Characteristics of Internet Background Radiation

Ruoming Pang*
rpang@cs.princeton.edu

Vinod Yegneswaran†
vinod@cs.wisc.edu

Paul Barford†
pb@cs.wisc.edu

Vern Paxson‡§
vern@icir.org

Larry Peterson*
llp@cs.princeton.edu

ABSTRACT

Monitoring any portion of the Internet address space reveals incessant activity. This holds even when monitoring traffic sent to unused addresses, which we term "background radiation." Background radiation reflects fundamentally nonproductive traffic, either malicious (flooding backscatter, scans for vulnerabilities, worms) or benign (misconfigurations). While the general presence of background radiation is well known to the network operator community, its nature has yet to be broadly characterized. We develop such a characterization based on data collected from four unused networks in the Internet. Two key elements of our methodology are *(i)* the use of *filtering* to reduce load on the measurement system, and *(ii)* the use of *active responders* to elicit further activity from scanners in order to differentiate different types of background radiation. We break down the components of background radiation by protocol, application, and often specific exploit; analyze temporal patterns and correlated activity; and assess variations across different networks and over time. While we find a menagerie of activity, probes from worms and autorooters heavily dominate. We conclude with considerations of how to incorporate our characterizations into monitoring and detection activities.

Categories and Subject Descriptors: C.2.5 [Local and Wide-Area Networks]: Internet

General Terms: Measurement

Keywords: Internet Background Radiation, Network Telescope, Honeypot

1. INTRODUCTION

In recent years a basic characteristic of Internet traffic has changed. Older traffic studies make no mention of the presence of appreciable, on-going attack traffic [9, 25, 34, 3], but those monitoring and operating today's networks are immediately familiar with the incessant presence of traffic that is "up to no good." We can broadly characterize this traffic as *nonproductive*: it is either destined for addresses that do not exist, servers that are not running, or servers that do not want to receive the traffic. It can be a hostile reconnaissance scan, "backscatter" from a flooding attack victimizing someone else, spam, or an exploit attempt.

The volume of this traffic is not minor. For example, traffic logs from the Lawrence Berkeley National Laboratory (LBL) for an arbitrarily-chosen day show that 138 different remote hosts each scanned 25,000 or more LBL addresses, for a total of about 8 million connection attempts. This is more than double the site's entire quantity of successfully-established incoming connections, originated by 47,000 distinct remote hosts. A more fine-grained study of remote scanning activity found (for a different day) 13,000 different scanners probing LBL addresses [14].

What is all this nonproductive traffic trying to do? How can we filter it out in order to detect *new* types of malicious activity?

Because this new phenomenon of incessant nonproductive traffic has not yet seen detailed characterization in the literature, we have lacked the means to answer these questions. In this study we aim to provide an initial characterization of this traffic. Given the traffic's pervasive nature (as we will demonstrate), we term it Internet "background radiation".

A basic issue when attempting to measure background radiation is how, in the large, to determine which observed traffic is indeed unwanted. If we simply include all unsuccessful connection attempts, then we will conflate truly unwanted traffic with traffic representing benign, transient failures, such as accesses to Web servers that are usually running but happen to be off-line during the measurement period.

By instead only measuring traffic sent to hosts that *don't exist—i.e.*, Internet addresses that are either unallocated or at least unused—we can eliminate most forms of benign failures and focus on traffic highly likely to reflect unwanted activity. In addition, analyzing unused addresses yields a second, major measurement benefit: we can safely *respond* to the traffic we receive. This gives us the means to not only passively measure unwanted traffic (for example, what ports get probed), but to then engage the remote sources in order to elicit from them their particular intentions (for example, what specific actions they will take if duped into thinking they have found a running server).

Given the newness of this type of Internet measurement, one of the contributions of our study is the set of methodologies we develop for our analysis. These include considerations for how to use *filtering* to reduce the load on the measurement system, how to construct *active responders* to differentiate different types of background radiation, and ways for interpreting which facets of the collected data merit investigation and which do not.

In some ways, the goals of our study are prosaic: we aim to char-

*Dept. of Computer Science, Princeton University
†Dept. of Computer Science, University of Wisconsin at Madison
‡International Computer Science Institute
§Lawrence Berkeley Laboratory

IMC'04, October 25–27, 2004, Taormina, Sicily, Italy.

acterize the nature of the *background*, which, by its very ubiquity, runs the risk of having a boring sameness to it. In fact, one measure of success for us would be to achieve a numbingly complete characterization of background radiation which could then facilitate the construction of *classifiers* to remove known elements of background radiation from a given set of observations. Such classifiers could both offload various types of network analyzers (for example, reducing the state a network intrusion detection system must track) and provide a means to return to the simpler world of a decade ago, by allowing us to recover a notion of "normal," attack-free traffic. Such attack-free traffic can be highly valuable for establishing baselines for types of analysis that flag departures from normality as harbingers of malicious activity meriting investigation.

We proceed with our study as follows. First, in § 2 we discuss previous work related to our efforts. In § 3 we describe the sources of data used in our study and our methodology related to capturing and analyzing this data. § 4 analyzes what we can learn from our monitoring when we use it purely passively, and § 5 then extends this to what we can learn if we also respond to traffic we receive. In § 6 we evaluate aspects of traffic source behavior. We conclude with a summary of the themes developed during our study.

2. RELATED WORK

Several studies have characterized specific types of malicious traffic. Moore *et al.* investigate the prevalence of denial-of-service attacks in the Internet using "backscatter analysis" [23], *i.e.*, observing not the attack traffic itself but the replies to it sent by the flooding victim, which are routed throughout the Internet due to the attacker's use of spoofed source addresses. Measurement studies of the Code Red I/II worm outbreaks [21] and the Sapphire/Slammer worm outbreak [20, 19] provide detail on the method, speed and effects of each worm's propagation through the Internet. Additional studies assess the speed at which counter-measures would have to be deployed to inhibit the spread of similar worms [22].

The empirical components of these studies were based largely on data collected at "network telescopes" (see below) similar to those used in our study, though without an active-response component. A related paper by Staniford *et al.* mathematically models the spread of Code Red I and considers threats posed by potential future worms [33]. A small scale study of Internet attack processes using a fixed honeypot setup is provided in [8]. Yegneswaran *et al.* explore the statistical characteristics of Internet attack and intrusion activity from a global perspective [43]. That work was based on the aggregation and analysis of firewall and intrusion detection logs collected by Dshield.org over a period of months. The coarse-grained nature of that data precluded an assessment of attacks beyond attribution to specific ports. Finally, Yegneswaran *et al.* provide a limited case study in [42] that demonstrates the potential of network telescopes to provide a broad perspective on Internet attack activity. We extend that work by developing a much more comprehensive analysis of attack activity.

Unused IP address space has become an important source of information on intrusion and attack activity. Measurement systems deployed on unused IP address ranges have been referred to as "Internet Sink-holes" [12], and "Network Telescopes" [18]. Active projects focused on unused address space monitoring include Honeynet [13] and Honeyd [27]. Honeynet focuses on the use of live VMware-based systems to monitor unused addresses. Honeyd uses a set of stateful virtual responders to operate as an interactive honeypot.

Finally, network intrusion detection systems, including Snort [29, 6], Bro [26], and a variety of commercial tools, are commonly used to detect scans for specific malicious payloads. An emerging area of research is in the automated generation of attack signatures. For example, Honeycomb [17] is an extension of Honeyd that uses a *longest common substring* (LCS) algorithm on packet-level data recorded by Honeyd to automatically generate signatures. Other recent work pursues a similar approach, including Earlybird [32] and Autograph [15]. Our study can inform future developments of such systems with respect to both the type and volume of ambient background attack activity.

3. MEASUREMENT METHODOLOGY

This section describes the methods and tools we use to measure and analyze background radiation traffic, addressing two key issues:

1. **Taming large traffic volume**: We listen and respond to background traffic on thousands to millions of IP addresses. The sheer volume of traffic presents a major hurdle. We handle this with two approaches: 1) devising a sound and effective filtering scheme, so that we can significantly reduce the traffic volume while maintaining the variety of traffic; and 2) building a scalable responder framework, so we can respond to traffic at a high rate.

2. **Building application-level responders**: We find that TCP SYN packets dominate background radiation traffic in our passive measurements, which means we need to accept connections from the sources and extend the dialog as long as possible to distinguish among the types of activities. This involves building responders for various application protocols, such as HTTP, NetBIOS, and CIFS/SMB, among others.

3.1 Taming the Traffic Volume

Responding to the entirety of background radiation traffic received by thousands to millions of IP addresses would entail processing an enormous volume of traffic. For example, we see nearly 30,000 packets per second of background radiation on the Class A network we monitor. Taming the traffic volume requires effective filtering, and it is also important to investigate scalable approaches to building responders. We discuss each in turn.

3.1.1 Filtering

When devising a filtering scheme, we try to balance trade-offs between traffic reduction and the amount of information lost in filtering. We considered the following strategies:

Source-Connection Filtering: This strategy keeps the first N connections initiated by each source and discards the remainder. A disadvantage of this strategy is that it provides an inconsistent view of the network to the source: that is, live IP addresses become unreachable. Another problem is that an effective value of N can be service- or attack-dependent. For certain attacks (*e.g.*, "Code Red"), $N = 1$ suffices, but multi-stage activities like Welchia, or multi-vector activities like Agobot, require larger values of N.

Source-Port Filtering: This strategy is similar except we keep N connections for each source/destination port pair. This alleviates the problem of estimating N for multi-vector activities like Agobot, but multi-stage activities on a single destination port like Welchia remain a problem. This strategy also exposes an inconsistent view of the network.

