WhoWas with blacklists. We summarize three IP usage behaviors of the discovered malicious webpages, and observe that the malicious webpages can stay up for a long time after being blacklisted.

(5) We find that a majority of web servers use dated versions of server and backend software, some of which have known vulnerabilities.

To facilitate future research, we have made the source code for WhoWas publicly available [3] and will make data available to researchers upon request.

The rest of this paper is structured as follows. We present background on IaaS clouds in §2, and related work in §3. We outline the WhoWas platform in §4, and its key analysis engines in §5. We describe how we collect data using WhoWas in §6, and the ethical and privacy considerations we take into account are discussed in §7. In §8, we show the broad utility of WhoWas via three usage cases: an analysis of the dynamics of cloud deployments, a study of malicious activity, and a characterization of in-use web software. We conclude in §9.

2. BACKGROUND

Infrastructure-as-a-Service (IaaS) systems allow users to rent virtual machines (VMs or instances) with flexible computing resources and operating systems. IaaS providers like Amazon EC2 and Azure typically provide a few default VM configurations with set resources (e.g., number of CPUs, storage, and memory), and many configuration options to adapt instances to the user's needs.

The most commonly offered instances are *on-demand*, which can be created by users as required, stopped when no longer needed, and are charged for the hours used. Azure only offers on-demand instances, while EC2 in addition offers *reserved* instances, which are rented for one or three years at lower cost.

Each instance has two associated IP addresses, a private IP that only works inside the cloud and a publicly routable IP. Both IPs are by default dynamic; they are released when an instance is stopped and can be reassigned to a different instance, possibly from a different customer. This creates IP churn, where the same public IP address may frequently change ownership.

In addition, EC2 and Azure provide up to 5 static public IP addresses to each account or service deployment[1]. These static IPs are not specific to an instance, but assigned to the user until explicitly released, and the user chooses the mapping between IP and instance. Both providers also offer load-balancing services to distribute requests to an IP across multiple instances.

In EC2, each instance is also assigned an externally resolvable DNS name, whose structure follows a provider-specific pattern: the prefix "ec2-" followed by a string that replaces the dots in the public IP with hyphens, and a region-specific suffix.

EC2 virtual networking. Amazon Virtual Private Cloud (VPC) allows users to create a fully-controllable and easy-to-customize virtual networking environment for their instances. Each VPC is a logically separate virtual network. We use the terms EC2-Classic and EC2-VPC to refer to classic networking and VPC networking, respectively. Compared to EC2-Classic, EC2-VPC provides extra features. For example, an EC2-VPC instance (also referred to as a VPC instance) can run on single-tenant hardware, while an instance launched with EC2-Classic (a classic instance) may run on shared hardware; a user can assign one or multiple specific private IP addresses to an EC2-VPC instance; classic instances can only have one public IP, while VPC instances can bind to multiple network interfaces and IP addresses (at most 240 IPs, the exact maximum depending on instance type) [4]. Because of the security and management benefits compared to EC2-Classic, Amazon has encouraged users to adopt VPC and announced that accounts created after Dec. 12, 2013 can only launch instances with VPC enabled.

[1]Called Elastic IPs in EC2 and Reserved IPs in Azure.

3. RELATED WORK

Prior measurement studies of web usage in the cloud primarily used active DNS probing and packet capture [2]. Such an approach provides only a limited view of website availability and deployment changes over time. Packet captures (along with similar techniques such as passive DNS [5]) are also subject to sampling bias.

Closely related to our work is the Wayback Machine, which archives old versions of websites [6]. While we investigated using its data for our study, this would only provide information on sites submitted and, in particular, provide relatively poor coverage of cloud usage.

Active probing is a popular technique that has been used for a variety of applications. Two decades ago, Comer and Lin proposed active probing to identify differences between TCP implementations [7]. Internet-wide active probing has been used for identifying vulnerable SSH servers [8], compiling a Web census [9], finding DNS servers providing incorrect resolutions [10], detecting weak cryptography keys in network devices [11], for identifying malicious servers [12, 13], and more. In our work we do not scan the full Internet but only IP ranges from cloud hosting providers. Localized scans have also been used for detecting network-based malware [14] in a local network. In contrast we scan remote networks.

Other works study how to scan fast. Staniford et al. described techniques that malware can use to quickly spread through scanning [15]; Netmap [16] proposed a software framework for fast packet I/O, enabling a single core to saturate a 10Gbps link; and Durumeric et al. released Zmap [17], a horizontal scanner that can probe the Internet in 45 minutes from a single host. WhoWas does not aim to scan fast or wide. Rather, it periodically scans the same IP ranges to analyze temporal evolution.

The ethical issues of active probing and sharing measurement data are treated by several prior works. Leonard et al. [18] describe how to perform maximally polite Internet-wide horizontal scans. We use their recommendations as guidelines for our probing. Allman and Paxson [19] discuss issues when sharing network measurement data. Currently, we do not make our data publicly available on the Internet, but will release it to researchers upon request, after privacy issues have been evaluated.

Prior works have studied malicious activity in cloud services. Balduzzi et al. [20] investigate security issues in the sharing of virtual images among EC2 users. Canali et al [21] install web applications on shared hosting providers and issue abuse reports on them. In contrast, we measure malicious activity by cloud users renting virtual machines. Nappa et al. [22] show that attackers are abusing cloud instances to host drive-by exploit servers. Our work further quantifies this trend by including multiple types of malicious activity, and analyzing the effect of cloud IP churn on blacklists.

4. THE WHOWAS PLATFORM

To track website deployments on IaaS clouds over time, we design and build a system that can perform lightweight probing of clouds, process the results, and enable a programmatic interface for conducting analyses on the gathered data. The basic functionality we seek is to be able to associate a (cloud) IP address with its status and content at some point in time, where status means whether it

is hosting a VM instance responding to network packets, running a web server, or otherwise.

To do so, WhoWas performs lightweight, active measurements over some user-specified time period and records the results for later analysis. A key design choice we faced involved how to balance the granularity of observations (how often we probe) of an IP address's status against not burdening customers of cloud services with many network requests. Another challenge is in identifying and extracting website features that may be useful for determining if two IP addresses are hosting the same web content. If successful, this would allow us to measure the "footprint" of a particular service: the number of IP addresses used by the service at a given period of time.

The main components of the WhoWas system are shown in Figure 1. They consist of a scanner, webpage fetcher, and feature generator, which are used in that order to populate a database with information about measurement results. WhoWas defines a Python library interface to allow programmatic access to the resulting data sets. This makes it easy to write analyses, and also to build extensions on top of the basic platform (see §5). All source code is publicly available [3].

We discuss each of the main components of WhoWas in turn.

Scanner. WhoWas is seeded with a list of IP address ranges to target. At the moment these must be manually gathered. Both EC2 and Azure make public the IP addresses used by their services [23, 24]. For clouds that do not, one could enumerate them via other means (e.g., via appropriate Whois interrogation) to seed a list. The scanner also takes a blacklist of IP addresses that should not be scanned. This enables users to avoid certain ranges, which enables operators to allow tenants to opt out from measurement studies.

The scanner translates these IP lists into a task list. The primary goal of the scanner is to ascertain what IP addresses are running typical services. In particular we focus on identifying instances running HTTP and/or HTTPS services. For each IP address, the scanner sends a TCP SYN (from now on, a probe) first to port 80 (HTTP default) and then to 443 (HTTPS default). If both probes fail to respond, then a probe is sent to port 22 (SSH). The SSH probe enables identifying live instances that are not hosting publicly accessible HTTP(S) servers.

The scanner times out a probe after two seconds and does not retry failed probes. The former increases speed of the scanner while the latter minimizes interaction with the cloud customers. To evaluate the effect of larger timeouts, we performed the following experiment. First, we randomly selected 5% of EC2 IPs from each /24 prefix used by EC2. This resulted in 235,070 IPs. For these IPs, we compared probes with timeouts of 2 s and 8 s. We observed an increase of only 0.61% in the number of IPs responding with the longer timeout. For the same set of IPs we also had the scanner probe with a 2 s timeout once, then again 200 s later and again three more times at intervals of 100 s for a total of 5 probes. Compared to the first round of scanning, only 0.27% more IPs responded if one took into account the additional 4 probes. We concluded that these failure rates will not unduly affect interpretation of scanner results.

By default, the scanner uses a global scan rate of 250 probes per second (pps) and only treats each IP once a day. This means that an IP address receives at most three probes (80/tcp, 443/tcp, 22/tcp) in a day. These scan rates are significantly smaller than those used in prior work that range from 1,000 pps up to 1.4 million pps [17, 18] (though we will admittedly be targeting smaller ranges). One round of scanning visits all the IP addresses input to WhoWas. We chose small defaults to balance burden on infrastructure with the temporal granularity of WhoWas measurements.

