

On the Design and Use of Internet Sinks for Network Abuse Monitoring

Vinod Yegneswaran¹, Paul Barford¹, and Dave Plonka²

¹ Dept. of Computer Science, University of Wisconsin, Madison

² Dept. of Information Technology, University of Wisconsin, Madison

Abstract. Monitoring *unused* or *dark* IP addresses offers opportunities to significantly improve and expand knowledge of abuse activity without many of the problems associated with typical network intrusion detection and firewall systems. In this paper, we address the problem of designing and deploying a system for monitoring large unused address spaces such as class A telescopes with 16M IP addresses. We describe the architecture and implementation of the Internet Sink (iSink) system which measures packet traffic on unused IP addresses in an efficient, extensible and scalable fashion. In contrast to traditional intrusion detection systems or firewalls, iSink includes an *active* component that generates response packets to incoming traffic. This gives the iSink an important advantage in discriminating between different types of attacks (through examination of the response payloads). The key feature of iSink's design that distinguishes it from other unused address space monitors is that its active response component is *stateless* and thus highly scalable. We report performance results of our iSink implementation in both controlled laboratory experiments and from a case study of a live deployment. Our results demonstrate the efficiency and scalability of our implementation as well as the important perspective on abuse activity that is afforded by its use.

Keywords: Intrusion Detection; Honeypots; Deception Systems

1 Introduction

Network abuse in the form of intrusions by port scanning or self propagating worms is a significant, on-going threat in the Internet. Clever new scanning methods are constantly being developed to thwart identification by standard firewalls and network intrusion detection systems (NIDS). Work by Staniford *et al.* [27] and by Moore *et al.* [18] project and evaluate the magnitude of the threat of new classes of worms and the difficulty of containing such worms. The conclusions of both papers is that addressing these threats presents the research and operational communities with serious challenges. An important step in protecting networks from malicious intrusions is to improve measurement and detection capabilities.

One means for improving the perspective and effectiveness of detection tools is to monitor both used *and* unused address space in a given network. Monitoring the unused addresses is not typically done since packets destined for those addresses are often dropped by a network's gateway or border router. However, tracking packets sent to unused addresses offers two important advantages. First, other than misconfigurations, packets destined to unused addresses are almost always malicious, thus false positives

– a significant problem in NIDS – are minimized. Second, unlike NIDS that monitor traffic passively, a detection tool that monitors unused addresses can actively respond to connection requests, thus enabling the capture of data packets with attack-specific information. The possibility for unused address space monitoring is perhaps most significant in class A and class B networks where the number of unused addresses is often substantial. The idea of monitoring unused address space has been adopted in a number of different studies and on-going projects including the DOMINO project [31], the HoneyNet project [29], LaBrea tarpits [14] and in the backscatter analysis conducted by Moore *et al.* in [19].

This paper makes two contributions. The first is our description of a new system architecture and implementation for measuring IP traffic. An *Internet Sink* or iSink, is a system we developed for monitoring abuse traffic by both active and passive means. The key design requirements of an iSink are extensibility of features and scalability of performance since it is meant to be used to monitor potentially large amounts of IP address space.

Our design of an iSink includes capabilities to trace packets, to actively respond to connection requests, to masquerade as several different application types, to fingerprint source hosts and to sample packets for increased scalability. The passive component of our implementation (which we call Passive Monitor) is based on Argus [3] – a freely available IP flow measurement tool. The active component of our implementation (which we call Active Sink) is based on the Click modular router platform [12]. Click is an open-source toolkit for building high performance network systems on commodity hardware. The focus of Active Sink’s development was to build a set of *stateless responder elements* which generate the appropriate series of application level response packets for connections that target different network services including HTTP, NetBIOS/SMB and DCERPC (Windows RPC Service).

The second contribution of this paper is a measurement and evaluation case study of our iSink implementation. We use the results from the case study to demonstrate the scale and diversity of traffic characteristics exposed by iSink-based monitoring. These results provide validation of our architectural requirements and rationale for subsequent evaluation criteria. We also deployed the iSink *in situ* to monitor four class B address spaces within our campus network for a period of 4 months and one entire class A address space to which we have access. From these data sets we report results that demonstrate the iSink’s capabilities and the unique information that can be extracted from this measurement tool. One example is that since the traffic characteristics from our class B monitor are substantially different from those on the class A monitor, we conclude that the location of the iSink in IP address space is important. Another example is that we see strong evidence of periodic probing in our class A monitor which we were able to isolate to the LovGate worm [2]. We also uncovered an SMTP hot-spot within the class A network that has been unreported prior to our study. We were able to attribute that anomaly to misconfigured wireless routers from a major vendor. Finally, we assess basic performance of the iSink in controlled laboratory experiments and show that our implementation has highly scalable response capability.

These results demonstrate that our iSink architecture is able to support a range of capabilities while providing scalable performance. The results also demonstrate that

Active iSinks are a simple and very useful way to extend basic intrusion monitoring capabilities in individual networks or in the Internet as a whole.

2 Related Work

The notion of monitoring unused IP addresses as a source of information on intrusions has been in use in various forms for some time. While we coin the terms “Internet Sink” and “iSink”, these monitors have variously been referred to as “Internet Sink-holes” [8], “Blackhole Routers” [9] and “Network Telescopes” [15]. Traditional *Honeypots* are defined as systems with no authorized activity that are deployed with the sole purpose of monitoring intrusions. *Honeynets* are network of honeypots (typically set up as VMware hosts). Their deployment is often associated with significant management and scalability challenges [29]. In [15], Moore raises the challenges of deploying honeypots in a class A network telescope. The systems that are perhaps most similar to the Active Sink have been developed in the Honeyd [10] and Labrea Tarpit projects [14]. Active Sink’s design differs in significant ways from these two systems. Much like the Active Sink, Honeyd is designed to simulate virtual honeypots over unused IP addresses, with the potential for a diverse set of interactive response capabilities. However, Honeyd’s stateful active responder design has significant scalability constraints that make it inappropriate for monitoring large IP address ranges which is one of iSinks primary objectives. LaBrea’s primary design objective is to slow the propagation of Internet worms (*i.e.*, a sticky honeypot), and as such, it lacks the richness of interaction capabilities that is required to gather important response information. In addition to a richer response set, our Active Sink’s performance greatly exceeds that of LaBrea as will be seen in Section 5.

There are a number of empirical studies of intrusion and attack activity that motivate and inform our work. In [33], the authors explore the statistical characteristics of Internet intrusion activity from a global perspective. That study is based on the use of intrusion logs from NIDS and firewalls located broadly across the Internet. Moore *et al.* examined the global prevalence of denial-of-service attacks using backscatter analysis in [19]. That work was conducted by gathering packet traces from a relatively quiescent class A network. Characteristics of the Code Red worm have been analyzed in a number of studies. In [17] the authors investigate the details of the Code Red outbreak and provide important perspective on the speed of worm propagation. Moore *et al.* provide further insights on the speed at which countermeasures would have to be installed to inhibit worms propagation [18]. While the prospects for successful containment are rather grim, it is clear that rapid detection will be a key component in any quarantine strategy.

