A Search Engine Backed by Internet-Wide Scanning

Zakir Durumeric[†] David Adrian[†] Ariana Mirian[†] Michael Bailey[‡] J. Alex Halderman[†]

[†] University of Michigan [‡] University of Illinois, Urbana Champaign

{zakir, davadria, amirian, jhalderm}@umich.edu mdbailey@illinois.edu

ABSTRACT

Fast Internet-wide scanning has opened new avenues for security research, ranging from uncovering widespread vulnerabilities in random number generators to tracking the evolving impact of Heartbleed. However, this technique still requires significant effort: even simple questions, such as, "What models of embedded devices prefer CBC ciphers?", require developing an application scanner, manually identifying and tagging devices, negotiating with network administrators, and responding to abuse complaints. In this paper, we introduce Censys, a public search engine and data processing facility backed by data collected from ongoing Internet-wide scans. Designed to help researchers answer security-related questions, Censys supports full-text searches on protocol banners and querying a wide range of derived fields (e.g., `443.https.cipher`). It can identify specific vulnerable devices and networks and generate statistical reports on broad usage patterns and trends. Censys returns these results in sub-second time, dramatically reducing the effort of understanding the hosts that comprise the Internet. We present the search engine architecture and experimentally evaluate its performance. We also explore Censys's applications and show how questions asked in recent studies become simple to answer.

1. INTRODUCTION

Fast Internet-wide scanning has opened new avenues for empirically-driven security research, as evidenced by the recent surge in publications based on the technique (e.g., [1, 7–11, 13, 14, 18, 19, 24–29, 38]). Yet while tools such as ZMap [20] have reduced the time required to conduct large-scale port scans, collecting meaningful data through Internet-wide scanning has remained a specialized and labor-intensive process. Answering simple questions, such as "What fraction of HTTPS servers prefer forward-secret key exchange methods?", can take weeks of implementation and debugging, reducing the time security researchers have to focus on more important questions. In this specific case, the researcher would need to develop a high-performance application scanner to make HTTPS connections to hosts listening on port 443, test and fix problems with hosts that do not fully follow the TLS specification, run the actual scan, and then process many gigabytes of resulting data.

Before beginning this process, security researchers must negotiate with their institution's legal and networking teams for permission to conduct the scan, coordinate with their upstream network providers, and later respond to resulting abuse complaints. Many institutions (and independent researchers) lack the network facilities or administrative backing to perform scans. For these reasons, Internet-wide scanning has remained the province of a small number of research groups, which severely limits the applications to which this powerful methodology is applied.

In order to democratize Internet-wide scanning and enable researchers to efficiently ask questions about how security protocols have been deployed in practice, we have developed Censys, a cloud-based service that not only maintains an up-to-date snapshot of the hosts and services running across the public IPv4 address space, but also exposes this data through a search engine and API. In contrast to existing scanning tools, which have primarily focused on performing host discovery, Censys immediately produces results based on full protocol handshakes, facilitates a community-driven approach to characterizing the exploding number of embedded devices and vulnerabilities on the Internet, and requires little or no user preparation.

To approximate a real-time "bird's eye view" of the Internet, Censys continually scans the public address space across a range of important ports and protocols. It validates this data and performs application-layer handshakes using a pluggable scanner framework, which dissects handshakes to produce structured data about each host and protocol. The resulting data is post-processed with an extensible annotation framework that enables researchers to programmatically define additional attributes that identify device models and tag security-relevant properties of each host. We operate Censys transparently and expose data back to the research community. In turn, we encourage external researchers to contribute both *application scanners* (to scan additional protocols) and *annotations* (to identify devices or properties) to Censys. In this way, Censys automates and centralizes the mechanical aspects of scanning.

Censys exposes data to researchers through a public search engine, REST API, publicly accessible tables on Google BigQuery, and downloadable datasets. The search interface enables researchers to perform full-text searches and query

CCS'15, October 12–16, 2015, Denver, Colorado, USA.

ACM 978-1-4503-3832-5/15/10.

DOI: http://dx.doi.org/10.1145/2810103.2813703.

any of the structured fields and tags produced during scanning and post processing (e.g., `443.https.cipher_suite`). It supports full-text searches, regular expressions, and numeric ranges, and queries can be combined with Boolean logic. These queries can be run against a current snapshot of publicly accessible IPv4 hosts, Alexa Top 1 Million websites, and known X.509 certificates. After running a query, users can interactively explore the hosts, sites, and certificates that match their query, as well as generate statistical reports suitable for direct use in research.

As a simple example, Censys can identify the set of hosts in the U.S. that are currently vulnerable to Heartbleed with the query, `443.https.heartbleed.vulnerable: true AND location.country_code: US`. From there, Censys can output a complete list of matching IP addresses and graph the distribution of the most common vulnerable device models. These queries complete in under one second.

To facilitate more complex analysis, we publish raw application handshakes and daily point-in-time snapshots of the structured data. These can be queried using SQL through publicly accessible Google BigQuery tables or downloaded in JSON form. Censys additionally exposes data through a public REST API that allows researchers to export raw query results, fetch statistical data, and view the historical state of specific hosts and networks.

We present Censys's data collection architecture in Section 3, explain how Censys presents data to researchers in Section 4, and describe our deployment in Section 5. We then illustrate Censys's potential in Section 6 by showing how it can be applied to easily answer a range of questions from recent security studies, including measuring the impact of POODLE and tracking vulnerable industrial control systems.

Internet-wide scanning has already shown great potential for uncovering security problems and understanding the security of complex distributed systems. By moving scanning to the cloud, Censys dramatically reduces the effort needed to investigate these questions, enabling researchers to focus on asking more important questions rather than on the mechanics of answering them. Further, Censys allows the security community to increase global protocol coverage and provides a tractable solution for understanding the increasing number of embedded devices on the Internet. Simultaneously, it minimizes redundant scanning by research groups and minimizes the incoming network traffic monitored by network operators.

Censys is available free to the public at https://censys.io.

2. GOOD INTERNET CITIZENSHIP

As with any research conducted through active network probing, our work raises important ethical considerations. We carefully considered the impact of our experimental measurements and disclosure of our results. When reasoning about our impact, we considered a variety of stakeholders, from our local institution to Internet service providers and the owners of the remote systems. Although the community has yet to derive robust ethical standards for active measurement, our reasoning was guided by broad ethical principles, such as those embodied in the Menlo Report [5], as well as by the guidelines for ethical scanning set forth in the original ZMap work [20].

We coordinated with network administrators and IT leadership at our department, college, and institution, as well as with our upstream ISP, to ensure that our scans do not adversely impact network operations and that all support centers can route external inquiries to our team. Second, we signaled the benign intent of our activities. All of the scanning hosts have WHOIS records and reverse DNS entries that describe the intent of the scanning. Further, each scanning host runs a simple website on port 80 that describes the goals of the research, including what data we collect, and how to contact us. Third, we invite user exclusion requests and respond to requests within 24 hours. Fourth, all scans perform standards-compliant handshakes; we do not send malformed packets or handshakes.

Disclosure of scan data also raises ethical questions, since it exposes information about potentially vulnerable systems. To minimize harms, we deliberately choose to collect and distribute data that is, at least in principle, already publicly visible. Our scanners do not perform login attempts, deploy any exploits, or try to access non-public resource paths. Furthermore, we treat opt-out requests for scanning as a request to be removed from the search index, which allows remote administrators to decide whether or not to be included in Censys's public interface. Many network operators, after understanding the goals of our measurement work, have responded supportively and invited us to continue scanning them. Finally, it is our hope that by publishing scan data, carefully acquired and properly curated, we can reduce the need for Internet scanning performed by other researchers, and thus reduce the overall burden on destination networks.