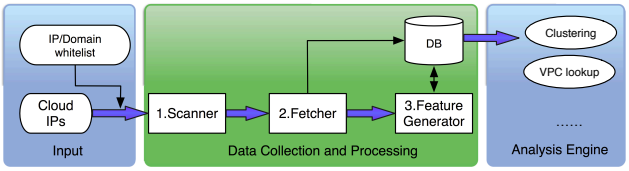

Figure 1: The main components of WhoWas.

We say an IP address is *responsive* if it responds to any of the network probes. Otherwise it is *unresponsive*.

HTTP requests. For any IP address that responds as being open on port 80 or port 443, the scanner submits the IP to the webpage fetcher, which will randomly pick up a worker process from a worker pool to gather further information from this IP. The worker generates a URL by concatenating the IP address to either "http://" (should port 80 and 443 have been seen as open) or "https://" (if only port 443 was open). A GET request is submitted to the resultant URL concatenating with "robots.txt". Then, a GET may be sent to the URL depending on the content of robots.txt. At most two GETs are made per round of scanning for a given IP address.

The fetcher records any received HTTP response code, response header, or error information in a MySQL database. Each round of scanning uses a distinct database table, with a timestamp in the table name to indicate the start of the round. If the type of the response content is "application/*", "audio/*", "image/*" or "video/*", the fetcher foregoes downloading the content as our analysis engine cannot yet process non-text data. For text content (e.g., HTML, JSON, XML) we store the first 512 KB of content into the database. The fetcher does not follow links in the collected content and does not execute active content. The fetcher is built using the Python Requests library [25].

An IP is *available* in a round if the HTTP(S) request for the URL without robots.txt succeeded, and *unavailable* otherwise. Note that unresponsive IPs are also unavailable.

Feature extraction. After a round of scanning is finished, WhoWas invokes a separate process to extract various features from the results. For each webpage successfully fetched, we extract and insert into the database: (1) the back-end technology from the "x-powered-by" field in HTTP response headers; (2) a description of a webpage from the "description" tag in the returned HTML; (3) a string that consists of all field names in the HTTP response header (sorted alphabetically and separated by "#"); (4) the length of the returned HTML; (5) the title string from the HTML; (6) the web template (e.g., Joomla!, WordPress, or Drupal) indicated by "generator" tags in the HTML; (7) the server type as reported in HTTP response headers; (8) the keywords as indicated in a keywords tag in the HTML; and (9) any Google Analytics ID found in the HTML. We mark entries as unknown when they are missing from the HTML or HTTP headers.

We additionally compute (10) a simhash [26–28] over the HTML of the returned webpage. This algorithm has been used previously for webpage comparison: two webpages are similar if their fingerprints have low Hamming distance. We use 96-bit hashes.

Limitations. We briefly highlight a few limitations of our platform. First, it can only detect a portion of web-facing instances hosted in the cloud. For example, the scanner fails to successfully discover a web server instance if the owner configures firewalls to only allow traffic from a given source; the instance is not bound to a public

IP address (e.g., instances in some EC2 VPC networks); and/or the ports used for SSH or web services are not set to the default.

A second limitation is that we can visit a website using only the IP address instead of the intended domain. PaaS providers or web hosting providers may use one instance for multiple websites or web applications, and not allow requests without specific URLs. In this case, they will return 404 pages or other customized pages to the fetcher. Even so, for some IPs, we can still identify their ownership by looking at unique features in the returned contents (e.g., if the 404 response contains the domain name of the website).

A third limitation is the granularity achieved by WhoWas. By only querying once per day, or even once per several days, we can only measure changes in status with that granularity. Network packet drops or other temporary failures will have WhoWas mark an IP as unresponsive or unavailable for the entire measurement period. More generally, EC2 and Azure (as well as other clouds) have per-hour charging models, meaning that we might expect an hourly granularity to better reflect the rapidity of changes.

Increasing the scan rate could be easily accommodated via parallelism. However, while fetching webpage content at such granularities might not represent a burden for large enterprise tenants (let alone the IaaS provider), it is impossible to know whether WhoWas is also interacting with tenants expecting very little, or even no, network traffic[2]. For these reasons we err on the side of having a relatively conservative scan rate.

5. ANALYSIS TOOLS

As described thus far, WhoWas can perform scans, fetch pages, and store them. To enable richer analyses, we build in extra tools to aid users. We first include a clustering algorithm to associate IP addresses that are likely to be hosting the same webpage or, at least, be operated by the same owner. We also build in some knowledge of the semantics associated with particular IP addresses by way of cloud cartography techniques.

Clustering. After collecting webpages for a period of time, we use clustering to help identify IP addresses that are likely to be hosting the same web applications. Features used for clustering are the title, template, server software, keywords, Google Analytics ID, and simhash. Other extracted features are not used for clustering because they are unstable (e.g., HTML length) or redundant. We then use a 2-level hierarchical clustering, using the features for IPs across all rounds of measurement. The clustering is run across all rounds, so that each entry in a cluster corresponds to an <IP, round> pair. In the first level clustering, we cluster solely on the first five features. At this level, strict equality is used for comparison, so each first-level cluster only has webpages with the same title, template, server, keywords, and Analytics ID.

In the second level clustering we use the simhashes. The distance measure here is the Hamming distance between two simhashes, which can take on a value from 0 (identical) to 96 (very dissimilar) since we use 96-bit hashes. We use a simple tuning procedure to pick a distance threshold for clustering based on the gap statistic, which is a common method for estimating the number of clusters when doing unsupervised clustering [29]. Each resulting second-level cluster is given a unique random identifier, and we associate to each IP's round record their cluster assignment.

If using this clustering approach to identify IPs belonging to the same website, an immediate limitation is that significant changes to the features (e.g., due to an update to the page or underlying server software) will result in IPs being placed into distinct clusters. We therefore provide some additional heuristics to help catch changes in features that may not be indicative of change of ownership of an IP address. In a post-processing step after clustering, we merge clusters using the following algorithm.

Consider two records in the database:
$DB[IP, T, simhash, cluster, f_a, f_b, f_c, f_d, f_e]$,
$DB[IP', T', simhash', cluster', f'_a, f'_b, f'_c, f'_d, f'_e]$
where we store the IP, the timestamp (T), the simhash of the content of the page and the cluster to which the IPs belong, and the other features (f_x) namely: title, server, template, keywords, and Google Analytics ID. We merge two records if the following conditions hold: the IPs are the same, the simhashes differ at most by 3 bits (used also in [30]), one feature is the same between the two records, and the two clusters are temporally ordered.

The 2-level clustering and merging heuristic above strive to identify web applications that are being used across distinct IPs or across time on the same IP address. It was the culmination of a manual refinement process, where we started with using just simhashes and then added top level features to help improve clustering results (as measured by manual inspection of samples). This was made easy by the programmatic WhoWas interface. The interface also makes it easy to modify, for example, the approach above to cluster with other goals in mind, such as simply finding related content (dropping the server feature) or only using Analytics IDs.

After merging, some clusters may still include IP addresses actually operated by different owners. For example, sites that lack a Google Analytics ID but reply with a default Apache page all end up in the same cluster. To not have such clusters bias analyses further in the paper, we perform some manual cleaning of the clusters. First, we write a script to remove the top-level clusters (and associated second-level clusters) whose title contains expressions indicating that WhoWas failed to fetch useful content. (e.g., "not found" and "error".) Then, for second-level clusters with over 20 IP addresses associated with them on average per day, we check the titles of their top-level clusters, and exclude them if they correspond to default server test pages. (e.g., with title "welcome-apache".)

Unless specified otherwise, from now on we refer to second-level clusters after merging and cleaning as simply clusters.

Cloud cartography. Previous work highlighted the utility of so-called cloud cartography, in which one uses measurements to associate cloud public IP addresses to types of infrastructure. For example, this has been used to estimate in EC2 the type [31] and availability zone [2,31] of a given instance. Different than previous works, we use cartography to help disambiguate distinct networking configurations associated with public EC2 IPs.

Recall that Amazon EC2 provides two ways to launch instances: classic and VPC. For classic instances, one public IP address represents one VM instance. VPC instances can instead have multiple public IPs associated to them. If we see a cluster of N IPs, if these IPs are VPC it may therefore be the case that these IPs are associated with fewer than N VM instances.

We would therefore like to have WhoWas be able to differentiate between VPC and classic instances. We use DNS interrogation to differentiate between public IP addresses used for classic instances and those used for VPC instances: Amazon's internal DNS always returns a public IP address when resolving the public domain name associated to a VPC IP address (c.f., [32]).

In more detail, we have the following procedure for a one-time measurement to label each public IP in a region with VPC or classic. In that region, we launch an instance in the classic network, and then perform a DNS query for each public IP in this region. For example, say the IP address is 1.2.3.4, then we

[2]Indeed, we received emails from a handful of tenants that did not realize their web servers were publicly accessible! See also §7.