Intrusion detection systems are a standard component in network security architectures. These tools typically monitor packet traffic at network ingress/egress points and identify potential intrusions using a variety of techniques. Standard methods for intrusion identification include misuse detection (*eg.* [21, 25]), statistical anomaly detection (*eg.* [26]), information retrieval (*eg.* [1]), data mining (*eg.* [13]), and inductive learning (*eg.* [28]). Our work is distinguished from general NIDS in that they operate on active IP addresses and must deal with the problem of identifying the nefarious traffic mixed

in with all of the legitimate traffic. We expect iSinks and NIDS to complement each other in future operational environment.

High performance packet monitors have been used for collecting packet traces in the Internet for years. These systems relate directly to our iSink design in that they must scale to reliably log packets on very high speed links. Examples of these include systems that have been developed with a variety of commodity and special purpose hardware such as [4, 7, 11]. Our iSink differs significantly from these systems (as well as the NIDS mentioned above) in that it not only passively monitors and logs packets, but it also *actively responds* to incoming TCP connection requests and has application level response capability.

3 Internet Sink Architecture

In this section we describe the iSink requirements, architecture and implementation. The implementation is described within the context of deployments on two different sets of address spaces.

3.1 Design Requirements

The general requirements for an iSink system are that it possess scalable capability for both passive and active monitoring and that it be secure. We discuss the issues of security in more detail in [32].

Passive monitoring capability must be able to capture packet header and payload information accurately. While there are many standard tools and method for packet capture, if either these or new tools are employed, they should be flexible and efficient in the ways in which data is captured and logged.

Active response capability is included in iSink's design as a means to gather more detailed information on abuse activity. This capability is enabled by generating appropriate response packets (at both transport and application levels) to a wide range of intrusion traffic. While active responses also have the potential to interfere with malicious traffic in beneficial ways such as tarpitting, this is not a focus of iSink's design.

We expect Internet Sinks to measure abuse activity over potentially vast unused IP address spaces. For example, in our experimental setup, we needed the ability to scale to an entire class A network (16 million addresses). With the continued growth in malicious Internet traffic, and transition to IPv6, we expect the scalability needs to grow significantly for both the active and passive components of our system. Our basic approach to scalability is to maintain as little state as possible in our active responders. Another means for increasing scalability is through the use of sampling techniques in both active and passive components of the system. If sampling is employed, then the measurement results must not be substantially altered through their use.

Finally, our intent is to develop iSink as an open platform, thus any systems that are used as foundational components must be open source.

3.2 Active Response: The Design Space

In this section we explore the architectural alternatives for sink-hole response systems. The choices we consider are LaBrea, Honeyd, Honeynets and Active Sink (iSink's active response system) as shown in Table 1. We compare these systems based on the following characteristics.

Table 1. Design Space of Sink-Hole Responders

	Configurability	Modularity	Flexibility	Interactivity	Scalability
Active Sink	High	High	High	Low-Medium	High
Honeyd	High	Low-Medium	High	Low-Medium	Low-Medium
Honeynet	Low	Medium	Medium	High	Low-Medium
LaBrea	Low	Low	Low	Limited	High

1. **Configurability** describes the ability of the configuration language to define the layout and components of response networks. Honeyd's strengths are in fine-grained control of virtual network topologies and network protocol stacks. However Honeyd's language does not provide support for assigning large blocks of IP addresses to templates (except for the default template)¹. Active Sink's configuration language (inherited from Click) uses a BPF like language and provides excellent support for both fine-grained and coarse-grained control of a virtual network topology. Active Sink's design is stateless and hence does not replicate network stack retransmission timers. LaBrea and Honeynets only allow for limited configurability.
2. **Flexibility** relates to the ability to mix and match services with operating systems. For example, the ability to define two types of Windows Servers: one with a telnet service and FTP service and another with NetBIOS Service and a Web server. The design of Honeyd and Active Sink both provide a high degree of flexibility. It is somewhat harder to do the same with Honeynets. LaBrea's flexibility in this regard is limited as it was designed with a different objective.
3. **Modularity** describes the ability to compose and layer services on top of one another. For example, layering Server Message Block (SMB) service over NetBIOS or layering Web services over SSL. Active Sink's design is inherently modular which directly facilitates service composition. In contrast, Honeyd's design is more monolithic and hence less straightforward to layer services.
4. **Interactivity** refers to the scope of response capability. The levels of interactivity of Honeyd and Active Sink are comparable. Obviously, Honeynets could provide more complete response capabilities. However, to mitigate the risk of Honeynets being used as a stepping-stone for additional attacks, data controls are required to be placed which limit interactivity. There are other practical configuration issues that also could limit interactivity. For example, Active Sink's NetBIOS responder grants session requests for all NetBIOS names and all user/password combinations, while a Honeynet Windows monitor would only allow NetBIOS session requests if it matches its list of valid names. Hence, the realized degree of interaction in Active Sinks are often higher than honeynets.
5. **Scalability** refers to the number of connections that can be handled in a given time period. In our monitoring environment we typically see hundreds of thousands of connection attempts per minute. Active Sink's stateless kernel module design provides high degree of scalability by eliminating unnecessary system calls and inter-

¹ This feature is particularly necessary for large network sinks.

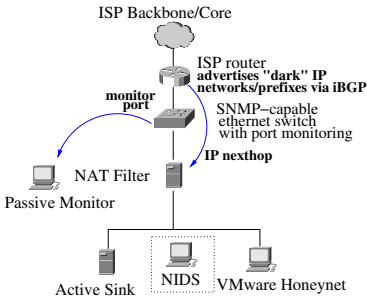


Fig. 1. Internet Sink Implementation. In our current implementation the NIDS is run offline

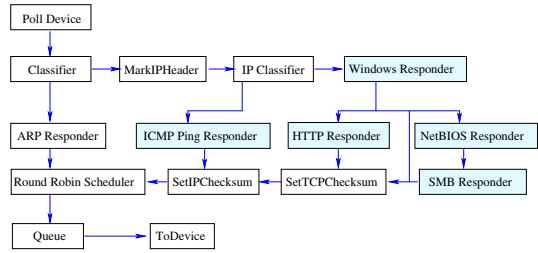


Fig. 2. Active Sink Configuration based on Click's modular design. Shaded elements are iSink extensions

rupt handling overheads². LaBrea's stateless design also provides reasonable scaling properties, however its user level implementation makes it inferior to the Active Sink. A weakness of Honeyd's design is its inherent statefulness that limits its scalability³. Our experience suggests that Honeyd works well in environments that see tens of connection attempts per minute. The scalability of Honeynet systems vary from low to medium depending on the service and licensing issues.

