Demystifying Service Discovery: Implementing an Internet-Wide Scanner

Derek Leonard and Dmitri Loguinov
Department of Computer Science and Engineering
Texas A&M University, College Station, TX 77843 USA
{dleonard,dmitri}@cse.tamu.edu

ABSTRACT

This paper develops a high-performance, Internet-wide service discovery tool, which we call IRLscanner, whose main design objectives have been to maximize politeness at remote networks, allow scanning rates that achieve coverage of the Internet in minutes/hours (rather than weeks/months), and significantly reduce administrator complaints. Using IRLscanner and 24-hour scans, we perform 21 Internet-wide experiments using 6 different protocols (i.e., DNS, HTTP, SMTP, EPMAP, ICMP and UDP ECHO), demonstrate the usefulness of ACK scans in detecting live hosts behind stateless firewalls, and undertake the first Internet-wide OS fingerprinting. In addition, we analyze the feedback generated (e.g., complaints, IDS alarms) and suggest novel approaches for reducing the amount of blowback during similar studies, which should enable researchers to collect valuable experimental data in the future with significantly fewer hurdles.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement Techniques

General Terms

Design, Measurement, Performance

Keywords

Horizontal Scanning, Service Discovery

1. INTRODUCTION

Characterizing visible services in the Internet (e.g., web sites [5], [16], [28], end-hosts [17], [18],[23]), discovering and patching servers with critical security vulnerabilities (e.g., SSH [42], DNS [13]), and understanding how Internet worms create massive botnets [10], [25], [31], [53] are important research topics that directly benefit from efficient and scalable scanning techniques that can quickly discover available services in the Internet.

IMC'10, November 1–3, 2010, Melbourne, Australia.

Our focus in this paper is on *horizontal scanning* [52], which is a method for enumerating (in some set $\mathcal{S}$) all remote hosts that support a given protocol/service p. This is accomplished by sending packets to destinations in $\mathcal{S}$ and counting positive responses within some time interval. We call a scan *complete* if each address in $\mathcal{S}$ is probed and *partial* otherwise. The latter type of scan significantly reduces the burden on remote networks and is useful when an estimate of the number of responsive hosts (rather than their IP addresses) is sufficient.

While several large-scale measurements have been conducted in the past [5], [13], [17], [43], researchers initially considering a similar project are often faced with delays on the order of months for individual tests to run [5], [43]. During this time, computational resources of potentially dozens [5] of local machines that could be put to other uses are tied up. Further complicating the issue is the possibility of facing a significant number of complaints from hostile network administrators [5], [8], [11], [13], [15], [17], [42], [43], even for partial measurements. Given these issues, questions arise about the feasibility of service discovery [9], especially on sensitive TCP ports. Evidence suggests that sometimes [17] researchers are even forced to abort planned activities due to the negative publicity generated by scan traffic.

Since previous work has not explicitly aimed to design a high-performance scanner and/or maximize its politeness, there is no standard by which to judge the quality or intrusiveness of a scanner. Our first step then is to propose three main objectives that a good Internet-wide scanning solution must satisfy: 1) efficient usage of resources (i.e., bandwidth, CPU, memory) in complete scans; 2) accuracy of extrapolation in partial scans; and 3) maximum politeness at remote networks. The first objective ensures that the implementation scales well when scan duration T is reduced to hours or even minutes. The second objective delivers a platform for extremely fast partial scans with accurate extrapolation of metrics of interest. The last objective maximally reduces the instantaneous load (i.e., burstiness) applied to target subnets and controls the rate of IDS activity (i.e., false-positive alarms, wasted investigation effort, and dynamic firewall blocks against the scanner network) in response to scan traffic.

We next build a scanner that satisfies these goals and evaluate its performance in real scans.

1.1 Our Contributions

The first part of the paper analyzes the approaches exposed in the literature to understand whether they can be used to optimize performance, politeness, and extrapolation

ability of an Internet-wide scanner. In addition to realizing that prior work was not driven by any particular objectives in designing their scanners (besides obtaining the data of interest in some finite amount of time), we also reach the conclusion that there is no consensus on such important parameters as scan scope, permutation, split among source IPs, implementation, timeouts, handling of complaints, and monitoring of the scan's intrusiveness at remote networks.

To overcome this problem, the second part of the paper presents our design of IRLscanner, which is a high-performance and source-IP scalable framework for service discovery in the Internet. The central element of the scanner is a novel permutation/split algorithm, which we show is optimally polite at each CIDR subnet s as it spaces probes arriving to s equally throughout scan duration $[0, T]$, even with multiple source IPs. Extrapolation results with IRLscanner running at its default rate r (at which it covers the Internet in $T = 24$ hours) demonstrate that partial scans of our approach are unbiased, leading to 1% estimation error in the number of live hosts in just 10 seconds.

Due to the goal of allowing faster scan durations (i.e., minutes/hours) and politeness concerns, IRLscanner incorporates additional features that help it achieve our objectives. These include a significantly reduced scope of measurements (i.e., one billion packets fewer) compared to previous scanners, absence of largely ineffective retransmissions, ability to run with any number of IPs aliased to the same server, capture of all back-scan and bogus traffic (e.g., from hackers and buggy implementations), and significantly higher timeouts for unresponsive targets, which allows it to capture a wider variety of busy/slow hosts in the Internet than was possible before.

Armed with IRLscanner, the third part of the paper highlights 21 Internet-wide scans across a wide range of protocols and ports, including DNS (port 53), HTTP (port 80), SMTP (port 25), EPMAP (port 135), and UDP ECHO (port 7). In addition to running over 20 times faster than any prior scanner and probing ports that have never been scanned in the literature (i.e., SMTP, ECHO, EPMAP), we experiment with several techniques never achieved on an Internet-wide scale in previous work (e.g., ACK scanning, testing ECN and TCP option support) and perform the first large-scale OS fingerprinting study of 44M hosts responding to port 80.

We finish the paper by analyzing the feedback generated from our experiments. This includes a detailed complaint analysis, techniques for monitoring the impact of scan traffic on IDS activity in the Internet, approaches for a-priori predicting the amount of blowback in response to scanning a particular port, and various ways for reducing the perceived maliciousness of the scan.

1.2 Ethical Implications

Over the last two years, we have closely worked with university officials and taken numerous steps (see below) in an effort to reduce investigation effort and aggravation for remote administrators. While our interest in this paper is purely to expose the underlying issues of service discovery and make it more accessible to researchers without damaging remote networks, one concern might be that our scan techniques are not only maximally polite, but also optimally stealthy against popular IDS packages. As a result, one could argue that attackers could benefit from our work and thus inflict certain damage that would not otherwise be possible.

However, we do not believe this to be the case. First, stealthier scanning by itself does not compromise hosts; in contrast, intrusion using malicious payload (e.g., delivered through unsolicited packets or email) does. As a result, many networks with patched hosts and up-to-date IDS signatures of various malware should remain well protected despite the findings of this paper. Second, botnets afford hackers such a diverse pool of IPs that they often do not care to remain stealthy and rely on the most basic sequential scanning [2], which apparently is sufficient for their purposes. Instead, we believe that the discussion and techniques exposed in this paper might be useful in building future defenses against scanning worms, should they choose to deploy similar techniques.

Another concern might be that we are collecting information about remote networks that their administrators do not wish to make public. We contend that such information is well-protected by firewalls, and this paper makes no attempt to trick or confuse network monitoring devices to gather sensitive data. Further, given the constant background scanning performed by attackers [41], any data (and likely much more) we collect is already available to them. However, to ensure privacy, we do not publicize any information about individual networks collected during our scans and instead rely only on summary statistics.

2. SCANNER DESIGN

Beyond the experiment-specific choice of the protocol/port pair that uniquely characterize a service, every researcher considering a horizontal scan must answer a common set of questions before proceeding. In this section, we turn to several recent studies [5], [13], [17], [43] that have performed large-scale service discovery to determine whether these design questions have been definitively answered and our objectives met.

2.1 Scan Scope

We start with the issue of which IP addresses to target when scanning. Define $\mathcal{F}$ to be the Internet IPv4 address space, which consists of $n = 2^{32}$ addresses available for scanning. While intuition may suggest to probe the entire space to ensure completeness, certain IPs may not be suitable for scanning. Before delving into details, define set $\mathcal{NR} \subseteq \mathcal{F}$ to be all non-reserved destinations [20], $\mathcal{I} \subseteq \mathcal{NR}$ to be all IANA-allocated blocks [19], and $\mathcal{B} \subseteq \mathcal{I}$ to be the set of IPs advertised in BGP prefixes [59] at the border router of the scanner network. Note that $\mathcal{B}$ must be verified against $\mathcal{I}$ to ensure that invalid advertisements are not included.

To capture the choice of which destinations to target, we define scan *scope* $\mathcal{S}$ to be the subset of $\mathcal{F}$ probed during measurement. As shown in the second column of Table 1, all previous Internet-wide service discovery projects scanned at least the IANA allocated space $\mathcal{I}$, which is justified [17], [34] by churn in BGP routing tables and desire to avoid losing responses during the period of the experiment (i.e., 30+ days in Table 1). In [5], however, no reason is given for scanning unallocated space, although there is a slight possibility of new blocks being allocated by IANA during the measurement.

As we discuss later, sets $\mathcal{I}$ and $\mathcal{NR}$ may be appropriate for slow scans; however, faster scanners have little incentive

Scanner	Scope	Permutation	Servers	Protocol	Port	Timeout	Duration	Blacklist	.0/.255	Exclude
Pryadkin [43]	$\mathcal{I}$	uniform	3	ICMP/TCP	–	10s	123d	yes	no	no
Benoit [5]	$\mathcal{NR}$	uniform	25	TCP	80	30s	92d	no	yes	no
Dagon [13]	$\mathcal{I}$	uniform	–	UDP	53	–	30d	–	yes	US Gov
Heidemann [17]	$\mathcal{I}$	RIS	8	ICMP	echo	5s	52d	yes	no	no

Table 1: Large-scale service discovery in the literature (dashes represent unreported values).

to utilize sets larger than $\mathcal{B}$ in the current Internet since performance concerns (i.e., volume of sent traffic) usually outweigh completeness of scan results.

