There’s a Hole in that Bucket! A Large-scale Analysis of Misconfigured S3 Buckets

Andrea Continella[*]
UC Santa Barbara
conand@cs.ucsb.edu

Mario Polino
Politecnico di Milano
mario.polino@polimi.it

Marcello Pogliani
Politecnico di Milano
marcello.pogliani@polimi.it

Stefano Zanero
Politecnico di Milano
stefano.zanero@polimi.it

ABSTRACT

Cloud storage services are an efficient solution for a variety of use cases, allowing even non-skilled users to benefit from fast, reliable and easy-to-use storage. However, using public cloud services for storage comes with security and privacy concerns. In fact, managing access control at scale is often particularly hard, as the size and complexity rapidly increases, especially when the role of access policies is underestimated, resulting in dangerous misconfigurations.

In this paper, we investigate the usage of Amazon S3, one of the most popular cloud storage services, focusing on automatically analyzing and discovering misconfigurations that affect security and privacy. We developed a tool that automatically performs security checks of S3 buckets, without storing nor exposing any sensitive data. This tool is intended for developers, end-users, enterprises, and any other organization that makes extensive use of S3 buckets. We validate our tool by performing the first comprehensive, large-scale analysis of 240,461 buckets, obtaining insights on the most common mistakes in access control policies. The most concerning one is certainly the (unwanted) exposure of storage buckets: These can easily leak sensitive data, such as private keys, credentials and database dumps, or allow attackers to tamper with their resources.

To raise awareness on the risks and help users to secure their storage services, we show how attackers could exploit unsecured S3 buckets to deface or deliver malicious content through websites that relies on S3 buckets. In fact, we identify 191 vulnerable websites. Finally, we propose a browser extension that prevents loading resources hosted in unsecured buckets, intended either for end-users, as a mitigation against vulnerable websites, and for developers and software testers, as a way to check for misconfigurations.

CCS CONCEPTS

• **Security and privacy → Systems security**; • **Networks →** *Cloud computing*;

KEYWORDS

Computer systems; security; Cloud computing; misconfigurations; vulnerabilities

ACM Reference Format:
Andrea Continella, Mario Polino, Marcello Pogliani, and Stefano Zanero. 2018. There’s a Hole in that Bucket! A Large-scale Analysis of Misconfigured S3 Buckets. In *2018 Annual Computer Security Applications Conference (ACSAC ’18), December 3–7, 2018, San Juan, PR, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3274694.3274736

1 INTRODUCTION

Due to their scalability, cost-effectiveness, and ease of use, cloud storage services are widely used and attractive for hobbyists and companies of all sizes alike. Remarkably, the spread of cloud storage services, accessible through simple-to-use APIs, is contributing to reduce the friction between developers and the need of skilled system administrators able to configure and secure complex in-house storage servers. Despite this advantage, the pervasiveness of cloud storage services—often replacing solutions once hosted on premise, as well as “behind the firewall” storage servers for internal data—comes with privacy concerns due to the possibility of involuntary data exposure. While most services offer fine-grained access control systems, these are not trivial to understand and use. Indeed, according to a recent study [10], one of the most frequent type of misconfiguration is the wrong assignment of permissions due to the lack of knowledge or experience. Unlike a permission error in a behind-the-firewall storage system, the same error in a public cloud storage service potentially exposes confidential information to the entire Internet, much like exposing an FTP server with anonymous access enabled to the Internet [26].

Storage services are a key part of the offerings of leading cloud services operators, e.g., Amazon S3 [2], Google Cloud Storage [13], Microsoft Azure Storage [18], and Digital Ocean Spaces [11]. Among these, arguably, one of the most widespread and well-known service is Amazon S3 (Simple Storage Service). In S3, users define storage containers (*buckets*) that contain arbitrary data objects (i.e., files). Access control rules can be defined at the bucket and object level. However, newly created buckets and objects are private and can be accessed only by their owner. Despite this default policy, many high-profile cases of security issues stem from permission misconfigurations: In October 2017, several S3 buckets belonging to a large technology consulting firm were left publicly accessible, which contained sensitive data, including access keys and credentials [22]; a similar data leak revealed a publicly-accessible database with information about customers of a large cable company [9]. Worse, misconfigurations may give attackers another opportunity to infect the users of popular websites with malicious code: In February 2018, a site belonging to the Los Angeles Times was found to

[*] Also with Politecnico di Milano.

