ZMap: Fast Internet-wide Scanning and Its Security Applications

Zakir Durumeric, Eric Wustrow, and J. Alex Halderman, *University of Michigan*

This paper is included in the Proceedings of the
22nd USENIX Security Symposium.

August 14–16, 2013 • Washington, D.C., USA

ISBN 978-1-931971-03-4

Open access to the Proceedings of the
22nd USENIX Security Symposium
is sponsored by USENIX

ZMap: Fast Internet-Wide Scanning and its Security Applications

Zakir Durumeric
University of Michigan
zakir@umich.edu

Eric Wustrow
University of Michigan
ewust@umich.edu

J. Alex Halderman
University of Michigan
jhalderm@umich.edu

Abstract

Internet-wide network scanning has numerous security applications, including exposing new vulnerabilities and tracking the adoption of defensive mechanisms, but probing the entire public address space with existing tools is both difficult and slow. We introduce ZMap, a modular, open-source network scanner specifically architected to perform Internet-wide scans and capable of surveying the entire IPv4 address space in under 45 minutes from user space on a single machine, approaching the theoretical maximum speed of gigabit Ethernet. We present the scanner architecture, experimentally characterize its performance and accuracy, and explore the security implications of high speed Internet-scale network surveys, both offensive and defensive. We also discuss best practices for good Internet citizenship when performing Internet-wide surveys, informed by our own experiences conducting a long-term research survey over the past year.

1 Introduction and Roadmap

Internet-scale network surveys collect data by probing large subsets of the public IP address space. While such scanning behavior is often associated with botnets and worms, it also has proved to be a valuable methodology for security research. Recent studies have demonstrated that Internet-wide scanning can help reveal new kinds of vulnerabilities, monitor deployment of mitigations, and shed light on previously opaque distributed ecosystems [10, 12, 14, 15, 25, 27]. Unfortunately, this methodology has been more accessible to attackers than to legitimate researchers, who cannot employ stolen network access or spread self-replicating code. Comprehensively scanning the public address space with off-the-shelf tools like Nmap [23] requires weeks of time or many machines.

In this paper, we introduce ZMap, a modular and open-source network scanner specifically designed for performing comprehensive Internet-wide research scans. A single mid-range machine running ZMap is capable of scanning for a given open port across the entire public IPv4 address space in under 45 minutes—over 97% of the theoretical maximum speed of gigabit Ethernet—without requiring specialized hardware [11] or kernel modules [8, 28]. ZMap's modular architecture can support many types of single-packet probes, including TCP SYN scans, ICMP echo request scans, and application-specific UDP scans, and it can interface easily with user-provided code to perform follow-up actions on discovered hosts, such as completing a protocol handshake.

Compared to Nmap—an excellent general-purpose network mapping tool, which was utilized in recent Internet-wide survey research [10, 14]—ZMap achieves much higher performance for Internet-scale scans. Experimentally, we find that ZMap is capable of scanning the IPv4 public address space over 1300 times faster than the most aggressive Nmap default settings, with equivalent accuracy. These performance gains are due to architectural choices that are specifically optimized for this application:

Optimized probing While Nmap adapts its transmission rate to avoid saturating the source or target networks, we assume that the source network is well provisioned (unable to be saturated by the source host), and that the targets are randomly ordered and widely dispersed (so no distant network or path is likely to be saturated by the scan). Consequently, we attempt to send probes as quickly as the source's NIC can support, skipping the TCP/IP stack and generating Ethernet frames directly. We show that ZMap can send probes at gigabit line speed from commodity hardware and entirely in user space.

No per-connection state While Nmap maintains state for each connection to track which hosts have been scanned and to handle timeouts and retransmissions, ZMap forgoes any per-connection state. Since it is intended to target random samples of the address space, ZMap can avoid storing the addresses it has already scanned or needs to scan and instead selects addresses according to a random permutation generated by a cyclic

multiplicative group. Rather than tracking connection timeouts, ZMap accepts response packets with the correct state fields for the duration of the scan, allowing it to extract as much data as possible from the responses it receives. To distinguish valid probe responses from background traffic, ZMap overloads unused values in each sent packet, in a manner similar to SYN cookies [4].

No retransmission While Nmap detects connection timeouts and adaptively retransmits probes that are lost due to packet loss, ZMap (to avoid keeping state) always sends a fixed number of probes per target and defaults to sending only one. In our experimental setup, we estimate that ZMap achieves 98% network coverage using only a single probe per host, even at its maximum scanning speed. We believe this small amount of loss will be insignificant for typical research applications.

We further describe ZMap's architecture and implementation in Section 2, and we experimentally characterize its performance in Section 3. In Section 4, we investigate the implications of the widespread availability of fast, low-cost Internet-wide scanning for both defenders and attackers, and we demonstrate ZMap's performance and flexibility in a variety of security settings, including:

Measuring protocol adoption, such as the transition from HTTP to HTTPS. We explore HTTPS adoption based on frequent Internet-wide scans over a year.

Visibility into distributed systems, such as the certificate authority (CA) ecosystem. We collect and analyze TLS certificates and identify misissued CA certs.

High-speed vulnerability scanning, which could allow attackers to widely exploit vulnerabilities within hours of their discovery. We build a UPnP scanner using ZMap through which we find 3.4 million UPnP devices with known vulnerabilities [25].

Uncovering unadvertised services, such as hidden Tor bridges. We show that ZMap can locate 86% of hidden Tor bridges via comprehensive enumeration.

High-speed scanning can be a powerful tool in the hands of security researchers, but users must be careful not to cause harm by inadvertently overloading networks or causing unnecessary work for network administrators. In Section 5, we discuss our experiences performing numerous large-scale scans over the past year, we report on the complaints and other reactions we have received, and we suggest several guidelines and best practices for good Internet citizenship while scanning.

Internet-wide scanning has already shown great potential as a research methodology [10, 12, 14, 25], and we hope ZMap will facilitate a variety of new applications by drastically reducing the costs of comprehensive network surveys and allowing scans to be performed with very fine time granularity. To facilitate this, we are releasing ZMap as an open source project that is documented and packaged for real world use. It is available at https://zmap.io/.

2 ZMap: The Scanner

ZMap uses a modular design to support many types of probes and integration with a variety of research applications, as illustrated in Figure 1. The *scanner core* handles command line and configuration file parsing, address generation and exclusion, progress and performance monitoring, and reading and writing network packets. Extensible *probe modules* can be customized for different kinds of probes, and are responsible for generating probe packets and interpreting whether incoming packets are valid responses. Modular *output handlers* allow scan results to be piped to another process, added directly to a database, or passed on to user code for further action, such as completing a protocol handshake.

We introduced the philosophy behind ZMap's design in Section 1. At a high level, one of ZMap's most important architectural features is that sending and receiving packets take place in separate threads that act independently and continuously throughout the scan. A number of design choices were made to ensure that these processes share as little state as possible.

We implemented ZMap in approximately 8,900 SLOC of C. It was written and tested on GNU/Linux.

2.1 Addressing Probes

If ZMap simply probed every IPv4 address in numerical order, it would risk overloading destination networks with scan traffic and produce inconsistent results in the case of a distant transient network failure. To avoid this, ZMap scans addresses according to a random permutation of the address space. To select smaller random samples of the address space, we simply scan a subset of the full permutation.

ZMap uses a simple and inexpensive method to traverse the address space, which lets it scan in a random permutation while maintaining only negligible state. We iterate over a multiplicative group of integers modulo p, choosing p to be a prime slightly larger than 2^{32}. By choosing p to be a prime, we guarantee that the group is cyclic and will reach all addresses in the IPv4 address space except 0.0.0.0 (conveniently an IANA reserved address) once per cycle. We choose to iterate over $(\mathbb{Z}/4,294,967,311\mathbb{Z})^{\times}$, the multiplicative group modulo p for the smallest prime larger than 2^{32}: $2^{32}+15$.