2.2 Scan Order

The next factor we consider is the order in which IP addresses are scanned. This is determined by the *permutation* [53] of space $\mathcal{S}$, which is simply a reordering of target addresses to achieve some desired result. The chosen permutation controls the burstiness of traffic seen by remote networks and is a significant factor in both the perceived politeness of scan traffic and estimation accuracy of Internet-wide metrics from partial scans. For all discussion below, we assume that target subnets s are full CIDR blocks (i.e., given in the $/b$ notation).

The most basic approach, which we call *IP-sequential*, does not shuffle the address space and probes it in numerical order (e.g., 10.0.0.1, 10.0.0.2, 10.0.0.3, etc.). It is not only simple to implement, but also routinely used in the Internet [2], [22], [30] and measurement studies [3], [16]. IP-sequential targets individual subnets s with a burst of $|s|$ consecutive packets at the rate of n/T before moving on to another network. Besides extremely high sending rates to s regardless of its size (e.g., 37 Kpps for $\mathcal{S} = \mathcal{I}$ and $T = 24$ hours), IP-sequential also suffers from poor extrapolation ability.

The main alternative to IP-sequential is the *uniform* permutation [53] also extensively used in the literature [5], [13], [42], [43]. This approach draws targets uniformly randomly from the full address space $\mathcal{S}$ and intermingles probes to many subnets, which reduces instantaneous load on individual networks and produces unbiased random sampling during partial scans. In the literature, the uniform permutation is usually accomplished by either an LCG (linear congruential generator) [29] or some encryption algorithm applied sequentially to each element of $\mathcal{S}$ (e.g., TEA [42]), both of which ensure that no IP is probed twice.

The final approach proposed in [17] we call *Reverse IP-sequential* (RIS) due to its reversal of the bits in the IP-sequential permutation and targeting the same address in each subnet (e.g., *.*.127.10, *.*.8.10, *.*.248.10, etc.) before moving on to another address. Intuition suggests that RIS is poorly suited for extrapolation (which we confirm below), while the uniform permutation fails to deliver packets with maximum spacing to each subnet. Since no analysis exists to further evaluate the differences between these three permutation algorithms, making an informed choice remains an open problem.

2.3 Scan Origin

In a bid to obtain multiple vantage points [17] and decrease the time required to complete the scan [5], past measurement studies have often distributed the burden of scanning amongst several hosts. We call this process a *split*, which in the literature parcels blocks of either contiguous [3], [5] or permuted [17], [43] IP addresses to m scanning nodes. Column four of Table 1 contains values for m used previously, with [17] being the only study that used multiple hosts residing on two different networks (four at each location).

The current consensus in the literature is that multiple scanning hosts on a single network are necessary only if the full assigned scanning bandwidth cannot be utilized by one host, a condition that is implied in [43] and mentioned explicitly in [3], [5], [17]. However, it is not clear from these studies how many hosts are needed to efficiently utilize a link or provide reasonably short scan durations. Further, the literature does not consider the split's impact on the perceived politeness of the scan, which we tackle later in the paper.

2.4 Extrapolation

In many research applications, especially those that monitor growth of the Internet [17], [23], it is sufficient to obtain the number of live hosts or estimate their characteristics (e.g., mean uptime) rather than a list of their exact IPs. The best approach in such cases is a partial scan, which produces a tiny footprint at remote networks and in many cases allows accurate extrapolation of metrics of interest. This requires that targets within each subnet be randomly selected, without any bias being given to certain parts of $\mathcal{S}$ or particular patterns within probed IP addresses (because the density of live hosts varies both across the Internet and the last $1-2$ bytes of the IP). In addition, non-random probing is often seen by administrators as purposefully malicious, which in turn leads to unnecessary investigation overhead, firewall blocks, and complaints.

2.5 Implementation

The next pressing issue of service discovery is the method used to send/receive packets, which significantly impacts the efficiency of the scanner. The easiest implementation method uses scripts that execute pre-written utilities [16], [43] or existing scanners [3]. An alternative is to write a custom scanner for a particular measurement, which opens the possibility of using connectionless sockets [13], [17] for ICMP or UDP-based scans, connection-oriented TCP sockets [5], and finally raw IP sockets for TCP SYN scans [42], [43]. While there is no consensus in the scanning literature on what method to use, [14] suggests that software limitations on packet sending rates can be overcome using a network subsystem that bypasses the default network stack.

2.6 Timeouts and Duration

The next two issues are when to mark a host as unresponsive and what aspects should determine scan duration T. The former issue comes down to two choices: 1) waiting a "safe" amount of time before retransmitting [5], [42], [43]; and 2) when to finally time out and declare targets dead [17]. Note that both incur substantial overhead due to the

need to remember all covered destinations and to maintain numerous timers. Furthermore, it is unclear what benefit retransmission carries given the low packet loss on the backbone and whether the increased overhead (i.e., doubling or tripling the number of sent packets) justifies the potentially minuscule accuracy gains.

Table 1 lists timeout values that range from 5 to 30 seconds for previous measurements studies. Given the limited number of outstanding sockets, finite bandwidth and CPU power, and a wide range of possible choices, it remains unclear how to choose timeouts to simultaneously allow for efficiency and accuracy (e.g., certain busy servers respond with a 60-second delay, but should they be captured by the scanner?).

After settling the above problems, it is important to ensure that the scan will complete in such amount of time T that produces the most relevant data without overburdening local/remote network resources. Of the measurement studies listed in Table 1, only [13] was unencumbered by software/hardware restrictions, while for others these issues dominated the choice of T. As such, previous measurement studies have generally been limited to a tradeoff between small T, few servers m, large timeouts, and large scan scope. Instead, our goal is to develop a scanner in which the researcher can control T independently of all other listed parameters.

2.7 Negative Feedback

Due to the unsolicited nature of the packets sent by service discovery measurement studies and the diversity of networks in the Internet, it is inevitable that some targeted hosts will take offense. This often manifests itself in the form of email/phone complaints from network administrators [5], [11], [13], [17], [42], [43], though the literature is lacking in details on the exact nature of complaints (e.g., frequency of legal threats) and specific techniques for dealing with them.

In Table 1, we list three of the methods used by previous studies to mitigate complaints. The first is the use of a blacklist by [17], [42], [43] to exclude the networks of sensitive/suspicious administrators, which generally avoids repeated complaints from the same network. The other two approaches boil down to a further reduction in scope by the omission of network/broadcast IP addresses (i.e., *.*.*.0 and *.*.*.255) [17], [43] and preemptively blacklisting networks before they complain (e.g., U.S. government) [13]. While blacklisting complaining parties is undoubtedly a sound approach, no reasoning or motivating factors have been provided for the other two methods and it is unclear whether they are indeed necessary.

3. IRLscanner

Based on our analysis of scanning literature in the last section, it appears that researchers interested in service discovery projects are faced with scan durations on the order of months, tying up several machines that could be dedicated to other projects, and the likelihood of significant negative feedback that could easily lead to the measurement being terminated. For example, [17] reports that TCP scans produce 30 times more complaints than ICMP scans, which has precluded the authors from conducting them after the first attempt.

Assuming a fixed amount of bandwidth available for scanning, in this section we seek to alleviate these concerns by designing a service discovery tool we call IRLscanner that allows for very small scan durations T, originates from a single inexpensive host, and minimizes aggravation of network administrators (both remote and local) by scanning as politely as possible for a given T.

Set $\mathcal{S}$	Size	Reduction	Rate (Mbps)	Rate (Kpps)
$\mathcal{F}$	4.29B	–	33.4	49.7
$\mathcal{NR}$	3.7B	14%	28.8	42.8
$\mathcal{I}$	3.27B	24%	25.4	37.8
$\mathcal{B}$	2.11B	51%	16.4	24.4

Table 2: Scan set size, Ethernet bandwidth requirement, and pps rate in a 24 hour TCP SYN scan.

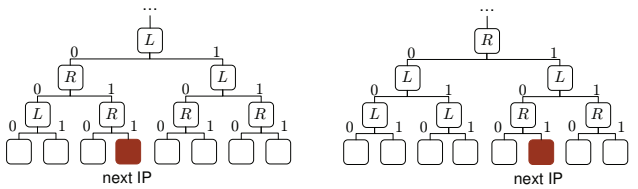
3.1 Scan Scope

We start by determining the scope of IRLscanner. While firewalls and routers routinely use the Bogons list [12] to filter nonsensical traffic (i.e., reserved and unallocated blocks), packets destined to unadvertised BGP networks are also dropped by the scanner's border router, but only after unnecessarily increasing router load and wasting scanner resources. Therefore, one expects that only set $\mathcal{B}$ should normally produce valid results or be used for discovering hosts responsive to unicast traffic. However, given that BGP tables change dramatically in the long-term [34], restricting the scope to only routable addresses either requires a live BGP feed or potentially allows for inaccurate representation of the Internet in the resulting measurement.

While this is definitely a concern for slow scanners (i.e., T is weeks or months), our goal is to complete measurements in much shorter periods (i.e., hours) during which BGP changes can often be neglected. For fast scans, updates pulled from RouteViews [46] at start time sufficiently approximate the routable space during the entire experiment. Our analysis of BGP tables during August 2009 discovered less than 0.1% difference in advertised prefixes over a 10-day period, with proportionally fewer changes during $T = 24$ hours used in our experiments. While for IRLscanner it makes sense to only probe $\mathcal{B}$, the tradeoff between scope, duration T, and BGP table accuracy must be determined on a case-by-case basis.