ACSAC ’18, December 3–7, 2018, San Juan, PR, USA

ACM ISBN 978-1-4503-6569-7/18/12…$15.00
https://doi.org/10.1145/3274694.3274736

be serving malicious crypto-mining JavaScript. The infection was caused by a misconfiguration of the Amazon S3 bucket that hosted the static assets, which enabled unauthenticated users to upload and overwrite arbitrary files [19]. More recently, information security experts have found publicly writable buckets and left alerts (i.e., uploaded text files) to notify the owners of the misconfiguration: According to the BBC [30], this happened in more than 50 cases.

The recent informal scans of the cloud storage ecosystem [24, 28], motivated us to create a tool for a comprehensive, large-scale, analysis of the usage of cloud storage in the wild, to help the users understand the source of configuration mistakes. Taking as a case study Amazon's S3, we present our approach and tool to quantify the prevalence of misconfigurations in the wild. Our tool can perform security checks and discover publicly exposed buckets, verify their access permissions and determine possible misconfigurations. Moreover, through web crawling and by leveraging data from the PublicWWW project [23], we study and quantify the problem of web resources loaded from publicly writable S3 buckets.

We found that, out of the representative subset of buckets we analyzed as of June 2018, and notwithstanding the public awareness raised by recent high-profile events, 11.01% of the S3 buckets are public (i.e., their files can be listed), 8.46% are readable, and 2.29% are writable. Among these, we found dangerous cases of readable buckets that could be leaking sensitive data, such as private keys and database dumps. Finally, we found 5,196 websites that load resources directly from S3 buckets into their homepage. Out of this set, 175 websites are vulnerable to defacement and 39 websites are vulnerable to malicious injection, putting end-users at risk.

Contributions. To the best of our knowledge, we are the first to comprehensively investigate the security issues caused by misconfigurations in the access control rules of a cloud storage service, and the chain of their consequences.

In summary, we make the following contributions:

- We build an automated tool to perform security checks on Amazon S3 buckets, discovering publicly accessible buckets, and verifying their access policies.
- We show how our tool can investigate the usage of the Amazon S3 service, analyzing more than 240,000 S3 buckets and discovering misconfigurations that impact security and privacy, including writable buckets containing resources loaded by several popular websites.
- We propose and develop a mitigation to protect end-users from those vulnerable websites that rely on writable buckets. Specifically, as part of this research, we release[1] a prototype of browser extension that checks if a website is requesting resources hosted in untrusted, writable buckets, and prevents the browser from loading such resources.

Ethical Considerations. Our large-scale scan raises important ethical considerations. In Section 6, we describe the countermeasures that we took to prevent potential harm. In particular, we do not publish our list of buckets as this could be abused by malicious attackers. Instead, as further discussed in Section 6, we notified responsible entities, including Amazon AWS, to give them the chance to timely fix the identified misconfigurations.

```
http[s]://<BUCKET_NAME>.s3[-region].amazonaws.com/
http[s]://s3[-region].amazonaws.com/<BUCKET_NAME>/
```

Figure 1: URLs format Amazon S3 buckets.

2 BACKGROUND

2.1 Amazon S3

Amazon S3 (Simple Storage Service) is a cloud-based service that stores arbitrary data objects organized into "buckets." Users can create buckets belonging to any of the available AWS regions, and assign a globally unique name (regardless of the region). Users can store and retrieve objects from their buckets either using the web-based console, or programmatically, using a REST interface or the AWS SDK. This gives application developers access to a highly reliable, scalable and fast data storage infrastructure through libraries available for a variety of programming languages.