To select a fresh random permutation for each scan, we generate a new primitive root of the multiplicative group and choose a random starting address. Because the order of elements in a group is preserved by an isomorphism, we efficiently find random primitive roots of the multiplicative group by utilizing the isomorphism $(\mathbb{Z}_{p-1},+) \cong (\mathbb{Z}_p^*,\times)$ and mapping roots of $(\mathbb{Z}_{p-1},+)$ into the multiplicative group via the function $f(x) = n^x$ where n is a known primitive root of $(\mathbb{Z}/p\mathbb{Z})^{\times}$. In our

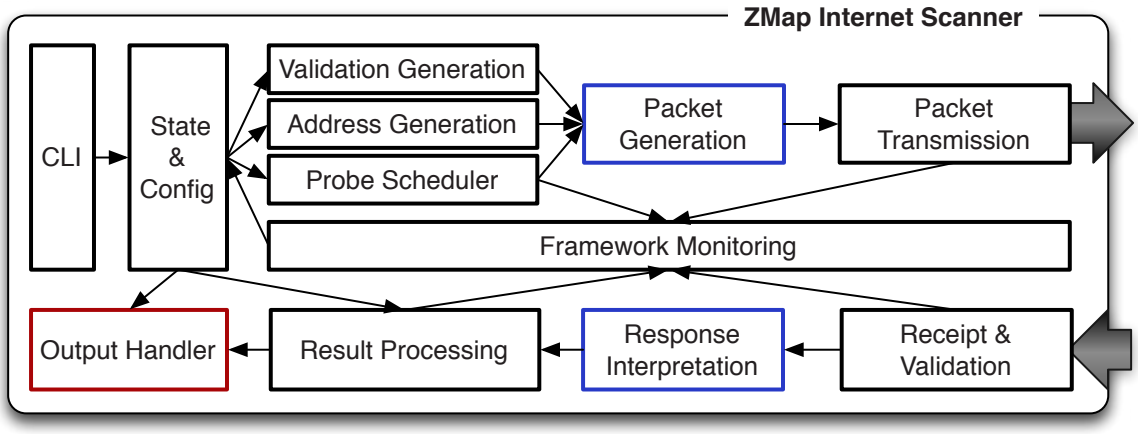

Figure 1: **ZMap Architecture** — ZMap is an open-source network scanner optimized for efficiently performing Internet-scale network surveys. Modular packet generation and response interpretation components (*blue*) support multiple kinds of probes, including TCP SYN scans and ICMP echo scans. Modular output handlers (*red*) allow users to output or act on scan results in application-specific ways. The architecture allows sending and receiving components to run asynchronously and enables a single source machine to comprehensively scan every host in the public IPv4 address space for a particular open TCP port in under 45 mins using a 1 Gbps Ethernet link.

specific case, we know that 3 is a primitive root of $(\mathbb{Z}/4,294,967,311\mathbb{Z})^{\times}$.

Because we know that the generators of $(\mathbb{Z}_{p-1},+)$ are $\{s|(s,p-1)=1\}$, we can efficiently find the generators of the additive group by precalculating and storing the factorization of $p-1$ and checking addresses against the factorization at random until we find one that is coprime with $p-1$ and then map it into $(\mathbb{Z}_p^*,\times)$. Given that there exist approximately 10^9 generators, we expect to make four tries before finding a primitive root. While this process introduces complexity at the beginning of a scan, it adds only a small amount of one-time overhead.

Once a primitive root has been generated, we can easily iterate through the address space by applying the group operation to the current address (in other words, multiplying the current address by the primitive root modulo $2^{32}+15$). We detect that a scan has completed when we reach the initially scanned IP address. This technique allows the sending thread to store the selected permutation and progress through it with only three integers: the primitive root used to generate the multiplicative group, the first scanned address, and the current address.

Excluding Addresses Since ZMap is optimized for Internet-wide scans, we represent the set of targets as the full IPv4 address space minus a set of smaller excluded address ranges. Certain address ranges need to be excluded for performance reasons (e.g., skipping IANA reserved allocations [16]) and others to honor requests from their owners to discontinue scanning. We efficiently support address exclusion through the use of radix trees, a trie specifically designed to handle ranges and frequently used by routing tables [32, 34]. Excluded ranges can be specified through a configuration file.

2.2 Packet Transmission and Receipt

ZMap is optimized to send probes as quickly as the source's CPU and NIC can support. The packet generation component operates asynchronously across multiple threads, each of which maintains a tight loop that sends Ethernet-layer packets via a raw socket.

We send packets at the Ethernet layer in order to cache packet values and reduce unnecessary kernel overhead. For example, the Ethernet header, minus the packet checksum, will never change during a scan. By generating and caching the Ethernet layer packet, we prevent the Linux kernel from performing a routing lookup, an arpcache lookup, and netfilter checks for every packet. An additional benefit of utilizing a raw socket for TCP SYN scans is that, because no TCP session is established in the kernel, upon receipt of a TCP SYN-ACK packet, the kernel will automatically respond with a TCP RST packet, closing the connection. ZMap can optionally use multiple source addresses and distribute outgoing probes among them in a round-robin fashion.

We implement the receiving component of ZMap using libpcap [17], a library for capturing network traffic and filtering the received packets. Although libpcap is a potential performance bottleneck, incoming response traffic is a small fraction of outgoing probe traffic, since the overwhelming majority of hosts are unresponsive to typical probes, and we find that libpcap is easily capable of handling response traffic in our tests (see Section 3).

Upon receipt of a packet, we check the source and destination port, discard packets clearly not initiated by the scan, and pass the remaining packets to the active probe module for interpretation.

While the sending and receiving components of ZMap operate independently, we ensure that the receiver is initialized prior to sending probes and that the receiver continues to run for a period of time (by default, 8 seconds) after the sender has completed in order to process any delayed responses.

2.3 Probe Modules

ZMap probe modules are responsible for filling in the body of probe packets and for validating whether incoming packets are responsive to the probes. Making these tasks modular allows ZMap to support a variety of probing methods and protocols and simplifies extensibility. Out of the box, ZMap provides probe modules to support TCP port scanning and ICMP echo scanning.

At initialization, the scanner core provides an empty buffer for the packet and the probe module fills in any static content that will be the same for all targets. Then, for each host to be scanned, the probe module updates this buffer with host-specific values. The probe module also receives incoming packets, after high-level validation by the scanner core, and determines whether they are positive or negative responses to scan probes. Users can add new scan types by implementing a small number of callback functions within the probe module framework.

For example, to facilitate TCP port scanning, ZMap implements a probing technique known as *SYN scanning* or *half-open scanning*. We chose to implement this specific technique instead of performing a full TCP handshake based on the reduced number of exchanged packets. In the dominant case where a host is unreachable or does not respond, only a single packet is used (a SYN from the scanner); in the case of a closed port, two packets are exchanged (a SYN answered with a RST); and in the uncommon case where the port is open, three packets are exchanged (a SYN, a SYN-ACK reply, and a RST from the scanner).

Checking Response Integrity ZMap's receiving components need to determine whether received packets are valid responses to probes originating from the scanner or are part of other background traffic. Probe modules perform this validation by encoding host- and scan-invocation–specific data into mutable fields of each probe packet, utilizing fields that will have recognizable effects on fields of the corresponding response packets in a manner similar to SYN cookies [4].

For each scanned host, ZMap computes a MAC of the destination address keyed by a scan-specific secret. This MAC value is then spread across any available fields by the active probe module. We chose to use the UMAC function for these operations, based on its performance guarantees [5]. In our TCP port scan module, we utilize the source port and initial sequence number; for ICMP, we use the ICMP identifier and sequence number. These fields are checked on packet receipt by the probe module, and ZMap discards any packets for which validation fails.

These inexpensive checks prevents the incorrect reporting of spurious response packets due to background traffic as well as responses triggered by previous scans. This design ultimately allows the receiver to validate responses while sharing only the scan secret and the initial configuration with the sending components.

2.4 Output Modules

ZMap provides a modular output interface that allows users to output scan results or act on them in application-specific ways. Output module callbacks are triggered by specific events: scan initialization, probe packet sent, response received, regular progress updates, and scan termination. ZMap's built-in output modules cover basic use, including simple text output (a file stream containing a list of unique IP addresses that have the specified port open), extended text output (a file stream containing a list of all packet responses and timing data), and an interface for queuing scan results in a Redis in-memory database [29].

Output modules can also be implemented to trigger network events in response to positive scan results, such as completing an application-level handshake. For TCP SYN scans, the simplest way to accomplish this is to create a fresh TCP connection with the responding address; this can be performed asynchronously with the scan and requires no special kernel support.

forge_socket Some ZMap users may wish to complete the TCP handshake begun during a TCP SYN scan and exchange data with the remote host without the extra overhead of establishing a new connection. While the initial SYN/SYN-ACK exchange has established a connection from the destination's perspective, ZMap bypasses the local system's TCP stack and as such the kernel does not recognize the connection.