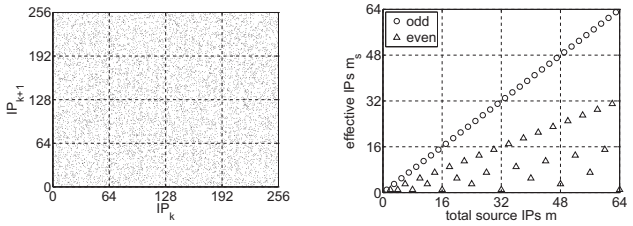
To gauge the potential gains from restricting the scope to routable destinations, we determine [19], [46] the current state of sets $\mathcal{B}$, $\mathcal{NR}$, and $\mathcal{I}$ in late August 2009 and list them in Table 2. While previous scanners achieve a significant reduction (i.e., by 24%) in the number of sent packets by omitting the reserved/unallocated space, probing only set $\mathcal{B}$ removes almost one billion *additional* targets and doubles the performance gains of previous work to 51%. The table also shows the bandwidth necessary to complete the scan in 24 hours, where all 40-byte SYN packets are padded to 84-byte minimum-size Ethernet frames, and the corresponding pps (packets per second) rate.

To implement a scanner with scope $\mathcal{B}$, it is necessary to obtain a timely BGP dump from either the RouteViews project [46] or the local border router. Given the desire for small scan durations on inexpensive hardware, checking individual IP addresses against a list of roughly $300,000$ prefixes must be very efficient. While IP checking can be accomplished with a balanced binary tree [13] with logarithmic lookup complexity, IRLscanner uses a 512 MB hash table, where each bit indicates whether the corresponding IP is allowed



(a) IP ends with 011 (b) IP ends with 101

Figure 1: Illustration of AGT.



(a) last-octet correlation (b) m_s in (1)

Figure 2: Randomness of RLCG and effectiveness of the GIW/RR split.

or not. This ensures that checks are accomplished in $O(1)$ time and improves lookup speed from 923 Kpps (balanced tree) to 11 Mpps (using a single core of a 2.2 GHz Opteron). Given that most commodity machines have at least 1 GB of RAM and the rest of our scanner requires only 2 MB of main memory, this tradeoff allows us to dedicate more computational power to sending packets and performing other processing as needed.

3.2 Scan Order

Despite the constant volume of scanning traffic in the Internet [41], network administrators generally view this activity as malicious and periodically complain to networks that originate such traffic [13], [17] [42]. Furthermore, many IDS tools [7], [51], [58] automatically generate firewall rules against scanning hosts, whether detected locally or through distributed collaborative systems [36], [47]. With this perception in mind, researchers must first weigh the benefit gained from performing a service discovery measurement with the possibility of negative publicity for their institution and/or its address space being blacklisted at remote networks.

Upon determining to proceed, these negative effects can be reduced for all involved parties by using an address permutation that avoids targeting individual subnets with large bursts of traffic, which often triggers Intrusion Detection Systems (IDS) and raises concerns of malicious/illegal activity. Since IDS predominantly operates on a per-IP basis, additional reduction in false-alarms is possible by using multiple source IPs at the scanner host, which we discuss later in this section. While the uniform permutation [53] is routinely used in scanning applications, no previous paper has examined the issue of achieving maximal politeness and whether such methods could be implemented in practice. We address this open problem next.

For a given CIDR subnet s in the Internet, our goal is to maximally reduce the burstiness of scan traffic seen by s, which is equivalent to maximizing inter-packet delays for *all* probes arriving to s. Recalling that $n = 2^{32}$, we define permutations that return to s with a period $n/|s|$ to be *IP-wide* at s and those that achieve this simultaneously for all possible CIDR subnets to be *globally IP-wide* (GIW). Note that GIW permutations spread probes to each s evenly throughout $[0, T]$, which ensures that all networks are probed at a constant rate $|s|/T$ proportional to their size and that no s can be scanned slower for a given value of T. This makes GIW optimally polite[1] across the entire Internet.

The simplest GIW technique, which we call an *alternating gateway tree* (AGT), is a binary tree of depth 32 where target IPs reside in leaves and all edges are labeled with 0/1 bits. Traversing the tree, the scanner accumulates individual bits along the edges into the next IP. Decisions to move left or right at internal nodes (gateways) v depend on their states θ_v, which are flipped during each visit to ensure that no IP is probed twice and that packets alternate between left/right children of each gateway. Figure 1 shows the bottom four levels of some random AGT, where the tree in part (a) generates an IP address ending with bits 011. Part (b) of the figure illustrates the next IP generated when the scanner returns to this portion of the AGT, which results in the address ending with bits 101.

Since balanced binary trees have well-defined rules for calculating the offset of each internal node, AGTs do not require storing child pointers. Thus, their RAM overhead is $(n-1)/8 = 512$ MB needed to store tuple $(\theta_1, \ldots, \theta_{n-1})$ and their computational complexity is 26 memory reads/writes (i.e., 52 total) per generated IP (assuming depth-31 traversal and 64-bit lookups that yield the first 5 levels of the tree in one RAM access).

Note that AGT provides the scanner with 2^{n-1} possible GIW permutations, which is enormous. In practice, one does not require this much diversity and other GIW algorithms may be sufficient. One reason to seek alternatives is that AGT requires saving 512 MB during checkpointing and transmission of the same amount of seed data to other scanner hosts in distributed implementations. Another reason is that AGT's CPU complexity is quite high, which we reduce in our next method.

Let ${}_b x$ be the lower b bits of an integer x and $\mathcal{R}(x)$ be the bit-reversal function. Then, we have the following result.

THEOREM 1. *Given a sequence of integers $\{x_k\}_{k=1}^n$, suppose $\{{}_b(x_k)\}_{k=1}^n$ has a full period for all $b = 1, 2, \ldots, 32$. Then, sequence $\{\mathcal{R}({}_{32}x_k)\}_k$ is GIW.*

PROOF. Assume that subnet s has depth b in the AGT (i.e., $n/|s| = 2^b$) and observe that GIW patterns must visit all remaining $2^b - 1$ subnets at depth b before returning to s. In practice, this means that the permutation must exhibit a full period in the upper b bits. Since for GIW this holds for all s, the full period must be simultaneously maintained at all depths $1 \le b \le 32$. Reversing the bits in each IP, we can replace this condition with a much simpler one – the full period must hold in the lower b bits. □

While there are many possible ways to construct $\{x_k\}_k$, an LCG of the form $x_k = ax_{k-1} + c$ is a natural choice due to its computational efficiency and need for only a single integer of state. To establish its suitability for Theorem 1, we note

[1]While completely refraining from scanning is even more polite, it does not produce any useful service-discovery results.

Type	Bit reversal	Rate (IP/sec)	State & seed
AGT	–	661,247	512 MB
LCG	Bit shifts	10,729,920	4 bytes
	Two-byte hash	21,263,889	4 bytes

Table 3: Benchmark of GIW address generation.

that the conditions for achieving a full period in $\{x_k\}_k$ with an LCG are well-known and require that $a-1$ be divisible by 4 and c be odd [26]. We call the resulting algorithm *Reversed LCG* (RLCG) and use it with $a = 214,013, c = 2,531,011$, which are well-known constants that produce an uncorrelated sequence of random variables (shown in Figure 2(a) for the last byte of generated IPs). Initial seed x_0 can then be used to change the scan order across multiple runs.

To efficiently reverse the bits, we use a 2-byte hash table that flips the order of bits in 16-bit integers. Therefore, any 32-bit IP can be processed in two memory lookups (i.e., 26 times faster than AGT); however, the CPU cache often makes this operation run even faster in practice. Table 3 benchmarks IP generation of AGT, naive bit-shifts (32 shifts down and 32 up), and the hash table technique. Observe that RLCG with a hash table runs at double the speed of bit-shifts and beats AGT by a factor of 32, which is slightly faster than 26 predicted by the analysis above. In more practical terms, AGT constrains the scanner to $T \geq 108$ minutes, while hash-based RLCG generates the entire permutation in just 3.4 minutes.

3.3 Scan Origin

While previous work has split the scan burden among m nodes to decrease total duration [3], [5], [17], [43] or obtain multiple vantage points [17], no apparent consideration has been given to the possible effect it has on the perceived politeness of the measurement. The main objective of a polite split in this paper is to maintain the GIW pattern across scanner IPs, which requires a mechanism for not only parceling the address space to m scanning hosts without burdensome message-passing, but also ensuring that each subnet s sees scanner IPs in a *perfectly alternating and equally-spaced fashion* (e.g., IP1, IP2, IP3, IP1, IP2, IP3, ...).

The rationale for using all m sources to scan each s lies in the fact that IDS (both open-source [7], [51] and commercial [24], [38]) detect scan traffic and throw alarms in response to perceived malicious activity based on individual source IPs. Therefore, a particular IP address sending its packets to s faster than other IPs is more readily detected as it simply stands out from the others. The reason for maximally spacing probes from different IPs is the same as before – reducing the overall burstiness at remote subnets – which for large m (i.e., hundreds or thousands) may become non-trivial. One example of an extremely impolite split is IP-sequential, which scans each s from a *single* source IP at rates similar to those in Table 2 (i.e., megabits per second and thousands of pps), regardless of subnet size.

Analysis shows that GIW split does not require a new permutation; however, individual source IPs must now return to s every $mn/|s|$ packets (i.e., alternating in some order with a full period m). Synchronizing m hosts using the block-split algorithms of previous work [3], [5], [17], [43], while sustaining the GIW split is a difficult problem. We instead introduce a new split algorithm that satisfies our conditions and requires low overhead/state.