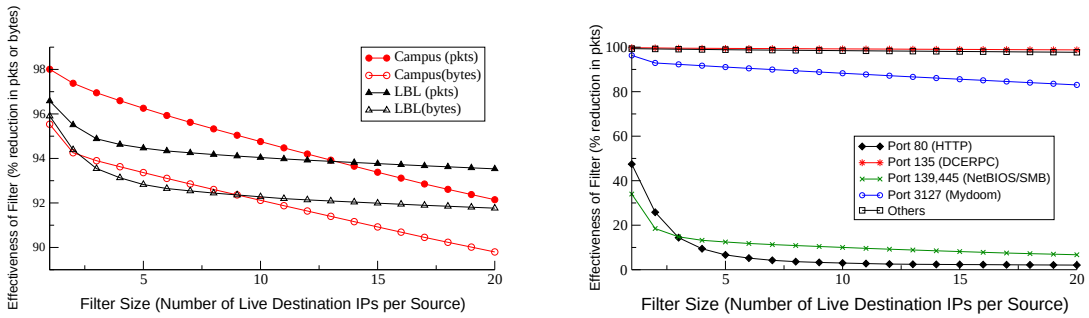


Figure 1: Effectiveness of Filtering, Networks (left) and Services (right)

Source-Payload Filtering: This strategy keeps one instance of each type of activity per source. From a data richness perspective, this seems quite attractive. However, it is very hard to implement in practice as we do not often know whether two activities are similar until we respond to several packets (especially true for multi-stage activities and chatty protocols like NetBIOS). This strategy also requires significant state.

Source-Destination Filtering: This is the strategy we chose for our experiments, based on the assumption that background radiation sources possess the same degree of affinity to all monitored IP addresses. More specifically, if a source contacts a destination IP address displaying certain activity, we assume that we will see the same kind of activity on all other IP addresses that the source tries to contact. We find this assumption generally holds, except for the case of certain multi-vector worms that pick one exploit per IP address, for which we will identify only one of the attack vectors.

Figure 1 illustrates the effectiveness of this filtering on different networks and services when run for a two-hour interval. The first plot shows that the filter reduces the inbound traffic by almost two orders of magnitude in both networks. The LBL network obtains more significant gains than the larger Campus networks because the Campus network intentionally does not respond to the last stage of exploits from certain frequently-seen Welchia variants that in their last step send a large attack payload ($>$ 30KB buffer overflow). The second plot illustrates the effectiveness of the filter for the various services. Since Blaster (port 135) and MyDoom (port 3127) scanners tend to horizontally sweep IP subnets, they lead to significant gains from filtering, while less energetic HTTP and NetBIOS scanners need to be nipped in the bud (low N) to have much benefit.

3.1.2 *Active Sink: an Event-driven Stateless Responder Platform*

Part of our active response framework explores a *stateless* approach to generating responses, with a goal of devising a highly scalable architecture. Active Sink is the active response component of iSink[42], a measurement system developed to scalably monitor background radiation observed in large IP address blocks. Active Sink simulates virtual machines at the network level, much like Honeyd [27], but to maximize scalability it is implemented in a stateless fashion as a Click kernel module [42] [16]. It achieves statelessness by using the form of incoming application traffic to determine an appropriate response (including appropriate sequence numbers), without maintaining any transport or application level state. A key question for this approach is whether all necessary responders can be constructed in such a stateless fashion. While exploring this issue is beyond the scope of the present work, we note that for all of the responders we discuss, we were able to implement a stateless form for Active Sink, as well as a stateful form based on Honeyd. (To facilitate the dual development, we developed interface modules so that each could use the same underlying code for the responders.)

3.2 Application-Level Responders

Our approach to building responders was "data driven": we determined which responders to build based on observed traffic volumes. Our general strategy was to pick the most common form of traffic, build a responder for it detailed enough to differentiate the traffic into specific types of activity, and, once the "Unknown" category for that type of activity was sufficiently small, repeat the process with the next largest type of traffic.

Using this process, we built an array of responders for the following protocols (Figure 2): HTTP (port 80), NetBIOS (port 137/139), CIFS/SMB [7] (port 139/445), DCE/RPC [10] (port 135/1025 and CIFS named pipes), and Dameware (port 6129). We also built responders to emulate the backdoors installed by MyDoom (port 3127) and Beagle (port 2745) [5], [24].

Application-level responders need to not only adhere to the structure of the underlying protocol, but also to know *what* to say. Most sources are probing for a particular implementation of a given protocol, and we need to emulate behavior of the target software in order to keep the conversation going.

The following example of HTTP/WebDAV demonstrates what this entails. We see frequent **`"GET /"`** requests on port 80. Only by responding to them and mimicking a Microsoft IIS with WebDAV enabled will elicit further traffic from the sources. The full sequence plays as:

```
GET /
⇒ |200 OK ... Server: Microsoft-IIS/5.0|
SEARCH /
⇒ |411 Length Required|
SEARCH /AAA... (URI length > 30KB)
⇒ (buffer overflow exploit received)
```

Some types of activity require quite intricate responders. Many Microsoft Windows services run on top of CIFS (port 139/445), which lead us to develop the detailed set of responses shown in Figure 3. Requests on named pipes are further tunneled to various DCE/RPC responders. One of the most complicated activities is the exploit on the SAMR and later on the SRVSVC pipe, which involves more than ten rounds exchanging messages before the source will reveal its specific intent by attempting to create an executable file on the destination host. Figure 4 shows an example where we cannot classify the source until the "NT Create AndX"

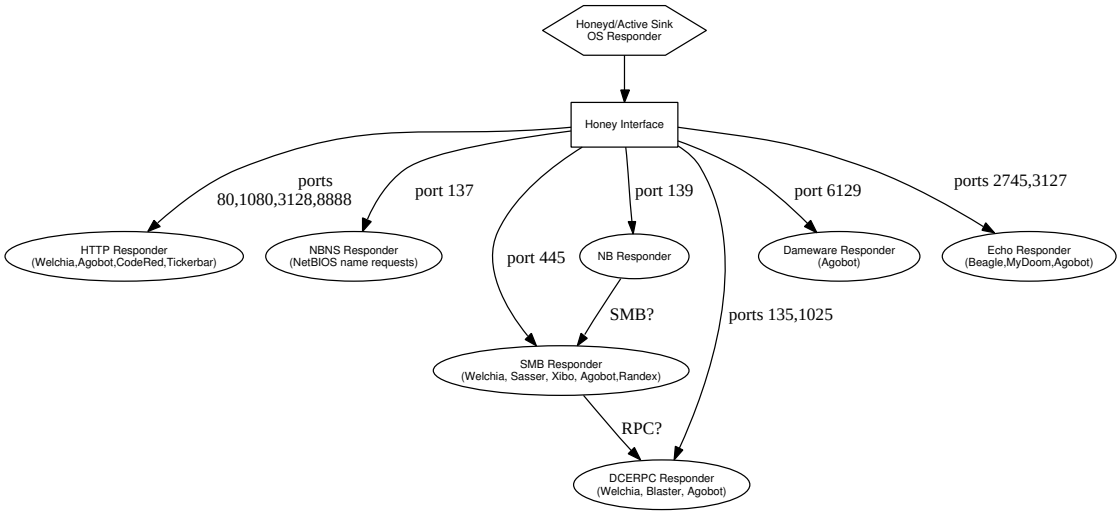


Figure 2: Top level Umbrella of Application Responders

request for **`msmsgri32.exe`**. (The NetrRemoteTOD command is used to schedule the worm process to be invoked one minute after TimeOfDay [4].) We found this attack sequence is shared across several viruses, including the Lioten worm [4] and Agobot variants [1].

Building responders like this one can prove difficult due to the lack of detailed documentation on services such as CIFS and DCE/RPC. Thus, we sometimes must resort to probing an actual Windows system running in a virtual machine environment, in order to analyze the responses it makes en route to becoming infected. We modified existing trace replay tools like `flowreplay` for this purpose [11].

More generally, as new types of activities emerge over time, our responders also need to evolve. While we find the current pace of maintaining the responders tractable, an important question is to what degree we can automate the development process.

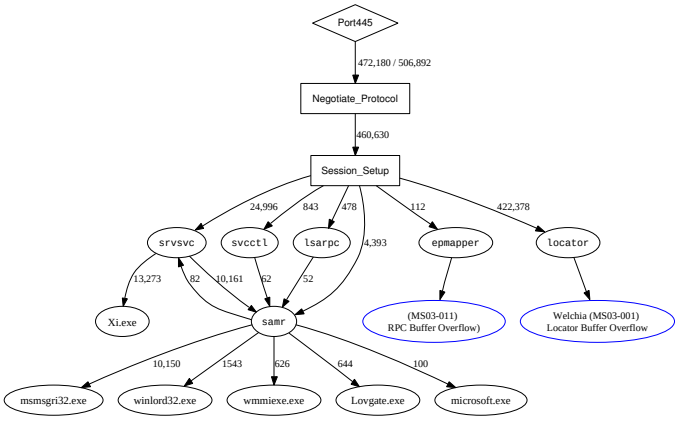


Figure 3: Example summary of port 445 activity on Class A (500K Sessions) Arcs indicate number of sessions

3.3 Traffic Analysis

Once we can engage in conversations with background radiation sources, we then need to undertake the task of understanding the traffic. Here our approach has two components: first, we separate traffic analysis from the responders themselves; second, we try to analyze the traffic in terms of its application-level semantics.

Regarding the first of these, while it might appear that the job of traffic analysis can be done by the responders(since the responders need to understand the traffic anyway), there are significant benefits to performing traffic analysis independently. We do so by capturing and storing `tcpdump` packet traces for later off-line analysis. This approach allows us to preserve the complete information about the traffic and evolve our analysis algorithms over time. The flip side is that doing so poses a challenge for the analysis tool, since it needs to do TCP stream reassembly and application-protocol parsing. To address this issue, we built our tool on top of the Bro intrusion detection system [26], which provides a convenient platform for application-level protocol analysis.

We found early on that in order to filter the background radiation traffic from the "normal" traffic, we need to understand the application semantics of the traffic. This is because the background radiation traffic has very distinctive application semantic characteristics compared to the "normal" traffic (as we will see in the following sections), but the differences are far more difficult to detect at the network or transport level.

Our analysis has an important limitation: we do not attempt to understand the binary code contained in buffer-overrun exploits. This means we cannot tell for sure which worm or autorooter sent us a particular exploit (also due to lack of a publicly available database of worm/virus/autorooter packet traces). If a new variant of an existing worm arises that exploits the same vulnerability, we may not be able to discern the difference. However, the analysis will identify a new worm if it exploits a different vulnerability, as in the case of the Sasser worm [30].