In contrast, it is well established that attackers already use Internet-wide scanning to find vulnerable machines from botnets and bullet-proof hosting providers [17]. Thus, systems that are configured to expose data publicly are already at risk. Censys helps level the playing field by enabling legitimate researchers to study and enhance the security of these hosts by providing a source of reliable and ethically collected data.

3. COLLECTING DATA

The data that powers Censys is collected through horizontal application scans of the public IPv4 address space, which are scheduled across a pool of scan workers. In the first step, we perform host discovery scans using ZMap [20], complete application handshakes with responsive hosts using pluggable application scanners, and derive structured fields (e.g., certificate subject or TLS cipher suite) from the handshake. We save and publish the raw handshakes, but continue further processing, validating the collected scan data, extracting valuable fields and annotating handshakes with additional metadata, such as device model and software version using user-defined annotations.

The structured, annotated data is then streamed to a central database, *ZDb*, which aggregates the horizontal scan results, pivoting the data and updating comprehensive records describing individual IPv4 hosts, Alexa Top 1 Million websites, as well as maintaining auxiliary collections of all found X.509 certificates and public keys. ZDb streams changes to downstream services and produces publishable point-in-time snapshots of hosts and websites, as well as differential updates to its collection of certificates and public keys.

There are several observations that led to this architecture. First, while horizontal scans measure a single aspect of a service (e.g., whether an HTTPS server supports SSLv3), research questions frequently depend on multiple scans. For example, calculating the percentage of HTTPS servers that

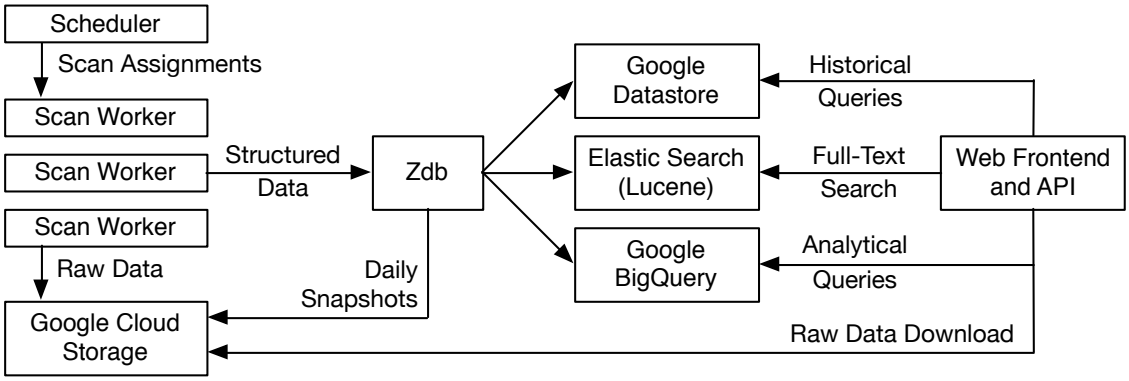

Figure 1: **Censys System Architecture** — Censys is driven by application scans of the IPv4 address space, which are scheduled onto a pool of scan workers. These workers complete scans, extract valuable fields, and annotate records with additional metadata in order to generate structured data about each host. These records are centrally managed in a custom database engine, ZDb, which maintains the current state of every host. ZDb feeds updated records to a web front-end where researchers can query the data.

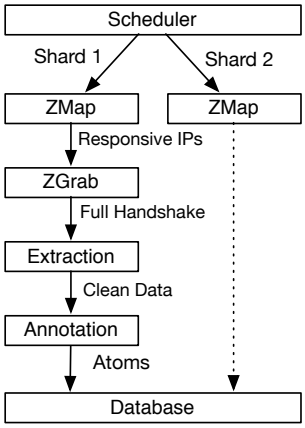

Figure 2: **Protocol Scanning and Annotation** — Each scan worker uses ZMap to perform host discovery for a shard of the IPv4 address, and completes protocol handshakes using pluggable application scanners. Censys extracts fields of interest and annotates records with additional metadata. The information from a protocol handshake is converted to an *atom*—a deterministic data structure describing a specific protocol on a host.

support SSLv3 requires a generic TLS scan and an SSLv3 scan. Similarly, a device model may only be identifiable based on its HTTP page, but this information is useful when studying any protocol. Therefore, despite being collected by protocol, data should be grouped by host. Second, our framework needs to be extensible, and facilitate community involvement. Much of ZMap's success is due to user contributed probe modules, and we believe the same will be true for Censys. This is particularly true for annotating hosts and services given the exploding number of embedded devices on the Internet. In turn, Censys needs to operate transparently and provide data back to the community. Third, the number of scans and annotations will both grow over time; our architecture should scale linearly to handle this increased load.

3.1 Internet-Wide Scanning

In the first step of data collection, we use ZMap [20] to perform single-packet host discovery scans against the IPv4 address space. The hosts found by ZMap seed pluggable application scanners, which perform a follow-up application-layer handshake and produce structured JSON data describing a certain aspect of how a host is configured. Typically, application scanners only perform a single handshake and measure one aspect of how a service is configured. For example, we perform separate horizontal scans and use different pluggable scanners to measure how HTTPS hosts respond to a typical TLS handshake, whether hosts support SSLv3, and whether a host is vulnerable to the Heartbleed attack.

Pluggable Scanners. While we can use ZMap to perform host discovery for many protocols, every application scanner requires protocol specific code and Censys's long-term success is dependent on easily adding new protocols. In order to reduce the effort required to scan new protocols, Censys handles the details of a scan and expects a minimally featured application scanner. Specifically, Censys expects a self-contained Linux executable that performs application-layer handshakes with IP addresses input on `stdin` and produces structured JSON output that describes how the protocol is configured on `stdout`. Censys controls network bandwidth by rate limiting the IP addresses provided to scanners, splits scans across multiple scan workers using ZMap's built-in sharding [2], and guarantees that application scanners do not scan networks that have requested exclusion using ZMap's blacklist.

In order to protect our infrastructure, we require that application scanners operate without root privileges or kernel modifications. Lastly, we encourage researchers to output metadata about the scan in JSON form and to log errors in a standard format, which enables Censys to confirm whether a scan completed successfully. We hope that by requiring minimal feature-set and allowing flexibility in language, we not only reduce the effort required for our team to add additional protocols, but also encourage an external community that develops new scanners.

Scheduling Scans. While it would be technically simplest to measure every aspect of a protocol at once, this frequently involves making multiple handshakes, which could

potentially inundate a host (e.g., an embedded device that only allows one or two simultaneous connection). Instead, we choose to perform scans independently, relying on our data processing pipeline to aggregate the data from different scans in order to reduce the load on individual hosts.

Internally, scans are referenced by the tuple (port, protocol, subprotocol, network destination), e.g., (443, https, heartbleed, ipv4_shard1), and individual scan executions are referenced by scan and timestamp. We currently maintain a master scan schedule; we plan to automatically schedule scans moving forward to better distribute network load.

3.2 ZGrab: Our Application Scanner

We are releasing a fast and extensible application scanner, ZGrab, which meets the previous specifications and facilitates the rapid development of new types of scans. At this time, ZGrab supports application handshakes for HTTP, HTTP Proxy, HTTPS, SMTP(S), IMAP(S), POP3(S), FTP, CWMP, SSH, and Modbus, as well as StartTLS, Heartbleed, SSLv3, and specific cipher suite checks. On a dual-Xeon E5-2640 (6-cores at 2.5 GHz) system with an Intel X520 ethernet adapter, ZGrab can complete HTTPS handshakes with the full IPv4 address space in 6h20m, and a banner grab and StartTLS connection with all publicly accessible SMTP hosts in 3h9m, 1.86k and 1.32k hosts/second respectively.