Algorithm 1 RLCG/RR at host $i \in \mathcal{M}$

```
1: x_0 = rand()                                  ▷ Set initial seed x_0
2: for k = 1 to n do                             ▷ Iterate through all IPs
3:     ip = k mod m                              ▷ Assigned source IP
4:     x_k = a x_{k-1} + c                       ▷ Advance LCG
5:     if (ip == i) then                         ▷ Our IP?
6:         target = R(x_k)                       ▷ Reverse bits
7:         if (BGP[target]==VALID) then          ▷ In BGP?
8:             probe(target)                     ▷ Hit destination
9:         end if
10:    end if
11:    sleep(T/n)                                ▷ Wait for next packet
12: end for
```

The intuition behind our split, which we call *round-robin* (RR), is to generate a single RLCG permutation $\{z_k\}$ and assign target z_k to host k mod m. Assuming $\mathcal{M}$ is the set of scanning hosts, RR sends the initial seed x_0 to every host $i \in \mathcal{M}$, its position i, and the number of sources m. Each host then generates the entire sequence $\{z_k\}_k$ locally and hits target z_{i+jm} at time $(i+jm)T/n$ for $j = 0, 1, \ldots, n/m$, the simplicity of which is demonstrated in Algorithm 1. Even with $T = 24$ hours, subnets are visited so infrequently (e.g., every 337 seconds for a /24) that perfect synchronization of start times is not necessary. Furthermore, in scanners running from a single location, all m IPs can be aliased to the same host and RR-split can be used locally to ensure perfect synchronization, which is the approach taken by IRLscanner later in the paper.

From the well-known properties of LCGs [4], we immediately obtain the following crucial result.

THEOREM 2. *RR-split with any GIW permutation scans s with* $\min(|s|, m_s)$ *sources, where*

$$m_s = \frac{m}{\gcd(\frac{n}{|s|}, m)} \tag{1}$$

and $\gcd(a,b)$ *is the greatest common divisor of* (a,b).

PROOF. Examine $\{z_k\}_k$ and assume it is GIW. For a given subnet s, observe that its IPs appear in this list with a period $n/|s|$, which follows from the definition of GIW. Assuming $w_j \in \mathcal{M}$ is the j-th IP that hits s, we have

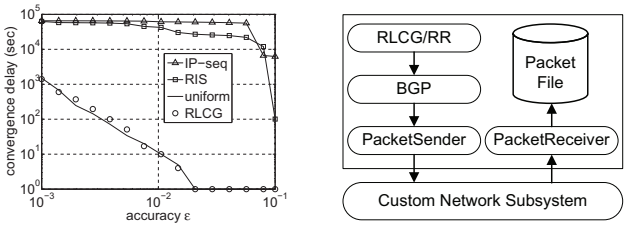
$$w_j = \left(w_{j-1} + \frac{n}{|s|}\right) \bmod m, \quad j = 1, 2, \ldots \tag{2}$$

This recurrence is an additive-only LCG whose period [4] is given by (1), which means that the number of sources scanning s is the smaller of its size and m_s. □

To better understand this result, examine Figure 2(b) that shows one example of (1) for $|s| = 65536$ (i.e., a /16 target subnet). Notice that even values of m lead to $m_s \leq m/2$ (triangles in the figure), which reduces the effective number of IPs seen by each subnet at least by half. The worst choice of m is a power of two, in which case $m_s = 1$ regardless of m. On the other hand, odd values of m produce the ideal $m_s = m$ (circles in the figure) and thus achieve a GIW split. We rely on this fact later in Section 4.

3.4 Extrapolation

Given the goal of being able to extrapolate the number of responsive hosts and other properties of open ports from a severely abbreviated scan (e.g., $1-10$ seconds instead of 24



(a) extrapolation delay (b) implementation

Figure 3: Extrapolation delay and IRLscanner implementation.

hours), we next examine how the existing and proposed approaches handle this problem. We split the allocated IANA space into three blocks (i.e., ARIN, RIPE, and APNIC), roughly corresponding to different geographical zones, and build three distributions of live IPs from our Internet measurements. Specifically, PMF function $p_j(x_3, x_4)$ specifies the probability that IP $x_1.x_2.x_3.x_4$ is alive in geographical zone $j \in \{1, 2, 3\}$. We then generate a Bernoulli random variable for each IP in the IANA space and make it alive using the corresponding probability $p_j(x_3, x_4)$.

Using a simulation with $T = 24$ hours, we scan the assigned distribution of live/dead hosts using four approaches – uniform, RLCG, IP-sequential, and RIS. Assuming A is the true number of live hosts in the assignment and $\tilde{A}(t)$ is an estimate at time t, define the relative extrapolation error $e(t) = |1 - \tilde{A}(t)/A|$. Convergence to threshold ϵ is established at such time t_ϵ when estimates for all $t \geq t_\epsilon$ have error smaller than ϵ.

Figure 3(a) plots the expected convergence delay t_ϵ averaged over 100 iterations. Observe that both RLCG and uniform converge to 1% error in 10 seconds, while RIS and IP-sequential take 11 and 16 hours, respectively. This result is easy to explain since IP-sequential gets trapped in certain CIDR blocks for an extended period and RIS hits the same last octet 16M times in a row. Furthermore, 0.1% error in Figure 3(a) can be achieved in 23 minutes for both uniform and RLCG, while the other two methods require 17+ hours. Even to arrive at 5% accuracy, which takes RLCG less than a second, RIS requires 6 hours, which makes this method unsuitable for all but most crude extrapolations.

3.5 Implementation

Figure 3(b) shows the general structure of IRLscanner. IPs generated by RLCG/RR are first checked against BGP prefixes and then delivered to the sending module, which forms raw Ethernet frames and transmits them to a custom network driver [49], which can transmit SYN packets at 5.25 Mpps using a modern quad-core CPU. Assuming 3.5 Gbps of available bandwidth, IRLscanner can cover the entire BGP space in 6.7 minutes from a single server.

The custom stack also allows us to intercept arbitrary incoming/outgoing packets and suppress RSTs from the OS TCP/IP stack, which we utilize later in the paper when profiling remote operating systems. All received packets are saved to disk without inspection and are processed offline. After completing each run, IRLscanner continues to listen for incoming packets for several hours to capture extremely slow hosts and record back-scanning packets from hackers and other potentially interesting entities.

3.6 Timeouts and Duration

Previous measurement studies [5], [42], [43] used retransmissions to the unresponsive set of target hosts to minimize false negatives, which we now evaluate in light of politeness and efficiency. Cursory inspection shows that retransmitting probes to unresponsive hosts is the violation of the GIW pattern, which is undesirable. Combining this with the likelihood that many false negatives in the unresponsive set are likely to be from persistently congested links or over-burdened servers [1], politeness concerns suggest that retransmission is not generally advisable.

From an efficiency standpoint, it should also be noted that the unresponsive set accounts for $90 - 99\%$ of $\mathcal{S}$ (depending on the protocol), which means that a single timeout-based retransmission would require almost doubling the number of sent packets. Our experiments show that retransmission not only yields a negligible increase in found hosts (i.e., by $0.3 - 1.7\%$ depending on the port and time of day), but also introduces bias by capturing hosts that come online within the retransmission timeout.

We next turn to the issue of *when* the status of an IP address can be determined, which in related work [5], [17], [42], [43] has occurred at some timeout after the initial probe was sent. Considerable effort has been spent deciding on appropriate timeout values [5], the choice of which affects the number of false negatives due to slowly responding hosts and the overhead of keeping large amounts of state for outstanding targets. Given that retransmissions are not required, we avoid this tradeoff entirely by delaying the classification of IP addresses until *after* the scan completes.

In practice, we accomplish this by saving all received packets (i.e., 25 GB per scan) to disk for later analysis. As there are many packets that are not relevant to the scan, we note that certain information can be embedded in the packets themselves to correlate responses with hosts scanned. This option has been used by encoding the target IP address in ICMP ID fields [17] and DNS queries [13]. For TCP scans, we take advantage of the sequence number field to encode the target IP, which allows us to detect invalid and/or malicious replies. Concerns about I/O speed do not arise until scan duration T drops below 60 seconds (i.e., assuming modest RAID write speed of 400 MB/s).

3.7 Negative Feedback

Throughout this section, we have explored and implemented several techniques to reduce the sending rate (i.e., BGP scope reduction), minimize the burden on remote networks (i.e., GIW), lower IDS false-alarm rates (i.e., RR-split), and avoid probing busy/firewalled/non-existent hosts with repeat packets (i.e., no retransmission).

In addition to technical solutions outlined above, a political strategy for reducing complaints and dealing with their aftermath is beneficial. Our general approach in this pursuit is to make the non-malicious purpose of our scans as transparent as possible to those remotely investigating our traffic. This includes providing scanning IPs with descriptive names (i.e., indicating their research purpose) in the forward/reverse DNS tree, as well as creation of TXT DNS records pointing to the project web-page with instructions on how to opt out. With over 123 IPs participating in this endeavor, special scripts have been written to manipulate IP assignment to various NIC interfaces and modify DNS records in our authoritative server.

Name	Protocol	Port	Type	Date	T	m	$\|\mathcal{O}\|$	$\|\mathcal{C}\|$	$\|\mathcal{U}\|$	pps	Mbps
DNS_1	UDP	53	DNS A	2-21-08	30d	1	15.2M	–	148M	709	0.48
DNS_2		53	DNS A	3-25-08	6d	5	15.2M	–	155M	3.5K	2.38
DNS_3		53	DNS A	5-07-08	1d	31	14.7M	–	168M	21.2K	14.28
DNS_4		53	DNS A	5-19-08	1d	31	14.5M	–	169M	21.2K	14.28
DNS_5		53	DNS A	5-20-08	1d	31	14.6M	–	168M	21.2K	14.28
DNS_6		53	DNS A	5-21-08	1d	31	14.5M	–	167M	21.2K	14.28
DNS_7		53	DNS A	5-22-08	1d	31	14.5M	–	169M	21.2K	14.28
ECHO		7	–	7-01-08	1d	31	322K	–	170M	22.1K	21.03
PING	ICMP	–	echo	6-24-08	1d	31	139M	–	99M	22.1K	14.85
$SMTP_S$	TCP	25	SYN	7-30-08	2d	61	17M	87.1M	119M	11.2K	7.55
$SMTP_A$		25	ACK	7-30-08	2d	61	–	116M		11.2K	7.55
$EPMAP_S$		135	SYN	8-05-08	2d	61	4.9M	40.2M	127M	11.3K	7.58
$EPMAP_A$		135	ACK	8-05-08	2d	61	–	68.4M		11.3K	7.58
$HTTP_1$		80	SYN	7-17-08	1d	123	30.3M	49.1M	78M	22.6K	15.19
$HTTP_2$		80	SYN	8-05-09	1d	61	44.3M	61.3M	97.1M	24.4K	16.39
$HTTP_3$		80	SYN	8-06-09	1d	61	44.0M	61.2M	85.1M	24.2K	16.26
$HTTP_4$		80	SYN	8-10-09	1d	123	44.2M	61.5M	94.7M	24.4K	16.39
$HTTP_5$		80	SYN	8-24-09	2d	123	44.5M	61.7M	96.4M	12.1K	8.15
$HTTP_6$		80	SYN	8-27-09	1d	61	44.1M	61.4M	80.7M	24.4K	16.37
$HTTP_{AS}$		80	ACK→SYN	9-02-09	1d	61	31.7M	49.6M	92M	25.8K	17.35
$HTTP_{OPT}$		80	SYN+OPT	7-15-10	1d	121	37.8M	48.1M	71.3M	26.3K	20.70

Table 4: Summary of scans performed.