Region	VPC prefixes	% all IPs in region
USEast	280	13.7
USWest_Oregon	256	36.4
EU	124	20.8
AsiaTokyo	98	32.0
AsiaSingapore	82	33.9
USWest_NC	72	22.5
AsiaSydney	64	33.3
SouthAmerica	56	31.9

Table 2: Breakdown of public IP prefixes in EC2 by VPC

form the EC2-style public domain name[3] "ec2-1-2-3-4.compute-1.amazonaws.com" and perform a DNS lookup from the instance. If the DNS response contains a start-of-authority (SOA) record (meaning no DNS information for that DNS name), then there is no active instance on this IP and also the IP is labeled as classic. If instead the response contains an IP address that is in EC2's public IP address space, then the IP is using VPC; otherwise, the IP is using classic networking.

After performing such queries for all EC2 public IP addresses (with a suitably low rate limit), we obtain a map of IP prefixes as being used for VPC or classic. Table 2 shows the results in terms of the number of /22 prefixes (c.f., [33]) used for VPC compared to all IPs associated to that region.

We include in WhoWas the resulting /22 map to add to any public EC2 IP address round record an indicator of whether the IP uses VPC or classic networking.

6. DATA COLLECTION

We seed WhoWas with the Amazon EC2 and Azure public IP address ranges as of Sep. 10, 2013 and use them as the target IP addresses [23, 24]. There are 4,702,208 target IP addresses in EC2 and 495,872 in Azure. The probing on EC2 began on Sep. 30, 2013 and lasted for 3 months, and the probing on Azure began on Oct. 31, 2013 and lasted for 2 months. In October and November, a round of measurements was performed every 3 days. We then increased the frequency to one round per day in December. The final round of measurement was on Dec. 31, 2013. In the end, we collected a total of 51 rounds of scanning in EC2 and 46 in Azure. The number of rounds are similar despite an extra month of probing on EC2 because of the slower rate in the first two months and because at the beginning we occasionally stopped the probing for updates on our infrastructure.

We run WhoWas from a pair of machines, one for EC2 and one for Azure. The machines had the same configuration: Ubuntu 12.10 i386 server, 8 core Intel i7 @ 3.40 GHz CPU, 32 GB memory, and 2 TB hard drive for storing data. The only software dependencies are Python Requests and Python MySQLdb libraries. The worker number for the crawler is set to 250 by default. The timeout for HTTP(S) connections is set to 10 seconds.

The resulting data set. We collected about 900 GB of data in total. Afterwards, we ran the analysis tools described in §5 to cluster the IP addresses and also label IP addresses as being VPC or classic. Recall that clusters contain <IP, round> pairs. As shown in Table 4, an average of 64.7% of the responsive EC2 IP addresses (60.6% for Azure) are available in each round of probing. Table 3 shows a break down of the ports open on responsive IPs. For the available IP addresses, Table 5 shows the top 5 text content types in the responses, as only text content is collected by WhoWas.

[3]Recall as per §2 that these differ slightly on a per-region and per-zone basis, but that it is easy to determine from looking at examples from one's own accounts.

provider \ port	22-only	80-only	443-only	80&443
EC2	25.9	38.0	5.5	30.6
Azure	9.3	45.8	16.5	28.4

Table 3: Average percentage of responsive IPs per measurement round that open a given port(s)

provider \ code	200	4xx	5xx	other
EC2	64.7	28.0	7.2	0.10
Azure	60.6	30.2	9.2	0.02

Table 4: Average percentage of responsive IPs per measurement round that respond with a given HTTP status code.

Table 6 summarizes the clustering for both datasets. First, it shows the number of responsive IP addresses. Then, the distinct simhashes for the content obtained from those IP addresses, which is larger than the number of IP addresses because the content collected from an IP address varies over time. Next, it shows the number of first-level and second-level clusters, which shows that some first-level clusters are split into multiple second-level clusters. The last row shows the final number of clusters, after merging and cleaning has been applied to the second-level clusters.

7. ETHICAL CONSIDERATIONS

The unsolicited nature of active probing can make some targets consider it offensive and can potentially place a burden on both the individual tenants and the cloud provider. We take this issue seriously in our study and seek a careful balance between the amount of interaction required and the merits of the obtained data. We guide our active probing on the recommendations of prior work [18].

Our scanner and fetcher are located at the premises of one of our institutions and their use was made in cooperation with the institution's IT staff. To minimize the impact on the EC2 and Azure networks we limit the probing to an average of once per day and constrain our probing to only 3 ports (80/tcp, 443/tcp, 22/tcp).

Our HTTP fetcher includes a brief note in the User-Agent string stating that the request is being made for a research project. The note includes an email address to contact in case of questions/concerns. This allows tenants to request opting out of future queries. We also exclude IPs that we have observed to disallow robots requesting the top-level content, as per the robots exclusion protocol [34]. In the course of operating WhoWas we received 84 emails from instance operators. Upon further explanation, most operators were happy to hear about our research project. Only 16 requested to have IPs excluded from future scans, which we promptly did. In total 67 IPs were excluded in the course of the experiment. In general, the reasons offered for requesting IPs be excluded were either: (1) the servers are not designed to be public, but with careless configurations, they are accidentally discovered by WhoWas; and (2) since IaaS providers charge tenants based on traffic, the traffic generated by WhoWas, though it is very small, is still unwelcome for some tenants.

While technically any data we obtain is public, in the sense that we are simply fetching webpages from publicly advertised IP addresses, it is clear that some cloud tenants inadvertently made accessible what should not be. So as to respect potential privacy issues, we will not make our data set publicly available on the In-

EC2		Azure	
text/html	95.9	text/html	97.8
text/plain	2.1	text/plain	1.0
application/json	1.0	application/xml	0.7
application/xml	0.3	application/json	0.2
text/xml	0.3	application/xhtml+xml	0.1
other	0.4	other	0.2

Table 5: Top 5 content-types in EC2 and Azure and the percentage of webpages of each content-type in all collected webpages.

	EC2	Azure
Responsive IPs	1,359,888	154,753
Unique simhashes	1,767,072	210,418
Top-level clusters	236,227	30,581
2nd-level clusters	256,335	39,183
Final clusters	243,164	31,728

Table 6: The number of responsive IPs and unique simhashes of their content, as well as the number of clusters output by the different steps of the clustering.

ternet. Interested researchers should contact the authors if they desire access to the data set, and we will evaluate the privacy issues of each request and use appropriate privacy safeguards (e.g., use anonymization tools [35]). The code and other tools that WhoWas consists of will however be released publicly, and in future work we may investigate making a public interface to our data set that provides appropriate privacy protections (e.g., only providing aggregate statistics).

8. ANALYSES USING WHOWAS

In this section we report on various analyses performed using WhoWas with the data sets gathered as discussed in §6. We start by using WhoWas to study usage and its change over time in EC2 and Azure. Then we use WhoWas to shed light on malicious activity in clouds, and finally we use WhoWas to describe the distribution of software in the web ecosystem in clouds.

8.1 Cloud Usage Dynamics

IP responsiveness and availability. Table 7 summarizes the usage statistics across all measurement rounds. The average number of responsive IPs is almost an order of magnitude larger in EC2 than in Azure, which illustrates their difference in popularity. The average IP address space usage is almost identical on both services and relatively low (23.7–23.9%), so both have ample room for growth. EC2 has on average 758 K available IPs, 7.6 times more than Azure with 99 K, which indicates that EC2 hosts a higher proportion of services running on ports other than 80/tcp and 443/tcp, which are less likely to be web services.

Figure 8 shows the IP responsiveness and availability over the two month measurement period. The variation of responsive and available IP addresses in both services over the two months is low, with a 0.3–0.5% standard deviation. On absolute numbers, EC2 grew more over this period, adding 36,414 responsive IPs by the end of the period, compared to 8,270 for Azure. However, the relative growth of Azure is larger, with an increase in 7.3% of responsive and 7.7% of available IPs, compared to 3.3% and 4.9% for EC2. Figure 8 shows several noticeable dips in the curves. In EC2, these dips correspond to the dates Oct. 4, Nov. 8, Nov. 30, Dec. 14, and Dec. 28, all in 2013. Interestingly, they are all Fridays or Saturdays. By appropriate queries to the WhoWas database, we found that 3,198, 2,767, 1,449, 983, and 1,327 clusters become unavailable on EC2 on these dates, respectively, and never return again. In Azure, the dips happened on Nov. 29 and Dec. 07, 2013, also Friday and Saturday. On these two days, 360 and 385 clusters become unavailable and never return again. A total of 15,295 IPs in EC2 and 775 IPs in Azure are associated with these clusters. These dips may be caused by web service maintenance, webpage update, web service migration (to other providers or to new added networks that we did not probe), or other reasons. It is possible that the servers that left the cloud were intended for other purposes instead of web services. Unfortunately, we could not figure out the exact reason for these departures.

