Using Honeybuckets to Characterize Cloud Storage Scanning in the Wild

Katherine Izhikevich
UC San Diego

Geoffrey M. Voelker
UC San Diego

Stefan Savage
UC San Diego

Liz Izhikevich
Stanford University

***Abstract*—In this work, we analyze to what extent actors target poorly-secured cloud storage buckets for attack. We deployed hundreds of AWS S3 honeybuckets with different names and content to lure and measure different scanning strategies. Actors exhibited clear preferences for scanning buckets that appeared to belong to organizations, especially commercial entities in the technology sector with a vulnerability disclosure program. Actors continuously engaged with the content of buckets by downloading, uploading, and deleting files. Most alarmingly, we recorded multiple instances in which malicious actors downloaded, read, and understood a document from our honeybucket, leading them to attempt to gain unauthorized server access.**

1. Introduction

This paper explores a simple but poorly understood question: *to what extent is insecure cloud storage actively targeted for attack?*

Storage in the cloud, such as Amazon Simple Storage Service (S3) and Google Cloud Storage, provides reliable, available, and elastic storage on demand to virtually anyone with the means to pay. Moreover, it is extremely easy to deploy. A client need only choose a name for their storage "bucket" and specify its access control list, before it is ready to serve files. This ease of use has made such services extremely popular; in 2021, Amazon's S3 service hosted more than 100 trillion files on behalf of its users [1]. However, this same flexibility has given rise to new risks. The confidentiality of each bucket is not governed by traditional enterprise security mechanisms, but by the correct configuration of individual access control settings by the bucket operator. Thus, if a bucket is misconfigured to be public, then any party guessing its *name* may gain access to all of its sensitive content. In 2018, Ero [2] found that unsolicited parties were indeed guessing the names of public storage buckets. Shortly after, a survey of almost 200,000 buckets revealed that 10% contained sensitive data, including passports and financial records [3].

However, it remains unclear how scanners find such buckets and how many of these are targeted in actual attacks. To put it another way: is this merely an abstract risk, or are concrete threat actors actively searching for such vulnerabilities and exploiting them in the wild? While more than a few high profile breaches have been publicly attributed to misconfigured cloud storage [4]–[9], none have documented how these attacks took place, nor the mechanism by which attackers identified the opportunity. Indeed, scanning for buckets is non-trivial as an attacker must correctly guess the bucket's full name, yet the potential search space of these names is 10^{62} times larger than IPv6 and no public repository of buckets-in-use exists. The names of misconfigured buckets must therefore either be guessed, or found in an unrelated passive data source (e.g., DNS).

In this work, we empirically analyze this question by deploying a range of "honeybuckets" on the AWS S3 platform, configured with names, permissions, and content to lure and measure different scanning strategies. By modulating how our buckets are named and whether they are leaked to other data sources, we have been able to identify the most widely used strategies employed by third parties to scan for misconfigured buckets. We identify that there are clear preferences for scanning particular kinds of organizations, notably commercial entities in the technology sector. Moreover, while we find that it is common for all such actors to hide behind proxy servers, we show how to automatically group seemingly disparate IP addresses by dynamically modulating filename content to create causal dependencies between metadata and attempts to access individual files.

To distinguish between potential benign actors who may be scanning to help notify vulnerable parties [10] (or at least to try to sell a subscription to such a security scanning service) and those who have malicious aims, we further configured our honeybuckets to create multiple opportunities for actors to engage in clearly malicious acts. First, we configure our buckets to allow actors to delete or upload data. We consider users who delete data or upload content that is designed to gain unauthorized access (i.e., to compromise a user who interacts with it and spawn a reverse shell) to be malicious. Second, we create lures whose value requires one to affirmatively violate a security norm (i.e., an unauthorized login to a third-party server exploiting an ssh credential extracted from one of our buckets). We show that all of these behaviors occur in our data.

In summary, we provide strong empirical evidence that shows how unsecured data on cloud storage is exploited today. While many targeted scans may reflect benign security interests, there is a range of malicious activity targeting commercial data. We conclude with a range of recommendations for how organizations might better protect such cloud assets in practice.

2. Background and Related Works

Cloud storage mimics a traditional file system interface. Files are stored in a file directory structure, with the top level directory referred to as a "bucket." Buckets are simple to create: a client must, at minimum, (1) choose a service-wide globally unique name that is 3–64 alphanumeric-symbolic characters long and (2) configure the bucket contents to be private (the default option)

or publicly accessible. Once a bucket is created, it is accessible via the cloud provider's API, cloud browser interface, or through a cloud-specific subdomain (e.g., mybucket.s3.amazonaws.com). A user with sufficient access can upload an unlimited number of files and delete files, among other bucket operations [11].

Once a bucket is accessible (i.e., its access controls allow public access), a third-party must know its *name* to access it. There is no mechanism for enumerating names, nor any public repository of bucket names (public or otherwise). Thus, any party who does not possess *a priori* knowledge of a bucket's name must either guess it—from a potential namespace of roughly 10^{101} possibilities—or find it in another data source.

An attacker seeking to narrow the search space for guessing such names—-to find and sell stolen data, for example [5]—can use one of several scanning methodologies. The simplest bucket scanners rely on a pre-defined list of strings and patterns (e.g., Slurp [12], s3enum [13], and BucketStream [14]), tailored towards targeting likely-popular bucket names. Similar to password guessing software, bucket scanners generate target names by mechanically combining dictionary words, technology terms, etc.

However, not all bucket names will be simple or use popular words—for example, a number of applications generate random bucket names (e.g., project IDs) for intermediate storage [15]. Continella *et al.* show that some such names can be identified using passive DNS collection services [16] (i.e., where another party exposes the name via their DNS lookups) and Cable *et al.* generalize this idea by showing that such data can be used to train a machine learning model to generate a large set of valid names [3].

Such approaches have been used by the research community to explore the storage bucket ecosystem and empirically establish the widespread existence of misconfigured buckets. Indeed, both Continella *et al.* [16] and Cable *et al.* [3] found thousands of public buckets that exposed sensitive data, including private keys and national defense documents. Ero [2] and Cable *et al.* further deployed empty storage buckets to establish the existence of online bucket scanning behavior. However, while both found evidence of unsolicited scans, neither investigated the method by which names were targeted, nor the actions taken once such buckets were found. We are the first to demonstrate how public storage buckets are attacked (i.e., when actors delete, modify, upload, and exploit content) in the wild.

Overall, attacks on cloud storage have been relatively understudied, especially when compared to the broad literature characterizing active attacks on other infrastructure namespaces such as the IPv4 address space [17]–[19], IPv6 address space [20]–[22], cloud compute services [23]–[26], DNS [27]–[29], and BGP [30]–[33]. In these other environments, researchers have also used honeypots—infrastructure deployed for the purpose of detecting malicious behavior [34]—to characterize IPv4 scanning [35]–[37], email security [38], leaked document activity [39], and DNS activity [40]. Our work brings this approach to the cloud storage context, particularly in service to understanding more about the nature of unsolicited visitors: how they target victims and the extent to which they reveal clear malicious intent.

3. Pilot study: How Buckets Are Targeted

This section describes our pilot experiment to broadly understand how, and to what extent, buckets are targeted for attack. Using a deployment of more than one hundred honeybuckets over a period of six months, we show that exploitation of misconfigured buckets is very real: hundreds of IP addresses attempted to download, delete, or upload objects including malicious shell scripts. Notably, the first honeybucket was scanned only 40 minutes after deployment. Buckets named after companies, universities, and government organizations were scanned and attacked the most, implying that actors intentionally scan for specific targets. We then use the results of the pilot study to inform a refined experiment in Section 4.

3.1. Methodology

To understand how actors scan and interact with sensitive bucket content, we deployed 112 unique buckets (i.e., "honeybuckets") on the AWS S3 platform on February 18, 2022, and hosted them for 6 continuous months. We configured the bucket names, permissions, and contents to attract bucket-scanning actors that use various scanning strategies to find public buckets. We focused only on the AWS S3 platform, as prior work found that AWS S3 receives the largest amount of unsolicited scanning traffic [3]. We do not use AWS decoy resources [41], as these decoys neither provide bucket naming schemes nor sample content.

3.1.1. Bucket Names. Scanning buckets is not a trivial task. The enormous search-space of bucket names is too large to exhaustively enumerate. Thus, actors must employ directed strategies for scanning for public buckets. We used five approaches to name buckets, each designed to measure a unique bucket-scanning strategy. We list all bucket names in the Appendix in Table 11.

Target Generated Buckets. To increase the success rate of finding public buckets, actors can use an open-source tool to discover commonly-named S3 buckets. These tools generate their scanning targets using target generation algorithms (TGAs), a method that uses a pre-defined list of strings and patterns—often common bucket names—and optionally concatenates them with a user-provided list of keywords.

To measure if actors use open-source tools to find buckets, we named 12 of the 112 buckets with names generated by three of the most popular bucket enumeration tools found on Github and Pastebin—Slurp [12], DNSpop [42], and bucket-stream-permutation-feature [14].[1] To identify an actor's use of a particular enumeration tool, we computed the disjoint set of target names that belong to each tool—and not to any other—to create a set of honeybucket names that are likely to be found by one—and only one—tool. We chose four unique bucket names from each disjoint set, for a total of 12 buckets covering the three tools. To act as a control group, we created four additional buckets named with strings that

1. We also considered s3enum [13] and s3Mining [43], but after comparing outputs we found that their generated bucket names were proper subsets of the other TGAs.

did not appear in any of the bucket enumeration tools. Crucially, all chosen target generated bucket names were a part of a fixed set of names that the TGA program constructs (i.e., concatenates with an empty string), no matter if the user provides the keyword; it is not clear if this is a bug or a feature of the TGAs.

Company, University, and Government Buckets. Rather than only scanning for popular bucket names, an actor might curate a list of names that target a specific entity. To detect whether actors explicitly search for buckets named after organizations, we named 48 of the 112 buckets after the names of companies (Tesla, Walmart, Tinder), government organizations (FBI, CIA, NYPD) and universities (UCSD and Stanford). We concatenated each of the eight organizations with six unique keywords that appeared in a subset of the bucket enumeration tools. The six unique keywords consisted of three sensitive ("hidden", "private", "security") and three non-sensitive ("production", "download", "public") keywords, to additionally measure if a sensitive bucket name influences the type of organizational bucket scanners search for. There was no overlap between the TGA names and the 48 organization names.

Cryptocurrency Buckets. If an actor targets specific content, rather than a specific entity, they might scan for buckets named after the content. To detect actors who may be searching for buckets storing cryptocurrency, we named four of the 112 buckets after two cryptocurrencies: Bitcoin and Ethereum. At the time of our experiment, many variations of "bitcoin" and "ethereum" bucket names already existed, so we hyphenated the names of the cryptocurrencies with keywords found in the most popular bucket enumeration tools (e.g.,"bitcoin-confidential").