However, the most widely-used means of investigation is through a whois lookup on offending IP addresses, followed by a direct email to the party listed therein. In the event a complaint is received, our policy is to reply as quickly as possible with an explanation of our traffic, a link to the project web-page, and an offer to blacklist the network. Dynamic blacklisting in IRLscanner is implemented through periodic reading of a flat file of blocked networks and simply removing them from the BGP hash table. Under the assumption that network administrators who complain will do so again later, blacklisted networks are maintained across scans. However, given that no analysis was provided in prior work [13], [17], [43] to justify preemptively removing subnets or addresses, our initial scan started with an empty blacklist.

The final issue one must also be aware of is that significant care should be taken to avoid negatively impacting the *local* network, where internal stateful firewalls and IDS are particularly vulnerable (from the load perspective) to large volumes of traffic destined to billions of unique destinations. We have experienced a number of issues with department and campus-wide IDS/firewall installations at our institution, which all had to be manually bypassed for this project to proceed.

4. EXPERIMENTS

In this section, we test our design decisions by performing several Internet-wide scans. We defer in-depth analysis of the actual scan data to a later paper, instead focusing on high-level observations and results.

4.1 Overview

As the goal of scanning is to produce the set of hosts offering a given service, each targeted IP address must eventually be classified into one of four categories. Define *open* set $\mathcal{O}$ to contain all hosts that responded positively to a scan packet (e.g., a SYN-ACK to a TCP SYN), *closed* set $\mathcal{C}$ to represent IPs responding negatively using the same protocol (e.g., a TCP RST to a SYN packet), *unreachable* set $\mathcal{U}$ to consist of IPs that return ICMP unreachable or TTL expired errors, and *dead* set $\mathcal{D}$ to designate hosts from which no reply was received at all. Note that excluding bogus responses and strange firewall/NAT behavior, $\mathcal{O} \cup \mathcal{C} \cup \mathcal{U} \cup \mathcal{D} = \mathcal{S}$ and the individual sets do not overlap.

Through development of IRLscanner and in the course of other projects, we have performed 21 Internet-wide scans since February 2008. To test a wide range of possibilities and demonstrate the general feasibility of service discovery, we targeted UDP, TCP, and ICMP protocols on both popular services (e.g., HTTP, DNS) and those often used for nefarious purposes (e.g., SMTP, EPMAP). Table 4 summarizes our scanning activity. We initially started slowly with a 30-day scan duration from a single source IP to gauge the feedback, then increased the sending rate over subsequent scans until we achieved a duration of 24 hours, which is over 20 times faster than any documented scan [17]. The number of source IPs m varied based on their availability in our subnet and specific goals of the measurement, generally ranging from 31 to 123. In comparison, the highest IP diversity in related work was $m = 25$ in [5], followed by $m = 8$ in [17].

4.2 UDP/ICMP Scans

We started with seven DNS scans due to an interest in public recursive DNS servers. These scans produced between 14.5M and 15.2M responses in each run, which represents a 30% growth from the 10.5M found in [13] less than 9 months prior. We discovered a stable set of 4.4M servers that responded to every DNS scan over a period of three months, which indicates that the number of consistently available hosts is far fewer than might be expected from the responses to a single scan.

Of further interest is the reduction in found hosts from 15.2M to 14.7M when scan duration reduced to 24 hours in DNS_3. This suggests that faster scan durations produce a lower cumulative response among the targets, which in part may be attributed to the lower possibility of counting the same host multiple times under different DHCP'ed IPs. To investigate whether previous scanning activity in some immediate past influences the response rate in subsequent scans, we probed DNS on four consecutive days in May 2008 (i.e., 96 hours of continuous scanning) and received roughly

Year	Data Source	IPs	ECN	MSS	TS	SACK	WS	EOL
2000	Padhye [40]	4.55K	1.15%	93.71%	–	40.75%	–	–
2004	Medina [33]	84.4K	2.09%	–	85%	67.79%	–	–
2005	Veal [54]	475	–	–	76.4%	–	–	–
2008	Langley [27]	1.35M	1.07%	–	–	–	–	–
2010	HTTP_{OPT}	37.8M	2.32%	99.47%	74.77%	60.94%	83.08%	6.21%

Table 5: Prevalence of TCP options in the Internet (dashes represent unreported values).

the same number of responses in each case, which indicates that the Internet is memoryless (at least at our scan rates).

Our last UDP scan was on ECHO port 7, which simply replies with a verbatim copy of the received packet and to our knowledge has never been scanned in the literature. We chose this port as a representative of a sensitive UDP service largely because of its notoriety for broadcast amplification attacks [32]. Later in the paper, we deal with the huge volume of complaints and speculation that ensued in the cooperative intrusion-detection community, but note that even though best practice is to disable this service, we nevertheless received replies from 321,675 unique IP addresses.

Our lone ICMP scan was a simple echo request [17], [43] that garnered 139M replies, representing a 20% gain over a similar scan performed in June 2007 [17].

4.3 TCP Scans

Our experiments targeted TCP with 12 scans using three target ports, two combinations of TCP flags, and one set of TCP options. TCP has not been scanned in the literature with T less than three months [5] and has not included any options or flags other than SYN [5], [17], [42], [43].

We start by describing the performed scans in an increasing order of their sensitivity. We initially scanned HTTP with a duration more than 90 times shorter than the only previous attempt [5], discovering 30.3M hosts in July 2008 and 44.5M in August 2009, the latter of which is a 140% increase compared to 18.5M IPs found in 2006 [5]. The other two services we targeted with SYN scans were SMTP, which is frequently probed by spammers searching for open relays, and EPMAP, which is heavily scanned for network reconnaissance prior to attack [32], discovering 17M and 4.9M hosts respectively. Given the large number of Windows hosts in the Internet, the EPMAP result seems low, which suggests that many ISPs filter SYN traffic on port 135.

To determine the feasibility of scanning with other types of TCP packets, we performed three measurements with ACK packets (i.e., SMTP_A, EPMAP_A, and HTTP_{AS}), which can be used not only to determine a host's liveness (i.e., an ACK normally elicits a RST from non-firewalled stacks), but also to bypass stateless firewalls. Both SMTP_A and EPMAP_A were executed *concurrently* with the corresponding SYN scan (i.e., two packets were sent to each IP) in order to allow us to detect and characterize firewalls. Observe in the table that SMTP_A found 12M more hosts (i.e., 116M total) than SMTP_S whose $|\mathcal{O}| + |\mathcal{C}|$ is only 104.1M. Without firewalls, the two sets should be identical. Similarly, EPMAP_A elicited 23M more responses (i.e., 68M total) compared to EPMAP_S (i.e., 45.1M). The EPMAP_A scan also suggests that filtering is heavily applied on port 135 not only for SYN packets, but for ACKs as well.

For HTTP_{AS}, we scanned the entire BGP space with ACK packets, then immediately followed the resulting RST responses with a SYN packet. We present our motivation and the results from this approach in a later section.

4.4 TCP Options

Over the last decade there has been an interest in the literature [27], [33], [40], [54] regarding the deployment of various extensions to TCP (e.g., SACK, ECN) in network stacks of both end-systems and intermediate network devices (e.g., routers, NATs, firewalls). While our first 20 scans did not utilize any options, the last scan HTTP_{OPT} not only attempted to negotiate ECN [45], but also transmitted four TCP options – MSS (maximum segment size), WS (window scaling), SACK (selective acknowledgments), and TS (timestamps), which are normally echoed in the SYN-ACK if the corresponding target supports it. The order of options transmitted by the scanner followed that in Windows Server 2008 (i.e., MSS, NOP, WS, SACK, TS).

Observe in Table 4 that in July 2010 HTTP_{OPT} yielded only 37.8M responses, which represents a nearly 15% reduction from HTTP scans a year earlier. This does not align well with the 30% annual growth rate during 2006-2009 and suggests that option-heavy packets indeed produce a dramatically lower response rate in the Internet.[2] While this single scan is insufficient to conclusively pinpoint which options are responsible for the dropped SYN packets, the most likely culprit is ECN. A similar result was found in [33], [40], where between 1 and 8% of the tested web servers were unable to accept ECN connections due to various protocol issues in end-systems and interference from ECN-challenged intermediate devices [33]. The larger percentage of ECN failures in our dataset is likely caused by the broader range of sampled embedded stacks (e.g., printers, cameras, modems).

Our next task is to analyze the number of responsive hosts that support each of the options. Prior work [27], [33], [40], [54] has examined this problem on a smaller scale using traditional (i.e., non-embedded) web servers and found that ECN support ranged from 1 to 2%, while each of SACK, MSS, and TS was enabled in at least 40% of the hosts. Table 5 summarizes these findings and shows our results, including two new fields – window scaling (WS) and end-of-options-list (EOL). Excluding a few nonsensical replies, the percentage of hosts that accept ECN still remains small (i.e., 2.32%), having grown by only 1.3% over the last 10 years.