In order to allow the scanning host to communicate over ZMap-initiated TCP sessions, we implemented forge_socket, a kernel module that allows user processes to pass in session parameters (e.g. initial sequence number) using setsockopt. This allows application-level handshakes to be performed using the initial ZMap handshake and does not require the unnecessary transmission of a RST, SYN, or SYN-ACK packet that would be required to close the existing connection and initiate a new kernel-recognized session. We are releasing forge_socket along with ZMap.

3 Validation and Measurement

We performed a series of experiments to characterize the performance of ZMap. Under our test setup, we find that a complete scan of the public IPv4 address space takes approximately 44 minutes on an entry-level server with a gigabit Ethernet connection. We estimate that a single-packet scan can detect approximately 98% of instantaneously listening hosts, and we measure a 1300 x performance improvement over Nmap for Internet-wide scanning, with equivalent coverage.

We performed the following measurements on an HP ProLiant DL120 G7 with a Xeon E3-1230 3.2 GHz processor and 4 GB of memory running a stock install of Ubuntu 12.04.1 LTS and the 3.2.0-32-generic Linux kernel. Experiments were conducted using the onboard NIC, which is based on the Intel 82574L chipset and uses the stock e1000e network driver, or a quad-port Intel Ethernet adapter based on the newer Intel 82580 chipset and using the stock igb network driver. For experiments involving complete TCP handshakes, we disabled kernel modules used by iptables and conntrack. Experiments comparing ZMap with Nmap were conducted with Nmap 5.21.

These measurements were conducted using the normal building network at the University of Michigan Computer Science & Engineering division. We used a gigabit Ethernet uplink (a standard office network connection in our facility); we did not arrange for any special network configuration beyond static IP addresses. The access layer of the building runs at 10 gbps, and the building uplink to the rest of the campus is an aggregated 2×10 gigabit port channel. We note that ZMap's performance on other source networks may be worse than reported here due to local congestion.

3.1 Scan Rate: How Fast is Too Fast?

In order to determine whether our scanner and our upstream network can handle scanning at gigabit line speed, we examine whether the *scan rate*, the rate at which ZMap sends probe packets, has any effect on the *hit rate*, the fraction of probed hosts that respond positively (in this case, with a SYN-ACK). If libpcap, the Linux kernel, our institutional network, or our upstream provider are unable to adequately handle the traffic generated by the scanner at full speed, we would expect packets to be dropped and the hit rate to be lower than at slower scan rates.

We experimented by sending TCP SYN packets to random 1% samples of the IPv4 address space on port 443 at varying scan rates. We conducted 10 trials at each of 16 scan rates ranging from 1,000 to 1.4 M packets per second. The results are shown in Figure 2.

We find no statistically significant correlation between scan rate and hit rate. This shows that our ZMap setup is capable of handling scanning at 1.4 M packets per

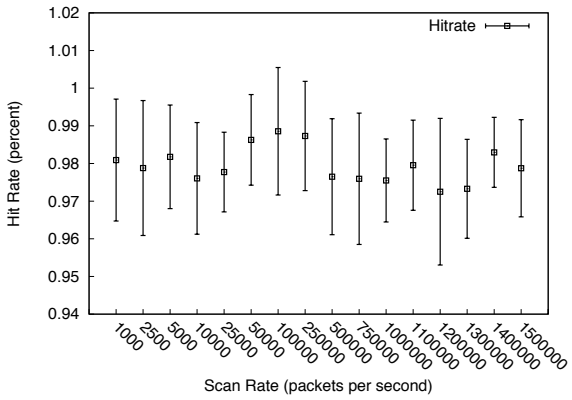

Figure 2: **Hit rate vs. Scan rate** — We find no correlation between hit rate (positive responses/hosts probed) and scan rate (probes sent/second). Shown are means and standard deviations over ten trials. This indicates that slower scanning does not reveal additional hosts.

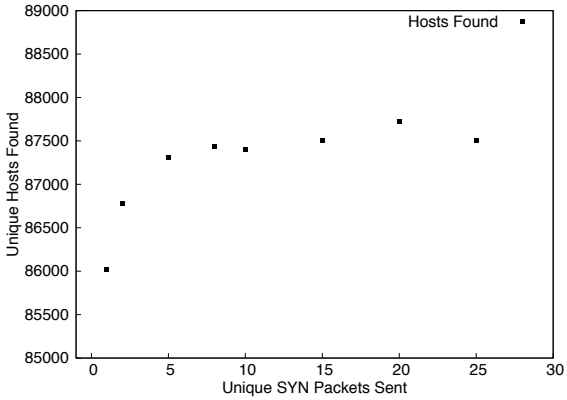

Figure 3: **Coverage for Multiple Probes** — Discovered hosts plateau after ZMap sends about 8 SYNs to each. If this plateau represents the true number of listening hosts, sending just 1 SYN will achieve about 98% coverage.

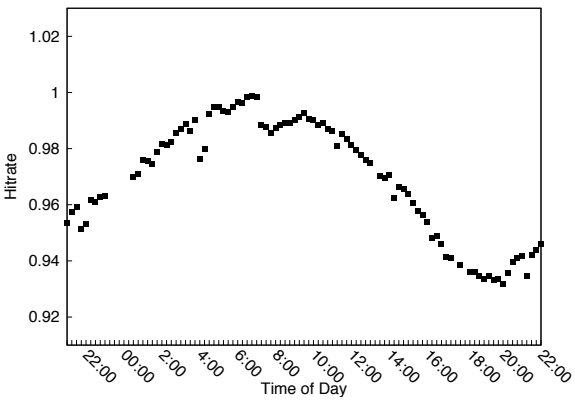

Figure 4: **Diurnal Effect on Hosts Found** — We observed a $\pm 3.1\%$ variation in ZMap's hit rate depending on the time of day the scan was performed. (Times EST.)

second and that scanning at lower rates provides no benefit in terms of identifying additional hosts. From an architectural perspective, this validates that our receiving infrastructure based on libpcap is capable of processing responses generated by the scanner at full speed and that kernel modules such as PF_RING [8] are not necessary for gigabit-speed network scanning.

3.2 Coverage: Is One SYN Enough?

While scanning at higher rates does not appear to result in a lower hit rate, this does not tell us what *coverage* we achieve with a single scan—what fraction of target hosts does ZMap actually find using its default single-packet probing strategy?

Given the absence of ground truth for the number of hosts on the Internet with a specific port open, we cannot measure coverage directly. This is further complicated by the ever changing state of the Internet; it is inherently difficult to detect whether a host was not included in a scan because it was not available at the time or because packets were dropped between it and the scanner. Yet, this question is essential to understanding whether performing fast, single-packet scans is an accurate methodology for Internet-wide surveys.

To characterize ZMap's coverage, we estimate the number of hosts that are actually listening by sending multiple, distinct SYN packets to a large address sample and analyzing the distribution of the number of positive responses received compared to the number of SYNs we send. We expect to eventually see a plateau in the number of hosts that respond regardless of the number of additional SYNs we send. If this plateau exists, we can treat it as an estimate of the real number of listening hosts, and we can use it as a baseline against which to compare scans with fewer SYN packets.

We performed this experiment by sending 1, 2, 5, 8, 10, 15, 20, and 25 SYN packets to random 1% samples of the IPv4 address space on port 443 and recording the number of distinct hosts that sent SYN-ACK responses in each scan. The results indicate a clear plateau in the number of responsive hosts after sending 8 SYN packets, as shown in Figure 3.

Based on the level of this plateau, we estimate that our setup reaches approximately 97.9% of live hosts using a single packet, 98.8% of hosts using two packets, and 99.4% of hosts using three packets. The single packet round-trip loss rate of about 2% is in agreement with previous studies on random packet drop on the Internet [12].

These results suggest that single-probe scans are sufficiently comprehensive for typical research applications. Investigators who require higher coverage can configure ZMap to send multiple probes per host, at the cost of proportionally longer running scans.

3.3 Variation by Time of Day

In previous work, Internet-wide scans took days to months to execute, so there was little concern over finding the optimal time of day to perform a scan. However, since ZMap scans can take less than an hour to complete, the question as to the "right time" to perform a scan arises. Are there certain hours of the day or days of the week that are more effective for scanning than others?