3.3 Implementation

The objective of our monitoring infrastructure implementation was to create a highly scalable backplane with sufficient interactivity to filter out known worms, attacks and misconfiguration. To accomplish this, the iSink design includes a Passive Monitor, an Active Sink and a Honeynet component. Unsolicited traffic can be directed to each of these components which provide unique measurement capabilities. These components, in addition to MRTG [20] and FlowScan [23], were run on Linux-based commodity PCs. Details of our implementation as illustrated in Figure 1 and include:

1. **Passive Monitor** – This component is based on Argus which is a generic libpcap based IP network auditing tool. It allows for flow level monitoring of sink traffic and can be interfaced with FlowScan which is a flow level network traffic visualization tool.
2. **Active Sink** – The standard collection of elements provided with Click enabled many of the basic capabilities required for building active responses in iSink. Figure 2 illustrates iSink's configuration based on Click's modular design. Some of the fundamental elements include: (i) Poll Device which constantly polls the interface for new packets; (ii) IP Classifier which routes ARP packets to the ARP Responder, ICMP ping packets to the Ping Responder and TCP packets to the Windows

² Click also provides the flexibility to be run as a userlevel module which greatly simplifies debugging and development.

³ Honeyd forks a process per connection attempt. A more recent version of Honeyd includes support for python threads. However, scalability improvements are limited by the overhead of the python interpreter.

Responder (all other packets are discarded); (iii) Windows Responder which responds to connection attempts on open ports and forwards HTTP requests to the Web Responder and SMB data packets to the NetBIOS Responder. The application responders developed specifically for iSink are shaded. As far as we know, we are the first non-commercial HoneyPot system to provide emulation capabilities for Windows Networking(NetBIOS/SMB/CIFS) and DCERPC. The current suite of responders that are available also includes an HTTP responder, an SMTP responder, an IRC responder, Dameware responder and a responder for backdoor ports such as MyDoom and Beagle.

Stateless responders are enabled by the following two observations:

- (a) It is almost always possible to concoct a suitable response just by looking at the contents of the request packet from the client – even for complex protocols like SMB. Knowledge of prior state is not compulsory.
 - (b) We need to continue the packet exchange only until the point where we can reliably identify the worm/virus.
3. **NAT Filter** – The motivation behind filtering is to reduce the volume of traffic generated by active responders. This module serves two purposes. It routes requests to appropriate responders (Active Sink or Honeynets) through network address translation. It also filters requests that attempt to exploit known vulnerabilities or misconfiguration. This makes mapping of iSinks more difficult and increases scalability of analysis daemons that have to process large volumes of data. We experimented with several filtering strategies:

For each source IP allow only:

- (a) first N connections
- (b) first N connections per <destination port>
- (c) connections to first N destinations IPs targeted by the source

Of the three strategies, **option (c)** [N destination IPs per source IP] seemed the most attractive. The performance of options (a) and (c) were comparable. They both provided two orders of reduction in the volume of packets and bytes) and were significantly better than option(b). We chose **option (c)** because it has the additional advantage of providing a *consistent view* of the network to the scan sources thus allowing the iSink to appear as if it were a subnet with N live hosts⁴.

- 4. **VMware Honeynets** – These are, quite simply, commodity operating systems running on VMware. Currently, we route packets of services for which we don't have complete responders to fully patched Windows systems.
- 5. **NIDS** – This system can be used to evaluate the packet logs collected at the filter. We plan to implement support for NIDS rules that can communicate with the filter and implement real time filtering decisions. For example, the decision to route packets or migrate connection to VMware Honeynet could be triggered upon the absence of a signature in the NIDS ruleset for the connection.

For this study, we built and deployed two separate iSinks: a “campus-enterprise” iSink and a “service-provider” iSink. These were used to assess our iSink design and demonstrate its capabilities.

⁴ The set of N destination hosts varies with each source depending on the order in which the source scans the address space.

3.4 Deployment: Campus-Enterprise Sink

The campus iSink received unsolicited traffic destined for approximately 100,000 unused IPv4 addresses within 4 sparsely-to-moderately utilized class-B networks that are in use at our campus. Essentially, these unused addresses are in the “holes” between active subnets, each of which typically contains 128 to 1024 contiguous host addresses (*i.e.*, 25 through 22-bit netmasks, respectively).

A so called “black-hole” intra-campus router was configured to also advertise the class B aggregate /16 routes into the intra-campus OSPF. The result was that there were persistent less-specific (16 bit netmask) routes for every campus address. Unsolicited traffic, whether from campus or outside sources, destined for unused campus IP addresses always “falls through” to those less-specific /16 routes, and therefore is routed to the iSink and measured. Furthermore, *occasionally* traffic destined for campus addresses that are normally in use can fall through to the iSink if its subnet’s more specific route disappears. Typically, this only happens during network outages, making the iSink a potential warning system of problems because it can passively detect routing failures. Whenever traffic that was destined for a campus IP address known to be in use reaches the iSink instead, the operators know that there is a problem.

It was important in our environment that the iSink machine was not capable of actively participating in the intra-campus routing, other than to respond via ARP as the IP nexthop on its transit link. The iSink is not an OSPF router, but instead is the destination of a static route. This limits the possible damage that could be caused if ever the iSink system was compromised and was attempted to be used maliciously.

3.5 Deployment: Service-Provider Sink

The service-provider iSink received unsolicited traffic destined for 16 million IPv4 addresses in one class A network. An ISP router, located at our campus’ service-provider, served as the gateway for the service-provider iSink. The service-provider was responsible for advertising the class A network via BGP to our service provider’s commercial transit providers, Internet2’s Abilene network, and to various other peers. SNMP-based measurements at the Ethernet switch’s ports were used to compute any packet loss by the libpcap-based Argus software.

4 Experiences with Internet Sink

Investigating Unique Periodic Probes. The periodicity observed in the service provider iSink data is an excellent example of the perspective on intrusion traffic afforded by iSink. The first step in our analysis of this periodicity was to understand the services that contributed to this phenomenon. We found that most of the periodicity observed in the TCP flows could be isolated to sources scanning two services (port 139 and 445) simultaneously. Port 139 is SMB (Server Message Block protocol) over NetBIOS and port 445 is direct SMB. However, this did not help us isolate the attack vector because it is fairly common for NetBIOS scanners to probe for both these services. Passive logs provided three additional clues: 1) scans typically involve 256 successive IP

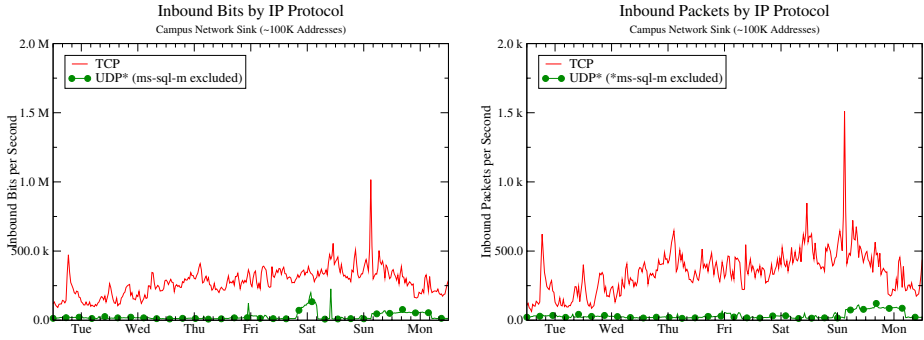


Fig. 3. Inbound Traffic for a Typical Week on Campus-Enterprise Sink (bits/pkts per second)

addresses that span a /24 boundary, 2) the probes had a period of roughly 2.5 hours, 3) the small timescale periodicity seemed to be super imposed over a diurnal periodic behavior at larger timescales.

