hashes to the controller which can be used to determine cache hit rates or to detect network anomalies. Monitoring the behavior of the backend Honeypots consists of three main components. The *network behavioral profiler* monitors the network traffic originating from the infected operating system. This module collects network connection attempts from the compromised machine for correlation with data collected from the IMS. The *file system behavioral profiler* is designed to detect differences between a clean, uninfected filesystem with the infected, compromised file system. Similar to tools such as tripwire, this module will be able to easily identify modified and added files for future infection detection and analysis. The final module is the *resource profiler* which profiles the resource usage of the infected operating system. This module is able to identify resource exhaustion based attacks as well as identifying which components of the operating system are affected by the malware.

The *virtual machine management module* takes care of running target operating systems on top of a virtual machine. This module starts a clean OS image for each infection attempt, and saves the infected state to disk for use by the other modules. In addition, instead of blocking outgoing connections, we redirect them back to the backend set. If a worm was to infect one of the machines, it would successfully infect more susceptible backends as long as uninfected backends are available. This mechanism is unique in that it allows us to watch actual worm propagation, including exploit as well as payload delivery in a controlled environment. Once this type of propagation is detected, the checkpointing features of VMware will not only allow us to save these infected states and corresponding forensic information, but also return the machines to a known good state for further use in analysis. Occasionally, these boxes may need to be reverted back to a compromised state in order to monitor secondary infections. By monitoring services added by the initial infection, the centralized control mechanism could use this information along with an increase in traffic on the new port to revert to an infected host state in order to monitor these secondary infections.

Coordinating connection handoff requires several key pieces of information. First the handoff manager maintains a global view of the payload cache state. The global view of the payload cache state allows us to avoid instructing sensors to offload payloads that may have already been handed off from another sensor. In addition, the manager maintains contact with the host system profiler, determining the load on the backend systems. This information is useful in helping to determine how aggressively to sample traffic. Finally, the handoff mechanism receive basic

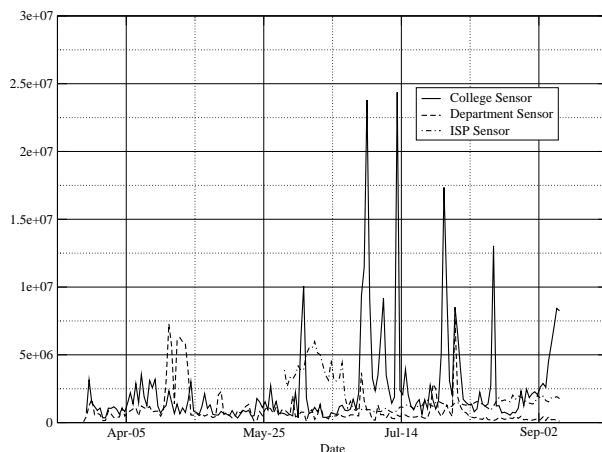


Figure 6: The packets seen at three IMS Sensors over a six month period.

network traffic characterization information in an aggregate form similar to netflow [12]. This information can be used to characterize changes that may guide the offload sampling mechanism.

One of the potential drawbacks of any handoff mechanism is that it may not accurately represent the effect of any given connection on the back end system. In order to monitor for the occurrence of new threats that have the same first payload, we support sampled connection handoff, periodically sending known signatures to the backend. This sampling is dynamic, relying on a feedback mechanism from the profiled backend honeypots. When the backends are lightly loaded we instruct the frontends to send signatures that have not been handed-off for the longest period of time. In addition to simple sampling, several other techniques can be applied to determine when to offload connections for existing payloads. The simplest approach is to watch traffic and alert when that traffic is over a certain threshold [28]. The problem with a static threshold is one of configuration and maintenance, that is, a threshold for every network segment is unique and subject to change. Adaptive thresholding provides a mechanism for understanding what is normal for a network and setting thresholds [13]. More refined notions of understanding traffic behavior include signal analysis [29], probabilistic approaches [16], and statistical approaches [11]. Each of these attempts to build a better notion of what is normal on a network. In addition to traffic based approaches, two recent works Autograph [15] and EarlyBird [33] use content prevalence as a mechanism for detection. Both systems use Rabin fingerprints of packet payloads to measure the prevalence of content across a network, alerting or detecting when a piece of content is higher than a threshold. All of these mechanisms are useful in identifying interesting traffic to redirect to the backend.