Cluster number and size. The average number of clusters for a round of probing in EC2 is 185,701 and in Azure 27,048. Over the measurement period, the number of clusters increased by 3.3% in EC2 and by 6.2% in Azure.

We define the average size of a cluster to be the average number of IPs in a cluster per round of measurement. The average cluster size distribution for EC2 is: 78.8% have average size of one IP address; 20.8% between 2 and 20; 0.28% between 21 and 50; and only 0.07% greater than 50. For Azure the distribution is quite similar, with a slightly larger number of small clusters: 86.2% (1), 13.6% (2–20), 0.1% (21–50), and 0.1% ($>$ 50). The numbers show that the vast majority of clusters in both clouds use just a small number of IPs.

The amount of overlap across the two clouds we study is small. We find 980 clusters are using both EC2 and Azure, and 85% (834) of them use the same average number of IPs in each cloud. These latter clusters all use a small number of IPs on average (at most 5 IPs). We also observe 110 clusters that use more IPs in EC2 than they use in Azure, on average. One of these clusters, a VPN service (cluster 3 in Table 15), uses over 2,000 IPs more in EC2 than it uses in Azure. We did not see any cluster migrations between EC2 and Azure during our measurement period.

The clustering heuristics we use can only provide estimates, and so we may be under- or over-counting cluster size in some cases. Web services may also be making use of other providers not included in our study, which would lead us to underestimate their overall resource usage. Indeed some larger services may use other hosting solutions such as content delivery networks, which are not included in our study. Despite these potential sources of error, the data still supports the conclusion that the majority of the services hosted in clouds are small. We delve more deeply into some of the largest clusters at the end of this section.

IP status churn. A key feature of WhoWas is our ability to estimate churn, meaning the quantity of change in status of IP addresses. The overall churn rate, meaning the number of IPs that change status in terms of responsiveness, availability, or which cluster they are associated to divided by the total number of IPs, is 3.0% for each of EC2 and Azure. Investigating individual types of status change, we have that for EC2 the average change in responsiveness was 2.5%, the average change in availability was 1.0%, and an average of 0.1% of IPs changed clusters between measurement rounds.[4] The same values for Azure are 2.2%, 1.7%, and 0.3%. A time series of churn rates is shown in Figure 9.

If one instead looks at percentages relative to all unique IP addresses that are responsive on either round $T-1$ or T, then overall churn rate becomes 11.9% in EC2 and 12.2% in Azure on average. The responsiveness, availability, and cluster change rates become

[4]Note that they do not sum to 3.0% since unavailable IPs include unresponsive IPs, as well as responsive but unavailable IPs.

	EC2			Azure		
	# Responsive IPs	# Available IPs	# Clusters	# Responsive IPs	# Available IPs	# Clusters
Minimum	1,061,834 (22.6%)	718,693 (15.3%)	179,870	111,851 (22.6%)	94,348 (19.0%)	25,903
Maximum	1,145,448 (24.4%)	773,832 (16.5%)	188,274	120,605 (24.3%)	102,953 (20.8%)	27,786
Average	1,113,599 (23.7%)	758,144 (16.1%)	185,701	118,290 (23.9%)	99,720 (20.1%)	27,048
Std. dev.	18,733 (0.4%)	13,232 (0.3%)	1,939	2,169 (0.4%)	2,640 (0.5%)	605
Overall growth	36,414 (3.3%)	35,442 (4.9%)	5,984	8,270 (7.3%)	7,299 (7.7%)	1,604

Table 7: Summary of overall usage of EC2 and Azure address spaces. Percentages are the fraction of the total number of IP addresses probed.

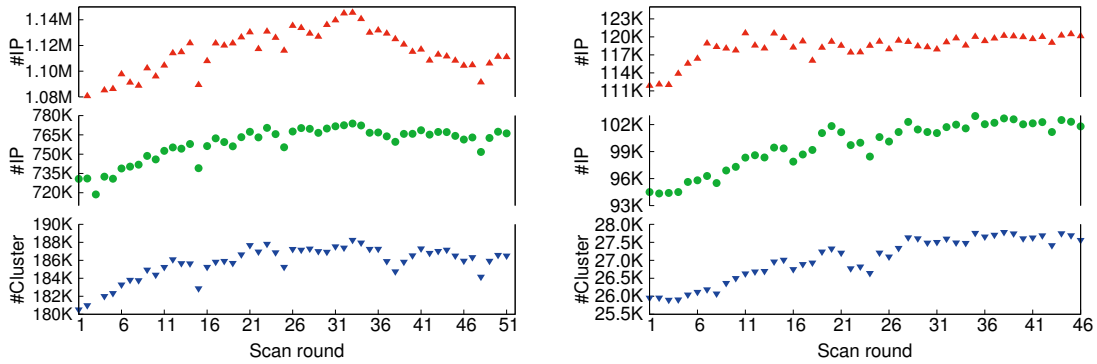

Figure 8: Change over time in responsive IPs, available IPs and clusters **(from top to bottom)** in **(left)** EC2 and **(right)** Azure. X-axis is the round of scanning and note that the Y-axes have gaps in the range covered.

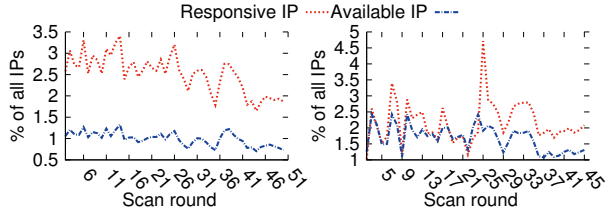

Figure 9: Change in status of responsive IPs and available IPs as a fraction of all IP addresses in **(left)** EC2 and **(right)** Azure.

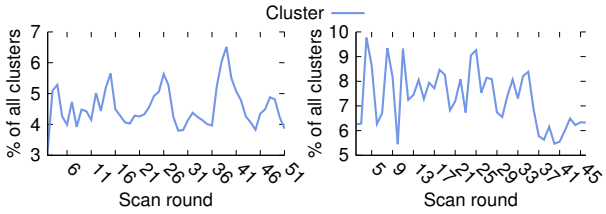

Figure 10: Change in availability of clusters as a percentage of the total number of clusters seen throughout the measurement period in **(left)** EC2 and **(right)** Azure.

10.1%, 4.0%, and 0.3% for EC2 and 8.7%, 6.8%, and 1.1% for Azure.

Even considering that these are just estimates (sources of error including incorrect clustering or transitory network failures), the per-round churn in these clouds appears quite small.

Cluster availability over time. A cluster is available in a given round if at least one of the IPs in the cluster is available in that round. Figure 10 plots for each round the fraction of clusters that changed from available to unavailable or vice versa, compared to the previous round, over the total number of unique clusters observed throughout the measurement period. On average across all rounds, the average percentage of cluster status changes is 4.6% on EC2 and 7.3% on Azure.

Cluster size over time. We now characterize the observed ways in which a cluster's size (the number of IPs associated to the cluster) changes over time. Table 11 shows the results of our analysis, in particular the top five size change patterns observed. Here a pattern of 0 means no change; 0,1,0 indicates an increasing trend; 0,-1,0 a decreasing trend; and so on. So about half of clusters are quite stable in terms of size. We now describe how we arrived at these patterns, and then discuss the results in further detail.

For each cluster, we first generate a vector D of the number of instances it used at each time point. So a vector of $(10, 3, 20)$ means that in time period one the cluster had 10 IPs associated to it, 3 in the next time period, etc. The dimension of the vector is the number of rounds of scanning. We reduce the dimensionality of the resulting vectors in order to more accurately compare and understand general usage patterns.

To do so we use piecewise aggregate approximation (PAA) [36], a method for reducing the dimensionality of time-series data. We divide the vector into frames and use the median value of a frame to represent the frame. Since the time interval of our data is not always constant, we could not simply use an equal number of data points in each frame. We use a 7-day window to generate frames. We use the 7-day time window because it can cover at least 3 data points and can catch more changes in IP usage. Each frame may contain different numbers of data points. For example, frame one may contain 3 data points from the first 7 days of scanning while frame two contains 4 data points from the next 7 days.

We therefore compute from D a new vector D' of dimension 13 for EC2 (since measurement lasts for 93 days) and 9 for Azure (62 days) whose values are the median of the values in each window. We use median so as to be robust in the face of outliers.

Using D', we then define a new vector D'', called the tendency vector, as shown in Algorithm 1. For example, and using smaller vector dimension for brevity, if $D' = (1, 2, 3, 1, 1, 1)$

pattern	description	# EC2 clusters	# Azure clusters
0	⎯	121,297 (49.9%)	17,092 (53.9%)
0,1,0	⟋	36,465 (15.0%)	4,406 (13.9%)
0,-1,0	⟍	33,400 (13.7%)	3,952 (12.5%)
0,1,0,-1,0	⎍	12,598 (5.2%)	1,189 (3.8%)
0,-1,1,0	⋁	9,857 (4.1%)	1,376 (4.3%)

Table 11: Top five size-change patterns observed in EC2 and Azure. Percentages in parentheses are with respect to total number of clusters.