Figure 6 shows the number of flows scanning both services in a week. To simplify our analysis we then focused on a single day’s data and classified scanners on these services based on their scan footprints. We defined scanners that match our profile (between 250-256 successive IP addresses spanning a /24 boundary) as *type-1* sources. We also defined sources that scan five or more subnets simultaneously as *type-5* sources. This includes processes that pick destination IP addresses randomly and others that are highly aggressive. Figure 7 shows a time-volume graph of the *type-1* and *type-5* scanners. **The interesting aspect of this figure is that the number of sources in each peak (around 100) is more than an order of magnitude smaller than the total number of participants observed in a day (2,177).** We can also see that most of the diurnal behavior could be attributed to type *type-5* sources.

This mystery motivated our development of NetBIOS and SMB responders. By observing the packet logs generated by the active response system we concluded that the scanning process was the LovGate worm [2] which creates the file `NetServices.exe` among others.

This section demonstrates iSink’s capabilities and illustrates the complementary roles of the Passive Monitor and the Active Sink using results from our two iSink deployments. We first discuss issues of perspective by comparing the passive-monitoring results observed in the campus-enterprise sink with that of the service-provider sink. We then demonstrate the utility of the Active Sink in investigating network phenomenon revealed by the Passive Monitor including periodic probing and SMTP hot-spots.

4.1 Campus Enterprise iSink Case Study

Because the campus iSink is located inside one autonomous system and advertised via the local interior routing protocol, this system sees traffic from local sources in addition to traffic from sources in remote networks. Traffic observed from local sources included:

- Enterprise network management traffic attempting to discover network topology and address utilization (such as ping sweeps and SNMP query attempts)

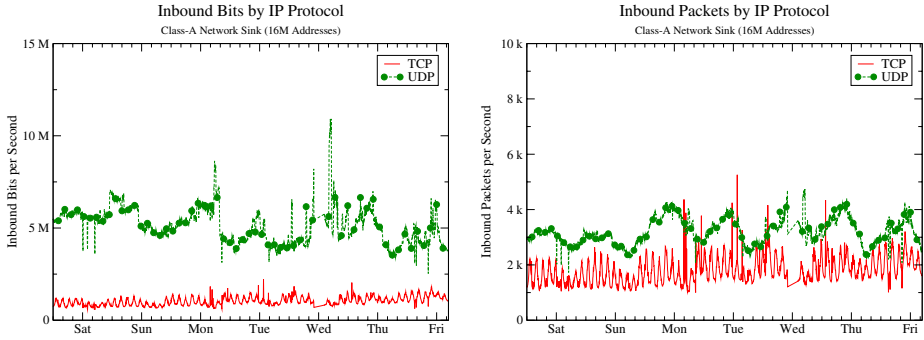


Fig. 4. Inbound Traffic for a Typical Week on Service Provider Sink (bits/pkts per second)

- Traffic from misconfigured hosts. For instance, a few hosts continually send domain queries to what is now an unused campus IP address. Presumably, an operational DNS server used to be at that address. We also see traffic from misconfigured AFS clients and NetBIOS name registration requests from local windows hosts with incorrect WINS address.
- Malicious probes and worm traffic that has an affinity for hosts within their classfull network.

Figure 3 shows the traffic observed from only remote sources in a typical week at the campus-enterprise iSink. There are several notable features. The dominant protocol is TCP since the campus border routers filter scans to port 1434 (ms-sql-m) that was exploited by the SQL-Slammer worm [16]. The peak rate of traffic is about 1Mb/s and 1500 packets per second. There is no obvious periodicity in this dataset. Finally, because TCP is the dominant protocol, the packet sizes are relatively constant and the number of bytes and packets follow a predictable ratio. Hence, the graphs of bit and packet rate show very similar trends.

4.2 Service Provider iSink Case Study

The volume of unsolicited inbound traffic to the class A network varied between average rates of 5,000 packets-per-second (pps) when we brought the system on line to over 20,000pps six months later at the end of our study. One consequence that was relayed to us by experienced network operators is that it is not possible to effectively operate even this relatively quiescent class A network at the end of a 1.5 megabit-per-second T1 link because the link becomes completely saturated by this unsolicited traffic.

To operate the service-provider iSink continuously, we originally assumed that we could safely introduce the class A least-specific /16 route for the iSink and still allow operators to occasionally introduce more-specific routes to draw the network’s traffic elsewhere in the Internet when need-be. While sound in theory (according to “CIDR and Classful Routing” [24]), it didn’t work in practice. Because today’s Internet is bifurcated into commercial/commodity networks and research/education networks (Internet2’s Abilene), some institutions connected to both types employ creative routing policies. We found that some sites prefer less-specific routes over more-specific *when*

Table 2. Top Services (Service Provider Sink)

Service:	Inbound flows per second
udp_netbios-ns_dst	1932
udp_ms-sql-m_dst	1187
http_dst	197
netbios-ssn_dst	133
microsoft-ds_dst	115
smtp_dst	67
http_src	44
https_dst	11
ms-sql-s_dst	10
telnet_dst	2

Table 3. Backscatter sources (victims) in service provider sink (12 hrs – 5 min avg)

Type	Num IPs	% IPs
TCP_RST	295	38%
TCP_SYN_RST	105	14%
TCP_ACK	81	10%
TCP_ACK_RST	80	10%
ICMP_INTRANS_TIME_EXCEEDED	58	7%
ICMP_PORT_UNREACH	29	4%
ICMP_PKT_FILTERED_UNREACH	23	3%
TCP_SYN_ACK	10	1%
ICMP_HOST_UNREACH	6	1%
OTHER	87	11%

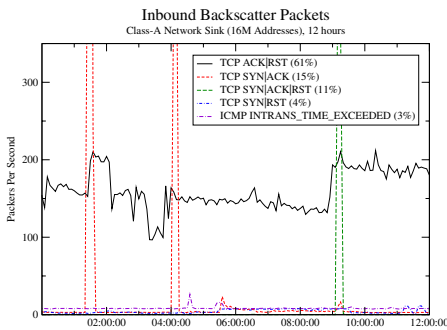


Fig. 5. Time-volume graph of backscatter packet types on service-provider sink over a typical 12 hour period

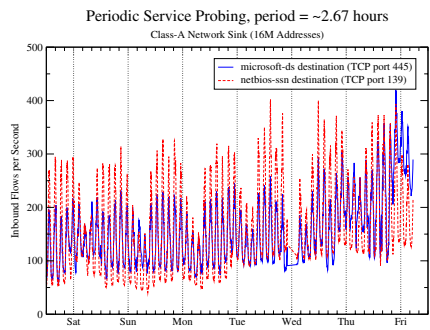


Fig. 6. Inbound flows (per second) observed at service-provider sink on ports 139 and 445 over a typical week

the less-specific route is seen on what is likely to be a higher-performance (or fixed cost) service such as Internet2.