ZGrab is implemented in Go, which we chose based on its native concurrency support [36], safety compared to other low-level languages, and its native cryptographic libraries [3]. The framework allows scanners to be defined as a serial chain of network events. By default, ZGrab will only perform one event per host, connect, which simply opens a TCP connection. Events can be as simple as reading or writing data, or more advanced, such as initiating a TLS handshake. For example, the HTTPS scanner is implemented as a connection event and a TLS handshake event. To extend ZGrab to support scanning for StartTLS support among SMTP servers, we added events to read the SMTP banner, write the SMTP EHLO command, read the SMTP response, send the StartTLS command, read the response, and perform the TLS handshake: a total 62 LoC. The ZGrab framework handles concurrent connections, as well as logging and generating JSON documents that describe the connection.

All of the protocols in the initial Censys deployment use ZGrab and we encourage other researchers to consider using it as a starting point for developing other application scanners. We are releasing and maintaining ZGrab as a standalone open-source tool as part of the ZMap Project[1]. ZGrab can be used independently of Censys and works in conjunction with ZMap: ZMap quickly identifies hosts and ZGrab produces structured data about each of those hosts.

3.3 Validation, Extraction, and Annotation

The raw, JSON data produced by pluggable application scanners is collected by Censys, where it is validated, transformed into a structured schema, and annotated with additional metadata (e.g., device manufacturer and model), before being streamed into our central database.

Validation. Censys validates scan data in two ways. First, we extended ZMap to detect variances in network responses during the host discovery stage. If the scan response rate falls below a set threshold at any time, varies more than a set amount during the scan, reaches a maximum number of sendto failures, or if *libpcap* is unable to keep up and drops a set number of packets, the scan automatically terminates and is rescheduled. Second, Censys validates a scan at its completion and rejects that scans where ZMap's or the application scanner's response rates fall outside of a static bound or deviate more than 10% from the median of the scans that completed over the last two weeks; rejected scans are manually inspected afterwards. These checks are primarily in place in order to detect transient network failures, human error in configuration, and coding errors.

[1]ZGrab is available at https://github.com/zmap/zgrab.

```
@tag(port=443, proto="https", subproto="tls")
def dell_idrac(d):
  subject = d.443.https.certificate.subject
  if subject.ou == "Remote Access Group" \
        and subject.org == "Dell Inc.":
    return {"hw_manufacturer": "Dell Inc.",
        "hw_model": "iDRAC",
```

Figure 3: **Dell iDRAC Annotation** — Censys supports community maintained annotations—simple Python functions—that append additional metadata and tags to records. Here, we show the tag for Dell iDRAC remote management cards.

Extraction. Application scanners output raw data about every aspect of an application handshake in a format analogous with the network handshake. For example, in the case of TLS, client and server randoms are output as part of the Client and Server Hello messages. While this data is needed for some research, many of these fields are not helpful when searching for hosts or identifying devices, and would cause unnecessary churn in our database. Similarly, commonly searched fields are nested deep within network protocol messages, making them hard to find. We save and publish the raw application scanner output, but then extract significant values and transform handshake data into consistent, structured records that conform to a published schema. We further output records in a deterministic manner during this process (i.e., the record has the same cryptographic hash if no configuration changes have occurred), which allows us to reduce load later by discarding records that contain no changes. We refer to these deterministic records that represent how a service is configured as *atoms*.

Annotation. While the output from application scanners can be used to identify a device model or version, these details are not directly exposed by a scanner. Instead, they frequently require a small amount of logic (e.g., running a regular expression against the HTTP server header or certificate subject). To facilitate adding this type of metadata, Censys allows researchers to define annotations—small functions—that can inject additional metadata fields (e.g., `device_module`) or attach simple tags (e.g., `IPMI` for server management cards) to hosts, websites, and certificates. Annotations are defined as standalone Python functions that are provided read-only access to the structured data that Censys generates from each scan. We show an example annotation for labeling Dell iDRAC remote management cards in Figure 3.

We encourage researchers (and end-users alike) to contribute annotations for new types of devices and vulnerabilities. We are hosting our repository of annotations, along with

Database	New Records (rec/sec)	No Differences (rec/sec)	Consecutive Day (rec/sec)	Size on Disk
ZDb	58,340	136,664	110,678	1.60 GB
MongoDB	1,059	1,441	1,392	13.67 GB
Cassandra	501	511	506	3.40 GB

Table 1: **NoSQL Engine Comparison** — We compare ZDb against the two leading NoSQL engines [37], MongoDB and Apache Cassandra by loading a full scan of FTP. We find that ZDb is 80× faster than MongoDB and 219× faster than Cassandra when updating a consecutive day. For context, a single HTTP scan produces 2.4 K records/second.

our transformations and schemas as a standalone open source project, ZTag, on GitHub (http://github.com/zmap/ztag). We note that when ZTag is paired with ZMap and ZGrab, researchers can independently reproduce the entire data processing pipeline that Censys uses and independently generate the same data (Figure 2).

3.4 Challenges in Aggregating Data

Scan workers act independently and statelessly; an individual scan worker is not aware of other scan workers nor does it have any prior knowledge of a host's state. Therefore, a worker does not know whether a scanned host has changed or moved since a previous scan. Instead, workers stream all structured data to a central database to be processed. This vastly simplifies the development of application scanners and facilitates linearly increasing computational power by adding additional scan workers. However, this abstraction requires a more complex data processing pipeline that can process the incoming stream of data. For just the five largest protocols in our initial deployment (Table 2), this amounts to processing at least 330m records per day—a sustained 3.8k writes/second.

Our database needs are theoretically simple: we need to (1) parse incoming records and update the relevant part of each host record, maintain the current state of hosts, (2) stream changes to downstream, user-facing, services, (3) efficiently dump the database to JSON to produce daily snapshots. At a small scale, this could be easily handled out-of-the-box by one of many NoSQL database engines. However, we find that popular NoSQL engines perform updates prohibitively slowly given our workload (Table 1).

We tested the two most popular NoSQL engines [37], MongoDB 2.6.7 and Apache Cassandra 2.1.2, under three scenarios: (1) importing a new protocol for the first time, (2) re-importing the same dataset, and (3) loading two consecutive days of scans. These tests approximate the worst case, best case, and typical daily use case for Censys. We specifically loaded a banner grab scan of FTP (one of our simplest protocols) from February 12 and 13, 2015, which contained an average 14.5m records. Apache Cassandra consistently updated at about 500 records/second for all cases. MongoDB updated at an average 1,400 records/second when updating between two consecutive days, but consistently slowed as the database grew. At these rates, Cassandra would require 37 hours to update a single HTTP scan on our server; MongoDB would require 13 hours. MongoDB and Cassandra further required 8.5× and 2.1× the disk space of the raw data. We performed these experiments on an Intel branded server with dual Xeon E5-2640 processors (12 cores at 2.50 GHz), 128 GB of DDR3 memory, and a Samsung 850 Pro 1 TB SSD drive. We ran MongoDB with `w=0` write concern, which provides acknowledgment from the server that the request was received and could be processed, but not that it has been flushed to disk.