Sensitive and Non-Sensitive Buckets. To search for sensitive content without restriction to a specific entity or content-type, an attacker might scan for bucket names with sensitive keywords. To compare the discovery rate of buckets named after sensitive keywords (e.g., "passport" and "bank") to non-sensitive keywords (e.g., "pictures" and "pretty"), we named four of the 112 buckets using each of these keywords.[2] At the time of our experiment, bucket names with only the keyword already existed, so we hyphenated all sensitive and non-sensitive keywords with a non-sensitive keyword found in the most popular bucket enumeration tools ("10").

Leaked-Alphanumeric Buckets. Rather than blindly guessing names, an actor might harvest bucket names from client activity (e.g., DNS queries) to increase the likelihood of discovering buckets. To measure if scanning actors are harvesting names, we assigned 40 out of the 112 honeybuckets with "unlikely-guessable" names: randomly generated alphanumeric names of length 16 (e.g., "q81osr2ba5wnid4g"). To identify potential sources of leaked buckets, we leaked 20 of our 40 unlikely-guessable honeybuckets across a variety of platforms, including a Github repository, a new Pastebin repository, a single tweet on a new Twitter account, and the HTML of an academic website (but not visible in a browser). We leaked two buckets at a time on each platform to verify whether a scanner likely guessed the bucket by chance (i.e., visited only one bucket) or likely found the bucket on the leaked platform (i.e., visited both buckets). To identify if scanners used passive DNS as sources for bucket names, we also queried the domains of two unlikely-guessable honeybuckets across DNS resolvers: two resolvers operated by Google, two by Spectrum, two by AT&T, two by a Russian ISP (ASN 12714), and two by a Chinese ISP (ASN 4134). Finally, to identify if scanners, such as Google bots, unsolicitedly download content from bucket names found in email, we saved two unlikely-guessable honeybuckets in a single email draft on Gmail. We withheld and did not leak the remaining 20 of 40 alphanumeric buckets to serve as a control group.

2. We used sensitive keywords defined by the U.S. Justice Department [44] and non-sensitive keywords defined by the disjoint set of those sensitive keywords and the top 1,000 most common English words [45].

3.1.2. Bucket Permissions. Having chosen candidate bucket names to scan, an attacker could use one or more of Amazon's 100 operations to interact with the bucket. To capture as many potential interactions as possible, we gave bucket-scanning actors considerable freedom to interact with our honeybuckets: all honeybuckets allowed any actor to read the contents of the bucket and write new content to the bucket. However, deleting files originally uploaded by us was forbidden (although we could detect deletion attempts). We enabled bucket versioning—a feature that saves all past versions of files in a bucket—to track how actors upload, delete, and modify the files they themselves upload. We recorded all available metadata of interactions with the honeybuckets, including the time of interaction, actor's IP address, actor's AWS account (if the actor sent an authenticated request [46]), request URI, and any error messages the actor received if their request was malformed or forbidden. To promote reproducibility, we share our raw data at https://github.com/kizhikevich/honeybuckets.

3.1.3. Bucket Contents. After listing the directory of a bucket, an actor might only choose to download a subset of "interesting" files. We uploaded files with a variety of enticing names and contents to each honeybucket to test for file-download preferences among scanning actors. Table 1 lists the nine files we placed in each honeybucket. To function as controlled variables when comparing download preferences among scanning actors, the file names and contents were identical across all buckets. For example, each honeybucket included a "Client_list_Dec_2021" file to lure scanning actors searching for sensitive client information. The file included fake names, home addresses, and social security numbers generated by Faker [47]. Files named "Backup.pst", "Outlook.pst", "id_ed25519", and "Inbox.mbox" were lures for actors who were searching for sensitive email folder names, SSH private keys, and Google takeout backups, respectively. Each file contained fake data in the expected format. We additionally included a file with the commonly abused .jar extension [48] to test for malicious actors who might wish to replace existing .jar files with hidden "trojan" malware. The content of the uploaded .jar file emulated a benign calculator program [49]. Finally, we included two README files—sized 0 bytes and 2 kilobytes—to test if actors checked for file size prior to downloading.

File Name	Content	Unique IPs	Unique ASNs	Downloads
Client_list_Dec_2021	Fake names, SSNs, addresses	88	56	160
Backup.pst	Sensitive mail folder names	69	47	155
README1	AAA...	64	41	127
Outlook.pst	Sensitive mail folder names	54	34	110
README2	Empty file	53	32	112
id_ed25519	SSH private key	53	31	117
Inbox.mbox	Google Takeout backup	50	32	107
UTC*	UTC wallet (keystore file)	48	30	112
javazoom.jar	Benign jar file	41	21	103

TABLE 1: **Honeybucket File Contents**—Each honeybucket hosted nine unique files that contained a variety of sensitive names and content, intended to attract scanning actors with different target preferences.

3.2. Pilot Study Results

In this section we characterize how scanning actors engaged with our honeybuckets. We investigate the most common methods actors used to scan for buckets, the type of abusive activities buckets received, the amount of time actors spent interacting with a public bucket, and who was hunting for buckets. Most notably, we found that **buckets named after companies were the most likely to be accessed** (Section 3.2.1). Although the majority of bucket interactions only checked for bucket existence, **hundreds of IP addresses attempted to download, delete, or upload objects**—including malicious shell scripts (Section 3.2.2). This activity happened quickly after the buckets become accessible: **actors scanned public buckets within 40 minutes of deployment and uploaded unsolicited content within 10 days** (Section 3.2.3).

3.2.1. How Buckets Were Found. Buckets named after companies were scanned with the highest number of operations and IP addresses. Table 2 lists the top 20 buckets with the most attempted operations. Half of these top 20 buckets had a company in their name, with the top five named after "Tesla." For a detailed breakdown, Table 3 presents the number of unique IP addresses and Autonomous Systems (ASes) that targeted each bucket on average per bucket type. Company buckets were targeted with statistically significantly[3] more IPs and ASes per day compared to all other bucket types. On average, at least one unique IP and AS visited a company bucket per day. In Section 4, we further investigate why actors are lured towards particular companies by deploying a second honeybucket experiment.

Seven of the top 20 buckets with the most number of interactions had names from open-source bucket target generators (TGAs); however, the majority of actors did not appear to use only the TGAs to generate these bucket names. Rather, actors were likely using their own list of target bucket names that coincided with the TGA's list. We considered an actor to be using a TGA if (1) a single IP address[4] targeted all four bucket names that belonged to that TGA; or (2) all TGA buckets were scanned by a uniform distribution of unique IP addresses (which accounts for actors using multiple IPs when scanning, e.g., from VPNs). In Table 3, we filtered for IP addresses that targeted all four buckets from a single TGA and found statistically significantly fewer actors that exhaustively used the TGA names compared to the number of unique IPs that scanned buckets named after organizations. Furthermore, TGA buckets were not targeted by a uniform number of IP addresses (e.g., "origin-www" from DNSpop was targeted

Bucket Name	Type	# Ops	# IPs
teslaproduction	Company	2538	467
teslapublic	Company	1620	355
tesladownload	Company	1601	375
teslasecurity	Company	1470	339
teslaprivate	Company	1456	342
origin-www	TGA (DNSpop)	1379	349
612	TGA (DNSpop)	1312	446
lyncdiscover	TGA (DNSpop)	980	278
www-download	TGA (Pastebin)	894	323
walmartproduction	Company	872	170
tinderproduction	Company	755	178
ucsdprivate	University	747	143
fbiproduction	Government	627	179
www-slack	TGA (Pastebin)	604	260
www-security	TGA (Pastebin)	572	220
screenshots-www	TGA (Pastebin)	552	245
tinderpublic	Company	542	139
tinderdownload	Company	527	129
walmartsecurity	Company	516	133
ciaproduction	Government	508	151

TABLE 2: **Top 20 Buckets With The Most Attempted Operations**—Five out of six buckets named after Tesla experienced the most attempted operations.

3. We used a one-sided Mann-Whitney U test to evaluate whether the volume of traffic per day that targeted a specific bucket type was stochastically greater than the volume that targeted the bucket type with the next greatest volume of traffic per day. We used $p < 0.05$ and additionally applied a Bonferroni correction—to account for multiple comparisons—when determining statistical significance.

4. The majority of actors used one IP address when scanning (Section 4).

	Total		Per Day	
Bucket Type	**IPs**	**ASNs**	**IPs**	**ASNs**
Companies	195.39	35.72	1.63*	1.44*
Universities	133.83	20.75	1.27	1.19
Government	100.22	19.50	1.08*	1.04*
Non-sensitive Keywords	74.50	15.00	0.95	0.90
Sensitive Keywords	43.50	8.50	0.74	0.72
Cryptocurrency	27.75	7.75	0.56*	0.54*
TGA (filtered)	12.67	12.67	0.42*	0.46*
Leaked	6.17	3.22	0.46*	0.45
Control	1.17	1.17	0.44	0.44*

TABLE 3: **Traffic Across Bucket Types**—Average number of unique IP addresses and associated ASNs that visit the various types of buckets. Buckets named after companies experienced the most traffic on average both overall and per day. Statistically significant increases of traffic per day, relative to all other types of buckets that experienced less average traffic per day, are marked with an $\star$. For example, while university buckets did not experience statistically significantly more traffic than government buckets, government buckets did experience significantly more traffic than all bucket types except for company buckets.

by 349 IPs, whereas "lyncdiscover" from DNSpop was targeted by 278 IPs).

Buckets named after universities or a government service were the second-most likely to be scanned, with no statistically-significant difference between the two (Table 3). Among the buckets associated with organizations, bucket names concatenated with the word "production" were scanned the most. Table 4 shows the relationship between the number of scanning IPs and the organization/keywords in the bucket name. While 467 unique IPs scanned "teslaproduction," only 375 unique IPs scanned "tesladownload." On the other hand, across all organization types, bucket names concatenated with the word "hidden" were an order of magnitude less likely to be targeted than all other keywords (e.g., 4 compared to 127 unique IPs for "fbihidden" vs. "fbisecurity").

Buckets leaked to passive data sources were the least likely to be scanned: an average of just 0.46 unique IP addresses visited a leaked bucket per day compared to an average of 1.63 IPs that visited a company bucket per day. The bucket names embedded in the website HTML content were scanned by just 45 IP addresses, and those leaked via Twitter by 22 IPs (compared to the 467 IPs that scanned teslaproduction). The buckets leaked via DNS queries were never scanned.

Since the search space for buckets is vast, prior work has found that targeting shorter and lower entropy names results in an overall higher hit rate when scanning buckets [3]. However, our results imply that actors are optimizing to find specific targets in addition to maximizing overall hit rate. For example, while bucket "612" contained the least amount of entropy in our honeybucket set, it was scanned by fewer unique IP addresses than five of the six Tesla buckets (Table 2), each of which was substantially longer and higher in entropy. Using the Kolmogorov-Smirnov test, we found no statistically significant difference between the number of unique IPs and ASes that targeted buckets "612" and "teslaproduction" (the Tesla bucket with the highest number of operations) per day, indicating that maximizing overall hit rate was not the only popular scanning strategy. Overall, buckets named after companies were scanned the most compared to TGA buckets, leaked buckets, and buckets with names of a lower entropy.