Figure 4 depicts the traffic observed in a typical week at the service-provider iSink. Unlike the campus-enterprise network, the dominant protocol is UDP, most of which can be attribute to Windows NetBIOS scans on port 137 and the ms-sql-m traffic from worm attempting to exploit the vulnerable MS-SQL monitor. Since UDP traffic with payloads of varying sizes dominates, there is no strong correspondence between the graphs for bytes and packets. The most interesting feature is the striking periodic behavior of the TCP flows, discussed in more detail in the section 4. Table 2 provides a summary of the inbound per second flow rate of the top services.

Analysis of Backscatter Packets. Backscatter packets are responses to spoofed DoS attacks and have been effectively used to project Internet wide attack behavior [19]. Figure 5 provides a time series graph of the backscatter packet volume observed in our service-provider sink. Noteworthy features include the following:

1. TCP packets with ACK/RST dominate as might be expected. This would be the most common response to a SYN flood from forged sources.

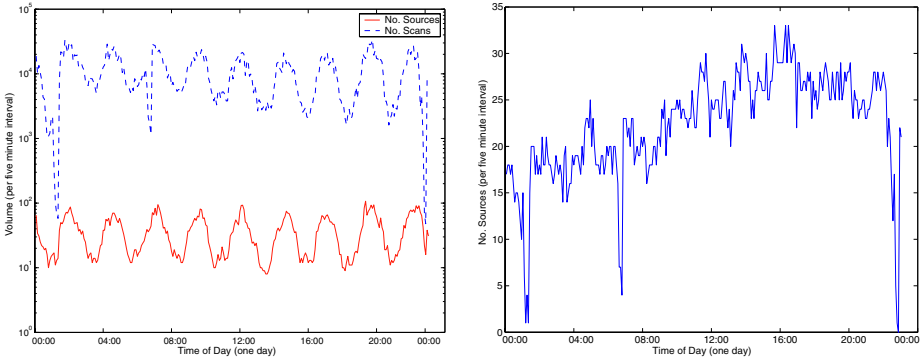


Fig. 7. Left: Volume/Count of *type-1* port 139 scanners: 24 hours, Dec 14, 2003, (no. sources per peak = 100, total sources = 2,177) Right: Volume of *type-5* port139 scanners

2. Vertical lines that correspond to less common short duration spikes of SYN/ACK and SYN/ACK/RST.
3. ICMP TTL exceeded packets could be attributed to either routing loops or DoS floods with a low initial TTL.

Table 3 provides a summary of the number of active sources of backscatter traffic, *i.e.*, the estimated count of the victims of spoofed source attacks. These numbers are an average during the 12 hours shown in Figure 5 of the number of sources in each 5 minute sample. In terms of the distribution of the volumes of Backscatter scan types, our results are consistent with those published in [19]. Backscatter made up a small percentage (under 5%) of the overall traffic seen on our service-provider sink.

We proceeded to setup a controlled experiment which began by trying to infect a Windows 2000 host running on VMware with LovGate. LovGate uses a dictionary attack, so we expected a machine with blank administrative password to be easily infected. However, the NetBIOS sessions were continually getting rejected due to NetBIOS name mismatches. So we modified the lmhosts file to accept the name *SMB-SERVER enabling us to capture the worm.

We verified that LovGate’s NetBIOS scanning process matched the profile of the *type-1* scanners⁵. To date, we have not been able to disassemble the binary as it is a compressed self-extracting executable. So we monitored the scans from the infected host. There were two relevant characteristics that provide insight into the periodicity: 1) The scanning process is deterministic, *i.e.*, after every reboot it repeats the same scanning order 2) During the course of a day there are several 5-10 minute intervals where it stops scanning. Our conjecture is that these gaps occur due to approximately synchronized clocks in the wide area thus producing the observed periodicity.

SMTP Hot-Spot. Analysis of SMTP (Simple Mail Transfer Protocol) scans in the service provider sink is another important demonstration of active sink’s capabilities. From passive measurements, we identified an SMTP hot-spot *i.e.*, there was one IP

⁵ Besides the NetBIOS scanning LovGate also sent SMTP probes to `www.163.com`.

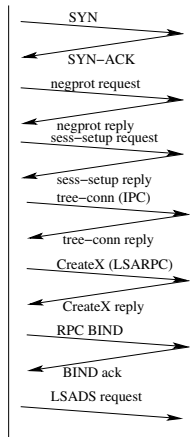


Fig. 8. RBOT.CC timeline of lsarpc exploit(port 445)

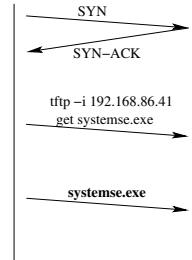


Fig. 9. RBOT.CC follow-up commands(port 44445)

address that was attracting a disproportionately large number of SMTP scans (20-50 scans per second). Hot-spots in unused address space are typically good indicators of misconfiguration. During a 10 day period in December we observed over 4.5 million scans from around 14,000 unique IP addresses all bound to one destination IP within our monitor. A cursory analysis suggested that these scans were all from cable-modem and DSL subscribers. Finally, the scans also seemed to have an uncommon TCP SYN fingerprint (win 8192, mss 1456).

The possibility of spam software as a source of this anomaly was ruled out due to the non-standard TCP fingerprint. We then hypothesized that this could be from a specific cable-modem or DSL device. We set up an SMTP responder on the target IP address and captured the incoming email. **This revealed the source of the email to be misconfigured wireless-router/firewall systems from a major vendor⁶.** The emails are actual firewall logs!

To better understand the reasons behind this SMTP hot-spot, we examined the firewall system’s firmware. The `unrarj` utility was used to extract the compressed binary. However, searching for the hot-spot IP address string in the binary proved fruitless. Examination of the firmware “application” revealed that there was an entry for SMTP server that was left blank by default. This led us to conjecture that the target IP address was the result of an uninitialized garbage value that was converted to a network ordered IP address. It also turns out that every byte in our hot-spot address is a printable ASCII character. So we searched for this four byte ASCII string and found a match *in almost all versions of firmware for this device*. The string occurred in both the extracted and compressed versions of the firmware. As a sanity check, we looked for other similar ASCII strings, but did not find them. These kind of hot-spots can have very serious ramifications in network operations. For example, one the authors discovered a similar problem with Netgear routers that inadvertently flood our campus NTP servers [22].

⁶ We are in the process of notifying the manufacturer and plan to reveal the name of the vendor once this is completed.