While it is possible to horizontally scale both database engines across a large number of servers, we observe that our needs differ greatly from a typical OLTP database and could likely be solved with a simple, custom database. Our first observation is that the majority of updates contain unchanged data (91.5% in the case of HTTPS; 88.5% for HTTP) and can be safely discarded. While MongoDB and Cassandra did not handle unchanged records significantly faster than actual updates, we could quickly discard these records. Second, if changed data is immediately streamed to downstream user-facing services, we do not need to quickly serve arbitrary reads. Instead, reads will only be necessary as part of large batch jobs. Therefore, we do not need to cache user data in memory in order to support fast queries. Instead, we should focus on optimizing for quickly processing incoming records and organizing the data to facilitate efficient batch jobs. Last, updates are streamed from scan workers, which are architected to be linearly scaled. If there are expensive operations to be performed on every record, these can be offloaded to the database client in order to reduce load on the central database.

With these considerations in mind we developed ZDb, which aggregates the records generated by scan workers, maintains the current state of hosts on the IPv4 address space, websites in the Alexa Top 1 Million Sites, and curates auxiliary collections of all X.509 certificates and public keys we've encountered. ZDb is able to process upwards of 110K records/second for a typical workload—a 219× speedup over Cassandra and 80× speedup over MongoDB. We describe ZDb's architecture and our data processing pipeline in the next section.

3.5 Censys Data Flow

After a scan worker finishes processing a host, it serializes the annotated, structured data into a Google Protocol Buffer message [22], which it sends to the central ZDb server along with the SHA-1 fingerprint of the message and a key describing what was scanned. These messages are queued in memory and processed by a pool of worker threads, which deserialize and validate the outer record, and check whether the record has changed since the last scan (using the attached SHA-1 fingerprint). If the record has changed, the new record is written to disk, and enqueued in external Redis queues for downstream services (e.g., the database of historical records, the search index, and other institutions subscribed to a live data feed). If the record has not changed since the latest scan, we simply mark the record as having been seen in the most recent scan. When the scan completes, we prune any records that were not updated or marked as having been seen in the scan. Analogous to the IPv4 database, we maintain a collection of records for the Alexa Top 1 Million domains, as

Port	Protocol	SubProtocol	Port Open (Hosts)	Full Handshake (Hosts)	Raw Record Size (KB)	Processed Size (KB)	% Diff 1 Day	Database Size on Disk
80	HTTP	GET /	77.3 M	66.8 M	.69 (1.8)	.32 (.096)	11.5%	10.9 GB
443	HTTPS	TLS	47.1 M	33.3 M	3.7 (4.9)	4.5 (1.4)	8.5%	50.1 GB
443	HTTPS	SSLv3	43.1 M	22.5 M	2.8 (3.9)	.08 (.0001)	6.8%	1.5 GB
443	HTTPS	Heartbleed	47.1 M	33.1 M	3.6 (4.8)	2.3 (.002)	4.4%	4.8 GB
7547	CWMP	GET /	55.1 M	44.3 M	.3 (.3)	.34 (.09)	28.1%	6.5 GB
502	MODBUS	Device ID	2.0 M	32 K	.19 (.20)	.10 (.08)	10.6%	0.0 GB
21	FTP	Banner Grab	22.9 M	14.9 M	.08 (.09)	.33 (.31)	7.5%	9.0 GB
143	IMAP	Banner Grab	7.9 M	4.9 M	2.8 (4.1)	2.2 (8.9)	3.3%	7.0 GB
993	IMAPS	Banner Grab	6.9 M	4.3 M	6.6 (4.2)	4.9 (8.4)	2.0%	11.5 GB
110	POP3	Banner Grab	8.8 M	4.1 M	2.5 (3.9)	2.3 (.44)	4.4%	6.9 GB
995	POP3S	Banner Grab	6.6 M	4.0 M	6.4 (4.1)	2.4 (.4)	1.9%	6.9 GB
25	SMTP	Banner Grab	14.7 M	9.0 M	1.9 (3.6)	1.5 (1.2)	5.8%	8.9 GB
22	SSH	RSA	14.3 M	14.3 M	1.0 (.2)	.6 (.2)	13.8%	5.8 GB
53	DNS	OpenResolver	12.4 M	8.4 M	.05 (.001)	.145 (0)	29.8%	0.7 GB
123	NTP	Get Time	1.6 M	1.2 M	.02 (.0006)	.145 (0)	92.8%	0.1 GB
1900	UPnP	Discovery	9.5 M	9.5 M	.051 (.002)	.10 (0.0)	37.2%	0.6 GB

Table 2: **Scanned Protocols** — We scan 16 protocols in our initial implementation. For each protocol and subprotocol we scan, we show the average size and standard deviation for raw and transformed records, as well as the percent-change in records across two days of scans. Most protocols have a less than a 15% turnover rate between consecutive days.

well as auxiliary collections of all seen X.509 certificates and public keys.

Internally, data is stored on disk using RocksDB [21], an embeddable key-value store optimized for flash storage. RocksDB buffers writes to a small in-memory table and on-disk journal, and, in another thread, flushes changes to a log-structured merge-tree on disk. The records stored in RocksDB consist of the serialized protobuf messages generated by the scan workers. We note that because data is written to disk as a log-structured merge-tree, we maintain a global ordering of all records, which we use to logically group multiple records that describe a single host. Similarly, records describing a single network are grouped together. This allows us to efficiently generate daily snapshots of the IPv4 address space by performing a single, linear pass of the database, grouping together all records describing a single host, and outputting a structured JSON document describing all measured aspects of each host.

All of the functionality in ZDb could be achieved using only RocksDB, but would require a disk read to process every incoming record. To improve performance, we cache the SHA-1 fingerprints of current records, along with whether the record were seen in the latest scan using an in-memory Judy Array. With this additional optimization, we no longer need to make random reads from RocksDB during processing and can process all incoming records that contain no changes without touching disk. We then update when each record was seen at the end of the scan during the prune process, which already performs a linear pass of the records on disk. With these optimizations, ZDb is able to process 58k records per second in the worst case, 137k records/second in the best case, and 111k records/second in the daily workload using the same metrics we used to measure MongoDB and Apache Cassandra.

We would be remiss not to mention that because records are queued in memory before they are processed, and because we cache which records have been seen in memory until a scan finishes, ZDb would lose the data associated with a particular scan if the server crashed (e.g., due to a kernel panic). While this is non-optimal, we find this risk acceptable, because a fresh scan can be completed in a matter of hours, which would likely be similar to the amount of time needed to investigate the crash, recover the database, and finish the remainder of a scan. We take a similar approach to managing failures during a scan. If a scan worker crashes, or if scan validation fails for any reason, we start a new scan rather than try to recover the previous scan.

4. EXPOSING DATA

To be successful, Censys needs to expose data back to the community, which ranges from researchers who need to quickly perform a simple query to those who want to perform in-depth analysis on raw data. In order to meet these disparate needs, we are exposing the data to researchers through several interfaces, which offer varying degrees of flexibility: (1) a web-based query and reporting interface, (2) a programmatic REST API, (3) public Google BigQuery tables, and (4) raw downloadable scan results. We further plan to publish pre-defined dashboards that are accessible to users outside of the research community. In this section, we describe each of these interfaces in depth.

4.1 Search Interface

The primary interface for Censys is a search engine that allows researchers to perform full-text searches and structured queries against the most recent data for IPv4 hosts, the Alexa Top 1 Million websites, and known certificates. For example, a researcher can find all hosts currently vulnerable to Heartbleed in the United States with the query: `443.https.heartbleed.vulnerable: True AND location.country_code: US`. This query executes in approximately 250 ms and users are presented with the hosts that meet the criteria, along with basic metadata, which includes the breakdown of the top ASes, countries, and tags. Users can view the details of any host, as well as generate statistical reports.