In order to measure any diurnal effects on scanning, we performed continuous scans of TCP port 443 targeting a random 1% sample of the Internet over a 24-hour period. Figure 4 shows the number of hosts found in each scan.

We observed a $\pm 3.1\%$ variation in hit rate dependent on the time of day scans took place. The highest response rates were at approximately 7:00 AM EST and the lowest response rates were at around 7:45 PM EST.

These effects may be due to variation in overall network congestion and packet drop rates or due to a diurnal pattern in the aggregate availability of end hosts that are only intermittently connected to the network. In less formal testing, we did not notice any obvious variation by day of the week or day of the month.

3.4 Comparison with Nmap

We performed several experiments to compare ZMap to Nmap in Internet-wide scanning applications, focusing on coverage and elapsed time to complete a scan. Nmap and ZMap are optimized for very different purposes. Nmap is a highly flexible, multipurpose tool that is frequently used for probing a large number of open ports on a smaller number of hosts, whereas ZMap is optimized to probe a single port across very large numbers of targets. We chose to compare the two because recent security studies used Nmap for Internet-wide surveys [10, 14], and because, like ZMap, Nmap operates from within user space on Linux [23].

We tested a variety of Nmap settings to find reasonable configurations to compare. All performed a TCP SYN scan on port 443 (-Ss -p 443). Nmap provides several defaults known as *timing templates*, but even with the most aggressive of these (labeled "insane"), an Internet-wide scan would take over a year to complete. To make Nmap scan faster in our test configurations, we started with the "insane" template (-T5), disabled host discovery and DNS resolutions (-Pn -n), and set a high minimum packet rate (--min-rate 10000). The "insane" template retries each probe once after a timeout; we additionally tested a second Nmap configuration with retries disabled (--max-retries 0).

We used ZMap to select a random sample of 1 million IP addresses and scanned them for hosts listening on port 443 with Nmap in the two configurations described above and with ZMap in its default configuration and in a

Scan Type	Coverage (normalized)	Duration (mm:ss)	Est. Time for Internet-wide Scan
Nmap, max 2 probes (default)	0.978	45:03	116.3 days
Nmap, 1 probe	0.814	24:12	62.5 days
ZMap, 2 probes	1.000	00:11	2:12:35
ZMap, 1 probe (default)	0.987	00:10	1:09:45

Table 1: **ZMap vs. Nmap Comparison** — We scanned 1 million hosts on TCP port 443 using ZMap and Nmap and averaged over 10 trials. Despite running hundreds of times faster, ZMap finds more listening hosts than Nmap, due to Nmap's low host timeout. Times for ZMap include a fixed 8 second delay to wait for responses after the final probe.

second configuration that sends two SYN probes to each host (-P 2). We repeated this process for 10 trials over a 12 hour period and report the averages in Table 1.

The results show that ZMap scanned much faster than Nmap and found more listening hosts than either Nmap configuration. The reported durations for ZMap include time sent sending probes as well as a fixed 8-second delay after the sending process completes, during which ZMap waits for late responses. Extrapolating to the time required for an Internet-wide scan, the fastest tested ZMap configuration would complete approximately 1300 times faster than the fastest Nmap configuration.[1]

Coverage and Timeouts To investigate why ZMap achieved higher coverage than Nmap, we probed a random sample of 4.3 million addresses on TCP port 80 and measured the latency between sending a SYN and receiving a SYN-ACK from responsive hosts. Figure 5 shows the CDF of the results. The maximum round-trip time was 450 seconds, and a small number of hosts took more than 63 seconds to respond, the time it takes for a TCP connection attempt to timeout on Linux. 99% of hosts that responded within 500 seconds did so within about 1 second, and 99.9% responded within 8.16 seconds.

As ZMap's receiving code is stateless with respect to the sending code, a valid SYN-ACK that comes back any time before the scan completes will be recorded as a listening host. To assure a high level of coverage, the default ZMap settings incorporate an empirically derived 8-second delay after the last probe is sent before the receiving process terminates.

In contrast, Nmap maintains timeouts for each probe. In the Nmap "insane" timing template we tested, the timeout is initially 250 ms, by which time fewer than 85% of responsive hosts in our test had responded. Over the course of a scan, Nmap's timeout can increase to 300 ms, by which time 93.2% had responded. Thus, we would expect a single-probe Nmap scan with these timing values to see 85–93% of the hosts that ZMap finds, which is roughly in line with the observed value of 82.5%.

With Nmap's "insane" defaults, it will attempt to send a second probe after a timeout. A response to either the first or second SYN will be considered valid until the second times out, so this effectively raises the overall timeout to 500–600 ms, by which time we received 98.2–98.5% of responses. Additional responses will likely be generated by the second SYN. We observed that the 2-probe Nmap scan found 99.1% of the number of hosts that a 1-probe ZMap scan found.

[1]The extrapolated 1-packet Internet-wide scan time for ZMap is longer than the 44 minutes we report elsewhere for complete scans, because this test used a slower NIC based on the Intel 82574L chipset.

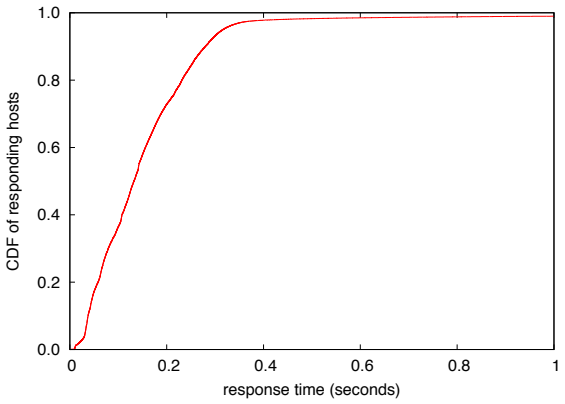

Figure 5: **SYN to SYN-ACK time** — In an experiment that probed 4.3 million hosts, 99% of SYN-ACKs arrived within about 1 second and 99.9% within 8.16 seconds.

3.5 Comparison with Previous Studies

Several groups have previously performed Internet-wide surveys using various methodologies. Here we compare ZMap to two recent studies that focused on HTTPS certificates. Most recently, Heninger et al. performed a distributed scan of port 443 in 2011 as part of a global analysis on cryptographic key generation [14]. Their scan used Nmap on 25 Amazon EC2 instances and required 25 hours to complete, with a reported average of 40,566 hosts scanned per second. A 2010 scan by the EFF SSL Observatory project used Nmap on 3 hosts and took 3 months to complete [10].

Scan	Date	Port 443 Open	TLS Servers	All Certs	Trusted Certs
EFF SSL Observatory [10]	2010/12	16.2 M	7.7 M	4.0 M	1.46 M
Mining Ps and Qs [14]	2011/10	28.9 M	12.8 M	5.8 M	1.96 M
ZMap + certificate fetcher	2012/06	31.8 M	19.0 M	7.8 M	2.95 M
ZMap + certificate fetcher	2013/05	34.5 M	22.8 M	8.6 M	3.27 M

Table 2: **Comparison with Prior Internet-wide HTTPS Surveys** — Due to growth in HTTPS deployment, ZMap finds almost three times as many TLS servers as the SSL Observatory did in late 2010, yet this process takes only 10 hours to complete from a single machine using a ZMap-based workflow, versus three months on three machines.

To compare ZMap's performance for this task, we used it to conduct comprehensive scans of port 443 and used a custom certificate fetcher based on libevent [24] and OpenSSL [37] to retrieve TLS certificates from each responsive host. With this methodology, we were able to discover hosts, perform TLS handshakes, and collect and parse the resulting certificates in under 10 hours from a single machine.

As shown in Table 2, we find significantly more TLS servers than previous work—78% more than Heninger et al. and 196% more than the SSL Observatory—likely due to increased HTTPS deployment since those studies were conducted. Linear regression shows an average growth in HTTPS deployment of about 540,000 hosts per month over the 29 month period between the SSL Observatory scan and our most recent dataset. Despite this growth, ZMap is able to collect comprehensive TLS certificate data in a fraction of the time and cost needed in earlier work. The SSL Observatory took roughly 650 times as much machine time to acquire the same kind of data, and Heninger et al. took about 65 times as much.