Algorithm 1 Algorithm for computing tendency vector given D'

```
for 1 ≤ i ≤ |D'| − 1 do
    if D'[i + 1] > D'[i] then
        set D''[i] = 1
    else if D'[i + 1] = D'[i] then
        set D''[i] = 0
    else
        set D''[i] = −1
    end if
end for
```

then $D'' = (1, 1, -1, 0, 0)$, or if $D' = (1, 10, 0, 5, 4, 2)$ then $D'' = (1, -1, 1, -1, -1)$. Finally we merge repeat values in D'', so for example, $(0, 1, 1, 0, -1, -1)$ becomes $(0, 1, 0, -1)$. The resulting merged tendency vector is the size-change pattern. Again the top five such patterns observed are given in Table 11.

As previously noted, about half the clusters in EC2 and Azure, respectively, follow pattern 0 (no change in cluster size). Clusters in pattern 0 can be further split into two groups:

(1) Ephemeral clusters: the median number of IPs they used is zero, because they just appeared in the cloud for a very short period of time. In EC2, 11.4% of all clusters are ephemeral while in Azure 13.1% are ephemeral. Of ephemeral clusters, 92.8% only use 1 IP, 99.1% use less than 3 IPs, and 24 clusters use more than 10 IPs. We arbitrarily chose 10 of these last 24 clusters for manual investigation, which revealed a few explanations that may generalize more broadly: (a) a website was under maintenance so it temporarily changed the top-level webpage; (b) a website just moved into the cloud and was under construction, and we caught the in-development webpage; or (c) the webpage was developed by tenants who may temporarily use the cloud for testing and left the cloud soon.

(2) Relatively stable clusters: The other 38.5% of clusters in EC2 (40.8% in Azure) instead use nearly the same number of IPs in each round over the entire measurement period. In this group, 38,466 (15.8%) clusters in EC2 and 7,789 (24.5%) in Azure haven't changed the number of IPs they used during measurement periods and are available in each scan round. Most of these clusters (92.7%) use the exact same IP addresses all the time.

Pattern 0,-1,1,0 represents clusters that have a drop in the instance usage, immediately followed by an increase. A short-term unavailability can cause such a drop. We define the lifetime of a cluster as the time period between the first time a cluster is available and the last time it was available. The cluster uptime is the fraction of time the cluster is available relative to its lifetime. For example, if a cluster was first observed on one day, and last observed 10 days later, and no IP associated with it responded on one of the days in between, then the lifetime is 10 days and the uptime is 90%. In EC2 54.3%'s of the singleton clusters have 100% uptime, 89.1% have $\geq$90% uptime, and 92.7% have $\geq$80%. Of clusters of size 2, 86.4% have a 100% uptime. The cluster uptime of all the clusters with size greater than or equal to 18 is 100%. Clusters in Azure show a similar pattern — in general, the more IPs a cluster uses, the higher cluster uptime it has. The cluster IP time of only a small fraction of clusters are less than 90% (9.4% of clusters in EC2 and 10.6% in Azure).

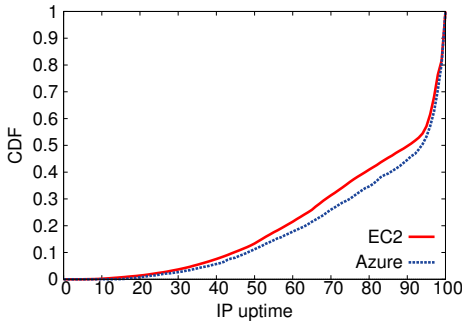

Figure 12: Distribution of average IP uptime of clusters of size two or larger.

Churn within clusters. We turn now to measuring the amount of turnover in IPs used by a given cluster. Website deployments may be designed to have relatively high turnover, taking advantage of cloud elasticity to scale with demand. For example, EC2 includes an auto-scaling feature, and several third party companies advertise similar services. Alternatively, churn may be indicative of instability in deployments such as VM instance restarts caused by crashes or other failures. Our measurements cannot distinguish between such causes of churn, but can give an overall estimate of the amount of churn.

We define the uptime of an IP for a given cluster as the fraction of days the IP is available and associated to the cluster divided by the total amount of time the cluster is available. The average IP uptime of a cluster is the average of IP uptimes for all IPs associated to the cluster. Average IP uptime therefore provides a measure of how much a cluster experiences churn within the IPs that it uses. If a cluster uses the same IP addresses all the time, then the average IP uptime of this cluster will be 100%. Low average IP uptime indicates the cluster frequently changes IPs.

A total of 183,119 (75.3%) clusters in EC2 and 25,037 (78.9%) in Azure have a 100% average IP uptime. The bulk of these are singleton clusters, meaning clusters of size one. In EC2, 71.4% of all clusters are singletons, and in Azure 76.2% are singletons. For non-singleton clusters, we find more evidence of churn. Figure 12 gives a CDF of the average IP uptimes across all clusters of size two or larger. We see that about half of the clusters have average IP uptime greater than 90%, in either cloud. A large fraction in fact (27.2% and 30.1% of clusters in EC2 and Azure, respectively) have average IP uptime between 95% and 99%, indicative of low churn. For the other half of clusters having below 90% average IP uptime, we see quite a spread of uptimes.

When the cluster size increases, the average IP uptime becomes smaller. For clusters with cluster size 50 or greater, the average IP uptime is 62%. For the top 10 websites by cluster size, about 90% of the IPs being used in one measurement round, are still used in the next round, but less than 50% of the IPs have been used for the entire 3 months.

Region and VPC usage. Prior work [2] showed that the vast majority of websites use a single region. Our measurements are inline with this: 97.0% of all clusters use a single region. Even among

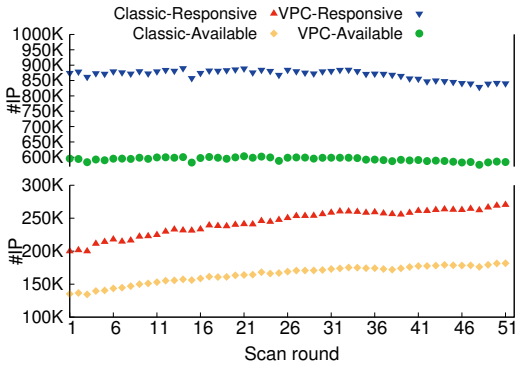

Figure 13: Change over time in responsive IPs, available IPs and clusters in EC2 VPC and classic instances. X-axis is the round of scanning. Note that Y-axes are different in each chart.

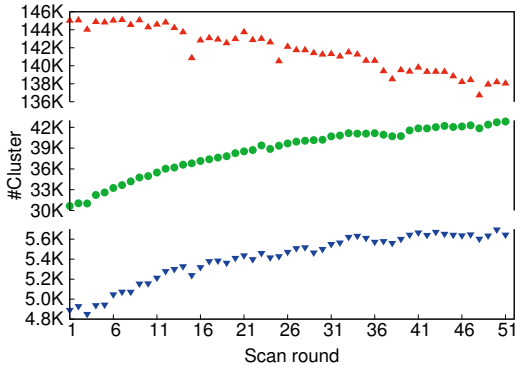

Figure 14: Change over time in classic-only, VPC-only, and mixed clusters **(from top to bottom)** in EC2.

the top 5% of all clusters (in terms of cluster size), only 21.5% use more than a single region. While most of the clusters use the same region(s) over time, a small number of them have a change in region usage. 98.37% of clusters use the same region(s), 0.7% use 1 extra region, 0.07% use 2 extra regions, 0.76% decrease by 1, and 0.07% decrease by 2 in EC2. In Azure, 97.33% use the same region(s), 1.28% use 1 extra region, 1.13% decreased by 1 and 0.09% decreased by 2.

For EC2, we can use the cartographic mechanisms described in §5 to examine the usage of VPC versus classic networking. This is particularly interesting as it shows the uptake of a newer feature, and one that is required for new accounts. Figure 13 shows a timeline of the number of responsive and available IPs as broken down by VPC/classic. We found in total 177,246 (72.9% of all clusters) use only classic IPs; 59,547 (24.5%) use only VPC IPs; and 6,371 (2.6%) use both classic and VPC IPs. As shown in Figure 14, the number of classic-only clusters is decreasing rapidly while VPC-only and mixed clusters are increasing. We also observed a total of 1,024 mixed clusters that transitioned from classic to VPC while 483 went in the other direction.

Large clusters. We found a number of very large clusters within the collected data. Table 15 gives details regarding the top ten clusters in EC2, by average cluster size (the clusters in Azure have relatively smaller sizes compared with those in EC2, so we did not examine them in detail). We manually investigated these ten clusters, and provide a categorization of the type of industry they are associated to. These clusters show a diversity of usage dynamics.