Effective coordination is essential for the efficient and accurate operation of the system. We have discussed three key features of the centralized coordination mechanism: this possible; the behavioral monitor, the virtual machine management module, and the connection handoff manager. These modules work in concert to coordinate connection handoff to the backends and build forensic information. New content is the primary handoff mechanism, but when backend resources become available, we support sampling of connection as well as traffic based triggers for connection offload.

## 4 Application of the Framework

In this section, we discuss the applicability of this framework to solving some of the standard problems in Internet threat detection and resolution. While we have completed many of the components of the proposed framework including distributed lightweight sensor deployment, a centralized VMWare host system, and the connection handoff mechanisms, these components have not been fully integrated. Subsequently, the full potential of the following applications has not been reached, but we believe they are representative of the unique capabilities enabled by this framework.

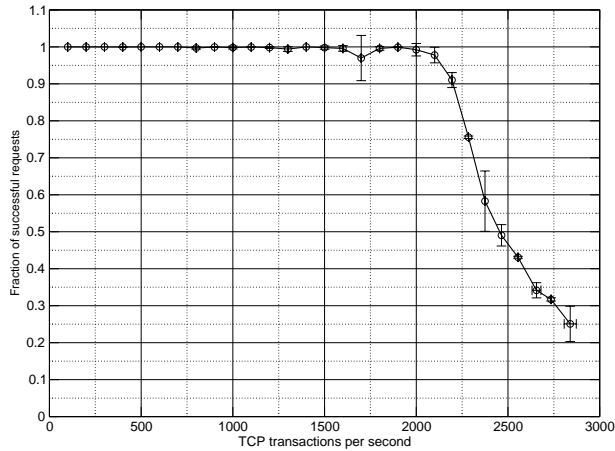


Figure 7: The graph shows the number of TCP transactions per second that Honeyd can support on 1 GHz Pentium III. Honeyd can support. In this case Honeyd is emulating a TCP echo server and therefore a transaction represents multiple packets.

## 4.1 Detection

Detection is the process of detecting and alerting network operators and researchers of brewing threats in the network. Traditional network approaches to this problem look at the amount of traffic, source and destination distribution, or the prevalence of content to determine the advent of a new threat. Host based techniques rely on antivirus, intrusion detection, or changes to the underlying operating systems to detect intrusions. Both of these provide value, but do not provide a high degree of confidence for previously unseen self propagating code. Network sensors do not support full compromise, e.g. a new worm would fail to propagate by just talking to a low-interaction honeypot. This is almost the reverse for high-interaction honeypots: a new worm can successfully infect such a machine and use it to propagate. Unfortunately, high-interactions system do not have optimal performance and would not scale to a large address space.

One unique application of the hybrid framework is to the area of worm detection. Recall that the backend component of this system consists of actual hosts serving as honeypots. Honeypots by their very nature do not have any real users that generate requests or load. The observable actions of the honeypot (network, filesystem, resources, etc.) are the result of either the standard operation of the OS or of external inputs. While honeypots have typically looked at traffic inbound to a honeypot to detect intrusions, worm detection is concerned with self-propagating code; code that exploits a vulnerability and uses the new host host as a new infection vector. Instead of watching the inbound traffic for signatures or unsolicited behavior, our novel worm propagation detection mechanism actually watches for the propagation of worms. As mentioned above, honeypots do not open outbound connections routinely as part of their operation and those connections that do occur are easily profiled. Therefore any *outbound* connection is a better indicator of an infection than an inbound analysis. In addition, because worms often use the same propagation method from host to host, we can apply the same content checksumming algorithm to packets out of the backend honeypot, and match them to the MD5 of the inbound connection. A matching outbound signature that matches an inbound handoff signature is even a higher indicator of self propagating code. Figure 9 shows an example of a worm that would be easily caught by this method.