4 Applications and Security Implications

The ability to scan the IPv4 address space in under an hour opens an array of new research possibilities, including the ability to gain visibility into previously opaque distributed systems, understand protocol adoption at a new resolution, and uncover security phenomenon only accessible with a global perspective [14]. However, high-speed scanning also has potentially malicious applications, such as finding and attacking vulnerable hosts en masse. Furthermore, many developers have the preconceived notion that the Internet is far too large to be fully enumerated, so the reality of high speed scanning may disrupt existing security models, such as by leading to the discovery of services previously thought to be well hidden. In this section, we use ZMap to explore several of these applications.

4.1 Visibility into Distributed Systems

High-speed network scanning provides researchers with the possibility for a new real-time perspective into previously opaque distributed systems on the Internet. For instance, e-commerce and secure web transactions inherently depend on browser trusted TLS certificates. However, there is currently little oversight over browser trusted certificate authorities (CAs) or issued certificates. Most CAs do not publish lists of the certificates they have signed, and, due to delegation of authority to intermediate CAs, it is unknown what set of entities have the technical ability to sign browser-trusted certificates at any given time.

Organization	Certificates	
GoDaddy.com, Inc.	913,416	(31.0%)
GeoTrust Inc.	586,376	(19.9%)
Comodo CA Limited	374,769	(12.7%)
VeriSign, Inc.	317,934	(10.8%)
Thawte, Inc.	228,779	(7.8%)
DigiCert Inc	145,232	(4.9%)
GlobalSign	117,685	(4.0%)
Starfield Technologies	94,794	(3.2%)
StartCom Ltd.	88,729	(3.0%)
Entrust, Inc.	76,929	(2.6%)

Table 3: **Top 10 Certificate Authorities** — We used ZMap to perform regular comprehensive scans of HTTPS hosts in order gain visibility into the CA ecosystem. Ten organizations control 86% of browser trusted certificates.

To explore this potential, we used ZMap and our custom certificate fetcher to conduct regular scans over the past year and perform analysis on new high-profile certificates and CA certificates. Between April 2012 and June 2013, we performed 1.81 billion TLS handshakes, ultimately collecting 33.6 million unique X.509 certificates of which 6.2 million were browser trusted. We found and processed an average of 220,000 new certificates, 15,300 new browser trusted certificates, and 1.2 new CA certificates per scan. In our most recent scan, we identified 1,832 browser trusted signing certificates from 683 organizations and 57 countries. We observed 3,744 distinct browser-trusted signing certificates in total. Table 3 shows the most prolific CAs by leaf certificates issued.

Wide-scale visibility into CA behavior can help to identify security problems [10, 18]. We found two cases of misissued CA certificates. In the first case, we found a CA certificate that was accidentally issued to a Turkish transit provider. This certificate, `C=TR, ST=ANKARA, L=ANKARA, O=EGO, OU=EGO BILGI ISLEM, CN=*.EGO.GOV.TR`, was later found by Google after being used to sign a Google wildcard certificate and has since been revoked and blacklisted in common web browsers [20].

In the second case, we found approximately 1,300 CA certificates that were misissued by the Korean Government to government sponsored organizations such as schools and libraries. While these certificates had been issued with rights to sign additional certificates, a length constraint on the grandparent CA certificate prevented these organizations from signing new certificates. We do not include these Korean certificates in the CA totals above because they are unable to sign valid browser-trusted certificates.

4.2 Tracking Protocol Adoption

Researchers have previously attempted to understand the adoption of new protocols, address depletion, common misconfigurations, and vulnerabilities through active scanning [2, 10, 12, 14, 15, 27]. In many of these cases, these analyses have been performed on random samples of the IPv4 address space due to the difficulty of performing comprehensive scans [15, 27]. In cases where full scans were performed, they were completed over an extended period of time or through massive parallelization on cloud providers [10, 14]. ZMap lowers the barriers to entry and allows researchers to perform studies like these in a comprehensive and timely manner, ultimately enabling much higher resolution measurements than previously possible.

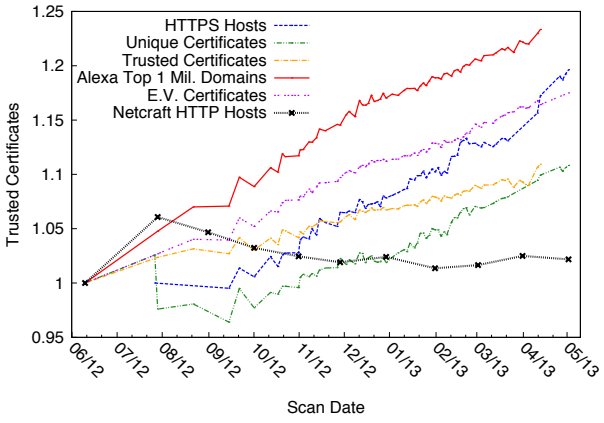

Figure 6: **HTTPS Adoption** — Data we collected using ZMap show trends in HTTPS deployment over one year. We observed 19.6% growth in hosts serving HTTPS.

Port	Service	Hit Rate (%)
80	HTTP	1.77
7547	CWMP	1.12
443	HTTPS	0.93
21	FTP	0.77
23	Telnet	0.71
22	SSH	0.57
25	SMTP	0.43
3479	2-Wire RPC	0.42
8080	HTTP-alt/proxy	0.38
53	DNS	0.38

Table 4: **Top 10 TCP ports** — We scanned 2.15 million hosts on TCP ports 0–9175 and observed what fraction were listening on each port. We saw a surprising number of open ports associated with embedded devices, such as ports 7547 (CWMP) and 3479 (2-Wire RPC).

To illustrate this application, we tracked the adoption of HTTPS using 158 Internet-wide scans over the past year. Notably, we find a 23% increase in HTTPS use among Alexa Top 1 Million websites and a 10.9% increase in the number of browser-trusted certificates. During this period, the Netcraft Web Survey [26] finds only a 2.2% increase in the number of HTTP sites, but we observe an 8.5% increase in sites using HTTPS. We plot these trends in Figure 6.

We can also gain instantaneous visibility into the deployment of multiple protocols by performing many ZMaps scans of different ports. We scanned 0.05% samples of the IPv4 address space on each TCP port below 9175 to determine the percentage of hosts that were listening on each port. This experiment requires the same number of packets as over 5 Internet-wide scans of a single port, yet we completed it in under a day using ZMap. Table 4 shows the top 10 open ports we observed.

4.3 Enumerating Vulnerable Hosts

With the ability to perform rapid Internet-wide scans comes the potential to quickly enumerate hosts that suffer from specific vulnerabilities [2]. While this can be a powerful defensive tool for researchers—for instance, to measure the severity of a problem or to track the application of a patch—it also creates the possibility for an attacker with control of only a small number of machines to scan for and infect all public hosts suffering from a new vulnerability within minutes.

UPnP Vulnerabilities To explore these applications, we investigated several recently disclosed vulnerabilities in common UPnP frameworks. On January 29, 2013, HD Moore publicly disclosed several vulnerabilities in common UPnP libraries [25]. These vulnerabilities ulti-

mately impacted 1,500 vendors and 6,900 products, all of which can be exploited to perform arbitrary code execution with a single UDP packet. Moore followed responsible disclosure guidelines and worked with manufacturers to patch vulnerable libraries, and many of the libraries had already been patched at the time of disclosure. Despite these precautions, we found that at least 3.4 million devices were still vulnerable to the problem in February 2013.

To measure this, we created a custom ZMap probe module that performs a UPnP discovery handshake. We were able to develop this 150-SLOC module from scratch in approximately four hours and performed a comprehensive scan of the IPv4 address space for publicly available UPnP hosts on February 11, 2013, which completed in under two hours. This scan found 15.7 million publicly accessible UPnP devices, of which 2.56 million (16.5%) were running vulnerable versions of the Intel SDK for UPnP Devices, and 817,000 (5.2%) used vulnerable versions of MiniUPnPd.[2]

Given that these vulnerable devices can be infected with a single UDP packet [25], we note that these 3.4 million devices could have been infected in approximately the same length of time—much faster than network operators can reasonably respond or for patches to be applied to vulnerable hosts. Leveraging methodology similar to ZMap, it would only have taken a matter of hours from the time of disclosure to infect every publicly available vulnerable host.

Weak Public Keys As part of our regular scans of the HTTPS ecosystem, we tracked the mitigation of the 2008 Debian weak key vulnerability [3] and the weak and shared keys described by Heninger et al. in 2012 [14]. Figure 7 shows several trends over the past year.