3.2.2. Bucket Interactions and Abuse. Over 100 unique IP addresses (of 6,567 total IPs) uploaded at least one file to a bucket, for a total of 206 files. Through manual investigation, we identified four unique files that hosted malicious content (uploaded across 16 unique buckets): (1) a "poc.jsp" JavaScript file that spawned a reverse shell to a server specified by a command-line argument; (2) a "test-file.svg" file that, when opened, re-directed to a suspicious domain ("ngrok.io"); and (3) two files, named "_snapshot/test" and "_snapshot/test2", that contained code to send the contents of the `/etc/passwd` file to the actor who uploaded the file.

Only two files, "upload.png" and "s3sec.txt", contained a message to the bucket owner as a warning that their bucket was public. Notably ironic, these good-samaritan warnings were shared through an unsolicited upload. Of the remaining uploaded files, six unique files uploaded across nine buckets contained benign content, six files were not accessible due to the object permissions set by the actor, and 188 files were empty. We summarize the contents of all uploaded files in Table 10 in the Appendix.

Over 700 unique IP addresses (of 6,567 total IPs) attempted to download a file from a bucket. The client list file was downloaded the most, with 160 total downloads across 88 unique IP addresses (Table 1). We did not find any file to have a statistically significantly[5] greater number of downloads per day, likely due to the infrequent nature of downloads per day (i.e., on average, there are just 0.00005 downloads per bucket per day). Thus, we cannot conclude that actors check for file size before downloading files. We summarize the remaining non-abusive bucket interactions in Table 9 in the Appendix, which often consists of checking a bucket's existence and/or listing files.

3.2.3. Bucket Time-to-Abuse. We consider a bucket to be abused when an actor attempted to upload a non-empty file, download a file, or delete a file from a bucket. While actors were quick to find buckets—within 40 minutes of deployment the bucket "walmartdownload" had its directory listed— it took over a week for an actor to abuse a bucket for the first time.

Buckets named after TGA targets or company names were the only two categories to experience uploads, with buckets receiving a first upload within an average of 71 days. The first successful upload across all 112 buckets occurred 10 days after deployment with the "lyncdiscover" bucket generated with the TGA DNSpop. An unauthenticated actor uploaded a file with instructions on how to

5. We calculated statistical significance using the methodology from Section 3.2.1.

	# Unique IPs Targeting Org-Keyword Bucket						
Organization	'production'	'download'	'public'	'private'	'security'	'hidden'	Total
Tesla	467	375	355	342	339	8	1886
Walmart	170	142	140	139	133	8	732
Tinder	178	129	139	125	108	5	684
UCSD	130	131	128	143	129	3	664
Stanford	163	173	162	158	158	3	817
FBI	179	122	119	116	127	4	667
CIA	151	116	116	117	118	4	622
NYPD	78	78	76	78	78	3	391
Total	1516	1266	1235	1218	1190	38	6463

TABLE 4: **The Impact of Bucket Name Construction on Received Scans**—Buckets that contained the keyword "production" were targeted by more unique IP addresses than buckets containing any other keyword, no matter the organization.

make the bucket private. The file was called "s3sec.txt" and can be found in the Github repository s3sec [50].

Compared to uploading content, actors were much slower to download content. The average time-to-first download across all bucket types was 78 days, with the first successful download of the file "Backup.pst," 27 days after deployment, in the bucket "tesladownload." Content inside the non-sensitive keyword and control buckets was never downloaded over the course of the experiment.

The most rare and slowest-to-occur abuse of a bucket was file deletion: the only attempt to delete a file from Table 1 occurred 134 days after bucket deployment. Thus we conclude that actors were quick to find buckets and overall performed abusive operations by uploading non-empty unsolicited files and downloading files, but rarely attempted to delete files from buckets.

3.2.4. Identifying Bucket-Scanning Actors. A total of 6,567 unique IP addresses performed at least one operation to at least one bucket. Nearly all (99.9%) IP addresses sent an unauthenticated AWS request,[6] allowing the user to remain anonymous. However, 27 actors authenticated themselves, and eight used non-alphanumeric usernames. Table 5 lists the eight actors, showing their username, IP addresses used, buckets visited, etc. Three actors had usernames that alluded to bugfinding (i.e., "s3bug", "bug", "pudsec"). The user "bug" uploaded a "Read.txt" file that described how to pen-test buckets for the purposes of receiving a bug bounty. Three authenticated users alluded to being "administrators," in which at least one suggested they were a "bot" for scanning (i.e., "Admin.../xbotusr"). The remaining 19 authenticated users used non-informative, random alphanumeric names. In Section 4, we deploy a new set of experiments to filter for only non-bot scanners.

Authenticated users also gave a glimpse into understanding if bucket-scanning actors often used multiple source IP addresses, and whether a unique IP address was likely to identify a unique scanning actor. We used the authenticated actor set as an approximate ground truth mapping of unique users to IP addresses.[7] IP addresses were a sufficient approximate indicator of unique scanning actors: 90% of authenticated actors used only one scanning IP address and 100% of authenticated actors used IP addresses from the same autonomous system (Table 5). In Section 4.1, we show that the vast majority of bucket scanning actors were likely only using one IP address.

All scanners originated from a set of 330 autonomous systems, a subset of which have security-critical reputations. Approximately 66% of scanners originated from three ASes: M247 (ASN 9009), HostRoyale (ASN 203020), and CHOOPA (ASN 20473). Clients using M247 and Choopa are known to be consistently engaged in high-risk and highly fraudulent behavior [52], [53]. HostRoyale's clients are known to use its anonymizing VPN services [54].

3.3. Summary

This pilot study systematically demonstrated that scanning strategies are not random: our honeybuckets named after well-known companies received the most activity (Section 3.2.1). Scanners were quick to discover new publicly accessible buckets, finding buckets within 40 minutes of deployment (Section 3.2.3). Finally, scanning actors actively interacted with the bucket contents in a variety of concerning ways: at least one file containing sensitive data was downloaded across nearly all buckets, many buckets had malicious files uploaded to them, and some had files targeted for deletion (Section 3.2.2).

While this experiment established many aspects of scanning activity, a number of questions remain. It is unclear why actors were lured towards particular companies, how actors might take further advantage of downloaded sensitive data, and whether identical files across buckets could cause actors to recognize the decoys and modify their behavior. Furthermore, actors can use multiple IPs and VPNs to mask their activity, and such aliasing leaves unresolved how to attribute multiple interactions to the same actor. In the next section, we build on our pilot

6. Amazon allows users to be unauthenticated, which is when a user does not have an AWS account [46] or when an authenticated user adds the flag "–no-sign-request" to their command line argument [51].

7. The set is likely biased towards users that did not care about concealing their identity and thus served as an expected upper-bound of the number of actors that used only one scanning IP address.

User	IPs	ASN	Buckets Visited	Operations
user/energi-0001	103.157.116.108/32	Cloud Teknologi (137331)	All DNSpop	Check exist, List dir, Get ACL
assumed-role/... Admin.../xbotusr	12 IPs in 148.177.96/24	DC Protection (198949)	origin-www (DNSpop)	Get ACL
user/Admin	186.29.129.113/32 190.25.111.135/32	ETB (19429)	612 (DNSpop)	List dir, Get ACL
user/bref-cli	159.89.129.123/32	DIGITAL OCEAN (14061)	lyncdiscover (DNSpop)	Check exist, List dir, Get ACL
user/Administrator	143.238.166.88/32	Telstra (1221)	origin-www (DNSpop)	Check exist, List dir, Get ACL
			lyncdiscover (DNSpop)	Check exist, List dir, Get ACL
user/s3bug	103.105.154.178/32	Global Ra Net (135692)	3/4 Tinder buckets	Get ACL
user/bug	103.79.171.204/32	MNR Broadband (133648)	tinderpublic	List dir, Upload object
user/pudsec	216.126.238.240/32	Hostodo (399804)	612 (DNSpop)	Check exist, List dir, Get ACL

TABLE 5: **Authenticated User Activity**— We present a list (8 out of 27) of all authenticated users who used non-alphanumeric usernames (omitting their unique account ID for brevity). The majority of authenticated users used only one IP address to scan, only visited buckets of one type (e.g., TGA, company), and did not interact with bucket content.

study and deploy a new, refined honeybucket experiment to provide more insight into precisely these questions.

4. Exploitation of Company Buckets

In our pilot study, actors were most likely to scan and download sensitive files from buckets named after companies, occasionally using multiple IPs to do so. In this section, we conduct a new, refined experiment to broadly investigate (1) what types of companies receive the most traffic (e.g., industry sector, Fortune 500 standing, having a vulnerability disclosure program), (2) if actors using multiple IP addresses can be more easily identified and, most importantly, (3) whether downloaded content is exploited. We found that companies with a vulnerability disclosure program were more likely to be scanned. Most alarmingly, though, we recorded eight instances of actors exploiting downloaded content from our buckets, which directly led to unauthorized attempts to login to a honeypot server.

4.1. Methodology for Bucket Configuration

We deployed 120 honeybuckets on the AWS S3 platform on October 2, 2022 and hosted them for 1 month. We configured buckets with three primary differences from the methodology in Section 3.1: (1) names followed a single enterprise-themed naming scheme, (2) buckets contained an informative document that tracked the longevity of information post-download, and (3) bucket contents used a unique identifier that helped track actors who used multiple IPs.

4.1.1. Bucket Names. We investigated what factors cause the buckets of one company to be at a higher risk of abuse (e.g., malicious uploads, malicious downloads) than another company. To study the cloud-storage attack surface of enterprises, we named 120 new buckets after 60 Fortune 500 companies [55]. We created the set of 60 companies by (1) removing company names with an "&", as that symbol is not allowed in bucket names, (2) randomly ordering the remaining Fortune 500 companies, and (3) selecting the first 30 companies that had a clear vulnerability disclosure procedure (VDP) and the first 30 companies that did not appear to have a VDP. To determine if a company hosted a VDP, we used Google to search for "<company name> vulnerability disclosure" and looked for a disclosure procedure within the top 10 results.[8]

To construct the bucket names, we concatenated (with no spaces) each chosen company with the two keywords from Section 3.2.1 that attracted the greatest number of actors—"production" and "download"—thereby assigning two buckets per company (e.g., "carvanaproduction" and "carvanadownload"). Table 12 in the Appendix lists the names of all chosen companies,[9] their offering of a VDP and/or bug bounty, and their 2022 Fortune 500 ranking. In Section 5, we discuss the ethics of this methodology.

4.1.2. Bucket Contents. Recall that in Section 3.2, some actors downloaded at least one file from the bucket. In this experiment, our goals were to (1) identify what actors do after downloading a file, and (2) better estimate the number of actors engaging with the buckets. To accomplish these goals, all buckets in our second experiment hosted (i) fictitious sensitive content to lure actors to interact with our honeybuckets in a way that allowed for tracking their actions, and (ii) a new text document that served as a source of information to trace the identity of actors.