Search Syntax. The search interface supports basic predicate logic (e.g. `(location.country_code: US OR location.country_code: CA) AND 80.http.server: Apache`), ranges (e.g., `80.http.http_status.code > 200`), wildcards (e.g., `443.https.certificate.certificate.issuer.*:GoDaddy*`) and regular expressions (e.g., `25.smtp.`

Interface	Query	Time
Web	80.http.get.headers.server:*	218 ms
Web	443.https.tls.signature.valid:true AND 443.https.tls.version.name:SSLv3	356 ms
API	25.smtp.banner.banner:gsmtp	82 ms
API	ip:1.2.3.4	12 ms
Report	443.https.tls.certificate.issuer_dn	417 ms
Report	502.modbus.device_id.product_name	11 ms

Table 3: **Censys Response Times** — Censys can be used to search records, and to create aggregations over fields in the search results. Here, we show example searches and aggregations along with their execution times. All of the queries completed in under 500 ms.

`banner: \Apache.*\`). Users can perform simple full-text searches as well as query any structured field generated during the scan process, including user annotations and system-maintained metadata (e.g., location and network topology).

Viewing Individual Records. Users can view the details any host, certificate, or domain returned by a query. This includes a user-friendly view of how each service is configured, the most recent raw data describing the host, user-provided metadata and tags, and historical scan data. We similarly display geographic location, routing, and WHOIS information.

Dynamic Reports. Once a query completes, users can generate reports on the breakdown of any field present on the resulting datasets. For example, users can view the breakdown of server chosen cipher suites for IPv4 HTTPS hosts with browser-trusted certificates by performing the query `443.https.tls.validation.browser_trusted: True` and generating a report on `443.https.cipher_suite.name`.

Backend. The search interface and reports are powered by Elasticsearch [6], an open-source project that front-ends Apache Lucene [4]. We maintain three indexes within Elasticsearch: IPv4 hosts, Alexa Top 1 Million websites, and all known certificates; ZDb updates the three indexes real time. All updates also appended to a Google Cloud Datastore collection, which is used to serve the history of each host. Our web front-end is implemented in Python using the Pylons Pyramid Framework, and is hosted on Google App Engine. We plan to rate-limit the web interface, using session-based token buckets, in order to prevent screen scraping and encourage developers to use the REST API we describe in the next section for programmatic access. We present the response time for sample queries in Table 3.

4.2 Programmatic Access

Censys has a programmatic API that provides equivalent functionality as the search interface, but presents JSON results and follows the semantics of a REST API. For example, researchers can get the history of an IPv4 host by doing a GET request for https://censys.io/api/ipv4/8.8.8.8/history. To prevent abuse, we require users to use a key, but are happy to provide these to researchers.

4.3 SQL Interface

We recognize that not all research questions can be answered through the search interface we described. This is particularly true for historical queries, because we only expose the most recent data. To support more complex queries, we are exposing Google BigQuery tables that contain the daily ZDb snapshots of the IPv4 address space and Alexa Top 1 Million Domains, along with our auxiliary collection of certificates and public keys. Google BigQuery is a Dremel-backed cloud database engine designed for performing large analytical queries. Queries require 10–20 seconds to execute, but allow a full SQL syntax and are not restricted to specific indexes. Authenticated researchers can perform queries through the Censys web interface, or access the tables directly using their own Google Cloud Accounts.

4.4 Raw Data

Lastly, we are publishing all of the raw data from our scans, along with our curated ZDb snapshots of the IPv4 address space, Alexa Top 1 Million websites, and known certificates. We will be posting these as structured JSON documents, along with data definitions, and schemas for common databases at censys.io/data. We previously posted scan data on https://scans.io, a generic scan data repository that our team hosts. We will continue to maintain the scans.io interface, provide continued access to our historical datasets, and allow researchers to upload other data. However, we will no longer post our regular scans to https://scans.io, but rather encourage users to download these directly from Censys's web interface.

4.5 Protocol Dashboards

While Censys's primary goal is to answer researchers' specific queries, the backend similarly supports the types of queries needed to generate pre-determined reports and dashboards. We plan to publish dashboards on Censys' website, which present various perspectives of how protocols are deployed in practice. At initial release, we will be releasing a *Global HTTPS Dashboard* that presents how well HTTPS has been deployed in practice, an *Alexa HTTPS Deployment Dashboard* that shows historical trends in HTTPS deployment and which high-ranking sites have not deployed HTTPS, and dashboards for each of the recent HTTPS vulnerabilities, which will supersede the *Heartbleed Bug Health Report*, *POODLE Attack and SSLv3 Deployment*, *Tracking the FREAK Attack*, and *Who is affected by Logjam?* sites. Initially, dashboards will be statically defined. However, we encourage researchers to contribute reports at the completion of research projects, and moving forward we hope to allow everyone to dynamically define new reports.

5. INITIAL DEPLOYMENT

We are releasing Censys with the sixteen protocols listed in Table 2, which we are scanning from the University of Michigan.

Scanning. We originally scheduled all protocols to run on a daily basis, but quickly reduced scan speed and frequency due a non-negligible uptick in complaints and exclusion requests. Instead, we have switched to scheduling scans based on protocol turnover. We specifically scan HTTP, HTTPS, and CWMP on a daily basis; SSH, Modbus, FTP, DNS, NTP, UPnP, SMTP, and SSH biweekly; and IMAP, IMAPS, POP3, POP3, POP3S, HTTPS/SSLv3, and HTTPS/Heartbleed weekly. All scans are performed over a 24 hour hour period. We plan to further fine tune this schedule based on which protocols are frequently searched and downloaded after public release, and will post updated schedules on the web interface.

Backend. During initial testing, when we scanned every protocol daily, we were able to consistently complete all sixteen scans from a pool of 12 scan workers and ZDb comfortably handled upwards of 40 full-speed scans on a single host—an Intel branded server with with two Intel Xeon E5-2640 (6 cores at 2.50GHz) processors, 192 GB of DDR3 memory, and RAID 1+0 with four Intel 850 Pro 1 TB SSD drives. Our scan workers are Dell PowerEdge 1950s, with a Quad-Core Intel Xeon X5460 at 3.16 Ghz, 16 GB of memory, and a local 1 TB 7200 RPM SATA drive.

Frontend. Our front-end is powered by a number of components. The web interface itself is implemented as a Python Pyramid project, served through Google App Engine, our search backend is powered by Elasticsearch and Apache Lucene, historical data is stored in Google Datastore, and historical and advanced queries utilize Google BigQuery. With the exception of Elasticsearch, these services will autoscale based on load. During private testing, we were comfortably able to serve Elasticsearch requests for a small number of internal users from a single server (2 x Intel Xeon E5-2640 (6 cores at 2.50GHz) processors, 192 GB of DDR3 memory, and RAID 1+0 with four Intel 850 Pro 1 TB SSD drives). However, it remains unclear exactly what Censys's load will look like after we launch publicly. Our initial public Elasticsearch deployment runs on Google Compute Engine and consists of six backend data nodes, each with 52 GB of memory, 8 VCPUs, and 500 GB solid state storage, and two front-end nodes. We can add additional nodes to the cluster as load dictates.

6. APPLICATIONS

Exposing scan data to new sets of researchers, who do not otherwise have access to scanning methodologies was one of the primary motivations for developing Censys. In this section, we show how Censys can be used to easily answer frequently asked Internet measurement questions, as well as questions from recent security studies.