In our most recent scan, we found that 44,600 unique certificates utilized factorable RSA keys and are served on 51,000 hosts, a 20% decrease from 2011 [14]. Four of these certificates were browser trusted; the last was signed in August 2012. Similarly, we found 2,743 unique certificates that contained Debian weak keys, of which 96 were browser trusted, a 34% decrease from 2011 [14]. The last browser trusted certificate containing a Debian weak key was signed in January 2012. We also observed a 67% decrease in the number of browser-trusted certificates that contained default public keys used for Citrix remote access products [14].

We created an automated process that alerts us to the discovery of new browser-trusted certificates containing factorable RSA keys, Debian weak keys, or default Citrix keys as soon as they are found, so that we can attempt to notify the certificate owners about the vulnerability.

[2]Moore reported many more UPnP hosts [25] but acknowledges that his scans occurred over a 5 month period and did not account for hosts being counted multiple times due to changing IP addresses.

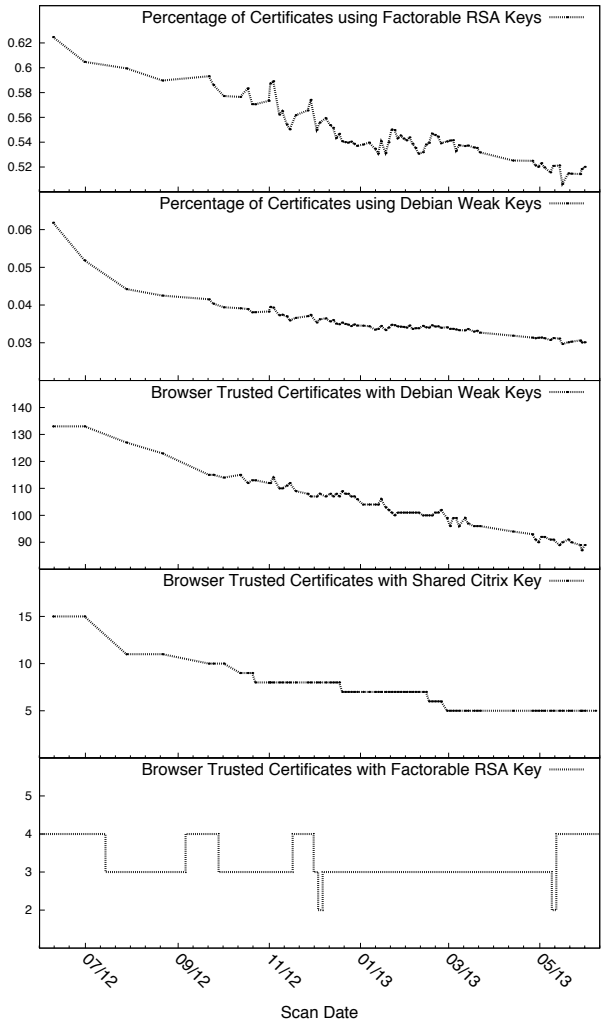

Figure 7: **Trends in HTTPS Weak Key Usage** — To explore how ZMap can be used to track the mitigation of known vulnerabilities, we monitored the use of weak HTTPS public keys from May 2012 through June 2013.

4.4 Discovering Unadvertised Services

The ability to perform comprehensive Internet scans implies the potential to uncover unadvertised services that were previously only accessible with explicit knowledge of the host name or address. For example, Tor bridges are intentionally not published in order to prevent ISPs and government censors from blocking connections to the Tor network [35]. Instead, the Tor Project provides users with the IP addresses of a small number of bridges based on their source address. While Tor developers have acknowledged that bridges can in principle be found by Internet-wide scanning [9], the set of active bridges is constantly changing, and the data would be stale by the time a long running scan was complete. However, high-speed scanning might be used to mount an effective attack.

To confirm this, we performed Internet wide-scans on ports 443 and 9001, which are common ports for Tor bridges and relays, and applied a set of heuristics to identify likely Tor nodes. For hosts with one of these ports open, we performed a TLS handshake using a specific set of cipher suites supported by Tor's "v1 handshake." When a Tor relay receives this set of cipher suites, it will respond with a two-certificate chain. The signing ("Certifiate Authority") certificate is self-signed with the relay's identity public key and uses a subject name of the form "CN=www.*X*.com", where *X* is a randomized alphanumeric string. This pattern matched 67,342 hosts on port 443, and 2,952 hosts on port 9001.

We calculated each host's identity fingerprint and checked whether the SHA1 hash appeared in the public Tor metrics list for bridge pool assignments. Hosts we found matched 1,170 unique bridge fingerprints on port 443 and 419 unique fingerprints on port 9001, with a combined total of 1,534 unique fingerprints (some were found on both ports). From the bridge pool assignment data, we see there have been 1,767–1,936 unique fingerprints allocated at any given time in the recent past, which suggests that we were able to identify 79–86% of allocated bridges at the time of the scan. The unmatched fingerprints in the Tor metrics list may correspond to bridges we missed, offline bridges, or bridges configured to use a port other than 9001 or 443.

In response to other discovery attacks against Tor bridges [38], the Tor project has started to deploy obfsproxy [36], a wrapper that disguises client–bridge connections as random data in order to make discovery by censors more difficult. Obfsproxy nodes listen on randomized ports, which serves as a defense against discovery by comprehensive scanning.

4.5 Monitoring Service Availability

Active scanning can help identify Internet outages and disruptions to service availability without an administrative perspective. Previous studies have shown that active surveying (ICMP echo request scans) can help track Internet outages, but they have either scanned small subsets of the address space based on preconceived notions of where outages would occur or have performed random sampling [9, 13, 31]. High speed scanning allows scans to be performed at a high temporal resolution through sampling or comprehensively. Similarly, scanning can help service providers identify networks and physical regions that have lost access to their service.

In order to explore ZMap's potential for tracking service availability, we performed continuous scans of the IPv4 address space during Hurricane Sandy to track its impact on the East Coast of the United States. We show a snapshot of outages caused by the hurricane in Figure 8.

4.6 Privacy and Anonymous Communication

The advent of comprehensive high-speed scanning raises potential new privacy threats, such as the possibility of tracking user devices between IP addresses. For instance, a company could track home Internet users between dynamically assigned IP addresses based on the HTTPS certificate or SSH host key presented by many home routers and cable modems. This would allow tracking companies to extend existing IP-based tracking beyond the length of DHCP leases.

In another scenario, it may be possible to track travelers. In 2006 Scholz et al. presented methods for fingerprinting SIP devices [30] and other protocols inadvertently expose unique identifiers such as cryptographic keys. Such features could be used to follow a specific mobile host across network locations. These unique fingerprints, paired with publicly available network data and commercial geolocation databases, could allow an attacker to infer relationships and travel patterns of a specific individual.

The ability to rapidly send a single packet to all IPv4 addresses could provide the basis for a system of anonymous communication. Rather than using the scanner to send probes, it could be used to broadcast a short encrypted message to every public IP address. In this scenario, it would be impossible to determine the desired destination host. If the sender is on a network that does not use ingress filtering, it could also spoof source addresses to obscure the sender's identity. This style of communication could be of particular interest to botnet operators, because it would allow infected hosts to remain dormant indefinitely while waiting for instructions, instead of periodically checking in with command and control infrastructure and potentially revealing their existence.

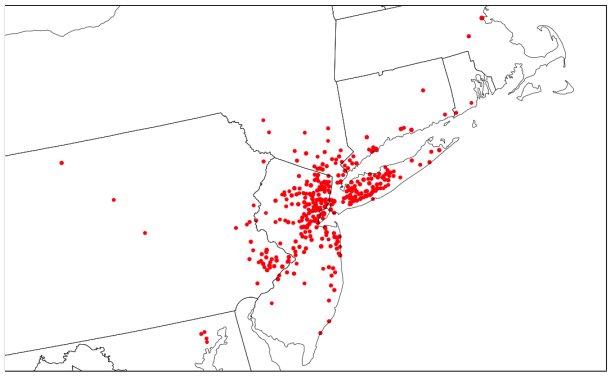

Figure 8: **Outages in the Wake of Hurricane Sandy** — We performed scans of port 443 across the entire IPv4 address space every 2 hours from October 29–31, 2013 to track the impact of Hurricane Sandy on the East Coast of the United States. Here, we show locations with more than a 30% decrease in the number of listening hosts.