This new approach is desirable because it combines the unique features of each sensor type. Actual infections provide a higher degree of confidence in actual self-propagating code, but scale to large enough monitored address blocks to be useful for early detection. Another interesting artifact of this approach is that there is now an infected host running the worm code. This code can be decompiled, the filesystem of the infected host can be compared to a known good state, and the many other behavioral and resource components of the worm can be analyzed.



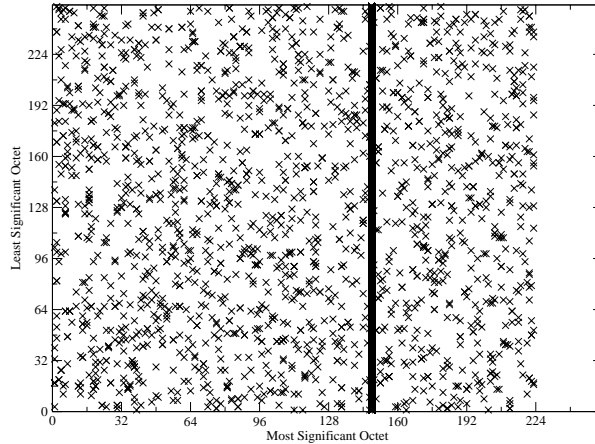


Figure 10: The scanning patterns of code red II. The graph shows the relationship between the most significant octet of a destination address and the least significant. The banding at  $x=148$  is the result of the  $/8$  preference of code red (the infected host is 148.233.157.181).

This means that specific hotspots might be missed by this approach. In contrast, host-based approaches are unable to account for policy filtering devices and/or network bottlenecks and outages that would affect the actual propagation of the worm.

Figure 10 shows the worm propagation from the host perspective by plotting the first 100,000 connection attempts from a jailed, infected honeypot machine infected with the CodeRed II worm. The graph shows the relationship between the first and last octet of a destination address for an attempted worm infection. Octet graphs of this nature are designed to show correlations in destination address selection and highlight address selection preferences. In the case of CodeRed II we see three interesting results. First, CodeRed II never selects  $/8$  address blocks greater than 224. While this behavior is not revealed in code analysis, and this class D space may not be routed due to the default routing behavior of a windows machine. Second, CodeRed II has a strong preference to the local  $/8$  (and also the local  $/16$ , although a separate octet graph is needed to show this). In this graph, this manifests itself as a dense collection of points along the vertical line for the  $/8$  block. Finally, a hardly visible gap is seen at the  $127/8$  block as this  $/8$  is also skipped by CodeRed II.

What this doesn't show is that even when these well defined propagation strategies are known through host-based analysis, observations at network sensors are wildly deviant from the controlled behaviors. Previous work has shown that these differences and conjectures are the result of errors in target selection, policy, topology, and statistical variance [6]. Without capturing both the observed network behavior as well as the host behavior, it is difficult to obtain a global perspective on worm propagation. This combination of both the network and on host sensors enables not only propagation strategy, but also the systems ability to determine the infection mechanism and key indicators or signatures that are needed to protect against future infections and clean up existing infections.

### 4.3 Signature Generation and Mitigation

Signature generation is the process of defining all the necessary characteristics of a new threat to be able to detect a new occurrence of the threat, identify existing infected hosts, and immunize against additional infections. This process is uniquely suited to the hybrid architecture as it requires; sufficient number of monitored hosts to catch the threat early in its growth phase and sufficient detailed behavioral analysis to identify previously unseen threats.

As an example, consider the cause of an auto immunizing system. Analogous to Moore et al., we can model the effect of immunization on worm propagation by using the classic SIR epidemic model. The model states that the number of newly infected hosts increases linearly with the product of infected hosts, fraction of susceptible hosts and contact rate. The immunization is represented by a decrease in new infections that is linear in the number of infected hosts.