3.4 Experimental Setup

We conducted our experiments at two different sites. These ran two different systems, *iSink* and *LBL Sink*, which conducted the same forms of application response but used different underlying mechanisms.

iSink: Our iSink instance monitored background traffic observed in a Class A network (/8, 2^{24} addresses), and two /19 subnets (16K addresses) on two adjacent UW campus class B net-

```
-> SMB Negotiate Protocol Request
<- SMB Negotiate Protocol Response
-> SMB Session Setup AndX Request
<- SMB Session Setup AndX Response
-> SMB Tree Connect AndX Request,
       Path:  \\XX.128.18.16\IPC$
<- SMB Tree Connect AndX Response
-> SMB NT Create AndX Request, Path: \samr
<- SMB NT Create AndX Response
-> DCERPC Bind: call_id: 1 UUID: SAMR
<- DCERPC Bind_ack:
-> SAMR Connect4 request
<- SAMR Connect4 reply
-> SAMR EnumDomains request
<- SAMR EnumDomains reply
-> SAMR LookupDomain request
<- SAMR LookupDomain reply
-> SAMR OpenDomain request
<- SAMR OpenDomain reply
-> SAMR EnumDomainUsers request
```

```
Now start another session, connect to the
SRVSVC pipe and issue NetRemoteTOD
(get remote Time of Day) request

-> SMB Negotiate Protocol Request
<- SMB Negotiate Protocol Response
-> SMB Session Setup AndX Request
<- SMB Session Setup AndX Response
-> SMB Tree Connect AndX Request,
       Path: \ \XX.128.18.16\IPC$
<- SMB Tree Connect AndX Response
-> SMB NT Create AndX Request, Path: \srvsvc
<- SMB NT Create AndX Response
-> DCERPC Bind: call_id: 1 UUID: SRVSVC
<- DCERPC Bind_ack: call_id: 1
-> SRVSVC NetrRemoteTOD request
<- SRVSVC NetrRemoteTOD reply
-> SMB Close request
<- SMB Close Response
```

```
Now connect to the ADMIN share and write the file


-> SMB Tree Connect AndX Request, Path: \\XX.128.18.16\ADMIN$
<- SMB Tree Connect AndX Response
-> SMB NT Create AndX Request,
    Path:\system32\msmsgri32.exe  <<<===

<- SMB NT Create AndX Response, FID: 0x74ca
-> SMB Transaction2 Request SET_FILE_INFORMATION
<- SMB Transaction2 Response SET_FILE_INFORMATION
-> SMB Transaction2 Request QUERY_FS_INFORMATION
<- SMB Transaction2 Response QUERY_FS_INFORMATION
-> SMB Write Request
....
```

Figure 4: Active response sequence for Samr-exe viruses

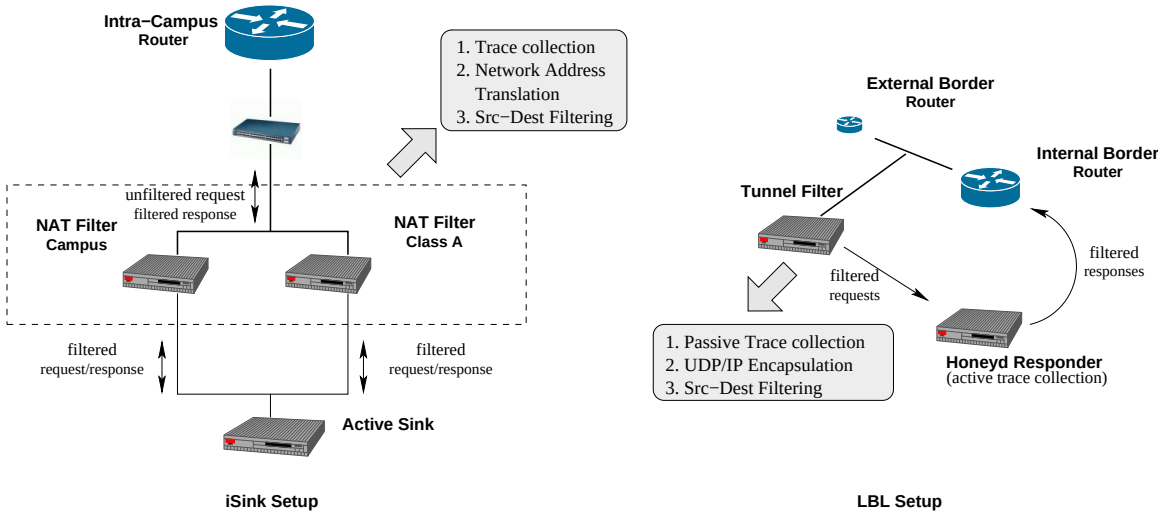


Figure 5: The Honeynet architecture at iSink and LBL

works, respectively. Filtered packets are routed via Network Address Translation to the Active Sink, per Figure 5. We used two separate filters: one for the Class A network and another for the two campus /19 subnets. We collected two sets of `tcpdump` traces for the networks: prefiltered traces with of packet headers, which we use in passive measurements (of periods during which the active responders were turned off), and filtered traces with complete payloads, which we use for active traffic analysis. The prefiltered traces for the Class A network are sampled at 1/10 packets to mitigate storage requirements.

LBL Sink: The LBL Sink monitors two sets of 10 contiguous /24 subnets. The first is for passive analysis; we merely listen but do not respond, and we do *not* filter the traffic. The second is for active analysis. We further divide it into two halves, 5 /24 subnets each, and apply filtering on these separately. After filtering, our system tunnels the traffic to the active responders, as shown in Figure 5. This tunnel is one-way—the responses are routed directly via the internal router. We use the same set of application protocol responders at LBL as in iSink, but they are invoked by Honeyd instead of iSink, because Honeyd is sufficient for the scale of traffic at LBL after filtering. We trace active response traffic at the Honeyd host, and unless stated otherwise this comes from one of the halves (*i.e.*, 5 /24 subnets).

Site	Networks (/size)	Datasets	Duration
iSink	UW-I (/19)	Active	Mar16–May14, 2004
		Passive	Mar11–May14, 2004
	UW-II (/19)	Active	Mar16–May14, 2004
		Passive	Mar11–May14, 2004
	Class A (/8)	Active	Mar12–Mar30, 2004
		Passive	Mar16–Mar30, 2004
LBL Sink	LBL-A (2 x 5 x /24)	Active	Mar12–May14, 2004
	LBL-P (10 x /24)	Passive	Apr 28–May 5, 2004

Table 1: Summary of Data Collection

Note that the LBL and UW campus have the same /8 prefix, which gives them much more locality than either has with the class A network.

Table 1 summarizes the datasets used in our study. At each network we collected passive `tcpdump` traces and filtered, active-response traces. On the two UW networks and the LBL network, we collected two months' worth of data. Our provisional access to the class A enabled us to collect about two weeks of data.

The sites use two different mechanisms to forward packets to the active responder: tunneling, and Network Address Translation (NAT). The LBL site uses tunneling (encapsulation of IP datagrams inside UDP datagrams), which has the advantages that: (i) it is very straightforward to implement and (ii) it does not require extensive

state management at the forwarder. However, tunneling requires the receive end to a) decapsulate traces before analysis, b) handle fragmentation of full-MTU packets, and c) allocate a dedicated tunnel port. NAT, on the other hand, does not have these three issues, but necessitates maintaining per-flow state at the forwarder, which can be significant in large networks. The stateless responder deployed at the UW site allows such state to be *ephemeral*, which makes the approach feasible. That is we only need to maintain a consistent flow ID for each outstanding incoming packet, so the corresponding flow record at the filter can be evicted as soon as it sees a response. Hence, the lifetime of flow records is on the order of milliseconds (RTT between the forwarder and active-sink) instead of seconds.

4. PASSIVE MEASUREMENT OF BACKGROUND RADIATION

This section presents a baseline of background radiation traffic on unused IP addresses *without actively responding to any packet*. It starts with a traffic breakdown by protocols and ports, and then takes a close look at one particular facet of the traffic: backscatter.

4.1 Traffic Composition

A likely first question about background radiation characteristics is "What is the type and volume of observed traffic?". We start to answer this question by looking at two snapshots of background radiation traffic shown in Table 2 which includes an 80 hour trace collected at UW Campus on a /19 network from May 1 to May 4, a one week trace at LBL collected on 10 contiguous /24 networks from April 28 to May 5, and finally a one-week trace at Class A with 1/10 sampling from March 11 to 18.

Protocol	UW-1		LBL-P		Class A	
	Rate	%	Rate	%	Rate	%
TCP	928	95.0%	664	56.5%	130	88.5%
ICMP	4.00	4.2%	488	39.6%	0.376	0.3%
UDP	0.156	0.8%	45.2	3.8%	16.5	11.3%

Table 2: Traffic rate breakdown by protocols. The rate is computed as number of packets per destination IP address per day, *i.e.*, with network size and sampling rate normalized

Clearly, TCP dominates more or less in all three networks. The relatively lower TCP rate at Class A is partly due to the artifact that the Class A trace was collected in Mar instead in May, when we see a few large worm/malware outbreaks (include the Sasser worm). Not shown in the table, about 99% of the observed TCP packets are `TCP/SYN`.

The large number of ICMP packets (of which more than 99.9% are `ICMP/echo-req`) we see at LBL form daily high volume spikes (Figure 6), which are the result of a small number of sources scanning every address in the observed networks. On the other hand we see a lot fewer ICMP packets at the Class A monitor which is probably because the Welchia worm, which probes with `ICMP/echo-req`, avoids the Class A network.

Finally, the surprising low rate of UDP packets observed at UW is largely due to the artifact that UW filters UDP port 1434 (the Slammer worm).