5 Scanning and Good Internet Citizenship

We worked with senior colleagues and our local network administrators to consider the ethical implications of high-speed Internet-wide scanning and to develop a series of guidelines to identify and reduce any risks. Such scanning involves interacting with an enormous number of hosts and networks worldwide. It would be impossible to request permission in advance from the owners of all these systems, and there is no IP-level equivalent of the HTTP robots exclusion standard [19] to allow systems to signal that they desire not to be scanned. If we are to perform such scanning at all, the most we can do is try to minimize any potential for harm and give traffic recipients the ability to opt out of further probes.

High-speed scanning uses a large amount of bandwidth, so we need to ensure that our activities do not cause service degradation to the source or target networks. We confirmed with our local network administrators that our campus network and upstream provider had sufficient capacity for us to scan at gigabit speeds. To avoid overwhelming destination networks, we designed ZMap to scan addresses according to a random permutation. This spreads out traffic to any given destination network across the length of the scan. In a single probe TCP scan, an individual destination address receives one 40 byte SYN packet. If we scan at full gigabit speed, each /24 network block will receive a packet about every 10.6 seconds (3.8 bytes/s), each /16 network every 40 ms (1000 bytes/s), and each /8 network every 161 μs (250,000 bytes/s) for the 44 minute duration of the scan. These traffic volumes should be negligible for networks of these sizes.

Despite these precautions, there is a small but nonzero chance that any interaction with remote systems might cause operational problems. Moreover, users or network administrators who observe our scan traffic might be alarmed, in the mistaken belief that they are under attack. Many may be unable to recognize that their systems are not being uniquely targeted and that these scans are not malicious in nature, and might waste resources responding. Some owners of target systems may simply be annoyed and want our scans to cease. To minimize the risks from these scenarios, we took several steps to make it easy for traffic recipients to learn why they were receiving probes and to have their addresses excluded from scanning if so desired.

First, we configured our source addresses to present a simple website on port 80 that describes the nature and purpose of the scans The site explains that we are not targeting individual networks or attempting to obtain access to private systems, and it provides a contact email address to request exclusion from future scans. Second, we set reverse DNS records for our source addresses to "researchscan*x*.eecs.umich.edu" in order to signal that traffic from these hosts was part of an academic research study. Third, we coordinated with IT teams at our institution who might receive inquiries about our scan traffic.

1. Coordinate closely with local network admins to reduce risks and handle inquiries.
2. Verify that scans will not overwhelm the local network or upstream provider.
3. Signal the benign nature of the scans in web pages and DNS entries of the source addresses.
4. Clearly explain the purpose and scope of the scans in all communications.
5. Provide a simple means of opting out, and honor requests promptly.
6. Conduct scans no larger or more frequent than is necessary for research objectives.
7. Spread scan traffic over time or source addresses when feasible.

Table 5: **Recommended Practices** — We offer these suggestions for other researchers conducting fast Internet-wide scans as guidelines for good Internet citizenship.

For our ongoing Internet-wide HTTPS surveys (our largest-volume scanning effort), we took additional steps to further reduce the rate of false alarms from intrusion detection systems. Rather than scanning at full speed, we conducted each of these scans over a 12 hour period. We also configured ZMap to use a range of 64 source addresses and spread out probe traffic among them. We recognize that there is a difficult balance to strike here: we do not want to conceal our activities from system administrators who would want to know about them, but we also do not want to divert IT support resources that would otherwise be spent dealing with genuine attacks.

We provide a summary of the precautions we took in Table 5 as a starting point for future researchers performing Internet-wide scans. It should go without saying that scan practitioners should refrain from exploiting vulnerabilities or accessing protected resources, and should comply with any special legal requirements in their jurisdictions.

5.1 User Responses

We performed approximately 200 Internet-wide scans over the course of a year, following the practices described above. We received e-mail responses from 145 scan traffic recipients, which we classify in Table 6. In most cases, these responses were informative in nature, notifying us that we may have had infected machines, or were civil requests to be excluded from future scans. The vast majority of these requests were received at our institution's WHOIS abuse address or at the e-mail address published on the scan source IP addresses, but we also received

Small/Medium Business	41
Home User	38
Other Corporation	17
Academic Institution	22
Government/Military	15
Internet Service Provider	2
Unknown	10
Total Entities	145

Table 6: **Responses by Entity Type** — We classify the responses and complaints we received about our ongoing scans based on the type of entity that responded.

responses sent to our institution's help desk, our chief security officer, and our departmental administrator.

We responded to each inquiry with information about the purpose of our scans, and we immediately excluded the sender's network from future scans upon request. In all, we excluded networks belonging to 91 organizations or individuals, totaling 3,753,899 addresses (0.11% of the public IPv4 address space). About 49% of the blacklisted addresses resulted from requests from two Internet service providers. We received 15 actively hostile responses that threatened to retaliate against our institution legally or to conduct a denial-of-service (DOS) attack against our network. In two cases, we received retaliatory DOS traffic, which was blacklisted by our upstream provider.

6 Related Work

Many network scanning tools have been developed, the vast majority of which have been optimized to scan small network segments. The most popular and well respected is Nmap ("Network Mapper") [23], a versatile, multipurpose tool that supports a wide variety of probing techniques. Unlike Nmap, ZMap is specifically designed for Internet-wide scanning, and it achieves much higher performance in this application.

Leonard and Loguinov introduced IRLscanner, an Internet–scale scanner with the demonstrated ability to probe the advertised IPv4 address space in approximately 24 hours, ultimately scanning at 24,421 packets per second [22]. IRLscanner is able to perform scanning at this rate by utilizing a custom Windows network driver, IRLstack [33]. However, IRLscanner does not process responses, requires a custom network driver and a complete routing table for each scan, and was never released to the research community. In comparison, we developed ZMap as a self-contained network scanner that requires no custom drivers, and we are releasing it to the community under an open source license. We find that ZMap can scan at 1.37 million packets per second, 56 times faster than IRLScanner was shown to operate.

Previous work has developed methods for sending and receiving packets at fast network line speeds, including PF_RING [8], PacketShader [11], and netmap [28], all of which replace parts of the Linux kernel network stack. However, as discussed in Section 3.1, we find that the Linux kernel is capable of sending probe packets at gigabit Ethernet line speed without modification. In addition, libpcap is capable of processing responses without dropping packets as only a small number of hosts respond to probes. The bottlenecks in current tools are in the scan methodology rather than the network stack.

Many projects have performed Internet-scale network surveys (e.g., [10, 12, 14, 15, 25, 27]), but this has typically required heroic effort on the part of the researchers. In 2008, Heidemann et al. presented an Internet census in which they attempted to determine IPv4 address utilization by sending ICMP packets to allocated IP addresses; their scan of the IPv4 address space took approximately three months to complete and claimed to be the first Internet-wide survey since 1982 [12]. Two other recent works were motivated by studying the security of HTTPS. In 2010, the Electronic Frontier Foundation (EFF) performed a scan of the public IPv4 address space using Nmap [23] to find hosts with port 443 (HTTPS) open as part of their SSL Observatory Project [10]; their scans were performed on three Linux servers and took approximately three months to complete. Heninger et al. performed a scan of the IPv4 address space on port 443 (HTTPS) in 2011 and on port 22 (SSH) in 2012 as part of a study on weak cryptographic keys [14]. The researchers were able to perform a complete scan in 25 hours by concurrently performing scans from 25 Amazon EC2 instances at a cost of around $300. We show that ZMap could be used to collect the same data much faster and at far lower cost.

Most recently, an anonymous group performed an illegal "Internet Census" in 2012, using the self-named Carna Botnet. This botnet used default passwords to log into thousands of telnet devices. After logging in, the botnet scanned for additional vulnerable telnet devices and performed several scans over the IPv4 space, comprising over 600 TCP ports and 100 UDP ports over a 3-month period [1]. With this distributed architecture, the authors claim to have been able to perform a single-port scan survey over the IPv4 space in about an hour. ZMap can achieve similar performance without making use of stolen resources.

7 Future Work

While we have demonstrated that efficiently scanning the IPv4 address space at gigabit line speeds is possible, there remain several open questions related to performing network surveys over other protocols and at higher speeds.

Scanning IPv6 While ZMap is capable of rapidly scanning the IPv4 address space, brute-force scanning methods will not suffice in the IPv6 address space, which is far too large to be fully enumerated [7]. This places current researchers in a window of opportunity to take advantage of fast Internet-wide scanning methodologies before IPv6-only services become common place. New methodologies will need to be developed specifically for performing surveys of the IPv6 address space.