Experiences with Recent Worms. Our iSink deployment has proved quite useful in detecting the advent of recent worms such as Sasser [5]. Without active response capability, such as that provided by the Active Sink, it would be impossible to distinguish existing worm traffic on the commonly exploited ports such as port 445 from new worm activity. Detection of such new worms is often possible without modifications to the responder, as was the case for the `lsarpc` exploit used by Sasser. Our active response system enabled accurate detection of not only Sasser, but also more fine-grained classification of several variants. Prior to the release of Sasser, we were also able to observe early exploits on the `lsarpc` service which could be attributed to certain strains of Agobot. Figures 8 and 9 illustrate the interaction of RBOT.CC [30], a more recent virus that also exploits the `lsarpc` vulnerability, with the Active Sink.

5 Basic Performance

One of the primary objectives of the iSink’s design is scalability. We performed scalability tests on our Active Sink implementation using both TCP and UDP packet streams. The experimental setup involved four 2GHz Pentium 4 PCs connected in a common local area network. Three of the PCs were designated as load generators and the fourth was the iSink system that promiscuously responded to all ARP requests destined to any address within one class A network. Figures 10 demonstrates the scalability under of LaBrea⁷ and Active Sink under TCP and UDP stress tests. The primary difference between the TCP and UDP tests is that the TCP connection requests cause the iSink machine to respond with acknowledgments, while the UDP packets do not elicit a response. Ideally, we would expect the number of outbound packets to equal the number of inbound packets. The Click-based Active Sink scales well to TCP load with virtually no loss up to about 20,000 packets (connection attempts) per second. LaBrea performance starts to degrade at about 2,000 packets. The UDP test used 300 byte UDP packets (much like the SQL-Slammer worm). In this case, both the LaBrea and Active Sink perform admirably well. LaBrea starts to experience a 2% loss rate at about 15,000 packets/sec.

6 Sampling

There are three reasons why *connection sampling* can greatly benefit an iSink architecture: (i) *reduced bandwidth requirements*, (ii) *improved scalability*, (iii) *simplified data management and analysis*. In our iSink architecture, we envision building packet-level sampling strategies in the Passive Monitor and source-level sampling in the NAT Filter.

We considered two different resource constraint problems in the passive portion of the iSink and evaluated the use of sampling as a means for addressing these constraints. We first considered the problem of a fixed resource in the iSink itself. Estan and Varghese in [6] describe sampling methods aimed at monitoring “heavy hitters” in IP flows through routers with a limited amount of memory. We adapted one of these methods for use in iSink. Second, we considered the problem of bandwidth as the limited resource.

⁷ We compare Active Sink with LaBrea because unlike LaBrea, Honeyd is stateful (forks a process per connection), and hence is much less scalable. Since Honeyd also relies on a packet filter LaBrea’s scalability bounds affect Honeyd as well.

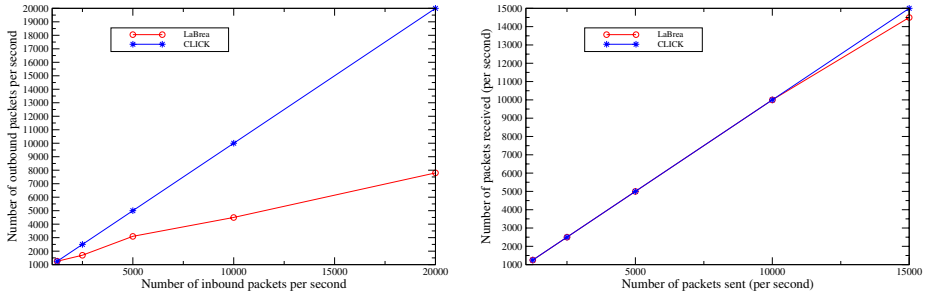


Fig. 10. Scalability of Click-based Internet Sink and LaBrea for TCP (left) and UDP (right) flows

In this case, the idea is to reduce the total amount of traffic routed to an iSink by selecting subnets within the total address space available for monitoring. These methods would be used in combination with the filtering methods described in Section 3.3.

Memory Constrained iSink Sampling. The method that forms the basis of our sampling approach with a memory constrained iSink is called *Sample and Hold* [6]. This method accurately identifies flows larger than a specified threshold (*i.e.*, heavy hitters). Sample and hold is based on simple random sampling in conjunction with a hash table that is used to maintain flow ID’s and byte counts. Specifically, incoming packets are randomly sampled and entries in the hash table are created for each new flow. After an entry has been created, *all* subsequent packets belonging to that flow are counted. While this approach can result in both false positives and false negatives, its accuracy is shown to be high in workloads with varied characteristics. We apply sample and hold in iSink to the problem of identifying “heavy hitters”, which are the worst offending source addresses based on the observed number of scans.

Adapting the sample and hold method to the iSink required us to define the size of the hash table that maintains the data, and the sampling rate based on empirical observation of traffic at the iSink. In [6], the objective is identifying accurately the flows that take over $T\%$ of a link’s capacity. An oversampling factor O is then selected to reduce the possibility of false negatives in the results. These parameters result in allocating $HT_{len} = 1/T * O$ locations in each hash table. The packet sampling rate is then set to HT_{len}/C where C is the maximum packet transmission capacity of the incoming link over a specified measurement period t . At the end of each t , the hash table is sorted and results are produced.

Bandwidth Constrained iSink Sampling. In the bandwidth constrained scenario, the sampling design problem is to select a set of subnets from the total address space that is available for monitoring on the iSink. The selection of the number of subnets to monitor is based on the bandwidth constraints. In this case we assume that we know the mean and variance for traffic volume on a “typical” class B or class C address space. We then divide the available bandwidth by this value to get the number of these subnets that can be monitored. The next step is to select the specific subnets within the entire space that will minimize the error introduced in estimates of probe populations.

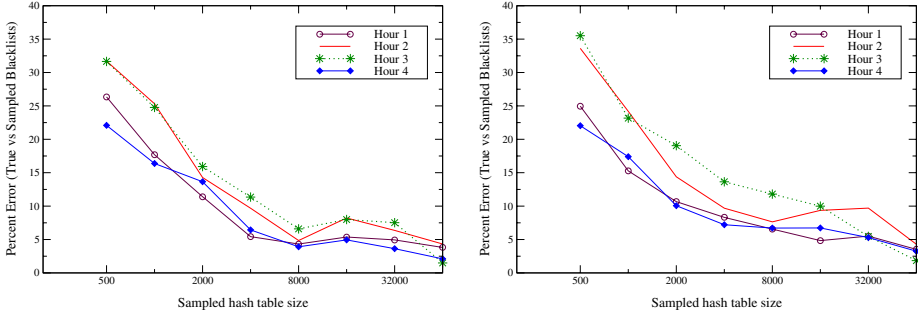


Fig. 11. Error rates for different hash table sizes (x-axis is log scale) using Sample and Hold method with rates of 1/100 (left) and 1/300 (right)

Our analysis in this paper is based on the use of random sampling as a means for subnet selection. Our rationale for this approach is based on the observation that overall traffic volumes across the service-provider class A address space that we monitor is quite uniform. The strengths of this approach are that it provides a simple method for subnet selection, it provides unbiased estimates and it lends itself directly to analysis. The drawback is that sampling designs that take advantage of additional information such as clustered or adaptive sampling could provide more accurate population estimates. We leave exploration of these and other sampling methods to future work.