6.1 Industrial Control Systems

SCADA (Supervisory control and data acquisition) systems provide a communication channel for computer systems to control industrial equipment, such as motors, generators, and physical sensors. SCADA-enabled devices are used extensively in industrial and infrastructure-critical systems, ranging from factories and power plants to HVAC systems and water treatment facilities. Attacks on these systems are particularly dangerous, because SCADA devices bridge the gap from virtual to physical and have devastating consequences. In one recent example, the compromise of a blast furnace control system resulted "massive damage" to a German steel mill in 2014 [39].

One of the primary SCADA protocols, Modbus, was originally designed for local communication over a serial connection, but has since been extended to operate over networks and the Internet [32]. The protocol contains no authentication, and as a result, publicly accessible Modbus devices are an inherent security risk. To show how Censys can be used characterize these devices, we implemented annotations to identify different types of Modbus devices. With 42 annotations, we categorized 92% of Modbus devices that responded to a device identification request. These annotations contain device classification, hardware manufacturer, software name, and version. We queried Censys for modbus devices and

Country	Modus Devices	
United States	4723	24.7%
Spain	1,448	7.58%
Italy	1,220	6.39%
France	1,149	6.02%
Turkey	884	4.63%
Canada	822	4.30%
Denmark	732	3.83%
Taiwan	682	3.57%
Europe	615	3.22%
Sweden	567	2.97%
Total	12,842	67.23%

Table 4: **Top Countries with Modbus Devices** — We identified Modbus hosts in 117 countries, with the top 10 countries accounting for 67% of the total costs, and nearly one-quarter of all Modbus hosts we identifed are located in the United States.

Device Type	Count
Modbus Ethernet Gateway	1,440
Programmable Logic Controller	1,054
Solar Panel Controller	635
Water Flow Controller	388
Power Monitor/Controller	158
Touchscreen System Controller	79
SCADA Processor/Controller	99
Environment/Temperature Sensor	10
Cinema Controller	5
Generic Modbus Device	28,750

Table 5: **Modbus Devices** — We used Censys to categorize publicly available industrial control systems that support the Modbus protocol.

Vulnerability	Alexa	IPv4	IPv4 Trusted
Heartbleed	1.16%	0.96%	0.19%
SSLv3 Support	46.0%	55.8%	34.7%
SSLv3 Only	0.05%	2.9%	0.07%

Table 6: **Heartbleed and SSLv3** — We show a breakdowns for the Heartbleed vulnerability and SSLv3 support for HTTPS hosts in the IPv4 address space and the Alexa Top 1 Million.

Expires	Count	%SHA-1	% Total	Chrome
2015	6.86 M	60.2%	46.0%	Secure
2016	2.84 M	25.0%	19.0%	Warning
2017+	1.69 M	14.8%	11.3%	Insecure

Table 7: **SHA-1 Prevalence** — Chrome is beginning to mark sites with SHA-1 signed certificates as insecure. We used Censys to measure the fraction of trusted sites with SHA-1 certificates and how they appear in Chrome.

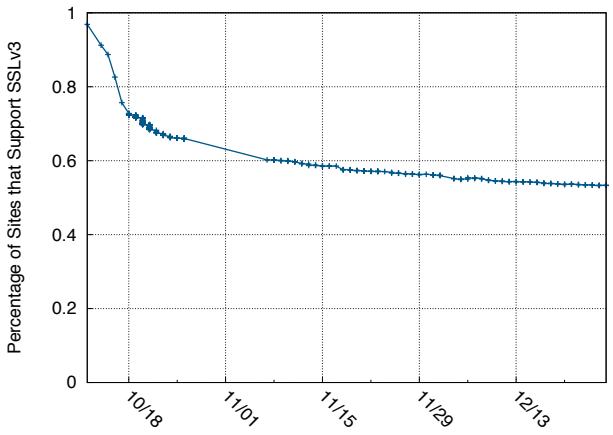

Figure 4: **SSLv3 Deprecation** — Censys tracks both the IPv4 and Alexa Top 1 Million websites. We track the deprecation of SSLv3 of both after the POODLE announcement using Censys. We find that the support for SSLv3 has dropped from 96.9% to 46.0% between October 2014 and February 2015 for the Top 1 Million Websites.

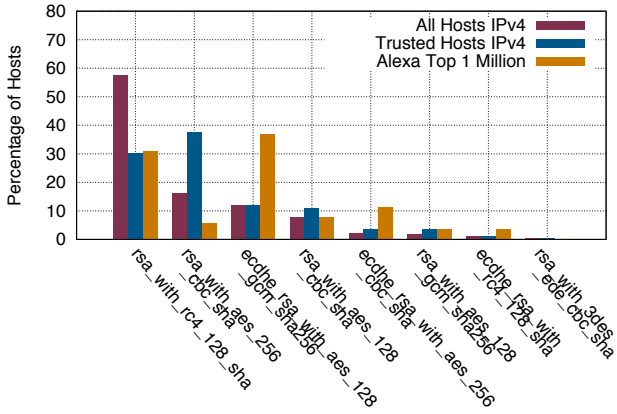

Figure 5: **HTTPS Cipher Suites** — We show the breakdown of cipher suites chosen by all IPv4 hosts, hosts with browser trusted certificates, and the Alexa top million domains using numbers returned by Censys's web interface.

aggregated hosts based on type, which identified 32,622 hosts and completed in 21 ms. We show the breakdown of device types in Table 5.

By querying Censys, we find that Internet-connected Modbus devices are pervasive, despite the total lack of security in the protocol. In situations where remote access is required, good security practice dictates that Modbus devices should be access-controlled through the use of a firewall or VPN [32]. Unfortunately, Censys identified many of the publicly accessible Modbus devices as network-enabled SCADA processors and gateways, capable of forwarding instructions to and controlling other networks of SCADA devices which might not be directly connected to the Internet. Censys located devices spread over 1,880 ASes and 111 countries, with the top countries accounting for 77% devices (Table 4).

6.2 Heartbleed, Poodle, and SSLv3

The security community watched two catastrophic TLS vulnerabilities unfold in 2014: Heartbleed [12] and Poodle [33]. Heartbleed was caused by an implementation error in OpenSSL that resulted in servers publicly leaking private data, including cryptographic keys and login credentials. Poodle was caused by fundamental flaw in the SSLv3 protocol, which allowed attackers to man-in-the-middle connections. Both of these vulnerabilities garnered both widespread research [15, 16, 19] and media attention. The fundamental question of "What hosts are still vulnerable?" surrounded the disclosures, and Internet-wide scanning was the key tool used for both understanding the impact of the vulnerabilities and facilitating Internet-wide notifications. Unfortunately, data describing the susceptibility of the Internet stagnated after initial publications. Despite this, the question of who is vulnerable remains important.

To determine what hosts with browser trusted certificates remained vulnerable to Heartbleed, we queried Censys for `443.https.certificate .signature.valid:true` and aggregated the `443.https.heartbleed_ vulnerable` field. Similarly, to determine what percentage of hosts only supported SSLv3 we queried for HTTPS hosts and aggregated the `443.https.tls_version.name` field, which completed in 229 ms. We provide breakdowns of the current state of Heartbleed and SSLv3 in Table 6.

Despite Heartbleed being disclosed over a year ago, over 1% of the Alexa Top 1M domains remain vulnerable. Additionally, 46% of HTTPS-enabled Alexa Top 1M sites still support SSLv3, down from 97% at the disclosure of POODLE four months ago [16]. All of the data used to make these measurements about the current state of the Internet can be publicly queried on Censys's web interface.