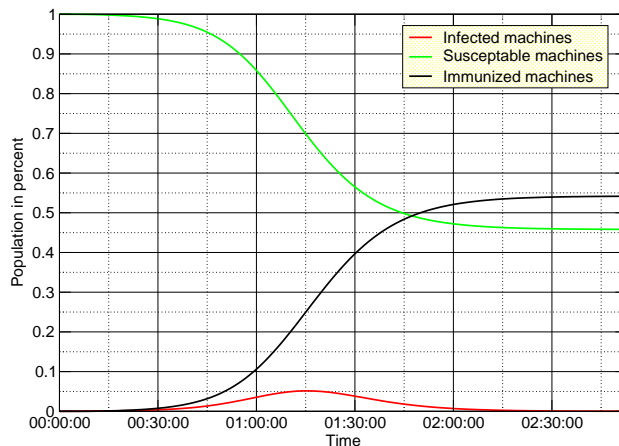


Figure 11: A simulated example of worm propagation when employing virtual honeypots to immunize infected hosts. We assume 360,000 susceptible machines in a 32-bit address space. With an initial worm seed of 150 infected machines that launch 50 probes per seconds, we can contain the worm if we employ 250,000 virtual honeypots with seven minutes.

The figure 11 shows a simulated example that tracks the change in the susceptible, infected and immunized populations. For example, if we assume 360,000 susceptible machines in a 32-bit address space, set the initial worm seed to 150 infected machines and each worm launches 50 probes per second. The simulation measures the effectiveness of using active immunization by virtual honeypots. The honeypots start working after a time delay. The time delay represents the time that is required to detect the worm and install the immunization code. We expect that immunization code can be prepared before a vulnerability is actively exploited. Time to detection and signature creation is key limited factor in this approach. If we wait for an hour, all vulnerable machines on the Internet will be infected. Our chances are better if we are able to begin immunization after 20 minutes. In that case, a deployment of about 262,000 honeypots is capable of stopping the worm from spreading. While beyond the scope of the three monitored blocks shown in this paper, we have shown that the existing infrastructure can support a minimum of 10x the amount of address space, more than enough to meet the requirements of our immunization example.

## 5 Related Work

Traditionally, approaches to threat monitoring fall into two broad categories, host based monitoring and network based monitoring.

Host based techniques fall into two basic approaches, forensics and host based honeypots. Antivirus software [4] and host based intrusion detection systems [10] seek to alert users of malicious code execution on the target machine by watching for patterns of behavior or signatures of known attacks. Host based honeypots [3, 5, 27, 9] track threats by providing an exploitable resource and monitoring it for abnormal behavior. A major goal of honeypots [35] is to provide insight into the motivation and techniques behind these threats.

The second monitoring approach is to monitor threats from the network perspective. Passive network techniques are characterized by the fact that they do little to intrude on the existing operation of the network. By far the most common technique is the passive measurement of live networks. They fall into three main categories: data from security or policy enforcement devices, data from traffic characterization mechanisms, and direct sensing or sniffing infrastructure. By either watching firewall logs, looking for policy violations, or by aggregating IDS alerts across multiple enterprises [30, 39], one can infer information regarding a worm's spread. Other policy enforcement mechanisms, such as router ACLs provide course-grained information about blocked packets. Instead of dropping these packets, CenterTrack [37] leveraged the existing routing infrastructure to collect denial of service traffic for analysis. Data collection techniques from traffic planning tools offer another rich area of pre-existing network instrumentation useful in characterizing threats. Course-grained interface counters and more fine-grained flow analysis tools such as NetFlow [12] offer another readily available source of information.

Another network monitoring approach is to passively collecting data from traffic to unused (or dark) address space. Because this space has no legitimate hosts, traffic destined to the space is the result of malicious activity or misconfiguration. The most common application of this technique is the global announcement and routing of unused space to a collection infrastructure that records the incoming packets [19, 23, 34].

The final networked monitoring approach uses active network perturbation to determine the scope and propagation of threats. This is typically done to elicit a behavior that is only visible by participating in a network or application session. Projects like honeynet [36] and iSink [38], and software like honeyd [26] are used to bring up networks of honeypots; places designed to capture information about intrusions in a controlled environments.