As shown in the next column of the table, MSS is now included in almost all connections, up from 94% in 2000. While in our earlier experiments the source port remained constant for the duration of each scan, HTTP_{OPT} randomly varied the source port of each connection (using a separate 2-byte LCG that skipped reserved ports), which was encoded in the TCP timestamp of the outgoing SYN packet and later checked against the destination port of the SYN-ACK. As

[2]Other reasons include increased deployment of ISP firewalls, new OS patches that block port 80, and fewer hosts on the Internet, none of which is historically probable.

Device	Found	%
Linux (2.4 or 2.6 kernel)	13.0M	32.9
Windows XP/Server 2003	6.3M	15.8
Windows Vista/7/Server 2008	5.6M	14.0
Windows Server 2003 SP2	3.5M	8.9
FreeBSD	1.5M	3.8

Table 6: Top 5 devices.

Device Type	Found	%
General purpose	32.4M	81.8
Network device	2.7M	6.8
Printer	1.8M	4.6
Networked storage	1.5M	3.7
Media	929K	2.3
Other embedded	287K	0.7
Total	39.6M	

Table 7: Summary of fingerprinted devices.

OS Class	Found	% of GP
Windows	16.3M	50.2
Linux	13.0M	40.2
BSD/Unix	2.2M	6.7
Mac	862K	2.7

Table 8: General purpose (GP) devices.

the table shows, approximately 75% of the responsive IPs returned the correct timestamp. Ignoring a handful of bogus replies, the remaining 25% of the hosts did not support TS.

Next, SACK results suggest a 20% increase from 2000 and a 6% decrease from 2004, though our numbers are again affected by the wider coverage of embedded devices than in prior work. While WS is often used to enable high-speed transfers in networks with large bandwidth-delay products, a surprising 83% of hosts support it. Finally, over 6% of Internet devices are compelled to use EOL, even though this optional field typically serves no practical purpose.

4.5 Remote OS Fingerprinting

While service discovery projects usually focus on enumerating open set $\mathcal{O}$, further information about the hosts themselves is often critical to the depth and usefulness of measurement studies [5], [13]. With the goals of resource efficiency and maximal politeness at remote networks, in this section we focus on determining the operating system of discovered hosts in $\mathcal{O}$, which could be used to estimate the global impact of known security vulnerabilities [35], approximate Internet-wide market share [37], or track hosts with dynamic IP addresses [13]. The main difficulty in executing such a study is that most existing tools [39], [57] not only trip IDS alarms and crash older end-hosts with unusual combinations of TCP/IP flags, but also require substantial overhead (e.g., 16 packets for Nmap) in Internet-wide use [50], [55]. It is thus not surprising that large-scale OS profiling has not been attempted in the literature.

Instead of traditional fingerprinting methods, we utilize a single-packet technique called Snacktime [50], which exploits OS-dependent features in SYN-ACKs such as the TCP window, IP time-to-live, and the sequence of RTOs (retransmission timeouts) of the SYN-ACK during TCP handshakes. While initial results on accuracy were promising [50], [55], Snacktime's requirement that outgoing TCP RST packets be dropped, long period needed to produce an answer (e.g., several minutes), and limited database (i.e., 25 signatures last updated in 2003) has previously restricted its usefulness. Given that we must already send a TCP SYN packet to every host in $\mathcal{O}$, modifying the Snacktime technique for use on an Internet-wide scale would result in *no additional sent packets* to enumerate remote OSes.

To implement a scalable Snacktime, we take advantage of our custom network driver to block outgoing TCP RST packets in response to arriving SYN-ACKs. Since IRLscanner already captures all incoming packets (including delayed retransmissions), it is a perfect platform for massively parallelizing the Snacktime technique. After a scan completes, we generate the RTOs of each target from the packet dump, then run a custom implementation of the Snacktime matching algorithm that gives preference to general classes of operating system in the case of ambiguity and reduces the microsecond precision of RTOs to manage random queuing delays in the Internet. To make the technique more useful, we processed almost 7K responsive hosts at a large university to manually verify and increase the database to 98 signatures, including the latest Windows versions (e.g., Vista, 7, Server 2008, Server 2003 SP2), webcams, switches, printers, and various other devices.

We applied the modified Snacktime technique to HTTP_2, which consisted of $|\mathcal{O}| = 44.3$M web servers that responded with at least one SYN-ACK on port 80. We achieved a Snacktime fingerprint for 39.6M hosts, with 2.3M being excluded due to insufficient retransmissions (i.e., none) and the remaining difference attributable to gaps in our signature database (the algorithm, signatures, and dataset are available [21]). According to Snacktime, the top 5 profiled OSes are given in Table 6, with Linux contributing 32.9% of the total and various Windows implementations consisting of the next several slots, which is indicative of their co-dominance in the web-server market. We provide more detail in Table 7, where we classified each signature into one of six categories and calculated summary statistics. Note that general-purpose systems (e.g., Linux, Windows) are responsible for nearly 82% of the total, with network devices (e.g., switches, routers, NAT boxes), networked storage (e.g., NAS, tape drives), and printers showing up with more than 1M devices each. The media category is comprised mainly of webcams and presentation devices (e.g., TVs, DVRs, projectors).

To finish this section, we present in Table 8 the total number of devices and their percentage attributed to each class of OS in the general-purpose category. Snacktime results suggest that approximately half of the total consists of Microsoft OSes (5.6% of which belong to Windows 2000 or older), which is likely due at least partially to individuals hosting personal web-sites on their home machines. It also shows that Linux hosts are responsible for 40%, which combined with the various related forms of BSD (e.g., OpenBSD, FreeBSD), SunOS, and Unix results in nearly 47% of the total and rivals Microsoft.

4.6 Service Lifetime

Another interesting property of Internet services is their average *lifetime* (uptime) $E[L]$, which is the mean duration of time a port stays active on a given IP. One technique [17] is to first estimate the CDF of lifetime L and then compute its mean $E[L]$. However, avoiding round-off errors and CDF

Service	Scans	Emails	Avg	IPs excluded	Avg
DNS	7	45	6.4	3.7M	530K
Echo	1	22	22	752K	752K
Ping	1	4	4	1K	1K
HTTP	8	27	3.4	459K	57K
SMTP	2	6	3	262K	131K
EPMAP	2	2	1	65K	32K
Total	21	106	5.05	5.3M	250K

Table 9: Emails and IPs excluded by service.

Source	Cease		FYI		Total
	Human	Script	Human	Script	
Individual	14	14	7	8	43
Government	6	2	6	2	16
Corporation	11	5	8	0	24
University	6	3	10	4	23
Total	37	24	31	14	106

Table 10: Email notices by complainant type.

tail cut-off often requires monitoring the pool of target IPs at frequent intervals (i.e., minutes) and for extended periods of time (i.e., days), all which contributes not only to higher bandwidth overhead, but also to more likely aggravation of remote network administrators.

We offer an alternative method that can estimate $E[L]$ using much lower overhead and overall delay. Modeling each host as an alternating ON/OFF process [56], a set $\mathcal{K}$ of uniformly selected live hosts exhibits a departure rate $\lambda = |\mathcal{K}|/E[L]$ hosts/sec (a similar result follows from Little's Theorem). Thus, by probing $\mathcal{K}$ twice at time t and $t + \Delta$, one can estimate λ as $p(\Delta)|\mathcal{K}|/\Delta$, where $p(\Delta)$ is the fraction of hosts that have disappeared in this interval. Solving $p(\Delta)|\mathcal{K}|/\Delta = |\mathcal{K}|/E[L]$, we obtain $E[L] = \Delta/p(\Delta)$.

The key to this technique is to uniformly randomly select $\mathcal{K}$ and simultaneously ensure maximal politeness of the scan. Leveraging the findings of Section 3.4 aimed exactly at this issue, we first use RLCG to scan the Internet for Δ time units at some constant rate r. We then re-generate the same sequence of IPs at the same rate, but actually send packets only to those targets that have responded in the first scan. Due to limited space, we omit simulations confirming the accuracy of this method and discuss only one extrapolation using port 80 and $\Delta = 45$ seconds. This experiment covered 1M targets, found $|\mathcal{K}| = 23.7$K live hosts, and yielded $E[L] = 50$ minutes (i.e., $p(\Delta) = 1.5\%$).

5. ANALYSIS

While it would be ideal to scan the Internet using different techniques (e.g., IP-sequential, uniform, GIW) and then assess the collected feedback as a measure of intrusiveness of each scan, certain practical limitations typically prevent one from doing so (e.g., our network administrators have explicitly prohibited scanning activity using certain non-optimal permutations). Thus, comparison is often only possible through feedback analysis exposed in publications, which unfortunately is very scarce in the existing literature. To overcome this limitation, this section introduces a number of novel metrics related to the perceived intrusiveness of Internet-wide scans, studies them in detail, and unveils certain simple, yet effective, techniques for reducing the blowback.

5.1 Email Complaints

One of the uncertainties we encountered when initially considering a service discovery project was the number of complaints to expect, particularly as they related to serious threats or resulted in widespread blacklisting of the scanner to the point of making Internet-wide measurements impossible. In this section, we attempt to clarify the issue by detailing the complaints we received and the effect they had on our measurements.

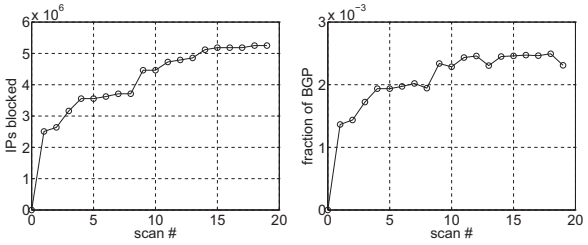
Table 9 contains a summary of email complaints broken down by service type. Over all 21 scans, we received 106 complaints for an average of 5.05 per scan. Our initial run (i.e., DNS_1) resulted in 10 complaints and more than 2.5M IP addresses blocked, which is nearly half the total of 5.3M blacklisted over the course of the project. Most of this initial number came from a single large ISP asking us to block several /16 residential networks. However, even with the initial burst removed from the calculation, DNS scans resulted in an average of 172K blacklisted IPs per scan. The most significant backlash we received was for the ECHO scan (UDP port 7), which led to 22 complaints and more than 750K blocked IP addresses. In the next section we provide an explanation for this significant increase, but note here that UDP scans account for 65% of all complaints, while being responsible for only 40% of the packets sent.