We acknowledge that not all vulnerabilities can immediately be detected upon disclosure without some level of code modification. However, the data processing and application scanner framework in Censys allows researchers to quickly respond to vulnerabilities and to easily develop a custom scan module, if necessary. For example, in the case of Heartbleed, a minor modification to ZGrab to send a custom Heartbeat packet was all that was needed in order to stream Heartbleed scan data into Censys. Realistically, automated measurement in Censys can be started within a few hours of vulnerability disclosure. As the protocol coverage of Censys increases, we except the need for custom scan modules will further decrease. In the case of POODLE, FREAK, and Logjam, measurement of supported TLS versions and cipher suites would have been sufficient in detecting vulnerability trends immediately at the time of disclosure.

6.3 Institutional Attack Surface

Managing large, publicly accessible networks is an involved and complicated process. Censys can be used by organizations to measure their external-facing attack surface. Network-connected devices can be difficult to keep track of, and users may mistakenly open up devices and services intended to be private. Censys supports queries based on network blocks and ASes, which an organization can use to easily export all of the data that Censys has gathered about their publicly-accessible services. This data can be used to identify mistakenly exposed or vulnerable devices, as well as identify devices that may have been overlooked when patching software.

Unfortunately, these mistakenly exposed devices might not only present a security risk to the institution hosting them, but also to the entire Internet. For example, misconfigured public NTP and DNS resolvers are the major cause of the new trend in amplification DDoS attacks [14, 27]. Amplification attacks can be globally prevented by eliminating publicly accessible open resolvers and NTP servers. As a result, several initiatives such as The Open Resolver Project [31] and The Open NTP Project [34] provide free scanning services on networks of up to 1,024 hosts for network administrators to use to identify misconfigured or publicly accessible devices that could be leveraged for amplification attacks. Censys removes the need for service-specific and vulnerability-specific scanning initiatives. Censys provides similar services as both of these initiatives, and is both real-time and Internet-wide.

6.4 Deprecating SHA-1

The Chrome Security team is spearheading an effort to deprecate HTTPS certificates signed using SHA-1, citing the decreasing cost of collision attacks [35]. Chrome now shows certificates signed with SHA-1 and expiring before 2016 as secure, displays a warning for those expiring between 2016 and 2017, and rejects SHA-1 signed certificates expiring in 2017 or later as insecure.

We used Censys to characterize the current prevalence of SHA-1 signed certificates for HTTPS hosts with browser trusted certificates. Specifically, we queried Censys to find all browser-trusted certificates expiring in 2015, 2016, and 2017 or later (e.g. `443.https.certificate.signature.valid: true AND 443.https.certificate.validity.end:[2017 TO *]`), and used Censys's aggregation feature to bucket results by signature algorithm. We find that 76.3% of trusted IPv4 hosts currently use a SHA-1 signature and show the breakdown of SHA-1 certificates and their status in Chrome in Table 7.

6.5 Cipher Suites

The security of the TLS protocol fundamentally relies on the use of strong cipher suites. However, servers and clients often have to make a tradeoff between level of security and compatibility. When performing a TLS handshake, ZGrab offers the cipher suites implemented by the Golang TLS library and logs the chosen cipher suite, which is then exported to Censys. Using Censys, we generated the distribution of selected cipher suites by all HTTPS hosts by querying for HTTPS hosts and aggregating on the `443.https.cipher_suite.name` field, which Censys completed in 212ms. We show these distributions in Figure 5. Even from these basic distributions, we can gain insights into cipher suite selection in the wild. The Alexa Top 1M domains prefer ECDHE key exchange, whereas IPv4 prefers RSA key exchange. Hosts without trusted certificates are much more likely to use RC4 rather than AES. We can also see that overall not only are RC4 and AES preferred to 3DES, but that 3DES ciphers are only chosen by less than 1% of hosts.

7. RELATED WORK

There have been a large number of research studies over the past several years that have been based on Internet-wide scanning [1, 7–11, 13, 14, 18, 19, 24–29, 38], which encouraged us to develop Censys.

Censys further enables these types of studies, and lowers the barriers to entry for utilizing Internet-wide scan data. Similarly, there have been several scanners designed for scanning the IPv4 address space, notably ZMap [20] and Masscan [23]. While we introduce ZGrab, an application scanner, Censys itself is not a new Internet scanner. Rather, it builds upon ZMap to provide a higher level interface to scan data.

University	Protocol	Shodan	Censys
Michigan	FTP	38	255
(141.212.0.0/16)	HTTP	274	987
	HTTPS	53	337
Iowa	FTP	12	98
(128.255.0.0/16)	HTTP	415	1,304
	HTTPS	30	662
Berkeley	FTP	84	582
(128.32.0.0/16)	HTTP	602	1,871
	HTTPS	158	1,188

Table 8: **Shodan Comparison** — We compared the number of hosts returned by Shodan and Censys for FTP, HTTP, and HTTPS on three different /16 network blocks, each belonging to a different public U.S. university. We find that, on average, Censys found 600% more FTP hosts, 220% more HTTP hosts, and 800% more HTTPS hosts.

7.1 Scan Driven Search Engines

The closest work to Censys is Shodan, which provides a text search of banners, primarily on FTP, SSH, Telnet, and HTTP [30]. Banners are searchable as plaintext, and searches can be filtered by CIDR range and location. While Censys and Shodan seek to to fulfill similar goals, they take different approaches, offer differing functionality, and fulfill different needs.

Unfortunately, it is unclear how Shodan performs banner grabs, from where, and how frequently. In order to compare Shodan's coverage with Censys, we compared the result sets for three academic institutions on three shared protocols: FTP, HTTP, and HTTPS. Shodan does not expire results, which makes it difficult to compare with our most recent scans. However, Shodan does include a last-seen-at timestamp on records and we compare Censys against any hosts Shodan has seen in the past month. In comparison, Censys only presents records from the latest scan. This will inflate the number of results presented by Shodan, given that hosts frequently move, but allows us to approximate the differences between the two services. On average, Censys found 598% more FTP hosts, 222% more HTTP hosts, and 808% more HTTPS hosts than Shodan. Spot-checks confirmed that these additional hosts were not false, positives, but rather hosts not recently found, or were missing from Shodan.

In order to measure how frequently Shodan scans the Internet, we provisioned a publicly available FTP server with a unique banner, and queried the Shodan search API every 10 minutes for the banner. Despite this, Shodan did not respond with our FTP host in its result set until 25 days after provisioning our server. In comparison, the host was present in Censys's public interface in under 48 hours. We also note that during this experiment, Shodan timed out for 2% of queries and returned an Invalid Query error for 1%.

Shodan's search interface and API further differ from Censys. In comparison to Censys's support for query statements on parsed out fields, Shodan only allows simple full-text

searches against the raw text of a host's banner. Further, Shodan limits anonymous users to 10 hosts per search and registered users to 50 hosts per search. All API calls require creating an account, and access to larger result sets and HTTPS requires purchasing an account (a minimum 50 USD). Any results from the API and results in the web interface beyond the 50 hosts per search require purchasing "query credits". Credits can be purchased at $2.50/credit, or $500/month for unlimited credits, and allow viewing. In comparison, Censys publicly provides a fully featured query interface to parsed application handshake data and it provides API results in paginated sets of 5k hosts. We further post all raw and parsed data from Censys on the Internet-Wide Scan Data Repository at https://scans.io.

While Shodan has frequently been used to show the existence of vulnerable systems, its lack of timeliness, coverage, and transparency prevents its use as a trusted tool by researchers. In contrast, Censys is architected to be a fully transparent, community-driven project. All of the code is available on GitHub, all results are publicly available, and we support a query syntax and API tailored to researchers.