Sensitive Information. To lure actors into interacting with our honeybuckets, we hosted a nested directory of fake financial data generated by the Faker tool [47]. Each honeybucket hosted unique data (unlike our first experiment in Section 3.1) to reduce the chance of actors finding multiple company-named honeybuckets and possibly growing suspicious if they encountered identical data. We named the nested directory with an hourly-changing hashed time stamp (i.e., "update_2022_chargeback_{unix time}"). As a result, an actor who used multiple IP addresses across multiple hours to list bucket contents and download individual files could be identified using the hashed time stamp. To avoid triggering alarms (see

8. This search methodology identified that 21% of the first set of the 60 randomly chosen companies had a vulnerability disclosure program, which is nearly identical to what prior work has found [56].

9. During bucket deployment, we encountered an already-existing bucket, "Blackrockproduction". We replaced Blackrock with another randomly-chosen Fortune 500 company.

Appendix A.2), we zip-encrypted the contents of individual sensitive files across all honeybuckets.

Informative Document. All buckets hosted a single, unencrypted text document that (1) encouraged actors to contact the bucket owner, (2) traced the longevity of sensitive information post-download, and (3) identified actors who used multiple IP addresses. To encourage actors to contact us, the document included an email address that we controlled, and attributed bucket ownership to an ambiguous financial analytics contractor of the Fortune 500 company. In this way, we could still measure interactions associated with buckets named after Fortune 500 companies, but—assuming the actor read the informative document—redirect follow-up email interactions to us.

We also wanted to infer whether actors who downloaded sensitive data had malicious intentions, such as using the downloaded information for nefarious purposes. For this goal, in the document we also included an SSH username, password, and IP address for a Cowrie SSH honeypot [37] that we hosted. To lure an actor to login via SSH, the file (falsely) stated that the encryption key to the sensitive files in the bucket could be found in our SSH honeypot. While no username or password combination granted entry into our Cowrie honeypot, we monitored login attempts (i.e., IP address, timestamp, username and password attempted) to see if any attempted SSH credentials matched the credentials provided in the honeybucket.

In the first experiment (Section 3.2), thousands of unique IP addresses interacted with our buckets. To identify individual actors who may have used multiple IP addresses to search for buckets and attempts to login into our honeypot, we updated the SSH password in the informative document every hour. Specifically, the document stated to "Concatenate your unique 3-digit token with the secure numeric key, like so: <token>62514653" where the "secure numeric key" was a hash of the current timestamp. We then used the SSH password as a link between the IP address that obtained the password and the IP address that used it. In a similar manner, we updated the name of the text document to trace actors who may have used different IP addresses to list directory contents and download individual files. To attract actors to the informative document, we named the document 'secure-encryption-ssh-quickstart-{unix time}.txt' to attract downloads with an enticing name. Relative to all other bucket contents, the document's name was alphabetically first, ensuring that the informative document appeared first when an actor listed the contents of the bucket. Appendix A.3 shows the exact text of the document.

4.1.3. Bucket Permissions. In Section 3.2.2, we documented actors uploading over 100 files—some of which were inaccessible to us—across different buckets. To preserve the integrity of our bucket configuration, in this experiment actors were only allowed to (1) list the bucket directory, (2) download all objects, and (3) upload an object if and only if the actor transferred ownership of the uploaded object to the bucket owner. Currently, this permission is the only mechanism to automatically ensure uploaded files can be accessed by the bucket owners, since files are automatically owned by the uploader [57], [58]. Deleting objects was forbidden.

Company Attribute	Avg # IPs per bucket
Has VDP	17.75*
No VDP	10.78
Technology	28.19*
Healthcare	16.20
Transportation	16.00
Financials	15.69
Retail	10.25
Chem./Energy/Industrial	8.78
Eng./Construction/Materials	5.83
Aerospace/Defense	5.50
Business Services	4.00

TABLE 6: **Company Attribute Impact On Scanning**—Buckets that were named after companies with vulnerability disclosure programs (VDPs), or were in the technology sector, attracted statistically significantly more IP addresses. Statistically significant increases of a metric, relative to all metrics with a smaller average value, are marked with an $\star$.

4.2. Results

In this section, we use the increased lures and tracking capabilities to further understand how bucket scanners operate. We start by investigating what kinds of companies lured the most actors and found **a significantly increased number of actors and significantly increased amount of abusive behavior correlated with companies with a vulnerability disclosure program** (Section 4.2.1). Certain scanners used VPNs, prompting us to develop a scalable approach to track and group colluding IPs as belonging to the same actor. Using this approach, we found most actors still only used one IP address (Section 4.2.3). Finally, we analyze the "abusiveness" of actor behavior, distinguishing between those actors who merely downloaded our purportedly sensitive information and those who used that data to login to another machine that they were not authorized to access. **We traced over 3000 login attempts to 8 unique actors who had previously downloaded content from our bucket** (Section 4.2.4).

4.2.1. Understanding Company Targets. Buckets that contained the names of companies with a vulnerability disclosure program ("VDP companies") were statistically significantly[10] more likely to be scanned. Table 6 presents the number of IP addresses which scanned different categories of companies. On average, VDP-company buckets were scanned by 60% more IP addresses compared to non-VDP company buckets, attracting roughly 18 IPs per day. VDP-company buckets were also responsible for the majority (6/8) of the abusive SSH behavior (Section 4.2.4). Our results are consistent with previous work [59] that found a positive correlation between bug bounty programs and data breaches when studying government-reported breaches. One possible explanation is that potential finan-

10. We used the one-sided Mann-Whitney U methodology from Section 3.

Bucket Name	# Unique IPs
americanexpressdownload	139
americanexpressproduction	140
oracledownload	112
intuitproduction	55
oracleproduction	54
intuitdownload	46
nvidiadownload	44
nvidiaproduction	44
targetdownload	43
polarisproduction	42

TABLE 7: **Top 10 Most Scanned Buckets**—Over 58% of the total number of IPs scanned the top 10 buckets (719 out of 1,228 total IPs), which spanned six companies. Every company, except Polaris, has a vulnerability disclosure program.

cial incentives attract scanners, but we cannot discount the possibility that companies with valuable online assets are simply more likely to institute VDPs (perhaps due to being attacked more frequently).

Additionally, buckets with names of companies in the technology sector were statistically significantly[11] more likely to be scanned by unique IP addresses compared to any other sector. Buckets named after technology companies[12] were scanned by 74% more IPs on average than healthcare buckets, the second-most scanned sector. Technology company buckets attracting the most IPs was not a symptom of having a VDP: while 5 out of 8 technology companies had a VDP, 10 out of 10 of the health companies and 10 out of 16 of the financial companies also had a VDP. We also found no correlation between a company's Fortune 500 rank and the number of IP addresses who scanned its corresponding bucket.

Table 7 lists the top ten buckets that were scanned by the greatest number of unique IP addresses,totaling 719 IPs out of the total 1228 unique IPs (58.55%). The two most-scanned buckets were both named after American Express, which has a VDP. Unique actors consistently scanned American Express buckets throughout our experiment. Six of the top ten most-scanned buckets were named after technology companies, and nine were named after companies with a VDP.

4.2.2. AWS Triggered Reports. We received two reports during the course of our experiment sent via AWS. However, these reports were not prompted by an internal AWS hygiene system. Instead, they were instigated by people (employees or contractors) who explicitly searched for buckets named after the corresponding companies. The first AWS report was for a bucket named after a technology company. AWS included the email addresses of the "original abuse reporter[s]" who were two employees of that company. The second report was for two buckets named after a healthcare company, and included the contact information of a security employee from the company. After removing the buckets, as per their request, we spoke with the healthcare security team and learned that the buckets were reported to them by a third party rather than an internal scanning system or team. These experiences highlight that actors explicitly searching for buckets trigger more warnings than AWS's internal, automated detection systems [61], [62].[13]

4.2.3. Identifying and Tracking Unique Actors. Since some actors are likely to use VPNs to interact with the buckets, we developed an algorithm to identify "colluding" IP addresses to better identify and track *at scale* a single actor's operations across multiple IP addresses. We define a single actor using colluding IP addresses to be either one scanner operating under a VPN, one host whose DHCP lease expired between actions, *or* multiple parties colluding by sharing or selling information to each other. We used this algorithm to better approximate the true number of actors interacting with the buckets, rather than just using the number of unique IP addresses.

Algorithm. To identify colluding IP addresses, we relied on the hourly updates of unique identifiers in bucket filenames (Section 4.1.2). These identifiers provided a link between colluding IP addresses that execute operations on behalf of each other. Concretely, there are three cases of operations that revealed colluding IPs:

- **Failure:** Without having ever listed the directory, there was an attempt to download a file that used to, but no longer, exists. For example, in Figure 1, IP9 never listed the directory, but attempted to download file *d*, which no longer existed. IP8 was the only IP that listed the directory during file *d*'s existence, thus IP8 and IP9 must have been colluding IPs.
- **Success:** Without having ever listed the directory, there was a successful download of a file. For example, in Figure 1, IP6 never listed the directory, but successfully downloaded file *c*. IP5 was the only IP address that listed the directory after file *c* was uploaded, but before IP6 downloaded file *c*. Thus IP6 must have been colluding with IP5.
- **SSH:** Without having ever downloaded the informative document, there was an attempt to login to the SSH honeypot. For example, in Figure 1, IP12 never listed the directory, but attempted to login to the SSH honeypot using the credentials only found in file *f*. IP11 was the only IP address that successfully downloaded file *f*, thus IP12 and IP11 must be colluding IPs. We use the success case above to determine that IP10 and IP11 must have been the same actor as well.

Algorithm 1 formalizes the logic to detect all three cases of likely-colluding events. It takes as input (1) an IP address and (2) a token associated with the IP address. The

11. Accounting for Bonferroni correction, the Mann-Whitney U methodology only returns significant p-values for sample sizes greater than 10. Thus, we excluded the following company sectors from statistical analysis: Healthcare (10 buckets total), Transportation (8), Engineering (6), Aerospace (4), and Business Services (2).

12. We determine sector by using the Fortune 500 sector label [60].

13. Indeed, to the best of our knowledge, neither the AWS security investigator, Detective [61], nor the AWS threat detection service, GuardDuty [62], specifically monitor for buckets hosting exposed sensitive data.