In Figure 6, we can also see that `TCP/SYN` packets seen at LBL arrive at a relatively steady rate, (and this is the case for the other two networks as well) in contrast to daily ICMP spikes. A closer look at the breakdown of `TCP/SYN` packets by destination port numbers at LBL (Table 4) reveals that a small number of ports are

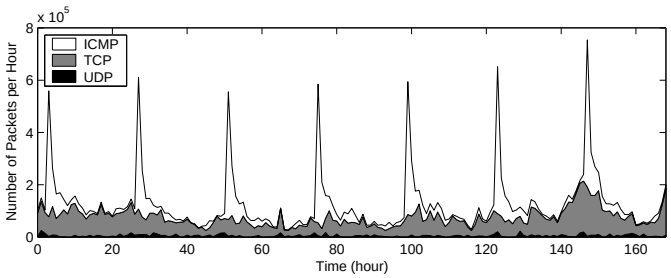


Figure 6: Number of background radiation packets per hour seen at LBL

the targets of a majority of `TCP/SYN` packets (the eight ports listed in the table account for more than 83% of the packets).

Table 3 shows the same traces from the perspective of the source of the traffic. Note that the rows are not mutually exclusive as one host may send both TCP and UDP packets. It is clear that TCP packets dominate in the population of source hosts we see. The distribution across ports of LBL traffic is shown in Table 4; as before, a small number of ports are dominant.

Protocol	UW		LBL	
	#SrcIP	Percentage	#SrcIP	Percentage
TCP	759,324	87.9%	586,025	90.0%
ICMP	109,135	12.6%	64,120	9.8%
UDP	4,273	0.5%	4,360	0.7%

Table 3: Traffic breakdown by number of sources.

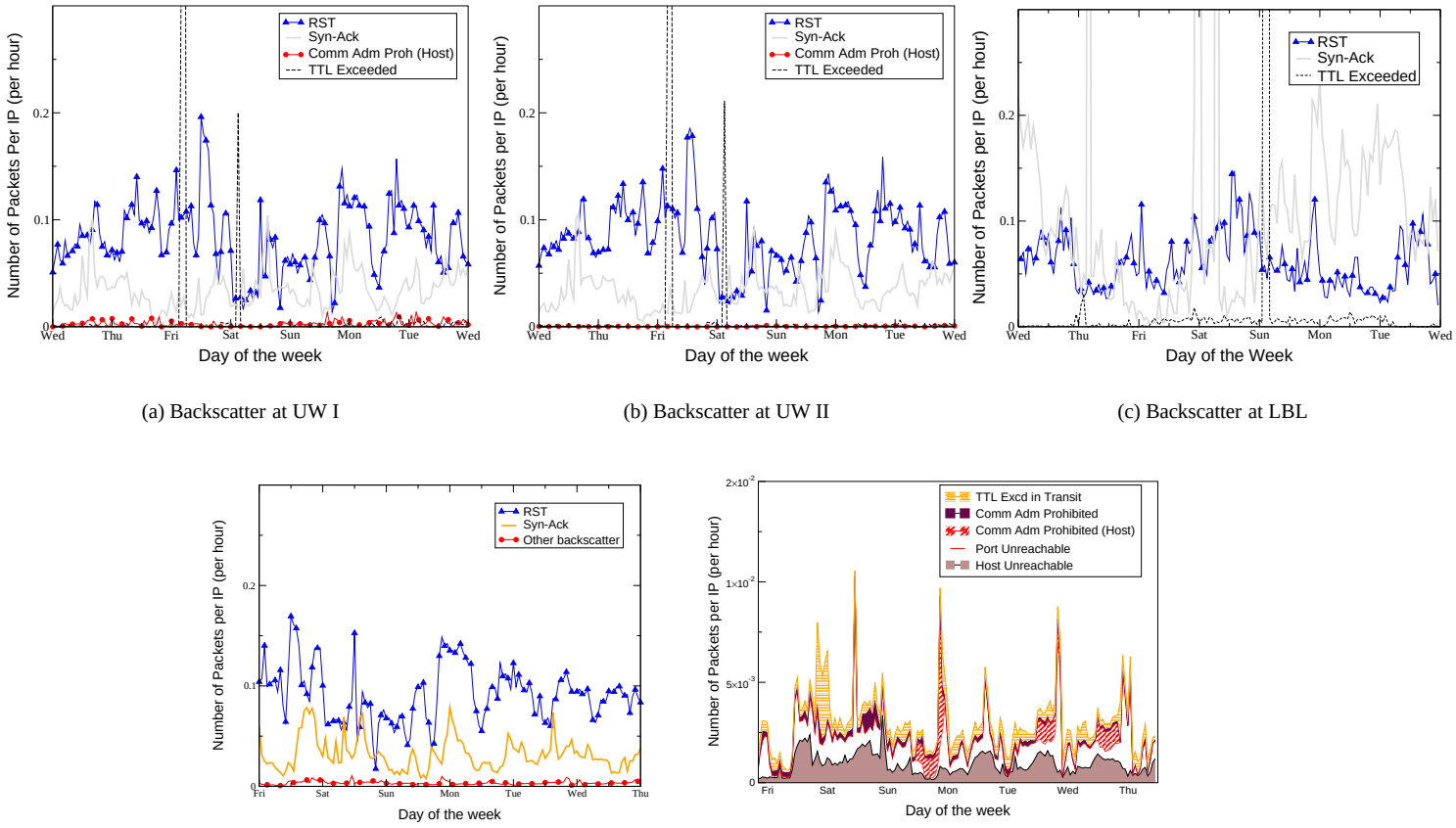
TCP Port	# Source IP (%)	# Packets (%)
445	43.4%	19.7%
80	28.7%	7.3%
135	19.1%	30.4%
1025	4.3%	5.8%
2745	3.2%	3.6%
139	3.2%	11.1%
3127	2.7%	3.2%
6129	2.2%	2.4%

Table 4: The Most Popular TCP Ports. Ports that are visited by the most number of source IPs, as in a one week passive trace at LBL. In total there are 12,037,064 packets from 651,126 distinct source IP addresses.

As `TCP/SYN` packets constitute a significant portion of the background radiation traffic observed on a passive network, the next obvious question is, *"What are the intentions of these connection requests?"*. We explore this question in Section 5 and 6.

4.2 Analysis of Backscatter Activity

The term Backscatter is commonly used to refer to unsolicited traffic that is the result of responses to attacks spoofed with a network's IP address. Figure 7 provides a time-series graph of the backscatter activity seen on the four networks. Not surprisingly, TCP RSTs and SYN-ACKs account for the majority of the scans seen in all four networks. These would be the most common responses to a spoofed SYN-flood (Denial of Service) attack. The figures for the two UW and the Class A networks span the same



(a) Backscatter at UW I

(b) Backscatter at UW II

(c) Backscatter at LBL

(d) Syn-Acks and RSTs in Class A

(e) Remaining Backscatter in Class A

Figure 7: Time series of weekly backscatter in the four networks. Note that Class A is shown in two charts, the second one (e) showing the other components of backscatter besides the dominant RST, SYN-ACK's.

two weeks. The backscatter in the two UW networks looks highly similar both in terms of volume and variability. This can be observed both in the TCP RSTs/SYN-ACKs and the two surges in ICMP TTL-Exceeded shown in Figures 7(a) and (b), and makes sense if the spoofed traffic which is eliciting the backscatter is uniformly distributed across the UW addresses. The only difference between the networks is that UW I tends to receive more "Communication administratively prohibited" ICMP messages than UW II. We do yet have an explanation why. While we see some common spikes in the SYN-ACKS at the Class A and UW networks, there seem to be significant differences in the RSTs. Another notable difference is that the Class A network attracts much more backscatter in other categories, as shown in Figure 7(e).

The LBL graph shown in Figure 7(c) belongs to a different week and displays a quite different pattern than that of UW. We note that the backscatter in the UW networks for the same week (not shown here) shows a very similar pattern as at LBL for the dominant traffic types (TCP RSTs/SYN-ACKs and ICMP TTL-Exceeded). This is not surprising, because the two UW networks and the LBL network belong to the same /8 network. On the other hand, the LBL network seems to receive far fewer scans in the other categories.

A significant portion of ICMP **`host-unreach`** messages we see at Class A are responses to UDP packets with spoofed source addresses from port 53 to port 1026. We first thought we were seeing backscatters of DNS poisoning attempts, but then we found that we are also seeing the UDP packets in other networks as well. Examining these packets reveals that they are not DNS packets, but rather Windows Messenger Pop-Up spams, as discussed in the next section.

5. ACTIVITIES IN BACKGROUND RADIATION

In this section we will first divide the traffic by ports and present a tour of dominant activities on the popular ports. Then we will add the temporal element to our analysis to see how the volume of activities vary over time.

5.1 Details per Port

We rank activities' popularity mostly by number of source IPs, rather than by packet or byte volume, for the following reasons. First, our filtering algorithm is biased against sources that try to reach many destinations, thus affects packet/byte volumes unevenly for different activities. The number of source IPs, however, should largely remain unaffected by filtering, assuming a symmetry among destinations. Also, number of source IPs reflects popularity of the activity across the Internet — an activity with a huge number of sources is likely to be prominent on the whole Internet. Finally, while a single-source activities might be merely a result of an eccentric host, a multi-source activity is more likely to be intentional.

Port/Abbrev.	Activity
80/Get	`"GET /"`
80/GetSrch	`"GET /"  "SEARCH /"`
80/SrchAAA	`"GET /"  "SEARCH /"  "SEARCH /AAA..."`
80/Srch64K	`"SEARCH /\x90\x02\xb1\x02\xb1..."` (65536 byte URI)
135/Bind1	`RPC bind: 000001a0-0000-0000-c000-000000000046`
135/RPC170	`Unknown RPC request (170 bytes)`
135/Bla	`RPC exploit: Blaster`
135/Wel	`RPC exploit: Welchia`
135/RPC-X1	`RPC exploit: Exploit1624a`
135/EP24-X2	`135/tcp/[empty] => 135/tcp/Probe24a => 135/tcp/RPC exploit: Exploit2904a`
445/Nego	`445/tcp/[session negotiation only]`
445/Locator	`"\\<ip>\IPC$ \locator"; RPC exploit: Exploit1896a`
445/Samr-exe	`"\\<dst-IP>\IPC$ \samr" "\\<dst-IP>\IPC$ \srvsvc" CREATE FILE: "[...].exe"`
445/Samr	`"\\<dst-IP>\IPC$ \samr"`
445/Srvsvc	`"\\<dst-IP>\IPC$ \srvsvc"`
445/Epmapper	`"\\<dst-IP>\IPC$ \epmapper"`

Table 5: Abbreviations for Popular Activities

When a source host contacts a port, it is common that it sends one or more probes before revealing its real intention, sometimes in its second or third connection to the destination host. A probe can be an empty connection, *i.e.* the source opens and closes the connection without sending a byte, or some short request, *e.g.*, an HTTP **`"GET /"`**. Since we are more interested in the intention of sources, we choose to look at the activities at a per-session (source-destination pair) granularity rather than a per-connection granularity. Otherwise one might reach the conclusion that the probes are the dominant elements. We consider all connections between a source-destination pair on the given destination port collectively and suppress repetitions. This approach usually gives us a clear picture of activity on each port.