In the table, the total number of unique IPs associated to the cluster across the entire measurement period is "Total #IP", while "Mean/Median/Min/Max #IP" indicate the basic statistics of number of IPs used by a cluster across all rounds. The average IP uptime of these clusters varies greatly, and is generally lower than the uptime of smaller clusters. As noted previously, most IPs are stable for short periods of time, meaning if they are used by the cluster in one round of measurement they are likely to be used in the next. The column "Max IP departure" gives the maximum, across all rounds, of IPs leaving a cluster as a fraction of the number of IPs used by the cluster in that round. An IP leaves a cluster in a given round if it was not associated with the cluster in that round, but was in the previous round. The column "Stable IP" gives the ratio of the number of IPs that were used throughout the entire measurement period by the cluster over the total number of unique IPs used by the cluster. As can be seen, long-run stability is low except for cluster 9. Finally, we give the number of regions used and the average number of VPC IPs per round for each cluster.

8.2 Finding and Analyzing Malicious Activity

We turn to using WhoWas to investigate malicious activity on IaaS clouds. We use two sources for flagging malicious behavior, Google Safe Browsing and VirusTotal. We will show that WhoWas can help investigate, as well as discover new, malicious activity. We will also quantify the time between a website appearing and it being marked as malicious and the time an IP address remains malicious after being dropped from a blacklist.

Google Safe Browsing. The Google Safe Browsing API allows one to submit a URL and obtain a status report indicating a status of either "phishing", "malware" (hosting malicious software), or "ok". To make use of the API, we extracted all the URLs contained in top-level webpages gathered by WhoWas and queried each against the safe browsing API after each round of scanning. In each round, there were about 3.2 M distinct URLs across the entire set of pages fetched, only a small fraction of which end up marked as malicious by Safe Browsing.

For EC2, we found 196 unique IPs (149 in classic and 47 in VPC) that contain at least one link labeled as phishing or malware, for a total of 1,393 distinct malicious URLs. Many of the webpages contain multiple malicious URLs: nine had a total of 63 URLs marked as phishing and 187 of them had a total of 1,330 URLs marked as malware. These 196 IPs are associated to 51 distinct clusters. A total of 19 IPs hosting malicious links were associated to different clusters (over time), but upon manual inspection of the webpages it was clear that the different clusters were in fact just revisions of the same, significant enough to push them into a distinct cluster. All but 18 of the IPs that the malicious URLs resolved to are outside EC2. Interestingly, one IP address removed the malicious URL around 3 days after being detected, changed the webpage to include a different malicious URL, and then changed it later to again include the first malicious URL. The webpage on this IP in fact appears to be legitimate (upon manual inspection), so it could be that the hosting server is compromised.

For Azure, we found 14 malicious URLs hosted on 13 distinct IPs that are associated to 11 clusters. All of these URLs are malware URLs. The malicious URLs were resolved to 19 IPs, and 7 of these IPs are in Azure. We did not find any malicious URLs that were linked to by webpages in both clouds.

Below we say that an IP is malicious if it hosts a webpage including a malicious URL. We define the lifetime of a malicious IP as the number of days that an IP address has a malicious URL within its

Cluster	Total #IP	Mean #IP	Median #IP	Min #IP	Max #IP	Avg IP uptime (%)	Max IP departure (%)	Stable IP (%)	Regions used	Mean #VPC IP	Category
1	51,211	33,145	33,513	30,624	34,509	73.8	6.6	49.9	2	1	PaaS
2	15,283	5,597	5,601	5,435	5,785	49.1	8.1	22.2	8	1,344	Cloud hosting
3	3,869	2,029	2,050	1,724	2,228	65.9	8.9	36.2	8	1,346	VPN
4	22,226	1,167	1,077	179	2,501	13.1	85.4	0.1	6	5	SaaS
5	8,488	617	403	57	1,836	14.1	81.3	0.1	1	0	Game
6	919	529	625	169	738	78.1	62.0	1.4	1	0	Shopping
7	1,928	370	320	141	622	26.2	86.3	1.6	1	0	PaaS
8	1,207	366	367	338	419	50.1	18.7	14.3	2	366	Video
9	303	281	285	269	291	94.5	6.3	89.1	1	0	Marketing
10	3,263	255	253	132	478	16.6	68.3	0.6	5	0	Cloud hosting

Table 15: Top 10 websites based on average number of IPs used per round, all in EC2.

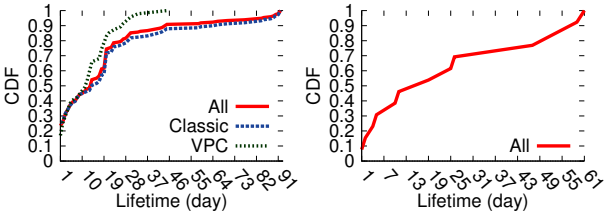

Figure 16: CDF of lifetimes of IPs hosting malicious content as indicated by Google Safe Browsing in **(left)** EC2 and **(right)** Azure.

HTML. We find the lifetime of malicious IPs to be relatively long; see Figure 16. In EC2, 62% of malicious IPs host malicious URLs for more than 7 days, and 46% host malicious URLs for over 14 days. The lifetime of malicious IPs in EC2-VPC is slightly shorter than in EC2-Classic. The longest availability of a malicious IP was 45 days in VPC and 93 days (the complete measurements period) for classic. In Azure, about 70% of malicious IPs host malicious URLs for more than 7 days, and 50% for at least 14 days.

We also find some IPs appear to serve as a "linchpin" for maliciousness. A linchpin IP hosts a webpage that aggregates many malicious URLs [37]. For example, we found at one point a particular IP address on EC2 contained 128 malware URLs (associated with different domains); all of these URLs linked to webpages that contained the Blackhole Exploit kit [38]. In our dataset, we identify five linchpin IPs that all host the same webpage. The malicious URLs in these webpages point to 23 different domains. The average lifetime of the IPs that host these malicious URLs is 20 days.

VirusTotal. VirusTotal aggregates information from multiple engines that detect malicious activity (e.g., AV products and blacklists). One service VirusTotal provides is to query an IP address to obtain a JSON report of malicious activity associated with the IP. Each report contains passive DNS information for the IP, and for each engine that detected malicious activity on that IP, the time and type of the malicious activity, e.g, malicious URLs and MD5 hashes of malware hosted on the IP. During Feb. 2014, we used this service to collect reports on all the IPs in EC2 and Azure.

VirusTotal aggregates information from the multiple engines without validating their detections, thus it is possible that an engine may (incorrectly) consider an IP address as malicious, when other engines do not. To reduce false positives, we only consider an IP address malicious when it is reported as malicious by two or more engines. Even with this constraint there could still be some false positives, which will be inherited by the following analysis.

After applying the above constraint, we find that from Sep. 30, 2013 to Dec. 31, 2013, there were 2,070 malicious IPs in EC2, hosting 13,752 malicious URLs. The malicious IPs represent 0.3% of the average available IPs in EC2. In contrast, no malicious IPs were found in Azure. Table 17 breaks down the malicious EC2 IPs by region and month, showing an overall increase of malicious activity over time.

	# Malicious IP			
Region	**Oct**	**Nov**	**Dec**	**Total**
USEast	544	847	728	1,422
EU	60	106	107	200
USWest_Oregon	50	94	110	192
USWest_NC	18	56	44	91
SouthAmerica	8	28	32	57
AsiaSingapore	13	26	22	51
AsiaTokyo	7	19	23	35
AsiaSydney	5	14	6	22

Table 17: Number of IPs in each EC2 region labeled as malicious by two or more engines in VirusTotal.

Domain	# URLs
dl.dropboxusercontent.com	993
dl.dropbox.com	936
download-instantly.com	295
tr.im	268
www.wishdownload.com	223
dlp.playmediaplayer.com	206
www.extrimdownloadmanager.com	128
dlp.123mediaplayer.com	122
install.fusioninstall.com	120
www.1disk.cn	119

Table 18: Top 10 domains associated with malicious IPs in EC2 by VirusTotal.

We extract 2,888 unique domains from the 13,752 malicious URLs. Of these, 222 (7.7%) contain the substring "download" in their domain names. Table 18 shows the top 10 domains by number of URLs that include the domain. The results show that Dropbox (which runs on EC2) and other file hosting providers have become a popular platform for distributing malware. The VirusTotal reports show that 371 IPs belonging to Dropbox have been used to distribute malicious content (the domains of these IPs all resolved to "dl.dropboxusercontent.com" or "dl.dropbox.com".) These 371 IPs hosted 1,929 malicious URLs and 2,421 pieces of malware. Tr.im provides a URL-shortening service [39] that attackers sometimes use to hide malicious URLs. We observe some engines only detecting the shortened URL, so the malicious page remains accessible via the malicious long URL. The remaining URLs in the top 10 correspond to sites distributing adware and other potentially unwanted software.