Algorithm 1: find_colluding_ips(IP,token):

```
// extract filename from token
if token==valid SSH password then
    file = unhash(token)
end
else if token==informative doc then
    file = token
else
    return // IP did not download a file or
        SSH
end
// base case: no collusion
if time('IP lists directory') < time('IP downloads file') then
    return IP // the IP listed directory for
        itself
end
// get time of first download
first_download=MIN(logs[logs['ips']==IP]['time'])
// get bucket of first download
bucket=logs[logs['time']==first_download]['bucket_name']
// find if download was successful
successful_download=
logs[logs['time']==first_download]['error']
// get upload time
upload_time=get_upload_time(file)
// get deletion time
delete_time=upload_time + 1 hour
if successful_download then
    cut_off=first_download // file not deleted by
        download time
end
if !successful_download then
    cut_off=delete_time // file already deleted
        by download time
end
// get all directory-listing IPs
ips_list_dir= logs[logs['operation']=='list directory']['ips']
for i in ips_list_dir do
    // if directory listing between upload
        and cut off time, mark colluding_ips
    if logs[logs['ips']==i]['time'] ≥ upload_time &
      logs[logs['ips']==i]['time'] ≤ cut_off then
        colluding_ips.append(i)
    end
end
return colluding_ips
```

algorithm returns as output a (potentially empty) set of IP addresses that colluded with the input IP address based upon the time of their operations. The token associated with the IP address is the informative document or SSH password that the IP address had downloaded or used. The token contains the unique time-based identifier that identifies which time frame another colluding IP could have downloaded the token from. Upon identifying the time frame, the algorithm simply searches for which other IP addresses downloaded the token in the target time frame, and assigns those IP addresses to the returned colluding set. We note that if the input IP address did not download a document or abuse an SSH password (i.e., "token == none"), then the algorithm will not return any colluding IPs for the current IP. Instead, if it was indeed a colluding IP, it may be returned as a colluding IP for another IP address that did download a document or abuse an SSH password.

Notably, the algorithm's accuracy directly depends upon the granularity of content updates. For example, since our methodology updates bucket file names every hour, IP addresses that appear together within the same hour—whether by accident or due to collusion—often cannot be differentiated and are all considered candidates for collusion. For example, this situation occurs with IP2, IP3, and IP4 in Figure 1, where IP2 and IP3 both listed the bucket directory and therefore both are considered candidates for colluding with IP4. Further, the algorithm cannot detect if two unique actors are using two VPNs within the same hour on the same bucket. Nevertheless, as shown in Table 3 the number of unique IPs that visited a bucket per day—let alone per hour—is often too low (i.e., $<$ one unique IP per day) for the algorithm's limitation to be an issue for our study.

Characterizing Actors. Applying this algorithm across all logged bucket operations, the vast majority (94.6%) of actors used only one IP address. In the most extreme case, one actor used 45 unique IPs—a subset of which map to known VPN products and Tor exit nodes—to download files. Section 4.2.4 shows that the actors who used more than one IP address were more likely to participate in security-critical behavior. The remaining 5% (61 out of 1228) of IP addresses clustered to at most six unique actors. Finally, we found no set of colluding IPs announced by the same Autonomous System, suggesting that the collusion was likely not a result of a host's DHCP lease expiration.

4.2.4. Abusive access. Among the challenges for a study like ours is that threat actors are anonymous and their underlying motivations are undeclared. Thus, it can be difficult to distinguish between the visits of a benign scanner (e.g., a security researcher or pentester), and a malicious party who is seeking to exploit exposed resources. We introduced the SSH honeypot component of our experiment to address this ambiguity. By creating an opportunity for scanners to "cross the line" and attempt an unauthorized login to third-party infrastructure using credentials harvested from our buckets, we created a measurement that is clearly interpretable. Independent of any motivation, such actions are widely understood to violate ethical norms [63] and, in most countries, civil and criminal law as well.[14] Put another way, there is no motivation that excuses an attempt to use purloined credentials to gain unauthorized access to an unknown server. Indeed, it would be particularly worrying if purportedly benign researchers were taking such action. Thus, we consider all such accesses to be malicious.

We tracked the downloads of the informative document and monitored our SSH honeypot over a span of 6 months. A total of 182 unique IP addresses downloaded at least one bucket's informative document, which contained an email address (i.e., a direct method to contact the bucket owner) and leaked SSH credentials. Eight unique IPs, traced to eight unique actors, collectively performed over 3,000 login attempts using the leaked SSH credentials. Only one actor contacted us directly.[15]

14. For example, in the US unauthorized access of this form is a violation of 18 USC 1030, the Computer Fraud and Abuse Act.

15. The party warned us, in our capacity as the purported financial analytics contractor, that our *usaadownload* bucket was public. This "Good Samaritan" requested that we pay a bug bounty and sent three follow-up email messages insisting on a reward. We note that USAA, which operates a monetary VDP [64], belongs to the category of organizations that attract the most scanning traffic.

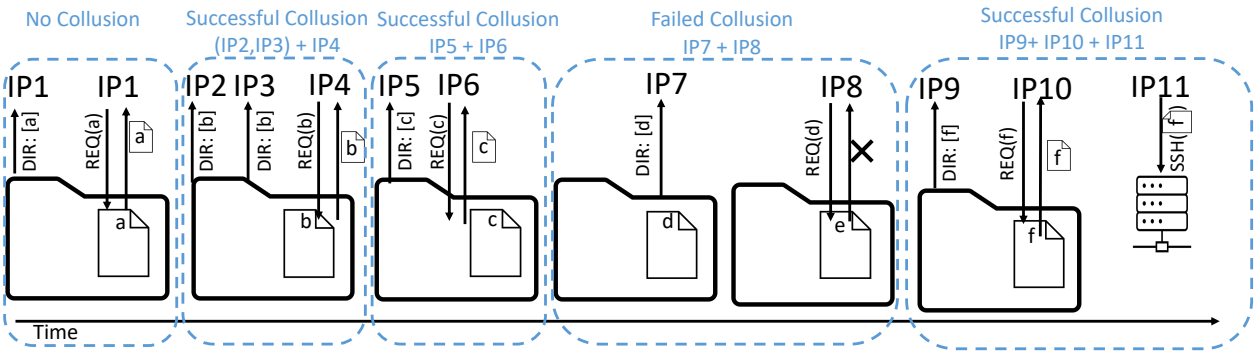

Figure 1: **Identifying Colluding IP Addresses**— We identified colluding IPs using the causal dependencies created by hourly-updated filenames. In our figure, "DIR: [a]" is the return of a directory listing with the filename "a", "REQ(a)" is the request of filename "a", and "SSH(a)" is the extraction of SSH credentials from filename "a". When an IP requested a file without having listed the directory (e.g., IP4), all IPs that listed the directory within the same hour (e.g., IP2 and IP3) were identified as potentially colluding with the file-requesting IP. An IP that SSH'ed into our honeypot server (e.g., IP12), without having downloaded the file with the SSH instructions, was identified as colluding with the IP that downloaded the file with the matching SSH credentials (e.g., IP11).

Using the algorithm outlined in Section 4.2.3 we place activity related to SSH abuse into two categories, "SSH Attempt" and "SSH Brute Force", based on the number of login attempts by an actor.

SSH Attempt. Five unique actors—whose IP addresses belong to Microsoft Cloud (AS 8075), Charter Communications (AS 11427), the high-fraud risk Cloudvider (AS 66240), TIAA (AS 2923), and NETSPI (AS 397919)—attempt at most eight passwords against our SSH honeypot (in particular, they do not exhaustively enumerate all possible passwords).

In the first SSH case (16 days after bucket deployment), a single actor used two IP addresses to find a company bucket, download the informative document, and attempt to login to the SSH honeypot—all within three minutes. The actor used address IP1 to list the directory of the bucket "tiaadownload" and download the informative document. Within two minutes, address IP2 attempted to login to the SSH honeypot using both the username and password originally displayed to IP1 a few minutes prior. IP1 was the only address to download the informative document with the credentials used by IP2. We therefore conclude that the two IPs were the same actor who downloaded and used the leaked SSH credentials.

The IP addresses both resolve to the "vpn.netspi.com" domain, which belongs to NetSPI, a penetration testing, threat and attack surface management company [65]. While NetSPI used the SSH credentials found in TIAA's bucket, we do not find public evidence that TIAA is a customer of NetSPI.[16] However, even if TIAA were a customer of NetSPI who authorized them to act on their behalf, neither the credentials we provided nor the domain accessed belonged to TIAA. Indeed, the downloaded informative document clearly stated that the SSH credentials belonged to a third-party contractor, and not to TIAA. Notably, NetSPI did not append a three-digit identification number to the SSH password, which the informative document instructed to do. While it is conceivable that NetSPI understood that no access would be accomplished, their attempt at unauthorized access violates ethical norms.

16. While TIAA is not a public customer of NetSPI, both companies do have another connection: NetSPI's current CTO previously worked for TIAA as the Head of Cybersecurity Technology [66], [67].

SSH Brute Force. Three unique actors—whose IP addresses belong to the high-fraud risk Packethub (AS 147049), Midco VPN (AS,11232), and 31173 Services VPN (AS 39351)—attempt all one thousand variations of the password leaked in the informative document.

In the first brute force event, an actor used multiple IP addresses to find a company bucket, download the informative document, and attempt to log into the SSH honeypot across a 24-hour period. We determined that two IP addresses used in these interactions belonged to the same actor. On October 30th, address IP1 listed the directory of the bucket "oracledownload." Nearly 24 hours later, address IP2 attempted to download the informative document, DocA, in "oracledownload" that was originally presented to IP1. However, IP2 was unsuccessful due to DocA having already been deleted because of our hourly updates (Section 4.1.2). Since IP1 was the only IP address to list the directory during DocA's existence, IP1 and IP2 must have been the same actor.

We determined that this actor also used two additional IP addresses. Since the actor realized that the original DocA was no longer in the bucket, within 10 seconds a new address, IP3, listed the bucket directory again. Yet another address, IP4, then downloaded the new informative document (DocB) originally listed to IP3. Having finally successfully downloaded DocB, which contained the leaked SSH credentials, IP1 attempted to login to the SSH honeypot using the username and password displayed to IP4 five minutes prior. We therefore conclude that these four IP addresses were all the same actor that ultimately found and used the leaked SSH credentials. All four IPs belong to AS39351, "31173 Services AB," which provides a VPN service [68].

However, unlike the other SSH-abuse attempts, this actor clearly understood the requirement to append a three-digit identification number to the SSH password, as they iterated through *all 1000 possibilities* while attempting to

login. It is difficult to construct a credible scenario in which the actor could have believed these actions were authorized.[17]

4.3. Summary

Actor engagement with company buckets increased with the presence of a vulnerability disclosure program (Section 4.2.1). To identify scanners using VPNs and to better approximate unique actor activity, we developed a scalable approach to identify colluding IP addresses (Section 4.2.3). Colluding IP addresses were more likely to participate in security-critical behavior. In the most abusive case, we identified eight separate events in which colluding IP addresses downloaded, read and understood our informative document, leading them to perform unauthorized login attempts into our honeypot server (Section 4.2.4).

5. Ethics

We considered two classes of ethical issues in this work: potential human subject issues and potential harms to companies. As per discussions with our university's human subjects office, our work does not constitute human subjects research for the purposes of the US HHS Common Rule (45 CFR 46) because we are not collecting information about individuals. Indeed, we have not solicited for any contact and any data that we receive is a byproduct of explicit actions taken to search for our buckets. We take no actions with this data (i.e., we do not act on the unauthorized logins to our infrastructure other than to log it) and thus we reason that any harms are minimal. This is consistent with a long line of research into third-party scanning behavior that has long been considered within the ethical norms of the community [69].