8. CONCLUSION

Until now, there remained a gap between the technical ability to perform host discovery scans on the IPv4 address space and answering meaningful research questions. In this paper, we introduced Censys, a public query engine and data processing facility backed by data collected from ongoing Internet-wide scans. Designed to help researchers answer security related questions, Censys collects structured data about the IPv4 address space and supports querying fields derived from scans and generating statistical reports. We explored several security applications of Censys and showed how Censys can be used to easily answer questions from recent studies. We hope that Censys enables researchers to easily answer questions about the Internet that previously required extensive effort, while simultaneously reducing duplicate effort and total scan traffic.

Acknowledgments

The authors thank Ben Burgess, Alishah Chator, Henry Fanson, and Harsha Gotur for their help building Censys. We thank the exceptional sysadmins at the University of Michigan for their help and support throughout this project, including Chris Brenner, Kevin Cheek, Laura Fink, Dan Maletta, Jeff Richardson, Donald Welch, Don Winsor, and others from ITS, CAEN, and DCO. We are extremely grateful to Elie Bursztein and the Google Anti-abuse team for their support and advice, without whose help this project would not have been possible. We also thank Brad Campbell, Aleksander Durumeric, James Kasten, Kyle Lady, Adam Langley, HD Moore, Pat Pannuto, Paul Pearce, Niels Provos, Mark Schloesser, Eric Wustrow, our anonymous reviewers for valuable feedback, and the many contributors to the ZMap and ZGrab open source projects. This material is based upon work supported by the National Science Foundation under grants CNS-1111699, CNS-1255153, CNS-1345254, CNS-1409505, CNS-1409758, and CNS-1518741, by the Google Ph.D. Fellowship in Computer Security, by the Morris Wellman Faculty Development Assistant Professorship, and by an Alfred P. Sloan Foundation Research Fellowship.

9. REFERENCES

[1] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[2] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman. Zippier ZMap: Internet-wide scanning at 10 Gbps. In *8th USENIX Workshop on Offensive Technologies*, Aug. 2014.

[3] S. Ajmani, B. Fitzpatrick, A. Gerrand, R. Griesemer, A. Langley, R. Pike, D. Symonds, N. Tao, and I. L. Taylor. A conversation with the Go team. Golang blog, June 2013. http://blog.golang.org/a-conversation-with-the-go-team.

[4] Apache. Lucene. https://lucene.apache.org.

[5] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan. The Menlo report. *IEEE Security and Privacy*, 10(2), Mar. 2012.

[6] S. Banon. Elasticsearch, 2013. https://www.elastic.co.

[7] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *36th IEEE Symposium on Security and Privacy*, May 2015.

[8] A. Bonkoski, R. Bielawski, and J. A. Halderman. Illuminating the security issues surrounding lights-out server management. In *8th USENIX Workshop on Offensive Technologies*, Aug. 2013.

[9] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow. Elliptic curve cryptography in practice. In *18th International Conference on Financial Cryptography and Data Security*, Mar. 2014.

[10] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov. Using Frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations. In *35th IEEE Symposium on Security and Privacy*, May 2014.

[11] S. Checkoway, M. Fredrikson, R. Niederhagen, A. Everspaugh, M. Green, T. Lange, T. Ristenpart, D. J. Bernstein, J. Maskiewicz, and H. Shacham. On the practical exploitability of Dual EC in TLS implementations. In *23rd USENIX Security Symposium*, Aug. 2014.

[12] Codenomicon. Heartbleed, Apr. 2014. http://heartbleed.com.

[13] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis. A large scale analysis of the security of embedded firmwares. In *23rd USENIX Security Symposium*, Aug. 2014.

[14] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir. Taming the 800 pound gorilla: The rise and decline of NTP DDoS attacks. In *14th ACM Internet Measurement Conference*, Nov. 2014.

[15] Z. Durumeric, D. Adrian, M. Bailey, and J. A. Halderman. Heartbleed bug health report, Apr. 2014. https://zmap.io/heartbleed.

[16] Z. Durumeric, D. Adrian, J. Kasten, D. Springall, M. Bailey, and J. A. Halderman. POODLE attack and SSLv3 deployment, Oct. 2014. https://poodle.io.

[17] Z. Durumeric, M. Bailey, and J. A. Halderman. An Internet-wide view of Internet-wide scanning. In *23rd USENIX Security Symposium*, Aug. 2014.

[18] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS certificate ecosystem. In *13th ACM Internet Measurement Conference*, Oct. 2013.

[19] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman. The Matter of Heartbleed. In *14th ACM Internet Measurement Conference*, Nov. 2014.

[20] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *22nd USENIX Security Symposium*, Aug. 2013.

[21] Facebook. RocksDB: A persistent key-value store for fast storage environments. http://rocksdb.org.

[22] Google. Protocol buffers. https://github.com/google/protobuf.

[23] R. Graham. Masscan: The entire Internet in 3 minutes. Errata Security blog, Sept. 2013. http://blog.erratasec.com/2013/09/masscan-entire-internet-in-3-minutes.html.

[24] X. Gu and X. Gu. On the detection of fake certificates via attribute correlation. *Entropy*, 17(6), 2015.

[25] 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*, Aug. 2012.

[26] M. Kranch and J. Bonneau. Upgrading HTTPS in mid-air: An empirical study of strict transport security and key pinning. In *2015 Network and Distributed System Security Symposium*, Feb. 2015.

[27] M. Kührer, T. Hupperich, C. Rossow, and T. Holz. Exit from hell? Reducing the impact of amplification DDoS attacks. In *23rd USENIX Security Symposium*, Aug. 2014.

[28] Y. Liu, A. Sarabi, J. Zhang, P. Naghizadeh, M. Karir, M. Bailey, and M. Liu. Cloudy with a chance of breach: Forecasting cyber security incidents. In *24th USENIX Security Symposium*, Aug. 2015.

[29] W. R. Marczak, J. Scott-Railton, M. Marquis-Boire, and V. Paxson. When governments hack opponents: A look at actors and technology. In *23rd USENIX Security Symposium*, Aug. 2014.

[30] J. Matherly. Shodan FAQ. http://www.shodanhq.com/help/faq.

[31] J. Mauch. Open resolver project. http://openresolverproject.org.

[32] Modbus IDA. MODBUS TCP implementation guide, Oct. 2006. http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.

[33] B. Moller, T. Duong, and K. Kotowicz. This POODLE bites: Exploiting the SSL 3.0 fallback, Sept. 2014. https://www.openssl.org/~bodo/ssl-poodle.pdf.

[34] Network Time Foundation. Open NTP project. http://openntpproject.org.

[35] C. Palmer and R. Sleevi. Gradually sunsetting SHA-1. Google Online Security Blog. http://googleonlinesecurity.blogspot.com/2014/09/gradually-sunsetting-sha-1.html.

[36] R. Pike. Concurrency is not parallelism. In *Heroku Waza*, Jan. 2012. http://talks.golang.org/2012/waza.slide.

[37] Solid IT. DB-Engines ranking. http://db-engines.com/en/ranking.

[38] E. Wustrow, C. Swanson, and J. A. Halderman. TapDance: End-to-middle anticensorship without flow blocking. In *23rd USENIX Security Symposium*, Aug. 2014.

[39] K. Zetter. A cyberattack has caused confirmed physical damage for the second time ever, Jan. 2015. http://www.wired.com/2015/01/german-steel-mill-hack-destruction.