In contrast to the experience of [17], where the authors received 30 times more complaints for a TCP scan than ICMP pings, our TCP measurements produced a *total* of 35 complaints over 12 scans, or about three per scan. This is an even more remarkable result given that we scanned two sensitive ports, used ACK packets that penetrate stateless firewalls, and clustered six scans in less than a month. While we cannot explain this discrepancy, our numbers do not support the notion that TCP scans are more invasive than the other protocols.

We next categorize the received emails in Table 10 to show the severity and type of each complaint. Out of 106 complaints, 61 were demands to cease the activity, while the other 45 were FYI notifications about a possible virus with no expectation that the measurement stop. The first row of the table shows that individual users who monitor a single IP address with a personal firewall (e.g., ZoneAlarm, Norton) represented 41% of the total complaints (i.e., 43 out of 106), which indicates that a large portion of these emails cannot be avoided by any means. The remaining three rows of the table represent complaints received from large network entities, with universities being the most likely to send an FYI notification and worldwide government entities comprising only 15% of the total complaints.

In contrast to [13], we received only four cease demands from U.S. Federal Government entities, none of which were defense-related. Another point of interest is the number of threats to pursue legal action, though of the three received none of them turned out to be legitimate. Finally, analysis of emails generated by an automated script suggests that a large chunk of all received complaints (i.e., 36% in our case) are seldom reviewed by an actual human given the large amount of background scan traffic their networks receive [41].

We now determine the impact of email complaints on the scope of subsequent measurements (i.e., size of $\mathcal{S}$ after removing blacklisted networks) by studying the progres-



(a) total (b) fraction of BGP

Figure 4: Progression of blacklisted IPs.

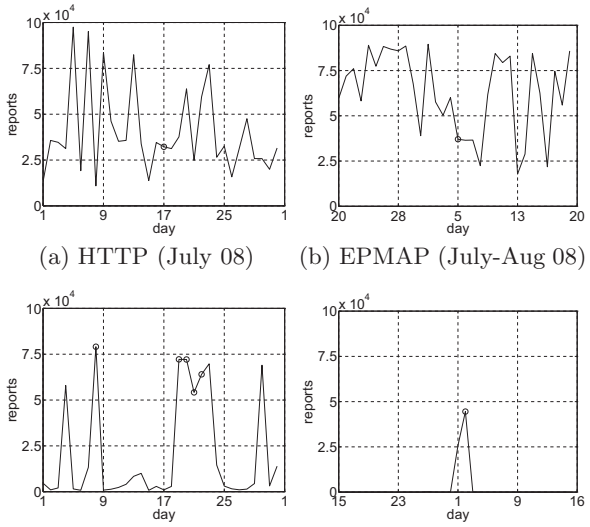
sion of blacklisted IP addresses in Figure 4, where scans are assigned numbers in chronological order. Note that the complaints for the two simultaneous scans (i.e., SMTP and EPMAP) are encompassed in a single data point due to our inability to tell whether the SYN or the ACK portion caused the complaint. Part (a) of the figure contains the raw number of blacklisted addresses, which did not increase significantly after we stopped scanning UDP. Part (b) shows the blocked addresses as a percentage of BGP, where the total number of 5.3M represents only 0.23% of the current space (the curve is non-monotonic due to the constant expansion of BGP).

5.2 Firewall Log Correlation

To gain a broader view of the Internet and decrease the amount of time required to detect large-scale attacks, online collaborative systems [36], [47] have been developed to pool data from strategically placed Internet sensors and firewall/IDS logs of various networks. We focus on the SANS Internet Storm Center (ISC) [47] due to its relatively large size of 500K monitored IP addresses and detailed publicly available data. An ISC report consists of an IP address detected as a scanner, its source port, and the target's (IP, port) pair. These reports are often shared publicly, although certain fields (e.g., destination IP) are obscured to protect the identity of subnets that submit their logs. Given that these reports represent information about unwanted traffic, they can be used to gain insight into how our scans are perceived by remote networks.

We examine ISC report summaries for several scans from Table 4. These summaries are compiled daily for each service (e.g., HTTP) and consist of the number of scanned targets, scanning hosts that targeted that service, and the ratio of packets that are TCP. We are particularly interested in the first metric as *all* reports related to our 24-hour scans should be contained in a single data point.

We downloaded summary data from ISC for one month surrounding a sample of our HTTP, EPMAP, DNS, and ECHO scans (i.e., 15 days prior and after). The result is plotted in Figure 5, where the x-axis labels days in the 30-day window surrounding each scan and the highlighted points represent the days our scanner was actively probing that particular port. We happened to scan both HTTP in part (a) and EPMAP in part (b) on days when ISC experienced roughly a third of its peak number of daily reports (i.e., 27K compared to $80 - 90$K), which is nevertheless an huge number. The figure also shows that EPMAP clearly stands out as being scanned with a consistently high amount of daily traffic.



(a) HTTP (July 08) (b) EPMAP (July-Aug 08)

(c) DNS (May 08) (d) ECHO (June-July 08)

Figure 5: ISC reports with our scans marked.

In contrast, parts (c) and (d) for DNS and ECHO show that IRLscanner spiked report levels well above those of surrounding days. In fact, in the case of ECHO we produced an extremely anomalous event, raising the total from almost zero to 50K. Our activity on that port created concerns among network administrators that a new exploit was under way and/or a virus outbreak was in progress. All this eventually drew the attention of one of the traffic monitors at ISC, who wrote an explanatory blog post to calm down the ISC community.

Given the large amount of background noise from many scanning sources (whose totals ISC also makes available) in parts (a) and (b) of Figure 5, we conjecture that network administrators are more likely to react only to traffic that clearly stands out (i.e., makes its presence known by its high signal-to-noise ratio) rather than to scans on sensitive ports. This is confirmed by the fact that attack-reconnaissance port 135 generated the *least* number of complaints and that the ECHO port, which inherently represents little real threat to administrators due to the lack of hosts offering this service and heavy firewall filtering, produced an unusually strong blowback. This relationship where higher background scan traffic seems to imply fewer complaints might benefit researchers considering scans on sensitive/popular ports in the future.

5.3 Enumerating Contributors

It is well-known [6], [48] that the contributors to ISC and other firewall log correlation systems are vulnerable to losing their anonymity due to the nearly real-time public display of firewall reports with only the destination IP address omitted. Several techniques for correlating reports with targeted subnets (which is called *contributor enumeration*) have been proposed [6]; however, they require tens of billions of packets, allow for false positives, and consume multiple days during full enumeration.

Given our high-performance scanner that is capable of locally using multiple IP addresses, a much simpler attack preys on the source port, destination port, and source IP ad-

dress reported in detailed ISC logs, which are displayed for all scanning hosts that ISC tracks. Probing each address in BGP set $\mathcal{B}$ with a unique combination of source/destination ports and source IP eliminates the possibility of false positives and the need to send any extra packets beyond those in $\mathcal{B}$. This can be accomplished for the current 2.1B hosts with 128 source IPs by simply rotating through all 64K source ports and roughly 250 destination ports, which can be handpicked from the most-scanned lists to minimize the likelihood of raising suspicion. However, by removing the source port from the public report, ISC can render this technique largely ineffective.

5.4 ACK Scans

To prepare their subnet's data for submission to ISC, many network administrators rely on firewall log analyzers such as psad [44] to separate scan traffic from innocuous packets dropped by the firewall. During our analysis, we discovered that many such tools ignore ACK packets, which suggests that network administrators often do not consider them to be particularly dangerous. To leverage this intuition, we propose a scan technique for cases where finding the majority of hosts in open set $\mathcal{O}$, while significantly reducing IDS detection, is beneficial (e.g., for rarely scanned ports).

The first phase of the technique is a simple ACK scan to every host in $\mathcal{B}$, which effectively discovers the subset of hosts that are not heavily protected by stateful firewalls. For every RST received, we verify that it has not been previously probed using a hash table and then immediately send it a SYN packet to establish whether the service is open or not. By only targeting hosts that previously responded, this type of scan reduces the SYN footprint by 94% for HTTP. We performed a single test measurement (HTTP_{AS} in Table 4), which discovered 31.7M of the 44M total responsive hosts, while requiring only 125M SYN packets to be sent. ISC data shows only $4,746$ reports for our IPs during HTTP_{AS} compared to $29,869$ reports collected for HTTP_2, which used the same T and m. This is significant as it amounts to an 84% decrease in the perceived intrusiveness of the scan.

5.5 DNS Lookups

We now turn to the last form of feedback we consider in this paper. While whois lookups seem to be the predominate form of reconnaissance performed by remote network administrators and individuals when they detect a scan, many specialized tools augment IDS reports and firewalls logs with DNS lookups on offending IPs to provide more information on the scanning host to the user. While for large networks this functionality should be disabled (as it allows remote hosts to DoS the network by loading it up with billions of useless DNS lookups), many personal firewalls and small subnets implement some form of it.

We tested the frequency of these additional lookups by collecting all incoming requests for IP addresses and hostnames at our locally controlled authoritative DNS server. To ensure that each request initiated by a remote entity contacted our nameserver, we set the DNS TTL to zero for both the reverse and forward lookups on scanner IPs/hostnames. After doing so, no RFC compliant nameserver should maintain our records in their cache.

The result of this collection process for three HTTP scans is contained in Table 11, which lists the number of reverse lookups for the IP addresses themselves and forward lookups on the names returned by those queries. We made sure that these IP addresses were not used for any other purpose but scanning and their names were not publicized beyond the project web-site. Therefore, forward lookups are almost certainly due to the common verification technique of determining the consistency between the forward and the reverse response. While the number of requests slightly declined for each subsequent scan, the last column shows that the number of unique servers in each dataset had the opposite trend. The decline in lookup rates can be attributed to random noise, long-term caching at non-compliant DNS servers, and users growing tired of looking up our IPs.