Each resource—be it a bucket, or an object—is identified by a resource URI. Amazon S3 supports virtual-host-style and path-style URLs (Figure 1). Additionally, in order to reference S3 resources from custom domains, if a bucket name has the format of a valid domain name (e.g., *storage.example.com*), users can set up a `CNAME` resource record in the domain's nameservers to point to the virtual-host-style bucket URL (e.g., *storage.example.com.s3.amazonaws.com*).

Amazon S3 supports various access control policies that can be attached either to resources (buckets and objects) or to users. It is possible to grant particular permissions to specific users and AWS accounts, to non-authenticated users (group `AllUsers`), or to any authenticated AWS account (group `AuthenticatedUsers`). There are two ways to specify resource permissions: First, each bucket and object is associated with an access control list (ACL), written in a S3-specific XML schema or built through a graphical user interface in the Amazon AWS web console; and, second, buckets can be associated with a bucket policy, specified as a JSON file according to the Amazon IAM policy language, used across AWS. Resource-based policies and ACLs are stored as bucket sub-resources. With respect to ACLs, policies allow to express more complex rules: While an ACL should be explicitly applied to each resource, a single policy may include wildcards and variables to match the desired resources (e.g., to match objects in a specific sub-folder), as well as more specific conditions. User policies, instead, can be attached to a specific Amazon IAM user, group or role, and are used to grant or restrict access to resources where the parent AWS account owns or is granted access. Resources are private by default: The owner AWS account has full permissions, and no other account has any access right.

2.2 Threats

In this paper, we investigate misconfigurations in the access control rules of Amazon S3 buckets found in the wild. Such misconfigurations may inadvertently allow unrestricted listing (e.g., by applying the `S3:ListBucket` policy action to anonymous, unauthorized users, or by granting the `READ` permission to all users in the bucket ACL), reading (`S3:ListBucket` policy action, or `READ` ACL permission in some or all objects of a bucket), or writing (`S3:PutObject`

[1] https://github.com/necst/truster

policy action, or WRITE bucket ACL permission), paving the way to security or privacy issues. Specifically, such misconfigurations enable the following scenarios:

- *Data Leakage* (readable buckets). A bucket whose content is unintentionally made public may lead to privacy issues due to the leakage of confidential or sensitive data.
- *Web Resource Infection* (writable buckets). Objects hosted in a publicly writable bucket can be overwritten with malicious content. This may pose different threats according to how the object is referenced [20]. Threats range from defacement [17] (e.g., a writable resource loaded by a website), to infection of end-user computers (e.g., a writable executable file referenced from a trusted website), to web site compromise (e.g., a writable JavaScript asset loaded by a web site). In the latter case, the malicious JavaScript may consume the resources of the end-user's computer (e.g., cryptocurrency mining script [14]), or target confidential information, given that it runs with the same origin of the compromised website.
- *Ransom Demand* (writable buckets). Attackers may, similarly to well-known ransomware [7], encrypt objects and ask owners for a ransom, following the attack scheme already employed for unsecured MongoDB instances [27].
- *Domain Name Trust Exploitation* (writable buckets). If a writable bucket is aliased to a custom subdomain of an organization that is usually considered trusted (e.g., a well-known company), the ability to serve arbitrary files from a "trusted" subdomain may be used to carry on a targeted phishing attack that exploits the implicit trust in the domain name (e.g., `http://storage.<trusted-corp>.com`), or bypass security whitelists based on the domain name. For example, this can happen with overly broad web content security policy (CSP) rules of the form `*.<trusted-corp>.com`, resulting in the bypass of the policy, or in case of domain-name based whitelists implemented at the firewall or application proxy level.
- *Dangling Subdomain Takeover* (unclaimed buckets). If a DNS CNAME record that points to an Amazon S3 bucket is not removed or updated when the pointed bucket is deleted, then it becomes dangling: It points to an abandoned service that can be trivially taken over by an attacker [16]. In fact, as Amazon S3 does not require domain ownership verification before allowing access to a bucket through a custom CNAME alias, an attacker can simply create a bucket under their AWS account with the (dangling) subdomain as a name, and take over the subdomain. This scenario is orthogonal to the previous ones, as it does not stem from a misconfiguration of the bucket permissions, but from a missing check in the deletion process when a custom domain is used. Furthermore, it is not a cloud storage-specific issues, but rather an issue stemming from the fact that the service supports custom subdomains, and that, to date, lacks domain ownership validation. However, this is a relevant threat model, as according to how the original bucket is used, this paves the way to resource infection, or to exploiting the users' trust in the domain name.