Below we examine the activities on popular destination ports, and for each port we will present the dominant activities. For convenience of presentation, we introduce abbreviations for activity descriptions, as shown in Table 5. We pick an arbitrary day, March 29, 2004, to compare the distribution of activities seen at different networks, LBL, UW (I,II), and the Class A network. We consider the two UW networks as a single network to eliminate possible bias that might occur due to a single filter.

The background radiation traffic is highly concentrated on a small number of popular ports. For example, on Mar 29 we saw 32,072 distinct source IPs at LBL,[1] and only 0.5% of the source hosts contacted a port not among the "popular" ports discussed below. Thus by looking at the most popular ports, we cover much of the background radiation activity.

Note that looking at the ports alone does not allow us to distinguish the background radiation traffic, because many of the popular ports, *e.g.*, 80/tcp (HTTP), 135/tcp (DCE/RPC) and 445/tcp (SMB), are also heavily used by the normal traffic. On the other hand, once we look at the background radiation traffic at application semantic level, it has a very distinctive modal distribution. For example, the activities on port 135 are predominantly targeted on two particular interfaces, and almost all buffer-overrun exploits are focused on one interface. It is worth noting that the activity composition may change dramatically over time, especially when new vulnerabilities/worms appear, *e.g.*, the dominant activity on port 445 is no longer "Locator" after the rise of the Sasser worm. However, we believe the modal pattern will last as long as the background radiation traffic remains highly automated.

[1] Here we ignore the effect of source IP spoofing, since our responder was able to establish TCP connections with most of the source hosts.

Activity	LBL	UW	Class A
Get	5.1%	2.9%	4.6%
GetSrch	5.2%	93.2%	93.4%
SrchAAA	84.2%	0.0%	0.0%
Srch64K	0.9%	1.1%	0.0%
CodeRed	0.6%	0.4%	0.5%
Nimda	0.2%	0.1%	0.2%
Other	3.8%	2.3%	1.3%

Table 6: Port 80 Activities (Mar 29, 2004) Note that to reduce trace size the active responders at UW and Class A do not respond to `"SEARCH /"` to avoid getting the large `SrchAAA` requests.

TCP Port 80 (HTTP) and HTTP Proxy Ports: Most activities we see on port 80 (Table 6) are targeted against the Microsoft IIS server. In most cases, imitating the response of a typical IIS server enables us to attract follow-up connections from the source.

The dominant activity on port 80 is a WebDAV buffer-overrun exploit [39] (denoted as SrchAAA). The exploit always makes two probes: **`"GET /"`** and **`"SEARCH /"`**, each in its own connection, before sending a **`"SEARCH"`** request with a long URI (in many cases 33,208 bytes, but the length can vary) starting with **`"/AAAA..."`** to overrun the buffer. Unlike exploits we see on many other ports, this exploit shows a lot of payload diversity — the URIs can be different from each other by hundreds of bytes, and the difference is not due to byte shifting. More interestingly, the URIs are composed solely of lower-case letters except for a few dozens of Unicode characters near the beginning. The URI appears to be constructed with the Venetian exploit [2], and it will become executable x86 code after Unicode encoding (inserting a byte 0 at every other byte). Besides this exploit, we also see other WebDAV exploits, *e.g.*, one popular exploit (Srch64K) from Agobot carries a fixed 65,536 byte URI.

Old IIS worms, Nimda and CodeRed II, remain visible in the datasets. The CodeRed II worm is almost the same as the original CodeRed II, except shift of a space and the change of expiration date to year 0x8888. We also often see a **`"OPTIONS /"`** followed by a **`"PROPFIND"`** request. As both requests are short, they look like probes. We have not been able to elicit further requests from the sources and do not yet fully comprehend the intention behind such probes. We suspect that they might be scanners trying to obtain a listing of list of scriptable files by sending "translate: f" in the header of the HTTP request [31].

An interesting component of background radiation observed across all networks on the HTTP proxy ports: 81/1080/3128/8000/8080/8888,[2] as well as on port 80, is source hosts using open-proxies to send probes to `tickerbar.net`. A typical request is shown in Figure 8. These requests are from sources abusing a "get rich quick" money scheme from greenhorse.com–a web site pays users money for running tickerbar while they surf the net. By using open-proxies, these sources can potentially appear to be running hundreds of nodes [35]. The Greenhorse website seems to have since been inactivated.

[2] Though some of these ports are not officially assigned to HTTP, the traffic we received almost contained only HTTP requests.

```
GET http://dc.tickerbar.net/tld/pxy.m?nc=262213531 HTTP/1.0
   Host: dc.tickerbar.net
   Connection: Close
```

Figure 8: Typical HTTP request of a tickerbar host

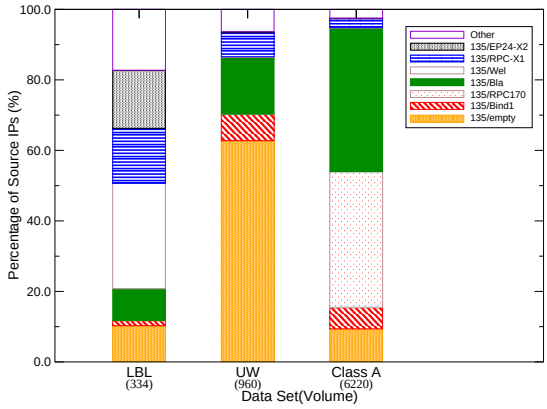


Figure 9: Port 135 activities on Mar 29

TCP Port 135/1025 (DCE/RPC): Port 135 is the Endpoint Mapper port on Windows systems [10] and one of the entry points to exploit the infamous Microsoft Windows DCOM RPC service buffer overrun vulnerability [37]. This vulnerability is exploited by the Blaster worm and the Welchia worm among others.

Figure 9 shows the dominant activities on the port. The Blaster worm was seen on all three networks, but strangely we only saw the Welchia worm at LBL. There were also a number of empty connections without follow-ups and a few types of probes (*e.g.*, 135/RPC170) we do not understand well. Comparing the activity distribution across three networks, the difference is striking and unlike what we see on other ports. This may be due to 1) lack of a single dominant activity and 2) that certain scanning and exploits might be targeted or localized.

On port 1025, which is open on a normal Windows XP host, we see a similar set of exploits. Further, DCE/RPC exploits are also seen on SMB name pipes on port 139 and 445. We will present a closer look of RPC exploit in Section 5.2.2.

TCP Port 139/445 (CIFS): Port 139 is the NetBIOS Session Service port and is usually used on Windows systems for CIFS (Common Internet File System) [7] over NetBIOS. Port 445 is for CIFS over TCP and is also known as Microsoft-DS. When used for CIFS sessions, the two ports are almost identical except that NetBIOS requires an extra step of session setup. Sources simultaneously connecting to both ports prefer port 445 and abandon the port 139 connection. Thus we frequently see empty port 139 connections.

As many Windows services run on top of CIFS there are a great variety of exploits we see on these two ports. Figure 3 shows a snapshot of exploits we see on port 445 at the Class A network. There are basically two kinds of activities: 1) buffer-overrun RPC exploits through named pipes, *e.g.* the Locator pipe [38] or the Epmapper pipe (connected to the endpoint mapper service); and 2) access control bypassing followed by attempts to upload executable files to the target host, *e.g.* as in exploit 445/Samr-exe.

As shown in Table 7, the Locator pipe exploit dominates port 445 activities at all four networks. Besides that, some sources did not go beyond the session negotiation step — the first step in a

Activity	LBL	UW	Class A
445/empty	2.4%	1.3%	0.9%
445/Nego	3.3%	2.4%	3.7%
445/Locator	72.7%	89.4%	89.3%
445/Samr-exe	11.6%	1.8%	1.1%
445/Samr	2.7%	0.8%	0.6%
445/Srvsvc	1.1%	0.4%	0.8%
445/Epmapper	0.8%	0.3%	0.0%
Other	5.4%	3.7%	3.5%

Table 7: Port 445 activities

CIFS session. We also see exploits that first connect to the SAMR (Session Account Manager) pipe, then connect to the SRVSVC pipe and attempt to create an executable file with names such as **`msmsgri.exe`**(W32 Randex.D) [28] and **`Microsoft.exe`** [1]. Finally, by connecting to the Epmapper pipe the sources are exploiting the same vulnerability as on port 135/1025 — note that this activity is not seen at the Class A network.

On port 139, 75% to 89% of source hosts either merely initiate empty connections or do not go beyond the NetBIOS session setup stage, and then migrate to port 445; The dominant activity that we accurately identify are attempts to create files on startup folders after connecting to the SRVSVC pipe **`Xi.exe`**(W32-Xibo) [41].Unlike port 445, we see few hosts attempting to exploit the buffer overflows on the Locator or Epmapper pipe. We also see Agobot variants that connect to the SAMR pipe and drop executables.

TCP Port 6129 (Dameware): Port 6129 is listened by Dameware Remote Control, an administration tool for Windows systems, which has a buffer overrun vulnerability in its early versions [36]. The Dameware exploits we see are similar to those of published exploit programs but do not have exactly the same payload. To launch an exploit, the source host will first send a 40 byte message to probe operating system version and then ship the exploit payload, which is almost always 5,096 bytes long.