Scan	Reverse	Forward	Req/sec	Servers
HTTP_2	3.03M	85.4K	36	47.8K
HTTP_3	2.89M	80.1K	34	48.2K
HTTP_6	2.85M	66.3K	33	49.2K

Table 11: DNS lookups on scanner source IPs.

It should be noted that performing DNS queries on scanner IPs potentially reveals the location of the IDS tool unless steps are taken to increase anonymity (e.g., using a DNS forwarding service at some distant location). The three scans in Table 11 have identified 64K unique DNS servers, out of which 35K were present in each dataset. Further analysis of this data is deferred to future work.

6. CONCLUSION

In this paper, we designed novel algorithms for maximally polite Internet scanning and documented our experience through extensive service-discovery statistics based on our implementation of the proposed methods. Future work involves exploring methods for reducing $\mathcal{B}$ to avoid scanning unproductive networks, expanding RLCG/RR to provide optimal spacing for multiple destination ports (i.e., in hybrid vertical/horizontal scanning), and more in-depth analysis of scan data.

7. ACKNOWLEDGMENT

We are grateful to the Network Security team at Texas A&M University for their helpful discussions and professional handling of network administrator complaints.

8. REFERENCES

[1] M. Allman, W. M. Eddy, and S. Ostermann, "Estimating loss rates with TCP," *ACM Performance Evaluation Review*, vol. 31, pp. 12–24, 2003.

[2] M. Allman, V. Paxson, and J. Terrell, "A Brief History of Scanning," in *Proc. ACM IMC*, Oct. 2007, pp. 77–82.

[3] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Understanding Passive and Active Service Discovery," in *Proc. ACM IMC*, Oct. 2007, pp. 57–70.

[4] H. Bauke and S. Mertens, "Random numbers for large-scale distributed Monte Carlo simulations," *Phys. Rev. E*, vol. 75, no. 6, p. 066701, 2007.

[5] D. Benoit and A. Trudel, "World's First Web Census," *Intl. Journal of Web Information Systems*, vol. 3, no. 4, pp. 378–389, 2007.

[6] J. Bethencourt, J. Franklin, and M. Vernon, "Mapping Internet Sensors with Probe Response Attacks," in *Proc. USENIX Security*, Jul. 2005, pp. 193–208.

[7] Bro IDS. [Online]. Available: http://bro-ids.org/.

[8] H. Burch and B. Cheswick, "Mapping the Internet," *IEEE Computer*, vol. 32, no. 4, pp. 97–98,102, 1999.

[9] M. Casado, T. Garfinkel, W. Cui, V. Paxson, and S. Savage, "Opportunistic Measurement: Extracting Insight from Spurious Traffic," in *Proc. HOTNETS*, Nov. 2005.

[10] Z. Chen and C. Ji, "Optimal Worm-Scanning Method using Vulnerable-Host Distributions," *ACM IJSN*, vol. 2, no. 1/2, pp. 71–80, Mar. 2007.

[11] B. Cheswick, H. Burch, and S. Branigan, "Mapping and Visualizing the Internet," in *Proc. USENIX Annual Technical Conference*, Jun. 2000, pp. 1–12.

[12] Team Cymru. [Online]. Available: http://www.team-cymru.org/Services/Bogons/.

[13] D. Dagon, N. Provos, C. P. Lee, and W. Lee, "Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority," in *Proc. NDSS*, Feb. 2008.

[14] L. Deri and F. Fusco, "Exploiting Commodity Multicore Systems for Network Traffic Analysis," Jul. 2009. [Online]. Available: http://ethereal.ntop.org/MulticorePacketCapture.pdf.

[15] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet Map Discovery," in *Proc. IEEE INFOCOM*, Mar. 2000.

[16] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang, "Accessing the Deep Web," *Comm. of the ACM*, vol. 50, no. 5, pp. 94–101, 2007.

[17] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister, "Census and Survey of the Visible Internet," in *Proc. ACM IMC*, Oct. 2008, pp. 169–182.

[18] Geoff Huston. [Online]. Available: http://bgp.potaroo.net.

[19] IANA, "IPv4 Global Unicast Address Assignments." [Online]. Available: http://www.iana.org/assignments/ipv4-address-space/.

[20] IANA, "Special-Use IPv4 Addresses," *IETF RFC 3330*, Sep. 2002.

[21] IRL Internet-Wide Fingerprinting Dataset. [Online]. Available: http://irl.cs.tamu.edu/projects/sampling/.

[22] B. Irwin and J. P. van Riel, "Using InetVis to Evaluate Snort and Bro Scan Detection on a Network Telescope," in *Proc. IEEE VizSEC*, Oct. 2007, pp. 255–273.

[23] ISC Internet Domain Survey. [Online]. Available: http://www.isc.org/index.pl?/ops/ds/.

[24] Juniper IDP. [Online]. Available: http://www.juniper.net/.

[25] M. G. Kang, J. Caballero, and D. Song, "Distributed Evasive Scan Techniques and Countermeasures," in *Proc. DIMVA*, Jul. 2007, pp. 157–174.

[26] D. Knuth, *The Art of Computer Programming, Vol. II*, 2nd ed. Addison-Wesley, 1981.

[27] A. Langley, "Probing the Viability of TCP Extensions," Google, Inc., Tech. Rep., Sep. 2008.

[28] S. Lawrence and C. L. Giles, "Accessibility of Information on the Web," *Intelligence*, vol. 11, no. 1, pp. 32Ű–39, 2000.

[29] D. H. Lehmer, "Mathematical Methods in Large-Scale Computing Units," in *Proc. Large-Scale Digital Calculating Machinery*, 1949, pp. 141–146.

[30] Z. Li, A. Goyal, Y. Chen, and V. Paxson, "Automating Analysis of Large-Scale Botnet Probing Events," in *Proc. ACM ASIACCS*, Mar. 2009, pp. 11–22.

[31] P. K. Manna, S. Chen, and S. Ranka, "Exact Modeling of Propagation for Permutation-Scanning Worms," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 1696–1704.

[32] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed: Network Security Secrets & Solutions*, 5th ed. McGraw-Hill/Osborne, 2005.

[33] A. Medina, M. Allman, and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet," *ACM SIGCOMM Comp. Comm. Rev.*, vol. 35, no. 2, pp. 37–52, Apr. 2005.

[34] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang, "IPv4 Address Allocation and the BGP Routing Table Evolution," *ACM SIGCOMM Comp. Comm. Rev.*, vol. 35, no. 1, pp. 71–80, 2005.

[35] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *IEEE S&P*, vol. 1, no. 4, pp. 33–39, 2003.

[36] myNetWatchman – Network Intrusion Detection and Reporting. [Online]. Available: http://www.mynetwatchman.com/.

[37] Netcraft Web Server Survey. [Online]. Available: http://news.netcraft.com/.

[38] NetVCR. [Online]. Available: http://www.niksun.com/.

[39] Nmap. [Online]. Available: http://nmap.org/.

[40] J. Padhye and S. Floyd, "On Inferring TCP Behavior," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 287–298.

[41] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of Internet Background Radiation," in *Proc. ACM IMC*, Oct. 2004, pp. 27–40.

[42] N. Provos and P. Honeyman, "ScanSSH - Scanning the Internet for SSH Servers," in *Proc. USENIX LISA*, Dec. 2001, pp. 25–30.

[43] Y. Pryadkin, R. Lindell, J. Bannister, and R. Govindan, "An Empirical Evaluation of IP Address Space Occupancy," USC/ISI, Tech. Rep. ISI-TR-2004-598, Nov. 2004.

[44] Psad: Intrusion Detection and Log Analysis with Iptables. [Online]. Available: http://www.cipherdyne.org/psad/.

[45] K. K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *IETF RFC 3168*, Sep. 2001.

[46] Route Views Project. [Online]. Available: http://www.routeviews.org/.

[47] SANS Internet Storm Center – Cooperative Network Security Community. [Online]. Available: http://isc.sans.org/.

[48] Y. Shinoda, K. Ikai, and M. Itoh, "Vulnerabilities of Passive Internet Threat Monitors," in *Proc. USENIX Security*, Jul. 2005, pp. 209–224.

[49] M. Smith and D. Loguinov, "Enabling High-Performance Internet-Wide Measurements on Windows," in *Proc. PAM*, Apr. 2010, pp. 121–130.

[50] Snacktime: A Perl Solution for Remote OS Fingerprinting. [Online]. Available: http://www.planb-security.net/wp/snacktime.html.

[51] Snort IDS. [Online]. Available: http://www.snort.org/.

[52] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical Automated Detection of Stealthy Portscans," *Computer Security*, vol. 10, no. 1–2, pp. 105–136, 2002.

[53] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," in *Proc. USENIX Security*, Aug. 2002.

[54] B. Veal, K. Li, and D. Lowenthal, "New Methods for Passive Estimation of TCP Round-Trip Times," in *Proc. PAM*, Mar. 2005, pp. 121–134.

[55] F. Veysset, O. Courtay, and O. Heen, "New Tool And Technique For Remote Operating System Fingerprinting," Intranode Software Technologies, Tech. Rep., Apr. 2002.

[56] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks," in *Proc. IEEE ICNP*, Nov. 2006, pp. 32–41.

[57] F. V. Yarochkin, O. Arkin, M. Kydyraliev, S.-Y. Dai, Y. Huang, and S.-Y. Kuo, "Xprobe2++: Low Volume Remote Network Information Gathering Tool," in *Proc. IEEE/IFIP DSN*, Jun. 2009, pp. 205–210.

[58] J. Zhang, P. Porras, and J. Ullrich, "Highly Predictive Blacklisting," in *Proc. USENIX Security*, Jul. 2008, pp. 107–122.

[59] C. C. Zou, D. Towsley, W. Gong, and S. Cai, "Routing Worm: A Fast, Selective Attack Worm based on IP Address Information," in *Proc. IEEE PADS*, Jun. 2005, pp. 199–206.