3 METHODOLOGY

To study the extent and the impact of misconfigurations in Amazon S3 buckets in the wild, we automatically discover and verify Amazon S3 buckets that have their content publicly listable. Our methodology is depicted in Figure 2. First, we collect candidate bucket names through three different methods (Section 3.1); then, we filter the candidates that correspond to an existing bucket (*scanner*), and verify whether the bucket is set as publicly readable or publicly writable (*inspector*). Last, we inspect if the writable buckets we found are referenced from publicly accessible websites, and, thus, can be used to carry out resource infection attacks (Section 3.2).

3.1 Enumeration & Data Collection

First of all, we enumerate a representative subset of Amazon S3 buckets. This step is not trivial: Indeed, a public directory of valid S3 bucket names is not available, and bucket names can be composed of an arbitrary sequence of three or more characters. We adopt three complementary approaches: First, we use mutations over dictionaries to generate a list of possible, candidate, bucket names, and verify whether they are valid through the Amazon S3 API. Secondly, we leverage web crawling to locate S3 buckets linked or referenced from popular websites. Third, we leverage publicly available sources (i.e., Amazon IP address ranges, VirusTotal Intelligence,[2] and RiskIQ[3]) to obtain passive DNS data, and look for resolutions pointing to S3 domains.

Candidate Generation. To generate candidate names for buckets, we use a mixture of acronym enumeration and generations of variations over a word dictionary. First of all, we generate bucket-name candidates by enumerating acronyms: We generate all the possible combinations of words composed of 3 and 4 characters. The rationale behind this step is that users tend to create short and easy to remember names, or to use acronyms. Second, we start from a dictionary of English words, and randomly mutate them to generate more bucket names: We concatenate multiple words, and randomly change one or more letters in a word. The rationale is that, when a desired bucket name is already registered, users tend to apply small modifications (e.g., removing a letter) to find an available one. In particular, we use two types of mutations: (a) we remove random letters from a word or concatenation of words, and (b) we duplicate a letter in a random position. The number of letters added or removed decreases exponentially in function of the number of mutations already happened. Finally, using the same enumeration and mutation techniques we generate others bucket names adding prominent top-level domains (TLDs) at the ends of generated names. TLDs are chosen with a probability proportional to their prominence, i.e., more frequent TLDs such as `.com` have higher probability to be used. This choice is due to the fact that, when using a custom CNAME alias, a bucket must be named as the subdomain pointing to it. This candidate generation process yields two lists: with and without TLDs.

Web Crawling. We crawled the home pages of the top 1 Million websites according to Alexa [1], to discover any resource linked from S3 buckets. For crawling, we used Google Chrome in headless mode. Using a full-fledged browser that included a JavaScript

[2] https://www.virustotal.com/#/intelligence-overview
[3] https://www.riskiq.com/