On Mar 29, 2004, 62% of the source hosts that connect to port 6129 at LBL[3] close the connections without sending a byte; another 26% abandoned the connections after sending the probe message; and we see exploit messages from the remaining 12% (the number is over 30% on Apr 29). It would be reasonable to question if the large number of abandoned connections suggest that the sources did not like our responders. However, we also find source hosts that would first connect with an empty connection and later came back to send an exploit. Port 6129 is associated with the Agobot that connects a variety of ports (see Section 6.1), and possibilities are that the bots may connect to a number of ports simultaneously and decide to exploit the port that they receive a response from first.

TCP Port 3127/2745/4751 (Virus Backdoors): Port 3127 and 2745/4751 are known to be the backdoor ports of the MyDoom virus and the Beagle viruses, respectively. On most port 3127 connections, we see a fixed 5-byte header followed by one or more Windows executable files uploads. The files are marked by **`"MZ"`** as the first two bytes and contain the string **`"This program cannot be run in DOS mode"`** near head of the file. Running several captured executable files in a closed environment reveals that the programs scan TCP ports 3127, 135, and 445.

On port 2745, the dominant payload we see at LBL and UW is the following FTP URL, which comes after exchanging of one or two short binary messages.

[3]Due to an iSink responder problem we do not have data for the UW and Class A network.

```
"ftp://bla:bla@<src-IP>:<port>/bot.exe\0"
```

On the Class A network, however, we do not see a lot of port 2745 activities. Interestingly, we see several source hosts that attempt to upload Windows executables. We also see many hosts that close the connection after exchange of an initial message.

On port 4751, in some cases we see binary upload after echoing a header, similar to what happens on port 3721, but in most cases we receive a cryptic 24-byte message, and are unable to elicit further response by echoing.

TCP Port 1981/4444/9996: (Exploit Follow-Ups): While worms such as CodeRed and Slammer are contained completely within the buffer-overrun payload, several of the other worms such as Blaster and Sasser infect victim hosts in two steps. First, the buffer-overrun payload carries only a piece of "shell code" that will listen on a particular port to accept further commands; Second, the source then instructs the shell code to download and execute a program from a remote host. For example, on port 4444, the follow-up port for the Blaster worm, we often see:

```
tftp -i <src-IP> GET msblast.exe
start msblast.exe
msblast.exe
```

Similarly, on port 1981 (Agobots) and 9996 (Sasser) we see sequences of shell commands to download and execute a **`bot.exe`**. In contrast, there is a different kind of shell code called "reverse shell" which does not listen on any particular port, but instead connects back to the source host ("phone home"). The port on the source host can be randomly chosen and is embedded in the shell code sent to the victim. The Welchia worm uses a reverse shell (though its random port selection is flawed). This makes it much harder to capture the contents of follow-up connections, because 1) we will have to understand the shell code to find out the "phone-home" port; and 2) initiating connections from our honeypots violates the policy of the hosting networks.

empty).

UDP Port 53: We expected to see a lot of DNS requests, but instead, find sources sending us non-DNS (or malformed) packets as shown below:

```
20:27:43.866952 172.147.151.249.domain > 128.3.x.x.domain: [udp sum ok]
    258 [b2&3=0x7] [16323a] [53638q] [9748n] [259au]
    Type26904 (Class 13568)? [|domain] (ttl 115, id 12429, len 58)
0x0000   (...)
0x0010   xxxx xxxx 0035 0035 0026 xxxx 0102 0007        ................
0x0020   d186 3fc3 2614 0103 d862 6918 3500 d54c        ..?.&....bi.5..L
0x0030   8862 3500 cb1f ee02 3500                       .b5.....5.
```

We do not know what these packets are. These requests dominate UDP packets observed in the LBL and UW (I,II) networks.

Table 8 provides a summary of the DNS activity observed in the Class A network during a 24 hour trace showing a more diverse activity. Much like the UW and LBL networks, sources sending malformed DNS requests dominate. However, in terms of packet counts other queries are substantial. We suspect these are possibly due to misconfigured DNS server IP addresses on hosts. These queries are sent to various destination IP addresses and originate from various networks. Hence it seems unlikely that these are a result of stale DNS entries.

The biggest contributor in terms of volume are standard A queries that resolve IP address for domain names. The SOA packets are "Start of Authority" packets used to register domain authorities. We observed 45 sources (out of total 95) registering different domain authorities in BGC.net. Other queries include PTR queries (used for reverse DNS lookups), SRV records (used to specify locations of services) and AAAA queries (IPv6 name resolution).

UDP Port 137: The activities are dominated by NetBIOS standard name queries (probes).

Type	Num packets	Num sources
Malformed packets	5755	3616
Standard (A) queries	10139	150
Standard query (SOA)	4059	95
Standard query (PTR)	1281	27
DNS Standard query SRV packets	785	20
DNS Standard query AAAA packets	55	16
DNS Standard unused packets	739	3
DNS Standard unknown packets	1485	3

Table 8: Summary of DNS activity seen in the Class A (24 hours)

UDP Port 1026, 1027 (Windows Messenger Pop-Up Spam): These appear as UDP packets with source port 53 and destination port 1026 (or 1027). While this port combination typically connotes a DNS reply, examination of packet contents reveal that they are in fact DCE/RPC requests that exploit a weakness in the Windows Messenger API to deliver spam messages to unpatched Windows desktops [40]. Figure 10 shows a trace of a typical packet. The source IP addresses of these packets are often spoofed, as suggested by the observed **`ICMP host-unreach`** backscatter of these attacks in the Class A. The choice of source port 53 is most likely to evade firewalls.

```
05:23:16.964060 13.183.182.178.domain > xxx.xxx.xxx.xxx.1026:  1024 op5
    [4097q] 68/68/68 (Class 0) Type0[|domain] (DF)
...
0x0010   .... .... .... .... .... .... 0400 a880        ................
0x0020   1001 000a 000a 000a 0000 0000 0000 0000        ................
0x0030   0000 0000 f891 7b5a 00ff d011 a9b2 00c0        ......{Z........
0x0040   4fb6 e6fc 4ba6 e851 f713 8030 a761 c319        O...K..Q...0.a..
0x0050   13f0 e28c 0000 0000 0100 0000 0000 0000        ................
0x0060   0000 ffff ffff 6400 0000 0000 0c00 0000        ......d.........
0x0070   0000 0000 0c00 0000 5265 616c 2057 6f6d        ........Real.Wom
0x0080   656e 0000 0400 0000 0000 0000 0400 0000        en..............
0x0090   596f 7500 3000 0000 0000 0000 3000 0000        You.0.......0...
0x00a0   5741 4e54 2053 4558 3f0d 0a0d 0a46 494e        WANT.SEX?....FIN
0x00b0   4420 5553 2041 543a 0d0a 0d0a 0977 7777        D.US.AT:.....www
0x00c0   2exx xxxx xxxx xxxx xx2e 4249 5a0d 0a00        .********.BIZ...
```

Figure 10: Observed Windows Messenger Pop-Up Spam packets.

UDP Port 1434: The Slammer worm is still alive and is the only background radiation we see on port 1434.

TCP Port 1433: We have not yet built a detailed responder for MS-SQL. It appears that most source hosts are trying to log in with blank passwords.

TCP Port 5000: We do not know enough about this port. The port is reserved for Universal Plug-and-Play on Windows Systems, but almost none of requests we see are valid HTTP requests. However, most requests contain a number of consecutive 0x90's (`NOP`) and thus look like buffer-overrun exploits.

All the ports we examine above exhibit a modal distribution at the application semantic level, *i.e.,* they all contain one or a few dominant elements. The only exception is the DCE/RPC ports, on which we see some diversity, but in some sense, the various exploits on DCE/RPC ports have a single dominant element on a higher level — they target the same vulnerability. As the dominant elements are quite different from what we see in the normal traffic, this suggests that we will be able to filter out the majority of background radiation traffic with a sound classification scheme at the application semantic level.

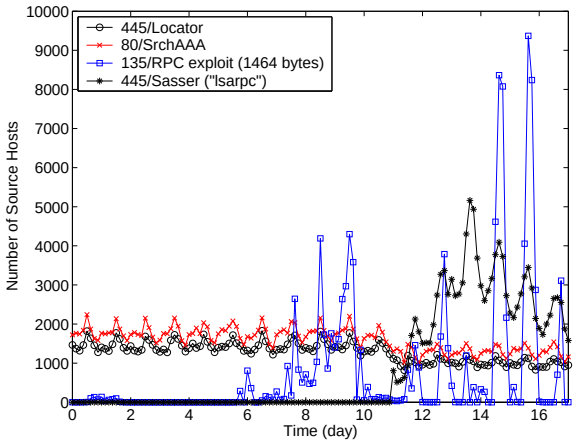


Figure 11: The Big Exploits (Apr 20 to May 7, 2004), as observed on 5 /C networks at LBL. The source hosts are counted every three hours.

5.2 Temporal Distribution of Activities

We will examine two cases of temporal activity. First, we will look at the exploits with the largest source population and consider the distributional variation over time; Second, we look at the exploits targeted at a particular vulnerability and consider how these exploits evolve and diversify over time. We will focus on the LBL network for this analysis.

5.2.1 The Dominant Exploits

Figure 11 shows how the numbers of source hosts vary over the course of 18 days for the four exploits with largest source population.

The source volumes for the SrchAAA and Locator exploits are relatively stable and close to each other over time. This is not surprising because these exploits are likely coming from the same worm, as we will see in Section 6.2.

The other two exploits, Exploit1464 and Sasser, show much a wider range of source volume dynamics — this is especially true for Exploit1464, which temporarily retreated to a much smaller scale around April 30th.

All four exploits demonstrate a strong diurnal pattern, with obvious peaks at local time noon. We do not have good explanations for this pattern. For SrchAAA, Locator, and the Sasser exploits, the peak might be due to hosts being turned on at daytime and doing local-biased search. However, for Exploit1464, the steep narrow peaks lead us to believe they could be caused by the scanning mechanism itself.

5.2.2 DCE/RPC Exploits

DCE/RPC exploits that target the Microsoft DCOM RPC vulnerability [37] present an interesting case of a single well-known vulnerability being used and reused by various worms and/or autorooters. This vulnerability is particularly attractive because it exists on every unpatched Windows 2000/XP system, in contrast to, *e.g.* vulnerabilities that exist only on IIS or SQL servers.