The second issue we considered was potential harm to companies due to trademark confusion or unwarranted reputational damage. Working with the guidance of our university's general counsel, we designed our methodology to minimize these issues in several ways. First, during the targeting phase of the study, we did not leak (i.e., advertise) buckets that included organizational names. Thus, organizational-named buckets could only be found by those actively and blindly guessing the names (minimizing any potential for confusion). Second, we provided multiple paths for resolving any confusion: a contact email address in the informative document, a university affiliation that became clear if a visitor requested the access control list, and the normal notification path via AWS. Finally, in the two instances in which we were contacted by brandholders (themselves notified by third parties, Section 4.2.2) we immediately removed the associated buckets as per their request.

As well, we adhered to AWS terms of service, did not host any (real) sensitive data, and did not allow unauthorized login attempts from the attackers who sought to exploit our seemingly misconfigured buckets.

17. Moreover, since in most countries such unauthorized accesses are criminal acts (e.g., in the US under the Computer Fraud and Abuse Act, 18 USC 1030), this action would be a substantial risk for a benign organization to take.

6. Recommendations

Our work demonstrates that buckets named after commercial entities are actively targeted and exploited.

Defending against cloud storage attackers is simple: configure buckets with sensitive information to be private. Unfortunately, prior work [16] has shown that thousands of buckets remain publicly exposed with sensitive information. Furthermore, over time, buckets are more likely to be misconfigured [3]. It is possible that misconfigured buckets are a symptom of owners who are unaware of the perils of exposing sensitive information, or are unaware that the bucket is theirs in the first place.

Nevertheless, we propose the following recommendations for decreasing the risk of bucket exploitation, beyond simply making buckets private and reiterating the empirical observation that buckets with high entropy names were less likely to be found.

Scan assets. Buckets named after the organization itself—as opposed to just low-entropy names—are the most attractive targets for scanners (Section 3.2.1). We recommend that organizations consistently scan for both known and unknown cloud storage assets to immediately detect misconfigurations. To scan for known assets, organizations can maintain a bucket bookkeeping system that periodically scans all buckets. To scan for unknown assets (e.g., [70]), organizations should scan for "easy-to-guess" buckets named after the organization itself. Scanning for unknown organization-named assets is not a new concept: prominent organizations (e.g., Levi's, New Balance, etc.) already use products (e.g. BrandShield [71]) that protect brand reputation by scanning for organization-named Internet domains (e.g., 'Levis.xyz') that might be engaging in phishing, fraud, or trademark infringement. Thus, such brand-reputation scanning protections can, with likely minimal overhead, also scan for the presence of organization-named buckets to help protect organizations from data breaches (another threat against brand reputation).

Weigh risk according to organization type. Companies with a VDP are more likely to be at risk than universities (Section 4.2.1). We recommend that security services which exist to help organizations trace stray and unknown assets (e.g., Censys [72]) weigh the risk of exposed bucket exploitation according to the organization type. Such services can more aggressively scan and push to patch according to the organization type.

Encourage cloud providers to protect customer assets. No matter the cause of misconfigured buckets, we believe cloud providers are in the best position to help trace and notify misconfigured bucket owners for two reasons. First, while it is challenging for a third party to identify who truly owns a misconfigured bucket, cloud providers can use the email address registered with the bucket owner's account to remind owners of buckets that are open to public access. Second, while no public repository enumerating all existing buckets exists, cloud providers likely have some internal bookkeeping of resources that can be used to exhaustively identify all public buckets. Currently, our experience with AWS is that it provided only limited proactive notifications: across the 232 honeybuckets deployed across a total of 8 months, AWS only notified our

account about four buckets that were publicly exposing sensitive data.

7. Conclusion

Actors scan and abuse the information found in cloud storage buckets. We deployed honeybuckets across two different experiments to measure how actors scan for buckets. Buckets named after companies—especially with a vulnerability disclosure program—were the most likely to be scanned and abused. Attackers constantly abused the permissions of the bucket they found: downloading files, uploading malicious executables, and even deleting existing content. Most concerning, actors read and exploited the contents they downloaded: in eight cases, SSH login instructions leaked from our honeybuckets were precisely followed and used to attempt to gain unauthorized server access. Given that attackers exploiting cloud storage is a reality, we hope our findings encourage both cloud storage operators and customers to track and secure their misconfigured buckets.

Data Availability

To promote reproducibility, we share our raw data and the source code used to identify colluding IPs at https://github.com/kizhikevich/honeybuckets. Raw data includes logs of all scans that target any honeybucket or SSH server.

Acknowledgements

The authors thank Tatyana Izhikevich for providing insightful comments on various versions of this work. Funding for this work was provided in part by NSF grant CNS-2152644, generous research support from Cisco Systems, a CRA-WP/ACSA Scholarship for Women Studying Information Security (SWSIS), the Irwin Mark and Joan Klein Jacobs Chair in Information and Computer Science, the CSE Professorship in Internet Privacy and/or Internet Data Security, and operational support from the UCSD Center for Networked Systems.

References

[1] J. Barr, "Celebrate 15 Years of Amazon S3 with 'Pi Week' Livestream Events," https://aws.amazon.com/blogs/aws/amazon-s3s-15th-birthday-it-is-still-day-1-after-5475-days-100-trillion-objects/, 2021.

[2] C. Ero, "The Bucket List: Experiences Operating S3 Honeypots," in *BSidesSF*, 2018.

[3] J. Cable, D. Gregory, L. Izhikevich, and Z. Durumeric, "Stratosphere: Finding Vulnerable Cloud Storage Buckets," in *Proc. of 24th RAID*, 2021.

[4] "A Cyberattack Illuminates the Shaky State of Student Privacy," https://www.nytimes.com/2022/07/31/business/student-privacy-illuminate-hack.html?referringSource=articleShare, 2022, [Accessed: 2023-05-26].

[5] "A Huge Data Leak of 1 Billion Records Exposes China's Vast Surveillance State — TechCrunch," https://techcrunch.com/2022/07/07/china-leak-police-database/?guccounter=1, 2022, [Accessed: 2023-05-26].

[6] "Microsoft Leaked 2.4TB of Data Belonging to Sensitive Customer. Critics are Furious." https://arstechnica.com/information-technology/2022/10/microsoft-under-fire-for-response-to-leak-of-2-4tb-of-sensitive-customer-data/?utm_brand=arstechnica&utm_source=twitter&utm_social-type=owned&utm_medium=social, 2022, [Accessed: 2023-05-26].

[7] "Misconfigured AWS S3 Bucket Leaks 36,000 Inmate Records," https://www.trendmicro.com/vinfo/ph/security/news/cybercrime-and-digital-threats/misconfigured-aws-s3-bucket-leaks-36-000-inmate-records, [Accessed on: 2023-05-21].

[8] "S3 misconfiguration exposes sensitive data on more than 3 million senior citizens," https://www.scmagazine.com/news/cloud-security/s3-misconfiguration-exposes-sensitive-data-on-more-than-3-million-senior-citizens, [Accessed on: 2023-05-21].

[9] "UK data protection regulator receiving 'large number of reports' about Capital," https://therecord.media/capita-data-breaches-information-commissioners-office, [Accessed on: 2023-05-29].

[10] "Grayhat Warfare," https://grayhatwarfare.com, [Accessed: 2023-05-02].

[11] "AWS S3 Actions," https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-with-s3-actions.html#using-with-s3-actions-related-to-buckets, [Accessed: 2023-05-1].

[12] Bharath, "Slurp's S3 string formatting permuations," https://github.com/0xbharath/slurp/blob/master/permutations.json, 2018, [Accessed: 2023-05-26].

[13] K. Rouwhorst, "s3enum," https://github.com/koenrh/s3enum, 2021, [Accessed: 2023-05-26].

[14] Moorer, "Pastebin bucket-stream-permutation-feature," https://pastebin.com/RpUkywbV, 2018, [Accessed: 2023-05-26].

[15] "Bucket of Staging files after deploying an app engine," https://stackoverflow.com/questions/42947918/bucket-of-staging-files-after-deploying-an-app-engine, 2017.

[16] A. Continella, M. Polino, M. Pogliani, and S. Zanero, "There's a Hole in that Bucket! A Large-Scale Analysis of Misconfigured S3 Buckets," in *Proc. of 34th ACM CCS*, 2018.

[17] M. A. T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, A. J. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in *Proc. of 26th USENIX Security*, 2017.

[18] K. Bock, A. Alaraj, Y. Fax, K. Hurley, E. Wustrow, and D. Levin, "Weaponizing Middleboxes for TCP Reflected Amplification," in *Proc. of 30th USENIX Security*, 2021.

[19] L. Izhikevich, R. Teixeira, and Z. Durumeric, "LZR Identifying Unexpected Internet Services," in *Proc. of 30th USENIX Security*, 2021.

[20] D. Barrera, G. Wurster, and P. C. van Oorschot, "Back to the Future: Revisiting IPv6 Privacy Extensions," *LOGIN: The USENIX Magazine*, vol. 36, no. 1, pp. 16–26, 2011.

[21] V. C. Perta, M. V. Barbera, G. Tyson, H. Haddadi, and A. Mei, "A Glance Through the VPN Looking Glass: IPv6 Leakage and DNS Hijacking in Commercial VPN Clients," in *Proc. of 15th PETS*, 2015.

[22] J. Ullrich, K. Krombholz, H. Hobel, A. Dabrowski, and E. Weippl, "IPv6 Security: Attacks and Countermeasures in a Nutshell," in *Proc. of 8th USENIX WOOT 14*, 2014.

[23] E. Izhikevich, *Building and Breaking Burst-Parallel Systems*. University of California, San Diego, 2018.

[24] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds," in *Proc. of 16th ACM CCS*, 2009.

[25] A. Yelam, S. Subbareddy, K. Ganesan, S. Savage, and A. Mirian, "Coresident Evil: Covert Communication in the Cloud with Lambdas," in *Proc. of 30th WWW*, 2021.

[26] L. Izhikevich, M. Tran, M. Kallitsis, A. Fass, and Z. Durumeric, "Cloud Watching: Understanding Attacks Against Cloud-Hosted Services," in *Proc. of 23rd ACM IMC*, 2023.

[27] G. Akiwate, R. Sommese, M. Jonker, Z. Durumeric, K. Claffy, G. M. Voelker, and S. Savage, "Retroactive Identification of Targeted DNS Infrastructure Hijacking," in *Proc. of 22nd ACM IMC*, 2022.

[28] A. Randall, E. Liu, R. Padmanabhan, G. Akiwate, G. M. Voelker, S. Savage, and A. Schulman, "Home is Where the Hijacking is: Understanding DNS Interception by Residential Routers," in *Proc. of 21st ACM IMC*, 2021.

[29] R. Sommese, M. Jonker, and K. Claffy, "Observable KINDNS: Validating DNS Hygiene," in *Proc. of 22nd ACM IMC*, 2022.