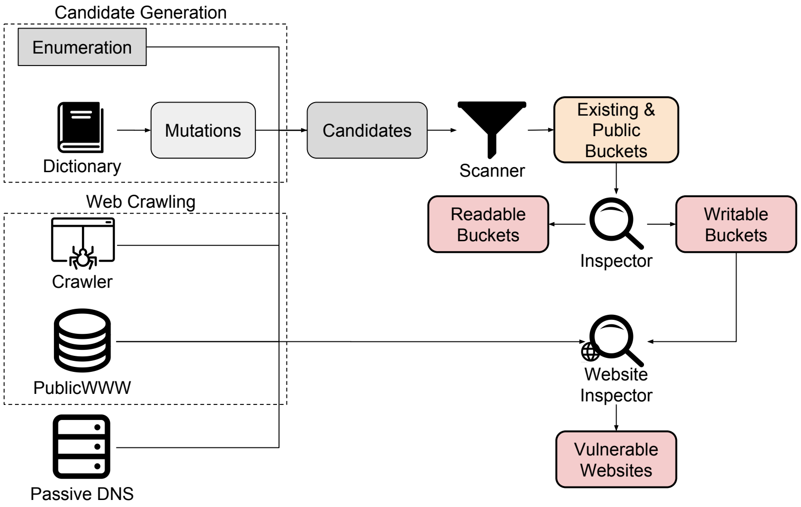

Figure 2: Overview of our methodology to discover misconfigured buckets. First, we collect candidate bucket names (Section 3.1); then, we discover existing buckets (Scanner), and verify whether they are publicly readable or writable (Inspector). Last, we check if any writable bucket is referenced from a website, and, thus, can be used to carry out a resource infection attack (Section 3.2).

interpreter allows the detection of links to S3 buckets built at runtime. We also resolve the CNAME records of domains linked in external resources and check whether they point to S3 buckets. Moreover, we leverage Google "dorks" and PublicWWW[4] to find further websites containing references to subdomains of Amazon S3 (i.e., bucket names).

Passive DNS. We downloaded the Amazon AWS IP address ranges,[5] and selected those assigned to the S3 service. Then, we leveraged VirusTotal Intelligence and RiskIQ to perform reverse lookup queries. In fact, VirusTotal runs its own passive DNS replication service, built by storing the DNS resolutions performed by visiting URLs and executing malware samples submitted by users [29]. For each IP address, VirusTotal and RiskIQ return a list of domains that resolve (or have resolved) to that IP address. Among such domains, we select those matching the S3 syntax (Figure 1) and extract the bucket names.

3.2 Security Analysis

Scanner. Starting from the list of candidate bucket names, our scanner builds the Amazon S3 bucket URLs according to the format shown in Figure 1. Then, it sends a request over HTTPS to the S3 API, and determines, according to the response, whether each candidate bucket name is an existing bucket or not and, if the bucket exists, whether it is public (i.e., object listing is enabled for non-authenticated users). If the bucket does not exist, the API replies with "`NoSuchBucket`" and a HTTP 404 status header; if the bucket exists and is not publicly listable, the API returns an `AccessDenied` error with a HTTP 403 status code (Figure 3); otherwise, it returns an XML document with the bucket file listing (including file names with extensions). Note that, as further discussed in Section 6, for privacy reasons we do not access (i.e., read or store) any personal file contained in such buckets.

```
HTTP/1.1 403 Forbidden
x-amz-bucket-region: ap-southeast-2
x-amz-request-id: 73E5693FA382F20D
x-amz-id-2: tMhK7yPboPa04kV/oypK9WVrYGqj8NR4QI7IP8QakzdCD
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 19 Mar 2018 13:22:24 GMT
Server: AmazonS3

<Error>
        <Code>AccessDenied</Code>
        <Message>Access Denied</Message>
        <RequestId>4C4B01F64E808F69</RequestId>
        <HostId>
                zPQX00xzxUJLH704xQLZFg9toDHjPOBk+E1JZOR5xouInq
        </HostId>
</Error>
```

Figure 3: Example of Access Denied response.

Inspector. The Inspector analyzes the discovered existing and public buckets, and checks their permission settings. Specifically, we want to verify if the files in the buckets are readable and/or writable. In order to safely test whether the files in a publicly listable bucket are readable, the Inspector queries the API using the minimum access level possible. Indeed, accessing users private files raises important privacy and ethical considerations. Thus, we determine whether a file is readable only by the returned status code. The Inspector sends a HTTP `HEAD` request, which returns only the response headers without retrieving the file content.

Since each file can have unique permissions, we cannot conclude that the entire bucket is readable by verifying only one file. However, verifying all the files would be too time-consuming (many buckets contain hundreds of thousands of files). Hence, we randomly sample and verify the readability of k files. If exactly k are readable, we

[4] https://publicwww.com/
[5] https://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html

assume the entire bucket is readable. If less than k are readable, we say that the bucket is partially readable. By setting the value of k, our tool can be configured according to different confidence levels. Otherwise, we conclude that the bucket is not readable. We also attempt to retrieve the access control list of each existing bucket with a HTTP GET request to /?acl, to check whether it is publicly readable and to study the access control policies. Note that the access control list does not contain any user personal file.