The hybrid system discussed in this paper falls into the the categories of active and passive measurement of unused address space. In the the area of multi-tiered sensing of unused space, there are two relevant projects. Collapsar is a centralized honeypot mechanism that relies on routing infrastructure to GRE tunnel ip space back to the honeypots [14]. One key differentiator between these approaches is that our hybrid approach focuses on filtering the interactions before they reach the backend, providing lighter requirements for the backend. In addition, because our hybrid approach responds to requests topologically next to the surrounding address space, it is not as susceptible to latency based fingerprinting as the tunneling approach. Another relevant project is that of iSink [38]. iSink focuses on scaling high interaction data collection to wide address ranges such as /8 blocks. There are two main areas in which this work differs. First and foremost, we are evaluating a distributed collection infrastructure that monitors numerous distributed blocks of varying size. Second, we use content prevalence as a mechanism for deciding when and how to filter interactions, unlike iSink which discusses only source destination filtering [25]. While unique in this context, content prevalence is not a new area of research. Two recent systems, Autograph [15], and EarlyBird [33] use content prevalence, in particular Rabin fingerprints, as mechanism for detecting worms. This work varies in two significant ways. One is that we monitor unused address space as opposed to live networks. This greatly filters the type and number of packet contents we receive and enables our second differentiator. Secondly, we employ a MD5 checksum to identify payloads, not LCS methods. The MD5 methods produce less overhead per packet and are scalable to the large amount of address space we monitor.

## 6 Conclusion

In this paper we have presented a novel, hybrid architecture for characterizing and tracking Internet threats such as worms or automated attacks. This architecture combines the wide coverage and quick assessment of network sensors with the behavioral insight provided by high-interaction honeypots. This hybrid system is scalable and is capable of providing fine grain detail about threats. Scale is achieved by a distributed collection of lightweight network sensors which filter the interactions to high-interaction backends. We discuss a novel mechanism for enabling the interaction between these two tiers of sensors and show how content prevalence can be used as an effective filtering mechanism. We use five months of deployment data and laboratory experiments to show how this hybrid architecture can scale to observing large numbers of IP addresses. Finally we demonstrate the utility of this framework by examining its application to threat detection, forensics, and mitigation.

## References

- [1] Iván Arce and Elias Levy. An analysis of the Slapper Worm. *IEEE Security & Privacy*, 1(1):82–87, January/February 2003.
- [2] Mohit Aron, Peter Druschel, and Willy Zwaenepoel. Efficient support for P-HTTP in cluster-based web servers, June 04 1999.
- [3] Bill Cheswick. An evening with berferd in which a cracker is lured, endured, and studied. In USENIX, editor, *Proceedings of the Winter 1992 USENIX Conference: January 20 — January 24, 1992, San Francisco, California*, pages 163–174, Berkeley, CA, USA, Winter 1992. USENIX.
- [4] Fred Cohen. *A Short Course on Computer Viruses*. John Wiley & Sons, 2nd edition, April 1994.
- [5] Fred Cohen. The deception toolkit (DTK), June 2004.
- [6] Evan Cooke, Michael Bailey, Z. Morley Mao, David Watson, and Farnam Jahanian. Toward understanding distributed blackhole placement. Submitted to WORM04.
- [7] Evan Cooke, Michael Bailey, David Watson, Farnam Jahanian, and Jose Nazario. The Internet motion sensor: A distributed global scoped Internet threat monitoring system. Technical Report CSE-TR-491-04, University of Michigan, Electrical Engineering and Computer Science, July 2004.