After selecting the sampling design, our analysis focused on the problem of *detectability*. Specifically, we were interested in understanding the accuracy of estimates of total probe populations from randomly selected subsets. If we consider $\hat{\tau}$ is an unbiased estimator of a population total τ then the estimated variance of $\hat{\tau}$ is given by:

$$var(\hat{\tau}) = N^2 \left[\left(\frac{N - n}{N} \right) \frac{\sigma^2}{n} + \left(\frac{1 - p}{p} \right) \frac{\mu}{n} \right]$$

where N is the total number of units (in our case, subnets), n is the sampled number of units, μ is the population mean (in our case, the mean number of occurrences of a specific type of probe), σ^2 is the population variance and p is the probability of detection for a particular type of probe. In the analysis presented in Section 6.1, we evaluate the error in population estimates over a range of detection probabilities for different size samples. The samples consider dividing the class A address space into its component class B's. The probabilities relate directly to detection of worst offenders (top sources of unsolicited traffic) as in the prior sampling analysis. The results provide a means for judging population estimation error rates as a function of network bandwidth consumption.

6.1 Sampling Evaluation

Our evaluation of the impact of sampling in an iSink was an *offline* analysis using traces gathered during one day selected at random from the service-provider iSink. Our objective was to empirically assess the accuracy of sampling under both memory constrained and bandwidth constrained conditions. In the memory constrained evaluation, we compare the ability to accurately generate the top 100 heavy hitter source list over four

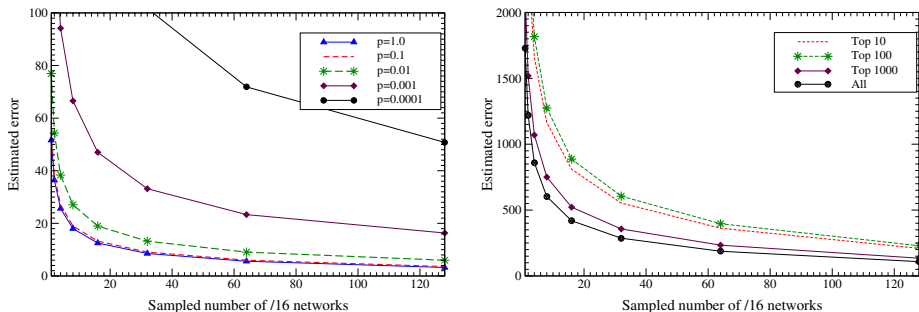


Fig. 12. Estimated error (σ/μ) for a given number of randomly selected /16’s. Included are error estimates for total probes from worst offending source IP over a range of detection probabilities (left), estimates for total probes for worst offender lists of several sizes (right)

consecutive 1 hour periods using different hash table sizes and different sampling rates. For each hour in the data set, we compare the percentage difference in the number of scans generated by the “true” top 100 blacklist and sampled top 100 blacklist sources. In the bandwidth constrained evaluation, we consider accuracy along three dimensions: 1) estimating the worst offender population with partial visibility, 2) estimating black lists of different lengths, 3) estimating backscatter population.

Our memory constrained evaluation considers hash table sizes varying from 500 to 64K entries where each entry consists of a source IP and a access attempt count. Note that the hash table required to maintain the complete list from this data was on the order of 350K entries We consider two different arbitrarily chosen sampling rates – 1 in 100 and 1 in 300 with uniform probability. In each case, once a source IP address has been entered into the table, all subsequent packets from that IP are counted. If tables become full during a given hour then entries with the lowest counts are evicted to make room for new entries. At the end of each hour, the top 100 from the true and sampled lists are compared. New lists are started for each hour. The results are shown in Figure 11. These results indicate that even coarse sampling rates (1/300) and relatively small hash tables enable fairly accurate black lists (between 5%–10% error). The factor of improvement between sampling at 1/100 and 1/300 is about 1.5, and there is little benefit to increasing the hash table size from 5,000 to 20,000. Thus, from the perspective of heavy hitter analysis in a memory constrained system, sampling can be effectively employed in iSinks.

As discussed in the prior section in our bandwidth constrained evaluation we consider error introduced in population estimates when using simple random sampling over a portion of the available IP address space. We argue that simple random sampling is appropriate for some analysis given the uniform distribution of traffic over our class A monitor. The cumulative distribution of traffic over a one hour period for half of the /16 subnets in our class A monitor is shown in Figure 13(right). This figure shows that while traffic across all subnets is relatively uniform (at a rate of about 320 packets per minute per /16), specific traffic subpopulations – TCP backscatter as an example – can show significant non-uniformity which can have a significant impact on sampling.

We use the mean normalized standard deviation (σ/μ) as an estimate of error in our analysis. In each case, using the data collected in a typical hour on the /8, we empirically

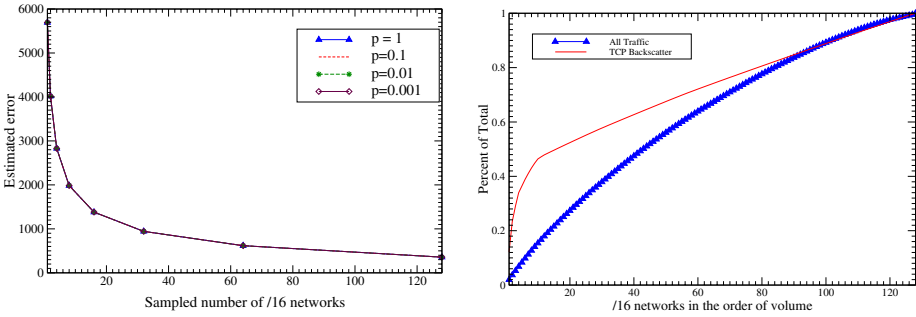


Fig. 13. Estimated error for TCP backscatter traffic (left). Cumulative distribution of all traffic and TCP backscatter traffic across half of the class A address space monitor over a one hour period. The average number of probes per /16 is 320 packets per minute (right)

assess the estimated error as a function of a randomly selected sample of /16 subnets. The results of this approach are shown in Figure 12. The graph on the left shows the ability to accurately estimate the number of probes from the single worst offending IP source over a range of detection probabilities (*i.e.*, the probability of detecting a source in a selected /16). This graph indicates that worst offenders are detectable even with a small sample size and error-prone or incomplete measurements. The graph on the right shows the ability to accurately estimate black lists from a selected sample of /16's. This graph indicates that it is easier to estimate larger rather than smaller black lists when sampling. We attribute this to the variability in black list ordering across the /16's. Finally, Figure 13(left) shows the ability to accurately estimate TCP backscatter traffic over a range of detection probabilities. The graph suggests that while backscatter estimates are robust in the face of error-prone or incomplete measurements, estimated error of total backscatter is quite high even with a reasonably large number of /16's. This can be attributed to the non-uniformity of backscatter traffic across the class A monitor shown in Figure 13(right) and suggests that alternative sampling methods for backscatter traffic should be explored. On a broader scale, this indicates that traditional backscatter methodologies that assumes uniformity could be error prone.