We have seen quite a few different exploit payloads in this data. There are at least 11 different payload lengths. This does not appear to be result of intentional polymorphism, for two reasons: 1) from almost every single source IP we see only one payload length; and 2) it would be easy to vary the length of payload by simply inserting `NOP`'s if the adversary wanted to incorporate some polymorphism. Thus we infer that the diversity of payloads is not due to deliberate polymorphism but due to different code bases. While the payloads themselves might not be very interesting, since the diversity is likely due to the various "shell code" they carry, the diversity offers us an opportunity to look at the rising and ebbing of different exploit programs.

Without a robust way to cluster payloads by contents (payloads of same length sometimes differ on tens to hundreds bytes and the differences are not merely byte shifting), we choose to cluster the exploits by lengths and the ports on which they appear, including port 135/1025 and the Epmapper pipes on port 445/139. Under this scheme, we see more than 30 different exploit types. We select 9 of the popular exploits and consider how the number of source IP addresses for each exploit varies over time during April 2004. The exploits have 4 different payload lengths: 1448, 1464, 2972, and 2904, and are seen on port 135, 1025, and 445.

We observe strong temporal correlation among exploits of the same length for lengths 2792 and 2904, while this is not the case for lengths 1448 and 1464. The four exploits also show some correlation in terms of activity to port 135 and to port 445, which is due to the same source probing both ports. We also find that even for multiple sources, activity for particular port/length pairs tends to come in bursty spikes, suggesting synchronized scanning among the sources.

6. CHARACTERISTICS OF SOURCES

In this section we examine the background radiation activities in terms of source hosts. We associate various activities coming from the same source IP to construct an "activity vector" for each source IP, which we then examine in three dimensions: 1) across ports, 2) across destination networks, and 3) over time.

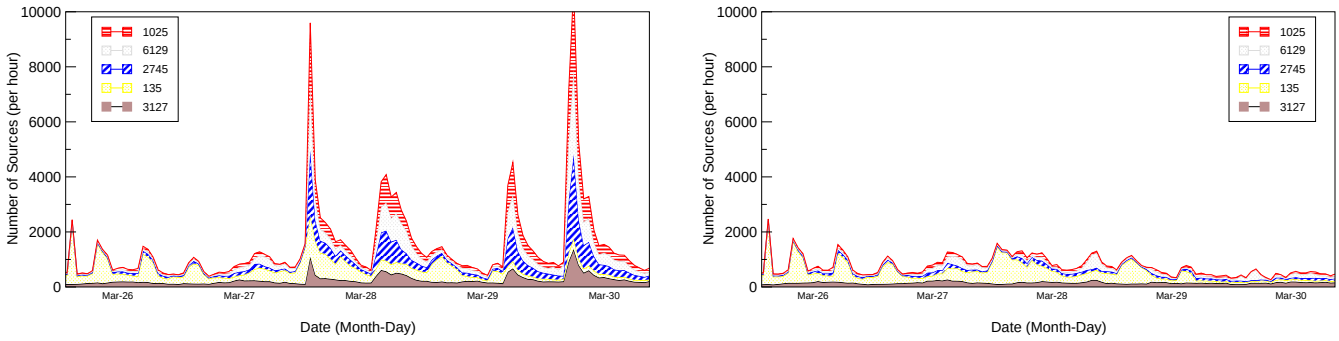
There is a caveat about identifying hosts with IP addresses: due to DHCP, hosts might be assigned different addresses over time. A study [21] concluded that "IP addresses are not an accurate measure of the spread of a worm on timescales longer than 24 hours". However, without a better notion to identify hosts, we will still use IP addresses to identify hosts, while keeping this caveat in mind.

6.1 Across Ports

Associating activities across ports sometimes gives us a significantly better picture of a source's goals. This especially helps with analyzing puzzling activities, because it puts behavior on individual ports in the context of collective activity. For example, simply looking at an RPC exploit may not readily reveal the worm or autorooter that sends it, but once we see a follow-up to port 4444 with `"tftp msblast.exe"`, we know that the earlier exploit comes from Blaster.

Table 9 provides a summary of the top multi-port scanning episodes seen in the four networks. The most common is sources that scan both 139 and 445. Many viruses that exploit NetBIOS/SMB (CIFS) exhibit this behavior since for propagation the ports can be used interchangeably. This next most common multi-port source behavior exploits the Microsoft DCE/RPC vulnerability [37] both via port 135 and by connecting to the Epmapper pipe through port 139 and port 445 during the same episode. These are likely variants of Welchia. Most port 5000 connections are empty and the rest small portion of them look like buffer overrun exploits. We also find Agobot variants that occasionally target these services. They connect to the SAMR pipe through CIFS to obtain registry information, following the sequence described in Figure 4 and drop the file `mdms.exe` into one of the startup folders. The least common profiles are used exclusively by Agobot variants (I, II and III).

We can examine the spatial variance of multi-port profiles by comparing the data collected at each network during the same



(a) Agobot Sources: UW I

(b) Agobot Sources: UW II

Figure 12: Time series of activity on Agobot ports in the two UW /19 networks (on adjacent Class B networks)

Name	Ports	Description	LBL num sources (rank)	UW I num sources (rank)	UW II num sources(rank)	Class A num sources (rank)
NB-1	139,445	Xi.exe (W32-Xibo), msmsgri32.exe (Randex.D), Antivirus32.exe (SDBot.JW)	4,310 (1)	4,300 (1)	4,313 (1)	7,408 (1)
NB-EP1	135,139,445	EP-2704, mdms.exe(Agobot)	1,187 (2)	1,028 (2)	1,046 (2)	537(4)
NB-EP2	135,139,445,5000	EP-2792, EP-2904	780 (3)	678(3)	721(3)	15
Agobot-1	1025,1981,2745,6129	Agobot variant-I	16	452 (4)	68(10)	0
Agobot-2	1025,2745,6129	Agobot variant-II	1	437 (5)	3	0
Agobot-3	80,139,1025, 2745,6129	Agobot variant-III	0	415 (6)	0	0

Table 9: 24 hours of multi-port source activity at the four sites

24 hour period. We order the profiles based on the ranks of all multi-port profiles collected at the UW I network, which showed the greatest affinity to Agobot. The table reveals several notable observations. First, the top two exploits are extensively observed across all four networks and their rankings are consistent in the Class B networks — a *spatial invariant*. Second, although the LBL network is much smaller, it observes the same number of sources as the other two UW networks for the top three exploits. This is probably due to the fact that the networks belong to the same /8, and suggests that these multi-port sources often sweep the address space. Third, UW I receives many more Agobot scans than the other networks, though we don't know why.

We explore Agobot scans in greater detail in Figure 12. The figures show the volume of unique sources per hour on several of the Agobot ports during a five day period in March at the two adjacent UW networks. The graph at UW I shows four visible spikes, indicating an Agobot attack. While UW II background radiation seems to otherwise closely follow UW I, these Agobot spikes are peculiarly absent. These graphs also provide a temporal perspective on the growth of Agobot, with a striking daily spike-followed-by-decay pattern, presumably as new machines are cleaned up over the course of the day.

We see little Agobot on the Class A network. This likely reflects Agobot's "maturity" as malwares go. It has gone through iterative enhancements and likely has (essentially) "evolved" to have been programmed to avoid telescopes; or perhaps it has become equipped with list of target networks, or the scanning is being consciously focused by a human operator via IRC control channels.

Activity	LBL	UW	Class A	LBL ∩ UW	LBL ∩ Class A
All	31K	276K	582K	15K	6.5K
Srch+Loc	76%	85%	57%	75%	91%
Samr-exe	1,601	2,111	2,012	1,634	116
Witty	72	241	162	61	18

Table 10: Traffic from sources seen across networks: intersections vs. individual networks

6.2 Sources Seen Across Networks

We now consider sets of source hosts seen on multiple networks at approximately the same time. We analyze source IPs seen across networks on an arbitrarily chosen day (Mar 29 GMT), characterizing them in terms of: 1) How many such source hosts are there? 2) Do they send the same traffic to different networks? 3) What does the activity distribution look like? and 4) how does it compare to the distribution on individual networks?

As shown in Table 10, source IPs seen at LBL and UW have a surprisingly large intersection set—almost half the source IPs seen at LBL are also seen at UW. In contrast, the intersection of LBL and the Class A is much smaller, even though we are seeing many more source IPs at the Class A than at UW.[4] This contrast may be due to some sources avoiding the Class A networks, and also the proximity of LBL and UW in the IP address space.

[4]Since we only see ICMP Unreachable backscatter only on the Class A network, and these constitute a significant number of source IPs, here we exclude them from the comparison to avoid skewing the activity distribution.

	Mar 29	Mar 30	Apr 29	1-Day ∩	1-Month ∩
All	31K	30K	62K	1,513	680
Srch+Loc	76%	83%	42%	68%	85%
Witty	72	64	0	15	0
Blaster	30	31	24	8	7

Table 11: Traffic from sources seen over time: intersections vs. individual periods

The next evaluation is to confirm that a given source indeed sends the same traffic to the different networks. We extract an activity vector for each source IP on the LBL and UW networks and compare, finding that indeed this is the case, with one peculiarity: while several thousand SrchAAA and Locator sources are common, we also find nearly two thousand Locator-only sources at one network that are SrchAAA-only sources at the other. This turns out to be due to the interaction between source-destination filtering and the scanning mechanism of the SrchAAA/Locator sources. These sources choose, apparently randomly, to send either SrchAAA or Locator to a given destination, but not both.

Finally, what does the activity profile of a given source tell us about how likely we are to see it elsewhere? As shown in Table 10, sources exhibiting the dominant activity profile—SrchAAA and Locator—are often seen at multiple locations in the network. On the other hand, Samr-exe and Witty present an interesting case. The Samr-exe sources we see in the intersection of LBL and UW are more than what we find at LBL alone! (1,634 vs 1,601) This seeming inconsistency is caused by a number of source hosts not completing the exploit when contacting LBL, and thus not being identified there, but doing so at UW. In addition, the Samr-exe population seen at UW is merely 2,061 (0.7%), so we see a surprisingly large overlap for it between LBL and UW. On the other hand, LBL and the Class A have only 116 Samr-exe sources in common, out of more than 2,000 seen on the Class A, suggesting that Samr-exe sources scan with a local bias.