[30] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill, "BGP Hijacking Classification," in *Proc. of 3rd IEEE TMA*, 2019.

[31] C. C. Demchak and Y. Shavitt, "China's Maxim–Leave No Access Point Unexploited: The Hidden Story of China Telecom's BGP Hijacking," *Military Cyber Affairs*, vol. 3, no. 1, p. 7, 2018.

[32] A. Gavrichenkov, "Breaking HTTPS with BGP hijacking," *Black Hat. Briefings*, 2015.

[33] P.-A. Vervier, O. Thonnard, and M. Dacier, "Mind Your Blocks: On the Stealthiness of Malicious BGP Hijacks," in *Proc. of 23rd NDSS*, 2015.

[34] C. Kiekintveld, V. Lisý, and R. Píbil, "Game-theoretic Foundations for the Strategic Use of Honeypots in Network Security," in *Cyber Warfare*. Springer, 2015, pp. 81–101.

[35] "Kippo," https://github.com/desaster/kippo, [Accessed: 2022-05-22].

[36] "T-Pot - The All In One Multi Honeypot Platform," https://github.com/telekom-security/tpotce, [Accessed: 2023-05-23].

[37] M. Oosterhof, "Cowrie SSH and Telnet Honeypot," https://github.com/cowrie/cowrie, [Accessed: 2023-05-30].

[38] A. Mirian, J. DeBlasio, S. Savage, G. M. Voelker, and K. Thomas, "Hack for Hire: Exploring the Emerging Market for Account Hijacking," in *Proc. of 25th WWW*, 2019.

[39] M. Lazarov, J. Onaolapo, and G. Stringhini, "Honey Sheets: What Happens to Leaked Google Spreadsheets?" in *Proc. of 9th USENIX CSET*, 2016.

[40] J. Oberheide, M. Karir, and Z. M. Mao, "Characterizing Dark DNS Behavior," in *Proc. 4th DIMVA*, 2007.

[41] M. Ranganath and M. Keating, "How to Detect Suspicious Activity in Your AWS Account by Using Private Decoy Resources," https://aws.amazon.com/blogs/security/how-to-detect-suspicious-activity-in-your-aws-account-by-using-private-decoy-resources/, 2022, [Accessed on: 2023-10-31].

[42] Bitquark, "DNSpop," https://github.com/bitquark/dnspop/blob/master/results/bitquark_20160227_subdomains_popular_1000000, 2016, [Accessed: 2023-05-26].

[43] Treebuilder, "S3 Mining," https://github.com/treebuilder/s3-mining/blob/master/s3.py, 2017, [Accessed: 2023-05-26].

[44] "Protecting Yourself While Using The Internet," https://www.justice.gov/usao-ndga/protecting-yourself-while-using-internet, 2015, [Accessed on: 2023-11-01].

[45] "1000 Most Common US English Words - GitHub Repository by Ddeekayen," https://gist.github.com/deekayen/4148741, [Accessed on: 2023-11-01].

[46] "Access Ccontrol List (ACL) Overview," https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl-overview.html, [Accessed: 2023-04-14].

[47] D. Faraglia and Other Contributors, "Faker," https://github.com/joke2k/faker, [Accessed: 2023-05-30].

[48] L. E. G. Medina, "A Brazilian Trojan Using A Jar File, VB Scripts And A DLL For Its Multi-Stage Infection," https://www.fortinet.com/blog/threat-research/a-brazilian-trojan-using-a-jar-file-vb-scripts-and-a-dll-for-its-multi-stage-infection, [Accessed: 2022-12-01].

[49] "Github - Java Calculator," https://github.com/arif98741/Java-Calculator/blob/master/Calculator.jar, [Accessed: 2024-05-13.

[50] "s3sec," https://github.com/0xmoot/s3sec, [Accessed: 2023-05-13].

[51] "Access S3 Public Data Without Credentials," https://dev.to/aws-builders/access-s3-public-data-without-credentials-4f06, [Accessed: 2023-04-14].

[52] "M247 - Fraud Risk," https://web.archive.org/web/20200806070926/https://scamalytics.com/ip/isp/m247, [Accessed: 2023-05-26].

[53] "Choopa - Fraud Risk," https://scamalytics.com/ip/isp/choopa, [Accessed: 2023-05-26].

[54] "Hostroyale - Fraud Risk," https://scamalytics.com/ip/isp/hostroyale, [Accessed: 2023-05-26].

[55] "Fortune 500 List," https://fortune.com/fortune500/2022/search, [Accessed: 2023-05-13].

[56] "Vulnerability Disclosure Programs Among the Fortune 500," https://www.rapid7.com/blog/post/2021/05/21/rapid7s-2021-icer-takeaways-vulnerability-disclosure-programs-among-the-fortune-500/, [Accessed: 2023-05-30].

[57] V. Shah, G. Sundaresan, and L. Kear, "Enforcing Ownership of Amazon S3 Objects in a Multi-Account Environment," https://aws.amazon.com/blogs/storage/enforcing-ownership-of-amazon-s3-objects-in-a-multi-account-environment/, 2021, [Accessed on: 2023-10-31].

[58] "How to Download Files That Others Put in Your AWS S3 Bucket," https://medium.com/artificial-industry/how-to-download-files-that-others-put-in-your-aws-s3-bucket-2269e20ed041, 2019, [Accessed on: 2023-10-31].

[59] A. Aaltonen and Y. Gao, "Does the Outsider Help? The Impact of Bug Bounty Programs on Data Breaches," in *Fox School of Business Research Paper*, 2021.

[60] "Visualize the Fortune 500," https://fortune.com/franchise-list-page/visualize-the-fortune-500-2023/, [Accessed: 2024-05-15].

[61] "Amazon Detective Documentation," https://docs.aws.amazon.com/detective/, [Accessed on: 2023-11-01].

[62] "Amazon GuardDuty Features," https://aws.amazon.com/guardduty/features/, [Accessed on: 2023-11-01].

[63] Z. Durumeric, E. Wustrow, and A. J. Halderman, "ZMap: Fast Internet-wide Scanning and Its Security Applications," in *Proc. of 22nd USENIX Security*, 2013.

[64] "USAA Vulnerability Disclosure Program," https://bugcrowd.com/usaa, [Accessed on: 2023-05-23].

[65] "NetSPI," https://www.netspi.com/, [Accessed: 2023-05-04].

[66] "NetSPI TIAA," https://www.netspi.com/webinars/financial-services-cybersecurity-part-two/, [Accessed: 2023-05-04].

[67] "Venturebeat: NetSPI expands," https://venturebeat.com/security/pentesting-firm-netspi-expands-into-attack-surface-management/amp/, [Accessed: 2023-05-04].

[68] "31173 Services AB," https://www.31173.se/, 2022, [Accessed: 2023-05-20].

[69] E. Pauley and P. McDaniel, "Understanding the Ethical Frameworks of Internet Measurement Studies," in *Proc. of 2nd EthiCS Workshop*, 2023.

[70] "Defending Assets You Don't Know About, Against Cyberattacks," https://threatpost.com/defending-unknown-assets-cyberattacks/175730/, [Accessed: 2023-05-23].

[71] "BrandShield, Anti-Phisihing Solutions," https://www.brandshield.com/products/anti-phishing/, [Accessed: 2023-05-26].

[72] "Attack Surface Management and Data Solutions," https://censys.io, [Accessed: 2023-05-30].

[73] "CVS Health Vulnerability Disclosure Program," https://www.cvshealth.com/vulnerability-disclosure-program, [Accessed: 2023-05-23].

[74] "Vulnerability Disclosure Program," https://www.optum.com/business/vulnerability.html#:~:text=Bug%20Bounties,in%20accordance%20with%20this%20policy., [Accessed: 2023-05-23].

[75] "Target Vulnerability Disclosure Program," https://security.target.com/vdp/, [Accessed: 2023-05-23].

[76] "State Farm Insurance Vulnerability Disclosure Program," https://www.statefarm.com/customer-care/privacy-security/security/vulnerability-disclosure-policy, [Accessed: 2023-05-23].

[77] "Pfizer Vulnerability Disclosure Program," https://hackerone.com/pfizer?type=team, [Accessed: 2023-05-23].

[78] "General Electric Vulnerability Disclosure Program," https://www.ge.com/gas-power/products/digital-and-controls/cybersecurity/vulnerability-response, [Accessed: 2023-05-23].

[79] "Goldman Sachs Vulnerability Disclosure Program," https://firebounty.com/416-goldman-sachs-vdp/, [Accessed on: 2023-05-23].

[80] "HCA Healthcare Vulnerability Disclosure Program," https://hcahealthcare.com/legal/index.dot#responsible-disclosure, [Accessed: 2023-05-23].

[81] "Deere Vulnerability Disclosure Program," https://www.deere.com/en/digital-security/responsible-disclosure/, [Accessed: 2023-05-23].

[82] "American Express Cybersecrurity," https://www.americanexpress.com/us/security-center/cybersecurity.html, [Accessed: 2023-05-20].

[83] "TIAA Vulnerability Disclosure Program," https://www.tiaa.org/public/support/security-center, [Accessed: 2023-05-23].

[84] "Oracle Vulnerability Disclosure Program," https://www.oracle.com/corporate/security-practices/assurance/vulnerability/disclosure.html, [Accessed on: 2023-05-23].

[85] "Northwestern Mutual VDP," https://bugcrowd.com/northwestern-vdp/, [Accessed: 2023-05-23].

[86] "Capital One Financial Vulnerability Disclosure Program," https://www.capitalone.com/digital/responsible-disclosure/, [Accessed on: 2023-05-23].

[87] "Nvidia Vulnerability Disclosure Program," https://www.nvidia.com/en-us/security/psirt-policies/, [Accessed: 2023-05-23].

[88] "PNC Financial Services Group Vulnerability Disclosure Program," https://www.pnc.com/en/security-privacy/responsible-disclosure-program.html, [Accessed on: 2023-05-23].

[89] "Charles Schwab Vulnerability Disclosure Program," https://www.schwab.com/schwabsafe/report-a-vulnerability, [Accessed on: 2023-05-23].

[90] "Otis Worldwide Vulnerability Disclosure Program," https://www.otis.com/en/uk/cyber-security, [Accessed on: 2023-05-23].

[91] "Discover Financial Services Vulnerability Disclosure Program," https://discover.responsibledisclosure.com/hc/en-us, [Accessed on: 2023-05-23].

[92] "Carvana Vulnerability Disclosure Program," https://www.carvana.com/responsible-disclosure, [Accessed: 2023-05-23].

[93] "Tractor Supply Vulnerability Disclosure Program," https://www.tractorsupply.com/policies-information_customer-solutions_responsible-disclosure-statement, [Accessed: 2023-05-23].

[94] "Keurig Dr Pepper Vulnerability Disclosure Program," https://hackerone.com/keurigdrpepper?type=team, [Accessed: 2023-05-23].

[95] "CSX Vulnerability Disclosure Program," https://www.csx.com/index.cfm/about-us/contact-us/csx-responsible-disclosure-policy/, [Accessed: 2023-05-23].