10gigE Surveys ZMap is currently limited by the speed of widely available gigabit networks, and we have not tested how well its architecture will scale as 10gigE and faster networks become available. There is motivation to perform the fastest scans possible as they will provide the truest sense of a snapshot of the Internet at a given point in time. However, these faster rates also open questions of overloading destination networks and hosts. The dynamics of performing scans at 10gigE have not yet been explored.

Server Name Indication Server Name Indication (SNI) is a TLS protocol extension that allows a server to present multiple certificates on the same IP address [6]. SNI has not yet been widely deployed, primarily because Internet Explorer does not support it on Windows XP hosts [21]. However, its inevitable growth will make scanning HTTPS sites more complicated, since simply enumerating the address space will miss certificates that are only presented with the correct SNI hostname.

Scanning Exclusion Standards If Internet-wide scanning becomes more widespread, it will become increasingly burdensome for system operators who do not want to receive such probe traffic to manually opt out from all benign sources. Further work is needed to standardize an exclusion signaling mechanism, akin to HTTP's robots.txt [19]. For example, a host could use a combination of protocol flags to send a "do-not-scan" signal, perhaps by responding to unwanted SYNs with the SYN and RST flags, or a specific TCP option set.

8 Conclusion

We are living in a unique period in the history of the Internet: typical office networks are becoming fast enough to exhaustively scan the IPv4 address space, yet IPv6 (with its much larger address space) has not yet been widely deployed. To help researchers make the most of this window of opportunity, we developed ZMap, a network scanner specifically architected for performing fast, comprehensive Internet-wide surveys.

We experimentally showed that ZMap is capable of scanning the public IPv4 address space on a single port in under 45 minutes, at 97% of the theoretical maximum speed for gigabit Ethernet and with an estimated 98% coverage of publicly available hosts. We explored the security applications of high speed scanning, including the ability to track protocol adoption at Internet scale and to gain timely insight into opaque distributed systems such as the certificate authority ecosystem. We further showed that high-speed scanning also provides new attack vectors that we must consider when defending systems, including the ability to uncover hidden services, the potential to track users between IP addresses, and the risk of infection of vulnerable hosts en masse within minutes of a vulnerability's discovery.

We hope ZMap will elevate Internet-wide scanning from an expensive and time-consuming endeavor to a routine methodology for future security research. As Internet-wide scanning is conducted more routinely, practitioners must ensure that they act as good Internet citizens by minimizing risks to networks and hosts and being responsive to inquiries from traffic recipients. We offer the recommendations we developed while performing our own scans as a starting point for further conversations about good scanning practice.

Acknowledgments

The authors thank the exceptional sysadmins at the University of Michigan for their help and support throughout this project. This research would not have been possible without Kevin Cheek, Laura Fink, Paul Howell, Don Winsor, and others from ITS, CAEN, and DCO. We thank Michael Bailey for advice on many aspects of the work and Oguz Durumeric for his discussion of generating permutations of the IPv4 address space. We also thank Brad Campbell, Peter Eckersley, James Kasten, Pat Pannuto, Amir Rahmati, Michael Rushanan, and Seth Schoen. This work was supported in part by NSF grant CNS-1255153 and by an NSF Graduate Research Fellowship.

References

[1] Anonymous. Internet census 2012. http://census2012.sourceforge.net/paper.html, March 2013.

[2] G. Bartlett, J. Heidemann, and C. Papadopoulos. Understanding passive and active service discovery. In *7th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 57–70, 2007.

[3] L. Bello. DSA-1571-1 OpenSSL—Predictable random number generator, 2008. Debian Security Advisory. http://www.debian.org/security/2008/dsa-1571.

[4] D. J. Bernstein. SYN cookies. http://cr.yp.to/syncookies.html, 1996.

[5] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and secure message authentication. In *Advances in Cryptology—CRYPTO '99*, 1999.

[6] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 3546 (Proposed Standard), June 2003.

[7] T. Chown. IPv6 Implications for Network Scanning. RFC 5157 (Informational), March 2008.

[8] L. Deri. Improving passive packet capture: Beyond device polling. In *4th International System Administration and Network Engineering Conference (SANE)*, 2004.

[9] R. Dingledine. Research problems: Ten ways to discover Tor bridges. http://blog.torproject.org/blog/research-problems-ten-ways-discover-tor-bridges, October 2011.

[10] P. Eckersley and J. Burns. An observatory for the SSLiverse. Talk at Defcon 18 (2010). https://www.eff.org/files/DefconSSLiverse.pdf.

[11] S. Han, K. Jang, K. Park, and S. Moon. PacketShader: A GPU-accelerated software router. In *ACM SIGCOMM*, September 2010.

[12] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister. Census and survey of the visible Internet. In *8th ACM SIGCOMM conference on Internet measurement (IMC)*, 2008.

[13] J. Heidemann, L. Quan, and Y. Pradkin. A preliminary analysis of network outages during hurricane sandy. Technical Report ISI-TR-2008-685b, USC/Information Sciences Institute, November 2012.

[14] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *21st USENIX Security Symposium*, August 2012.

[15] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL landscape: A thorough analysis of the X.509 PKI using active and passive measurements. In *11th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 427–444, 2011.

[16] IANA. IPv4 address space registry. http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml.

[17] V. Jacobson, C. Leres, and S. McCanne. libpcap. Lawrence Berkeley National Laboratory, Berkeley, CA. Initial release June 1994.

[18] J. Kasten, E. Wustrow, and J. A. Halderman. Cage: Taming certificate authorities by inferring restricted scopes. In *17th International Conference on Financial Cryptography and Data Security (FC)*, 2013.

[19] M Koster. A standard for robot exclusion. http://www.robotstxt.org/orig.html, 1994.

[20] A. Langley. Enhancing digital certificate security. Google Online Security Blog, http://googleonlinesecurity.blogspot.com/2013/01/enhancing-digital-certificate-security.html, January 2013.

[21] E. Law. Understanding certificate name mismatches. http://blogs.msdn.com/b/ieinternals/archive/2009/12/07/certificate-name-mismatch-warnings-and-server-name-indication.aspx, December 2009.

[22] D. Leonard and D. Loguinov. Demystifying service discovery: Implementing an Internet-wide scanner. In *10th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 109–122, 2010.

[23] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, USA, 2009.

[24] N. Mathewson and N. Provos. libevent—An event notification library. http://libevent.org.

[25] HD Moore. Security flaws in universal plug and play. Unplug. Don't Play, January 2013. http://community.rapid7.com/servlet/JiveServlet/download/2150-1-16596/SecurityFlawsUPnP.pdf.

[26] Netcraft, Ltd. Web server survey. http://news.netcraft.com/archives/2013/05/03/may-2013-web-server-survey.html, May 2013.

[27] N. Provos and P. Honeyman. ScanSSH: Scanning the Internet for SSH servers. In *16th USENIX Systems Administration Conference (LISA)*, 2001.

[28] Luigi Rizzo. netmap: A novel framework for fast packet I/O. In *2012 USENIX Annual Technical Conference*, 2012.

[29] S. Sanfilippo and P. Noordhuis. Redis. http://redis.io.

[30] H. Scholz. SIP stack fingerprinting and stack difference attacks. Talk at Blackhat 2006. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Scholz.pdf.

[31] A. Schulman and N. Spring. Pingin' in the rain. In *11th ACM SIGCOMM conference on Internet measurement (IMC)*, pages 19–28, 2011.

[32] K. Sklower. A tree-based packet routing table for Berkeley Unix. In *Winter USENIX Conference*, 1991.

[33] M. Smith and D. Loguinov. Enabling high-performance Internet-wide measurements on Windows. In *11th International Conference on Passive and Active Measurement (PAM)*, pages 121–130. Springer, 2010.

[34] W. R. Stevens and G. R. Wright. *TCP/IP Illustrated: The Implementation*, volume 2. Addison-Wesley, 1995.

[35] Tor Project. Tor Bridges. https://www.torproject.org/docs/bridges, 2008.

[36] Tor Project. obfsproxy. https://www.torproject.org/projects/obfsproxy.html.en, 2012.

[37] J. Viega, M. Messier, and P. Chandra. *Network Security with OpenSSL: Cryptography for Secure Communications*. O'Reilly, 2002.

[38] T. Wilde. Great Firewall Tor probing. https://gist.github.com/twilde/da3c7a9af01d74cd7de7, 2012.