6.3 Sources Seen Over Time

To characterize sources seen at the same network over time, we analyze activity seen at LBL on three days: March 29, March 30, and April 29. This gives us comparisons for adjacent days and one month apart, respectively. Table 11 characterizes the variation. We see that the intersection of source hosts—even in the case of only one day apart—is much smaller than the intersection across networks. While this is partly because the UW network is larger than LBL, looking at the set of sources seen on another LBL network of the same size on March 29 we find more than 5,000 hosts in common. This confirms that we tend to see a larger intersection of source IPs across networks than over time. One effect we have not controlled for here, however, is DHCP artifacts: a host might be assigned different addresses on different days. We also note that one month's time does not greatly further reduce the intersection size, suggesting that if a host does not have the DHCP artifact, then it tends to stay in the intersection. The initial steep decaying of source IP sets also suggest that it will be easier to track a (malicious) host across space than across time.

The number of source hosts seen over time also varies by activity. For example, Witty did not persist over a month (nor could it, as it was a rare instance of a worm that deliberately damages its host); Blaster's grip on hosts is quite tenacious; and the SrchAAA/Locator sources fall in between.

7. SUMMARY

Previous studies of Internet traffic have identified a number of now well-established properties: diurnal cycles in volume; variability in mix across sites and over time; bursty arrivals; the ubiquity of heavy-tailed distributions. Over the past several years, however, an important new dimension of Internet traffic has emerged, and it has done so without any systematic observation or characterization. The gross features of this new breed of traffic are that it is complex in structure, highly automated, frequently malicious, potentially adversarial, and mutates at a rapid pace. Each of these characteristics motivates the need for a deeper understanding of this "unwanted" traffic.

We have presented an initial study of the broad characteristics of Internet *background radiation*. Our evaluation is based on traffic measurements from four large, unused subnets within the IPv4 address space. We developed filtering techniques and active responders to use in our monitoring, analyzing both the characteristics of completely unsolicited traffic (passive analysis) and the details of traffic elicited by our active responses (activities analysis).

Passive analysis demonstrates both the prevalence and variability of background radiation. Evaluation of destination ports reveals that the vast majority of traffic targets services with frequently-exploited vulnerabilities. Analysis of backscatter traffic shows the overall dominance of TCP SYN-ACK/RST packets, but otherwise we do not find a great deal of consistency across the monitored subnets.

Our activities analysis focused on the most popular services targeted by background radiation, finding a rich variegation. Activities across all of the monitored services include new worms released during our study, vestiges of old worms such as Code Red and Nimda, the frequent presence of "autorooter" scans (similar to worms, but without self-propagation), and a noticeable number of connections that are simply empty even when given an opportunity to send data. As with the passive analysis, we find significant diversity across the subnets we monitored, and also over time.

We also examined background radiation from the perspective of source host behavior. Considering source activities across ports reveals consistent behavior in each of the measurement sites for the most prevalent multi-port scan type (scans to both ports 139 and 445). Furthermore, there was an appreciable intersection of sources across measurement sites. This can be explained by the random scanning behavior of worms like Welchia. However, there was a much smaller set of sources common to all measurement sites when they are considered over time.

Perhaps the most striking result of our analysis is the extreme dynamism in many aspects of background radiation. Unlike benign traffic, which only shows major shifts in constituency when new applications become popular (which happens on fairly lengthy time scales), the mix of background radiation sometimes changes on a nearly-daily basis. This dynamism results in a potpourri of connection-level behavior, packet payloads, and activity sessions seen in different regions of the address space.

Our efforts have implications for both the research and operational communities. The ubiquity of background radiation presents significant difficulties for those who monitor Internet traffic: it can clog stateful analyzers with uninteresting activity, and due to its variety it can significantly complicate the detection of new types of activity (for example, a new worm using the same port as existing worms). It is clear from the highly diverse and dynamic activity we have found that further work is needed to both assess the evolution of background radiation over time and to develop more detailed characterizations. We believe that our framework—prefiltering the traffic, using lightweight responders to engage sources in enough

detail to categorize them, analyzing the resulting traffic along the axes we have explored—is an important first step towards comprehensively studying this new phenomenon.

Acknowledgments

Our thanks to the System and Network Security group and the network operations staff of the Lawrence Berkeley National Laboratory for hosting the LBL monitoring setup. We are grateful to Dave Plonka, Jeff Bartig, Geoff Horne and Bill Jensen for their continuing support of the iSink project. We would also like to thank Niels Provos for his help in setting up the Honeyd monitor at LBL, and the anonymous reviewers for their constructive critiques.

This work was supported by ARO grant DAAD19-02-1-0304, NSF grant CCR-0325653, NSF grant ITR/ANI-0205519 and NSF grant EIN-0335214.

8. REFERENCES

[1] W32 Agobot IB. http://www.sophos.com/virusinfo/analyses/trojagobotib.html.

[2] C. Anley. Creating arbitrary shellcode in unicode expanded strings, January 2002. http://www.nextgenss.com/papers/unicodebo.pdf.

[3] M. Arlitt and C. Williamson. Web server workload characterization: The search for invariants. In *Proceedings of SIGMETRICS*, Philadelphia, May 1996.

[4] L. Baldwin, P. Sloss, and S. Friedl. Iraqi Trace. http://www.mynetwatchman.com/kb/security/articles/iraqiworm/iraqitrace.htm.

[5] W32 Beagle.J. http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.j@mm.html.

[6] B. Caswell and M. Roesch. The SNORT network intrusion detection system. http://www.snort.org, April 2004.

[7] Common Internet File System. http://www.snia.org/tech_activities/CIFS/CIFS-TR-1p00_FINAL.pdf.

[8] M. Dacier, F. Pouget, and H. Debar. Attack processes found on the internet. In *Proceedings of NATO Symposium*, 2004.

[9] P. Danzig, S. Jamin, R. Cáceres, D. Mitzel, , and D. Estrin. An empirical workload model for driving wide-area TCP/IP network simulations. *Internetworking: Research and Experience*, 3:1-26, 1992.

[10] DCE 1.1: Remote procedure call. http://www.opengroup.org/onlinepubs/9629399/toc.htm.

[11] Flowreplay Design Notes. http://www.synfin.net/papers/flowreplay.pdf.

[12] B. Greene. BGPv4 Security Risk Assessment, June 2002.

[13] The Honeynet Project. http://project.honeynet.org, 2003.

[14] J. Jung, V. Paxson, A. Berger, , and H. Balakrishnan. Fast portscan detectionusing sequential hypothesis testing. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

[15] H. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *13^{th} USENIX Security Symposium*, San Diego, California, August 2004.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3), August 2000.

[17] C. Kreibich and J. Crowcroft. Honeycomb–creating intrusion detection signatures using honeypots. In *2^{nd} Workshop on Hot Topics in Networks (Hotnets-II)*, Cambridge, Massachusetts, November 2003.

[18] D. Moore. Network telescopes: Observing small or distant security events. Invited Presentation at the 11th USENIX Security Symposium, 2002.

[19] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. In *Proceedings of IEEE Security and Privacy*, June 2003.

[20] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The spread of the sapphire/slammer worm. http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html, 2003.

[21] D. Moore, C. Shannon, and J. Brown. Code red: A case study on the spread and victims of an internet worm. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[22] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of IEEE INFOCOM*, April 2003.

[23] D. Moore, G. Voelker, and S. Savage. Inferring internet denial of service activity. In *Proceedings of the 2001 USENIX Security Symposium*, Washington D.C., August 2001.

[24] W32 Mydoom.A@mm. http://securityresponse.symantec.com/avcenter/venc/data/w32.mydoom.a@mm.html.

[25] V. Paxson. Empirically-derived analytic models of wide-area tcp connections. *IEEE/ACM Transactions on Networking*, 2:316-336, 1994.

[26] V. Paxson. Bro: A system for detecting network intruders in real time. *Computer Networks*, December 1999.

[27] N. Provos. The Honeyd Virtual Honeypot. http://www.honeyd.org, 2003.

[28] W32 Randex.D. http://www.liutilities.com/products/wintaskspro/processlibrary/msmsgri32.

[29] M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proceedings of USENIX LISA*, 1999.

[30] W32 Sasser.Worm. http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html.

[31] Security Focus. Microsoft IIS 5.0 “translate: f” source disclosure vulnerability. http://www.securityfocus.com/bid/1578/discussion/, April 2004.

[32] S. Singh, C. Estan, G. Varghese, and S. Savage. The Earlybird system for real-time detection of unknown worms. Technical Report CS2003-0761, University of California, San Diego, August 2003.

[33] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[34] K. Thompson, G. Miller, and R. Wilder. Wide area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, November 1997.

[35] Make your richer! Get more money easily! http://www.per.rcpt.to/lists/rlinetd/msg01850.html.

[36] DameWare Mini Remote Control Server $<=$ 3.72 buffer overflow. http://www.securityfocus.com/archive/1/347576.

[37] Microsoft Windows DCOM RPC interface buffer overrun vulnerability (MS03-026). http://www.securityfocus.com/bid/8205.

[38] Microsoft Windows Locator Service buffer overflow vulnerability (MS03-001). http://www.securityfocus.com/bid/6666.

[39] Microsoft Windows 2000 WebDAV buffer overflow vulnerability (MS03-007). http://www.securityfocus.com/bid/7116.

[40] Windows Messenger Popup Spam. http://www.lurhq.com/popup_spam.html.

[41] W32 Xibo. http://www.sophos.com/virusinfo/analyses/w32xiboa.html.

[42] V. Yegneswaran, P. Barford, and D. Plonka. On the design and use of internet sinks for network abuse monitoring. In *Proceedings of Recent Advances in Intrusion Detection*, 2004.

[43] V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: Global characteristics and prevalence. In *Proceedings of ACM SIGMETRICS*, June 2003.