7 Summary and Future Work

In this paper we describe the architecture and implementation of an Internet Sink: a useful tool in a general network security architecture. iSinks have several general design objectives including scalability, the ability to passively monitor network traffic on unused IP addresses, and to actively respond to incoming connection requests. These features enable large scale monitoring of scanning activity as well as attack payload monitoring. The implementation of our iSink is based on a novel application of the Click modular router, NAT Filter and the Argus flow monitor. This platform provides an extensible, scalable foundation for our system and enables its deployment on commodity hardware. Our initial implementation includes basic monitoring and active response capability which we test in both laboratory and live environments.

We report results from our iSink's deployment in a live environment comprising four class B networks and one entire class A network. The objectives of these case studies were to evaluate iSink's design choices, to demonstrate the breadth of information available from an iSink, and to assess the differences of perspective based on iSink location in IP address space. We show that the amount of traffic delivered to these iSinks can be large and quite variable. We see clear evidence of the well documented worm traffic as well as other easily explained traffic, the aggregate of which can be considered Internet background noise. While we expected overall volumes of traffic in the class B monitors and class A monitor to differ, we also found that the overall characteristics of scans in these networks were quite different. We also demonstrate the capability of iSinks to provide insights on interesting network phenomenon like periodic probing and SMTP hot-spots, and their ability gather information on sources of abuse through sampling techniques. A discussion of operational issues, security, and passive fingerprinting techniques is provided in [32].

The evaluation of our iSink implementation demonstrates both its performance capabilities and expectations for live deployment. From laboratory tests, we show that iSinks based on commodity PC hardware have the ability to monitor and respond to over 20,000 connection requests per second, which is approximately the peak traffic volume we observed on our class A monitor. This also exceeds the current version of LaBrea's performance by over 100%. Furthermore, we show that sampling techniques can be used effectively in an iSink to reduce system overhead while still providing accurate data on scanning activity.

We intend to pursue future work in a number of directions. First, we plan to expand the amount of IP address space we monitor by deploying iSinks in other networks. Next, we intend to supplement iSink by developing tools for datamining and automatic signature generation.

Acknowledgements

The authors would like to thank Jeff Bartig, Geoff Horne, Bill Jensen and Jim Martin for all of their help. Exploration of the filtering techniques grew out of fruitful discussions during Vinod's internship with Vern Paxson. We would also like to acknowledge contributions of Ruoming Pang in the development of the DCERPC responder. Finally, we would like to thank the anonymous reviewers for their insightful comments and Diego Zamboni for his excellent shepherding.

References

1. R. Anderson and A. Khattak. The Use of Information Retrieval Techniques for Intrusion Detection. In *Proceedings of RAID*, September 1998.
2. Network Associates. LovGate Virus Summary. <http://vil.nai.com/vil/content/Print100183.htm>, 2002.
3. C. Bullard. Argus Open Project. <http://www.qosient.com/argus/>.
4. C. Cranor, Y. Gao, T. Johnson, V. Shkapenyuk, and O. Spatscheck. Gigascope: High Performance Network Monitoring with an SQL Interface.
5. E-eye. Analysis: Sasser Worm. <http://www.eeye.com/html/Research/Advisories/AD20040501.html>.

6. C. Estan and G. Varghese. New Directions in Traffic Measurement and Accounting. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.
7. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. NetScope: Traffic Engineering for IP Networks. *IEEE Network Magazine*, Special Issue on Internet Traffic Engineering, 2000.
8. B. Greene. BGPv4 Security Risk Assessment, June 2002.
9. B. Greene. Remote Triggering Black Hole Filtering, August 2002.
10. Honeyd: Network Rhapsody for You. <http://www.citi.umich.edu/u/provos/honeyd>.
11. G. Iannaccone, C. Diot, I. Graham, and N. McKeown. Monitoring very high speed links. In *SIGCOMM Internet Measurement Workshop*, November 2001.
12. E. Kohler, R. Morris, B. Chen, J. Jannotti, and F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, August 2000.
13. W. Lee, S.J. Stolfo, and K.W. Mok. A Data Mining Framework for Building Intrusion Detection Models. In *IEEE Symposium on Security and Privacy*, 1999.
14. T. Liston. The Labrea Tarpit Homepage. <http://www.hackbusters.net/LaBrea/>.
15. D. Moore. Network Telescopes. <http://www.caida.org/outreach/presentations/2003/dimacs0309/>.
16. D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The Spread of the Sapphire/Slammer Worm. Technical report, CAIDA, 2003.
17. D. Moore, C. Shannon, and K. Claffy. Code Red: A Case Study on the Spread and Victims of an Internet Worm. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Marseilles, France, November 2002.
18. D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *Proceedings of IEEE INFOCOM*, April 2003.
19. 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.
20. T. Oetiker. The multi router traffic grapher. In *Proceedings of the USENIX Twelfth System Administration Conference LISA XII*, December 1998.
21. V. Paxson. BRO: A System for Detecting Network Intruders in Real Time. In *Proceedings of the 7th USENIX Security Symposium*, 1998.
22. D. Plonka. Flawed Routers Flood University of Wisconsin Internet Time Server. <http://www.cs.wisc.edu/plonka/netgear-sntp>.
23. D. Plonka. Flowscan: A network traffic flow reporting and visualization tool. In *Proceedings of the USENIX Fourteenth System Administration Conference LISA XIV*, December 2000.
24. Y. Rekhter. RFC 1817: CIDR and Classful Routing, August 1995.
25. M. Roesch. The SNORT Network Intrusion Detection System. <http://www.snort.org>.
26. S. Staniford, J. Hoagland, and J. McAlerney. Practical Automated Detection of Stealthy Portscans. In *Proceedings of the ACM CCS IDS Workshop*, November 2000.
27. S. Staniford, V. Paxson, and N. Weaver. How to Own the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, August 2002.
28. H.S. Teng, K. Chen, and S. C-Y Lu. Adaptive Real-Time Anomaly Detection Using Inductively Generated Sequential Patterns. In *IEEE Symposium on Security and Privacy*, 1999.
29. The HoneyNet Project. <http://project.honeynet.org>.
30. Trend Micro. WORM_RBOT.CC. http://uk.trendmicro-europe.com/enterprise/security_info/ve_detail.php?vname=WORM_RBOT.CC.
31. V. Yegneswaran, P. Barford, and S. Jha. Global Intrusion Detection in the DOMINO Overlay System. In *Proceedings of NDSS*, San Diego, CA, 2004.
32. V. Yegneswaran, P. Barford, and D. Plonka. On the Design and Use of Internet Sinks for Network Abuse Monitoring. University of Wisconsin Technical Report #1497, 2004.
33. V. Yegneswaran, P. Barford, and J. Ullrich. Internet Intrusions: Global Characteristics and Prevalence. In *Proceedings of ACM SIGMETRICS*, San Diego, CA, June 2003.