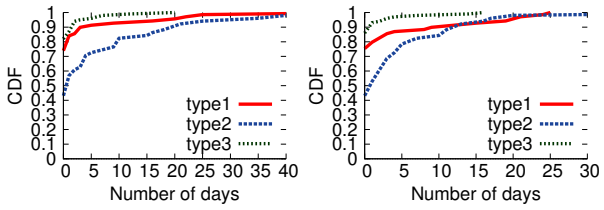

Figure 19: CDFs of the number of days an IP address hosts the same webpage **(left)** before first being labeled as malicious by VirusTotal and **(right)** after last being detected as malicious by VirusTotal.

WhoWas. We use WhoWas to analyze the content hosted over time by the malicious EC2 IPs obtained from VirusTotal. Only 98 of the malicious EC2 IP addresses appear in one of the final clusters in our analysis. The others return default webpages or errors. These 98 IPs are associated to 166 different clusters: 63 IPs are associated to only one cluster, 21 to two clusters, and five to three clusters. Two of the IPs are associated to 19 clusters. Among these 98 IPs, we observe three types of malicious behaviors over time. The first type corresponds to 34 IPs that host the same malicious webpage without any changes over time. The second type are 42 IPs where the same malicious webpage repeatedly appears and disappears. Once detected, the malicious webpage is temporarily removed from the IP, but returns several days later. The third type corresponds to 22 IPs that host multiple malicious webpages.

Next, we examine the lag time of the blacklists, defined to be the time a blacklist takes to detect a malicious website after it goes up. The engines in VirusTotal may scan an IP multiple times, generating multiple records, which may be associated with different malicious webpages. For our purposes, there are two import time points for any given webpage: (1) the first detection time, which is the first time an engine labels a webpage on an IP as malicious, and (2) the last detection time, which is the last time the malicious webpage on that IP was reported as being malicious by an engine. The last detection time also indicates when the engines believe the webpage has become benign.

Using WhoWas, we compare the lifetime of a cluster associated with a malicious IP address to those two time points. Figure 19 shows CDFs of the time between a webpage becoming available and being marked as malicious by the blacklist (left graph), and the time after being removed from blacklists that a malicious webpage remains active (right graph). Overall, about 90% of type 1 and type 3 webpages, and 50% of type 2 are detected within three days. Most of the type 1 and type 3 webpages are removed after the last detection. But in type 2, only about 40% of webpages are actually removed. There are some webpages that have a very long lifetime. For example, a particular type 1 IP hosted malware for at least 82 days (at the time of writing it is still available, in fact). Anecdotally, we also found that it frequently changes its domain name, presumably to avoid detection.

We can also use the clustering mechanism of WhoWas to find new malicious IP addresses, by labeling as malicious IP addresses in a cluster with a VirusTotal-labeled malicious IP address. In this way, we label an additional 191 IPs. As a breakdown, 15 VirusTotal-labeled IPs each had one additional IP; 10 IPs had 2 additional IPs, 6 had 3, and 16 had 4–13. We manually inspected the WhoWas records for these additional IPs, verifying the correctness of the approach in every case. All this shows that WhoWas can help find malicious activity.

8.3 Characterizing the Web Software Ecosystem on Clouds

As a final case study to exercise WhoWas, we study the software ecosystem of web services running the cloud. This includes the a census covering the kinds of web servers, back-end software languages, website templates, and third party-trackers running on top of EC2 and Azure.

In the rest of this section all the percentages reported are averages taken over all rounds of measurement; we omit repeatedly referring to them as averages for brevity.

Web servers and backend techniques. For EC2, we identified the server software running on 89.9% (682 K on average) of the available IPs. Apache (55.2%), nginx (21.2%), and Microsoft-IIS (12.2%) are the most popular web servers, accounting for over 88.6% of the identified servers. MochiWeb is also used (30,372 servers on average, 4.4%), but the servers running it are only associated with 47 clusters in total. Its use is almost entirely accounted for by a single PaaS provider. We further identify the backend techniques that support about 32% (215 K on average) of servers. Of these servers, 52.6% are using PHP, followed by ASP.NET (29.0%) and Phusion Passenger (8.1%). For the server usage and backend techniques, we did not see any significant difference between VPC and classic.

In Azure, we find, perhaps unsurprisingly, that the software ecosystem displays less diversity. Microsoft-IIS runs on 89% (about 86 K on average) of the identified servers in Azure. Only about 7.6 K IPs are running Apache and 1.7 K are running nginx, accounting for a total of 9.7% of identified servers. For back-ends in Azure, we found that 94.2% are using ASP.net, followed by PHP (4.3%) and Express (0.6%).

Examining the versions of commonly used servers, we find new versions of servers are not being adopted quickly. Of the Apache servers in EC2, 24.6% are using Apache/2.2.22, followed by Apache-Coyote/1.1 (15.0%) and Apache/2.2.25 (7.6%). More than 40% of identified Apache servers are using Apache/2.2.*. A few IPs (208 in total) in EC2 use Apache 1.3.*. The most recent version of Apache used in EC2 is Apache 2.4.7, and this is used by 147 IPs in the last round of measurement (the highest measured). Surprisingly, some of the websites that employ a large number of IPs are still using Apache/2.2.22.

In Azure, the top three IIS releases being used are 8.0 (39.0%), 7.5 (23.7%), and 7.0 (19.8%). Only 3.4% are using IIS 8.5.

The Security Engineering Research Term (SERT) reported the ten most vulnerable server versions as determined by a large-scale scan [40]. We find seven of the ten vulnerable servers are also being used in both clouds. For example, in EC2 in total across all rounds of measurement 2,641 IPs are running Microsoft-IIS/6.0, and 56 IPs are running ‘‘Apache/2.2.24 (Unix) mod_ssl/2.2.24 OpenSSL/1.0.0-fips mod_auth_passthrough/2.1 mod_bwlimited/1.4 FrontPage/5.0.2.2635’’. Most servers also use old releases of PHP. Of the servers that are using PHP in EC2, 60% are using PHP 5.3.*. PHP 5.3.10 (24.5%), 5.3.27 (16.22%) and 5.3.3 (9.7%) are the 3 top used PHP releases. The most recent version of PHP used in EC2 is PHP 5.4.23, used by 845 servers in the last round of measurement.

Website templates. We identify the template of webpages on an average of about 26 K IPs (3% of available IPs) running on EC2. The top 3 templates being used are: WordPress (71.1%), Joomla! (9.7%) and Drupal (4.1%). In Azure, we only find templates for an average of about 950 webpages. WordPress (55%) and Joomla! (12%) are still the most popular templates in this smaller set.

WordPress versions below 3.6 contain a series of known cross-site scripting (XSS) vulnerabilities [41]. More than 68% of the sites using WordPress are running vulnerable versions of WordPress. Checking the distribution of WordPress major versions we observed from Sep. 30, 2013 to Dec. 31, 2013, WordPress 3.5.* (6.5 K on average) and 3.6.* (5 K on average) are used by most of the identified WordPress-using webpages. We saw that one month after WordPress 3.7.* (released on Oct. 14, 2013) and 3.8.* (released on Dec. 12, 2013) were released, around 3,500 webpages and 2,500 webpages on average are using the new WordPress version, respectively, but 3.5.* and 3.6.* are still the most popular WordPress releases.

Third-party trackers. Many third party trackers work by having site owners embed a specific piece of JavaScript, called tracking code, into their webpage. We can determine if a website uses a given tracker by searching for the tracker's corresponding tracking code in the page's HTML contents. This method will miss trackers loaded dynamically or in the case that the code is modified by the site operator. Tracking code always contains a special URL which can serve as a fingerprint. Using MySQL regular expressions to search these fingerprints in the WhoWas database provides a lower bound on the number of websites using a given tracker. As a concrete example, for the tracker associated with scorecardresearch.com, we construct a query string

''<script.*http:\\/\\/b\\.scorecardresearch \\.com.*\\/script>''

which means search for ''http://b.scorecardresearch.com'' in the JavaScript section of a page. The URL is the link to the tracker's service extracted from its tracking code.

We search for use of the trackers previously identified by Mayer et al. [42]. Table 20 shows the Top 10 trackers used by websites in EC2 and Azure on Dec. 31, 2013 (the last round of our measurements). In both clouds, Google-Analytics, Facebook and Twitter sharing widgets, and doubleclick are the most widely used third-party trackers.