- [8] Microsoft Corporation. What you should know about the sasser worm. May 2004.
- [9] Symantec Corporation. Symantec decoy server (a.k.a mantrap), June 2004.
- [10] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for Unix processes. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 120–128, Oakland, CA, May 1996. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.
- [11] Joseph L. Hellerstein, Fan Zhang, and Shahabuddin Shahabuddin. A statistical approach to predictive detection. *Computer Networks (Amsterdam, Netherlands: 1999)*, 35(1):77–95, January 2001.
- [12] Cisco Systems Inc. Netflow services and applications. 2002. <http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neftct/tech/napps.wp.htm>.
- [13] Chuanyi Ji and Marina Thottan. Adaptive thresholding for proactive network problem detection, March 15 2000.
- [14] Xuxian Jiang and Dongyan Xu. Collapsar: A VM-based architecture for network attack detention center. In *Proceedings of the 11th USENIX Security Symposium*. USENIX, August 2004.
- [15] Hyang-Ah Kim and Brad Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proceedings of the 2004 USENIX Security Symposium*, San Diego, CA, USA, August 2004.
- [16] C. Leckie and R. Kotagiri. A probabilistic approach to detecting network scans. In *In Proceedings of the Eighth IEEE Network Operations and Management Symposium (NOMS 2002)*, pages 359–372, Florence, Italy, April 2002.
- [17] John Levine, Richard LaBella, Henry Owen, Didier Contis, and Brian Culver. The use of Honeynets to detect exploited systems across large enterprise networks. In *Proceedings of the 4th IEEE Information Assurance and Security Workshop*, West Point, NY, USA, June 2003.
- [18] Tom Liston. Labrea - homepage, June 2004.
- [19] David Moore. Network telescopes: Observing small or distant security events. In *11th USENIX Security Symposium, Invited talk*, San Francisco, CA, August 5–9 2002. Unpublished.
- [20] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the slammer worm. *IEEE Security & Privacy*, 1(4):33–39, 2003.
- [21] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. The Spread of the Sapphire/Slammer Worm. In *Proceedings of the 27th NANOG Meeting*, Phoenix, Arizona, February 2003.
- [22] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Network telescopes. <http://www.caida.org/outreach/papers/2004/tr-2004-04/>, July 2004.
- [23] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet denial-of-service activity. In *Proceedings of the Tenth USENIX Security Symposium*, pages 9–22, Washington, D.C., August 13–17 2001. USENIX.
- [24] Jose Nazario, Michael Bailey, and Farnam Jahanian. The spread of the Blaster worm. Submitted to IEEE Security and Privacy.
- [25] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of internet background radiation. To Appear in Proceedings of IMC 2004 (October 2004), June 2004.
- [26] Niels Provos. A Virtual Honey-pot Framework. In *Proceedings of the 13th USENIX Security Symposium*, pages 1–14, San Diego, CA, USA, August 2004.
- [27] Marcus Ranum and the NFR folks. nfr security resource center: back officer friendly, June 2004.
- [28] S. Robertson, E. Siegel, M. Miller, and Stolfo Stolfo. Surveillance detection in high bandwidth environments. In *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX 2003)*. IEEE Computer Society Press, April 2003.
- [29] Amos Ron, David Plonka, Jeffery Kline, and Paul Barford. A signal analysis of network traffic anomalies. *Internet Measurement Workshop 2002*, August 09 2002.
- [30] SANS. Sans - Internet storm center - cooperative cyber threat monitor and alert system, June 2004.
- [31] Colleen Shannon and David Moore. The spread of the Witty worm. <http://www.caida.org/analysis/security/witty/>, June 2004.
- [32] Colleen Shannon, David Moore, and Jeffery Brown. Code-red: a case study on the spread and victims of an internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)*, December 02 2002.
- [33] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. The earlybird system for real-time detection of unknown worms. Technical report, University of California at San Diego, August 2003.
- [34] Dug Song, Rob Malan, and Robert Stone. A snapshot of global internet worm activity. Arbor Network Technical Report, June 2002.

- [35] Lance Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley, 2002.
- [36] Lance Spitzner et al. The honeynet project, June 2004.
- [37] R. Stone. Centertrack: An ip overlay network for tracking dos floods, 2000.
- [38] Paul Barford Vinod Yegneswaran and Dave Plonka. On the design and use of internet sinks for network abuse monitoring. In *Recent Advances in Intrusion Detection — Proceedings of the 7th International Symposium (RAID 2004)*, Lecture Notes in Computer Science, Sophia Antipolis, French Riviera, France, October 2004. Springer-Verlag, Berlin Germany.
- [39] Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global intrusion detection in the DOMINO overlay system. In *Proceedings of Network and Distributed System Security Symposium (NDSS '04)*, San Diego, CA, February 2004.