To test whether an existing bucket is writable, there are no other choices than trying to create a new object using a HTTP PUT request, and checking if the operation is allowed. Note that our Inspector does not modify or overwrite any existing object, thus it is harmless for the bucket owner. Actually, it tries to create a plaintext object containing a message that responsibly discloses the security issue and explains how to fix it. To minimize the chances of colliding with an existing object, the file name is a 128-byte randomly generated string, and the Inspector checks whether an object with the generated name exists before attempting to write.

Website Inspector. To study the prevalence of potential resource infection issues, we leverage our crawler and data from PublicWWW to identify websites that load resources from S3 buckets. Specifically, our crawler inspects the following HTML elements: `script::attr(src)`, `link::attr(href)`, `a::attr(href)`, `img::attr(src)`. Moreover, we queried PublicWWW for pages containing S3 URIs, and then crawled the matched pages. If the S3 buckets referenced by a website happen to be writable, then that website is vulnerable to resource infection: Attackers could tamper with existing, loaded resources to deface the website (e.g., replacing an image) or deliver malicious content (e.g., injecting an exploit kit in a client-side loaded resource).

4 EXPERIMENTAL RESULTS

4.1 Experimental Setup

We implemented our tool in about 1,600 lines of Python code, leveraging its `multiprocessing` module to parallelize the analysis of buckets. We leveraged Selenium WebDriver to automate a headless instance of Google Chrome in our crawling. We used $k = 30$ in our Inspector as the sample set size to determine if a bucket is fully readable. Also, we leverage VirusTotal APIs to retrieve passive DNS data. Finally, as an input dictionary to generate our candidates, we leveraged the SCOWL (Spell Checker Oriented Word Lists) list of American English words, as implemented in the Debian wamerican package.[6]

We deployed our tool on a 40-core Intel Xeon machine with 378GB RAM, running Ubuntu 16.04. We performed the scanning of S3 buckets and the analysis of the results between February and June 2018.

4.2 Enumeration & Data Collection

In this section, we evaluate how each of our data collection method contributed in discovering existing buckets.

Candidate Generation. We generated 8,600,448 candidate bucket names. The enumeration of all possible combination of 3 and 4 characters covered 5.5% of the candidates. Single words from the dictionary covered 32.74%. Concatenation of two words covered 61.76% of the candidates. Among all of them, we applied a mutation (i.e., adding or removing characters) to 31.64% of the candidates, and we also duplicated candidates adding a TLD in 50% of the cases. From this list of candidates, we identified 132,686 existing buckets. Among these the acronyms contributed to the 3.76%, single words to the 93.41%, and concatenation of two words to 2.83%. Of the entire list of discovered bucket names, 62,061 (46.77%) were generated applying at least one mutation. Then, we studied how much each mutation contributed to the discovery of existing buckets. Specifically, removing one or more characters from English words contributed to the 37.55% of the discovered buckets, adding one of more characters contributed to the 4.38%, and adding a TLD contributed to the 4.84%.

Table 1: Scanning results summary.

Scan Data	No. Elements
Generated Candidates	8,783,964
Existing Buckets	240,461
Public Buckets	34,145
Readable Buckets (Total)	27,492
Fully Readable Buckets	20,496
Partially Readable Buckets	6,996
Writable Buckets	6,599
Buckets with readable ACL	13,046
Non-listable buckets with readable ACL	5,843

Web Crawling. We collected 7,627 candidates from which we identified 2,468 existing S3 buckets. Specifically, we identified 2,196 distinct S3 buckets from crawling the Alexa top 1 Million visited websites, and 353 different S3 buckets querying PublicWWW.[7]

Passive DNS. From the Amazon S3 IP address ranges, we collected 318,976 IP addresses. Among these, VirusTotal and RiskIQ had data available for 21,558 IP addresses. However, among the 297,418 missing IP addresses, 296,520 are probably not allocated to any machine. In fact, we sent ICMP echo requests (ping) to each IP address to verify whether such hosts were available.[8] Then, by querying VirusTotal and RiskIQ for passive DNS data, from the 20,558 IP addresses we obtained 651,643 domain names, 217,111 of which matched the S3 syntax (`*.s3[-region].amazonaws.com`). Finally, from such domains we discovered 105,307 additional existing buckets.