The Google Analytics service is the most popular tracker being used in EC2-hosted and Azure-hosted websites. In EC2, we can see on average 55 K clusters use this service. Using the service requires a user to apply for a Google Analytics ID and put that ID in her tracking code. We extract a total of 71,363 unique IDs from the webpages over 275,513 unique IPs in EC2. These IDs belong to 106,871 top-level clusters and 111,696 webpage clusters. In Azure, we collect 6,254 IDs from 11,840 IPs. We also find 166 IDs appeared in both clouds. According to Google [43], a Google Analytics ID can be split into two parts: user account and profile identifier. For example, in an ID "UA-0000-*", the "0000" is an ID for the user. The user might use "UA-0000-01", "UA-0000-02", and so on with different websites or applications. By checking IDs we can quickly estimate a lower bound on the number of unique websites in the two clouds, as well as the number of websites that belong to the same user. In EC2, the collected IDs belonging to 64,716 user accounts. Of these accounts, 93.5% use only one profile, 4.8% have two profiles, 1.6% have 3–11 profiles, and the rest (0.1%) use 14–35 profiles. In Azure, the IDs belong to 5,794 user accounts, 94.4% have one profile and 3.9% have two. An account in Azure is at most associated with 13 profiles.

Websites may use multiple trackers. We observed 77% of tracker-using webpages only use a single tracker, 16% use two trackers and 6% use three trackers in EC2. In Azure, 81% of tracker-using webpages use a single tracker, 15% use two trackers and 3.8% use three trackers.

EC2			Azure		
Tracker	**#IP**	**#Clust.**	**Tracker**	**#IP**	**#Clust.**
google-analytics	127,604	55,406	google-analytics	6,839	5,148
facebook	24,130	13,462	facebook	1,615	1,253
twitter	14,706	8,520	twitter	1,115	696
doubleclick	5,342	2,189	doubleclick	316	178
quantserve	2,243	742	atdmt	50	31
scorecardresearch	1,509	475	scorecardresearch	40	29
imrworldwide	474	95	quantserve	35	25
serving-sys	383	136	serving-sys	11	7
atdmt	275	69	adnxs	10	6
yieldmanager	188	54	imrworldwide	7	4
Total	176,852	81,147	Total	10,037	7,376

Table 20: Top 10 third party trackers in EC2 and Azure (on Dec. 31, 2013)

9. CONCLUSION

A growing number of web services are being hosted on public IaaS clouds such as EC2 and Azure. These cloud providers offer a variety of features, such as elastic scale in/out and pay-as-you-go payment models. Understanding how different tenants exercise these features can inform a variety of work on workload modeling, data center compute/network provisioning and placement, as well as on securing tenants' deployments. Unfortunately, however, very little is known about prevalent usage patterns and their evolution over time.

To shed light on this, we designed a measurement platform called WhoWas. Using carefully designed active probes, WhoWas enables direct measurements of what web services are running on cloud-associated IP addresses. This data can be used to perform lookups on particular IP addresses or web services, and retrieve a history of their activities over the period of measurement.

This paper presented the design of WhoWas and reported on using WhoWas to perform a first-of-its-kind measurement studies of usage patterns over time in EC2 and Azure, efficacy of blacklists for malicious activities in clouds, and adoption of new Web software by cloud customers.

Future work may include expanding WhoWas to analyze non-web services, evaluate improved clustering heuristics and provide more rigorous estimates of clustering error, provide deeper crawling of websites by following links in HTML, and correlate WhoWas data with other sources such as passive or active DNS interrogation. To facilitate such work, we have made the software underlying WhoWas public and open source [3].

Acknowledgements

The authors would like to especially thank the system administrators at the University of Wisconsin who helped ensure the experiments went smoothly. This work was supported in part by the National Science Foundation under grants CNS-1253870 and CNS-1330308. This research is also partially supported by the Spanish Government through Grant TIN2012-39391-C04-01 and a Juan de la Cierva Fellowship for Juan Caballero. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

10. REFERENCES

[1] Alexa. Alexa. `http://alexa.com/`, 2014.

[2] Keqiang He, Alexis Fisher, Liang Wang, Aaron Gember, Aditya Akella, and Thomas Ristenpart. Next stop, the cloud: understanding

modern web service deployment in EC2 and Azure. In *IMC 2013*, pages 177–190. ACM, 2013.

[3] Whowas project. http://www.cloudwhowas.org/.

[4] Amazon. Amazon EC2 instance IP addressing. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html#MultipleIP, 2013.

[5] Farsight Security. Passive DNS data. https://www.dnsdb.info/, 2014.

[6] Wayback machine. http://archive.org/web/.

[7] Douglas E. Comer and John C. Lin. Probing TCP implementations. In *USENIX Summer Technical Conference*, Boston, MA, June 1994.

[8] Niels Provos and Peter Honeyman. Scanssh - scanning the internet for ssh servers. Technical Report CITI TR 01-13, University of Michigan, October 2001.

[9] Darcy Benoit and André Trudel. World's first web census. *International Journal of Web Information System*, 3, 2007.

[10] David Dagon, Chris Lee, Wenke Lee, and Niels Provos. Corrupted Dns resolution paths: The rise of a malicious resolution authority. In *Network and Distributed System Security Symposium*, 2008.

[11] Nadia Heninger, Zagir Durumeric, Eric Wustrow, and J.Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *USENIX Security Symposium*, 2012.

[12] Antonio Nappa, Zhaoyan Xu, M. Zubair Rafique, Juan Caballero, and Guofei Gu. Cyberprobe: Towards internet-scale active detection of malicious servers. In *Network and Distributed System Security Symposium*, 2014.

[13] Zhaoyan Xu, Antonio Nappa, Robert Baykov, Guangliang Yang, Juan Caballero, and Guofei Gu. AutoProbe: Towards Automatic Active Malicious Server Probing Using Dynamic Binary Analysis. In *Proceedings of the 21st ACM Conference on Computer and Communication Security*, Scottsdale, AZ, November 2014.

[14] Zhaoyan Xu, Lingfen Chen, Guofei Gu, and Christopher Kruegel. Peerpress: Utilizing enemies' p2p strength against them. In *ACM Conference on Computer and Communications Security*, 2012.

[15] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the internet in your spare time. In *USENIX Security Symposium*, San Francisco, CA, August 2002.

[16] Luigi Rizzo. Netmap: A novel framework for fast packet i/o. In *usenixatc*, 2012.

[17] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *Proceedings of the 22nd USENIX Security Symposium*, August 2013.

[18] D. Leonard and D. Loguinov. Demystifying service discovery: Implementing an internet-wide scanner. In *Internet Measurement Conference*. ACM, 2010.

[19] M. Allman, V. Paxson, and J. Terrell. A brief history of scanning. In *Internet Measurement Conference*. ACM, 2007.

[20] Marco Balduzzi, Jonas Zaddach, Davide Balzarotti, Engin Kirda, and Sergio Loureiro. A security analysis of amazon's elastic compute cloud service. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1427–1434. ACM, 2012.

[21] Davide Canali, Davide Balzarotti, and Aurelien Francillon. The Role of Web Hosting Providers in Detecting Compromised Websites. In *Proceedings of the 22nd World Wide Web Conference*, 2013.

[22] Antonio Nappa, M. Zubair Rafique, and Juan Caballero. Driving in the cloud: An analysis of drive-by download operations and abuse reporting. In *SIG SIDAR Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, 2013.

[23] Amazon. Amazon EC2 public IP ranges. https://forums.aws.amazon.com/ann.jspa?annID=1701, 2013.

[24] Microsoft. Azure datacenter IP ranges. http://msdn.microsoft.com/en-us/library/azure/dn175718.aspx, 2013.

[25] Kenneth Reitz. Requests: Http for humans. http://docs.python-requests.org/en/latest/, 2014.

[26] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[27] Monika Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291. ACM, 2006.

[28] Mat Kelcey. The simhash algorithm. http://matpalm.com/resemblance/simhash/, 2013.

[29] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[30] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *WWW 2007*, pages 141–150. ACM, 2007.

[31] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS 2009*, pages 199–212. ACM, 2009.

[32] Amazon. Public IP addresses and external DNS hostnames. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html, 2014.

[33] Huan Liu. Amazon data center size. http://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/, 2012.

[34] Robot exclusion. http://www.robotstxt.org/faq/prevent.html.

[35] Anonymization tools taxonomy. http://www.caida.org/tools/taxonomy/anonymization.xml.

[36] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[37] Zhou Li, Sumayah Alrwais, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *S&P 2013*, pages 112–126. IEEE, 2013.

[38] Fraser Howard. Blackhole exploit kit analysis. http://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit/, 2012.

[39] tr.im. tr.im. http://tr.im, 2013.

[40] SERT. Sert quarterly threat intelligence report q4 2013. Technical report, Solutionary Security Engineering Research Team, 2013.

[41] CVE. Cve-2013-4338. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-4338, 2013.

[42] Jonathan R Mayer and John C Mitchell. Third-party web tracking: Policy and technology. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 413–427. IEEE, 2012.

[43] Google. Accounts and views. https://developers.google.com/analytics/resources/concepts/gaConceptsAccounts, 2013.