[96] "Boston Scientific Vulnerability Disclosure Program," https://www.bostonscientific.com/en-US/customer-service/product-security/responsible-disclosure.html, [Accessed on: 2023-05-23].

[97] "Booking Holdings Vulnerability Disclosure Program," https://raw.githubusercontent.com/arkadiyt/bounty-targets-data/master/data/hackerone_data.json, [Accessed on: 2023-05-23].

[98] "Intuit Vulnerability Disclosure Program," https://security.intuit.com/responsible-disclosure, [Accessed on: 2023-05-23].

[99] "Dover Vulnerability Disclosure Program," https://www.dover.com/privacy-policy, [Accessed on: 2023-05-23].

[100] "Analog Devices Vulnerability Disclosure Program," https://www.analog.com/en/support/technical-support/product-security-response-center/vulnerability-disclosure-policy.html, [Accessed on: 2023-05-23].

[101] "Regions Financial Vulnerability Disclosure Program," https://www.regions.com/about-regions/privacy-security/vulnerability-disclosure-program#:~:text=Regions%20Bank%20does%20not%20operate,a%20code%20or%20configuration%20change., [Accessed on: 2023-05-23].

Leaking Service	Leaking Method
Github repository	Both links in README.md
Pastebin	One link per paste
Twitter	One link in a tweet
Lab website	Both links in HTML
Google's DNS	Query link's A Record
Spectrum's DNS	Query link's A Record
AT&T's DNS	Query link's A Record
Open DNS (Russian AS12714)	Query link's A Record
Open DNS (Chinese AS4134)	Query link's A Record
Gmail drafts	Both links in email

TABLE 8: **Leaking Bucket Names**—We leaked alphanumeric names of length 16 on a variety of platforms (including public media, DNS, and email) to determine if scanning actors are harvesting candidate names from passive sources.

Operation	Occurrence (n=12233)	Cumulative Percentage
Check bucket existence	59.23% (7246)	59.23%
List bucket directory	29.03% (3551)	88.26%
Get object metadata	2.49% (305)	90.75%
Check bucket existence + List bucket directory	1.51% (185)	92.26%
Upload object	1.37% (167)	93.63%
Fail to download an object	0.89% (109)	94.52%
Check bucket existence + Get ACL	0.83% (102)	95.35%
List bucket directory + Fail to download an object	0.76% (93)	96.11%
Successfully download object	0.62% (76)	96.73%
Get ACL + Check bucket existence	0.36% (44)	97.09%

TABLE 9: **Most Common Bucket Interactions**—The majority of bucket-IP pairs only checked for the bucket's existence, but did not pursue further action. Note that the respective bucket-IP pairs listed completed *only* those operations.

A. Appendix

We provide additional details regarding our honeybucket deployment, analysis, and results from Section 3 and Section 4.

A.1. Bucket Scanning Ecosystem Experiment

In Section 3, we conducted an initial experiment to broadly study the bucket scanning ecosystem. Table 8 describes the platforms on which we leak our random alphanumeric buckets. Table 9 provides a general breakdown of the most popular operations performed by actors. Notably, 60% of all operations only checked for the bucket's existence and 30% only listed the files in a bucket without pursuing a further action (e.g., such as downloading a file).

Table 10 provides information about the 18 unique files that were uploaded by other users. We give the exact name of the file; an abbreviated description of its contents; whether the upload is a warning, malicious, or appears to serve no purpose; and the number of buckets the file appeared in. To protect researchers from any harm these unsolicited uploads may cause, researchers should download files in a protected and isolating computer environment to minimize the potential for harm introduced by these unsolicited uploads. However, simply listing the bucket contents can be done on a research machine.

Finally, Table 11 lists the exact names of all honeybuckets. For brevity, we exclude the exact names of the 40 random-alphanumeric buckets that we leaked on various platforms.

File Name	Description	Type	#
upload.png	Warning	Warn	1
s3sec.txt	Warning [50]	Warn	1
poc.jsp	reverse shell	Malicious	5
_snapshot/test	request /etc/password	Malicious	5
_snapshot/test2	request /etc/password	Malicious	5
test-file.svg	re-direct to (ngrok.io)	Malicious	1
bucket.png	No access	-	1
test	No access	-	1
test.txt	No access	-	1
indexx.html	No access	-	1
hello.txt	No access	-	1
s3-test.txt	No access	-	1
xss.svg	image	Benign	1
xss1.svg	image	Benign	1
Read.txt	pen-test [50]	Benign	1
testfile-nullg0re.txt	“this is a test”	Benign	4
test-test-nullg0re.txt	“this is a test”	Benign	1
testfile	“this is a test”	Benign	1

TABLE 10: **Uploaded Files With Content**—Actors uploaded a total of 206 unique files, 188 (91.3%) of which were empty. The table summarizes the 18 files uploaded with content. Two files warned that our buckets were misconfigured, four had malicious code, and six were inaccessible due to the uploader not permitting read access.

Type	Bucket name
Cryptocurrency	bitcoin-confidential bitcoin-secret ethereum-wallet ethereum-passwords
Sensitive Keywords	passport10 bank10
Non-sensitive keywords	pretty10 pictures10
DNSpop TGA	lyncdiscover 612 origin-www liboyulecheng
Slurp TGA	advogado applogie blognovo click1mail
Pastebin TGA	screenshots-www www-slack www-download www-security
Comparison Set (not in any TGA)	confidentialfiles dont-open ignore-me pretty-pictures
Organization	teslaproduction, tesladownload teslapublic, teslaprivate teslasecurity, teslahidden walmartproduction, walmartdownload walmartpublic, walmartprivate walmartsecurity, walmarthidden tinderproduction, tinderdownload tinderpublic, tinderprivate tindersecurity, tinderhidden ucsdproduction ucsddownload ucsdpublic ucsdprivate ucsdsecurity ucsdhidden stanfordproduction stanforddownload stanfordpublic stanfordprivate stanfordsecurity stanfordhidden fbiproduction, fbidownload fbipublic, fbiprivate fbisecurity, fbihidden ciaproduction, ciadownload ciapublic, ciaprivate ciasecurity, ciahidden nypdproduction, nypddownload nypdpublic, nypdprivate nypdsecurity, nypdhidden

TABLE 11: **Bucket Names**—The list of bucket names for our first experiment from Section 3, in which we broadly studied the bucket scanning ecosystem. For brevity, we exclude the 40 alphanumeric leaked buckets.

Vulnerability Disclosure Program			No Vulnerability Disclosure Program	
Company	**Rank**	**VDP**	**Company**	**Rank**
CVS Health	4	[73] $	Raytheon Technologies	58
United Health Group	5	[74]	Charter Communications	69
Target	32	[75]	Tyson Foods	81
State Farm Insurance	42	[76]	3M	102
Pfizer	43	[77]	Applied Materials	156
General Electric	48	[78]	Lithia Motors	158
Goldman Sachs Group	57	[79] $	Hartford Financial Services	160
HCA Healthcare	62	[80]	Lincoln National	187
Deere	84	[81]	Wesco International	200
American Express	85	[82]	L3 Harris Technologies	206
TIAA	90	[83]	Automatic Data Processing	242
Oracle*	91	[84]	Pioneer Natural Resources	248
USAA	96	[64] $	Pulte Group	267
Northwestern Mutual	97	[85]	Oreilly Automotive	279
Capital One Financial	108	[86]	Rocket Companies	282
Nvidia	134	[87]	Vistra	315
PNC Financial Services	178	[88]	Unum Group	317
Charles Schwab	188	[89]	Altice USA	355
Otis Worldwide	254	[90]	ODP	379
Discover Financial Services	281	[91]	Delek US Holdings	346
Carvana	290	[92]	Univar Solutions	369
Tractor Supply	294	[93]	Burlington Stores	377
Keurig Dr Pepper	296	[94]	Jefferies Financial Group	387
CSX	298	[95]	Polaris	419
Boston Scientific	319	[96]	MasTec	429
Booking Holdings	340	[97] $	GXO Logistics	430
Intuit*	366	[98]	Westinghouse Air Brake Tech	439
Dover	433	[99]	Hertz Global Holdings	462
Analog Devices	463	[100]	Graphic Packaging Holding	466
Regions Financial	489	[101]	Landstar System	491

TABLE 12: **Company Names**—The final list of companies used to name our buckets, with each name concatenated with 'download' and 'production' (e.g., 3mproduction, 3mdownload). Names indicated with a * were originally Comcast and Equinix (Section 4.1.1). Rank indicates the company's rank on the Fortune 500 list. In the vulnerability disclosure program column we cite the source explaining the company's VDP and indicate with a $ if the company has a monetary bug bounty.

A.2. Fortune 500 Companies

Table 12 lists the companies we name our buckets after for the experiment in Section 4, including buckets we took down and replaced. After nine days of deployment, Amazon sent a note to an email address associated with our lab's AWS account, warning us that two out of 120 honeybuckets were "publicly hosting highly sensitive and confidential information." We immediately configured these buckets to be private and renamed them to incorporate the names of two new Fortune 500 companies that fit our criteria from Section 4.1.1. We list these replaced buckets with an asterik in Table 12. We also separate buckets that do and do not have vulnerability disclosure programs, as well as indicate whether a monetary bug bounty is offered.

A.3. Informative Document

In Section 4, we conducted a follow-up experiment with additional lures to understand to what extent will actors engage with the content they download. We provide the exact text of the Informative Document, which is uploaded to all honeybuckets, below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Financial Analytics Storage Quick Start
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
This private storage bucket is used as a
temporary  mechanism to transfer data from the
on-premise  storage server to the client.
Transaction data provided is only from the past
three months. For  security reasons, each csv
file is zip  encrypted. See Encryption Key
Access tounzip. To provide the client with more
historical data, an analysis consultant  must
manually copy the data from the on-premise
storage server to the storage bucket.
%%%%%%%%%%
Login instructions:
%%%%%%%%%%
To access the storage server, add the
following entry to your ~/.ssh/config file.

Host transaction_storage
Hostname 35.162.99.242
User bain_fin_analytics

Password Instructions:
Concatenate your unique 3-digit token
with the secure numeric key, like so:
<token>62514653

If you do not have your unique 3-digit
token, please ask your manager or email
fanalytics.bain.IT@gmail.com.
%%%%%%%%%%%%
Encryption Key Access:
%%%%%%%%%%%%
Once logged into the server, navigate to
`/etc/s3_zip_encrypt`, in which the
encryption key can be found.
%%%%%%%%%%%%%%%%
Data transfer instructions:
%%%%%%%%%%%%%%%%
Once logged into the server, navigate to
`/mnt/transactions/<client name>/raw/`,
in which the raw historical data will be found.
Transferring data to and from the storage server
can be done through the following command:

`aws s3 sync /path/to/dir/ s3://COMPANY_BUCKET'
%%%%%%%%%%%%%%%%
Questions:
%%%%%%%%%%%%%%%%
For all other questions, email
fanalytics.bain.IT@gmail.com.
```