4.3 Scanning Results

Table 1 shows a summary of the results of our scan. We generated a total of 8,783,964 unique candidates, which allowed our Scanner to discover 240,461 existing buckets. 34,145 of them (14.20%) host publicly listed content. Out of these, we found that 27,492 buckets (80.51% of the publicly listable buckets, or 11.43% of the existing buckets analyzed) contain readable files. Among these,

[6] https://packages.debian.org/sid/wamerican

[7] We leveraged the free version of PublicWWW, which indexes the top 3M websites and limits the results of complex queries.

[8] While we cannot certainly affirm that the hosts that did not respond to our ping were unallocated or down, none of the IP addresses missing from VirusTotal and RiskIQ replied, while other S3 addresses did, suggesting that Amazon S3 servers are configured not to block ICMP requests.

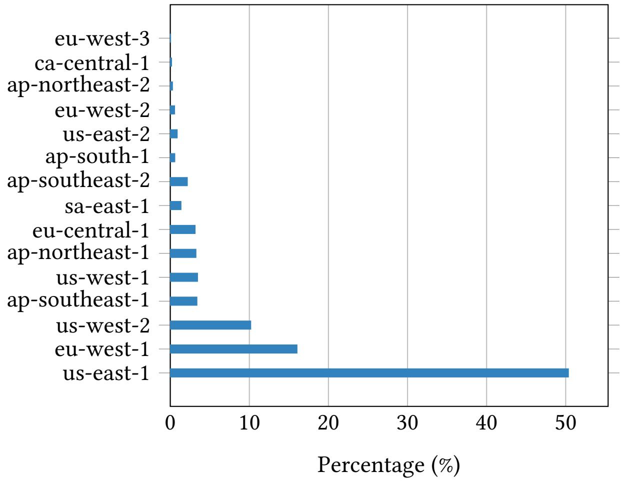

Figure 4: Distribution of the 240,461 buckets that we analyzed over the different regions.

20,496 (74.55%) are fully readable, while 6,996 (25.44%) are only partially readable (i.e., less than *k* files are readable). Also, our Inspector found 6,599 (2.74% of the existing bucket analyzed) writable buckets.

Furthermore, we found that 13,046 buckets (5.43%), almost evenly distributed between publicly listable and non-publicly listable, have their access control list publicly readable (i.e., READ_ACL). We do not deem this a security issue per se, as the permissions for the anonymous account are easy to infer, and the ACL discloses only the canonical ID and display name of the owner AWS account, as well as the IDs of other authorized users. However, by analyzing these ACL policies, we found that 3,415 buckets have their ACL writable by the anonymous user and 3,446 buckets have their ACL writable either by the anonymous user or by any authenticated AWS user (we remark that checking whether an ACL is writable is a "passive" operation, as it only requires to read the ACL XML file). Out of these, 93 buckets are not writable neither by the anonymous user nor by any authenticated user, and 101 buckets are not readable. This means that, for this small set of buckets, an attacker may easily upload a new ACL, read or write any object, and change the ACL back to the previous state. We also found out that, out of the buckets with publicly readable ACLs, only 1,647 buckets are granting any kind of permission to the generic authenticated user. We suspect that this low number is due to the fact that the AuthenticatedUser group is not readily accessible from the AWS web console. Instead, 777 buckets are explicitly granting permissions to other AWS accounts; although most bucket owners grant permission only to a small number of accounts (median: 1 account, average: 1.83 accounts), the maximum number of other accounts we found explicitly listed in an ACL is 22. Only 1,128 buckets grant write permission to the LogDelivery group: The great majority of the buckets with readable ACLs has server access logging disabled (the default option).

AWS Regions. Figure 4 shows the distribution across 15 AWS regions of the 240,461 buckets that we discovered during our scan.

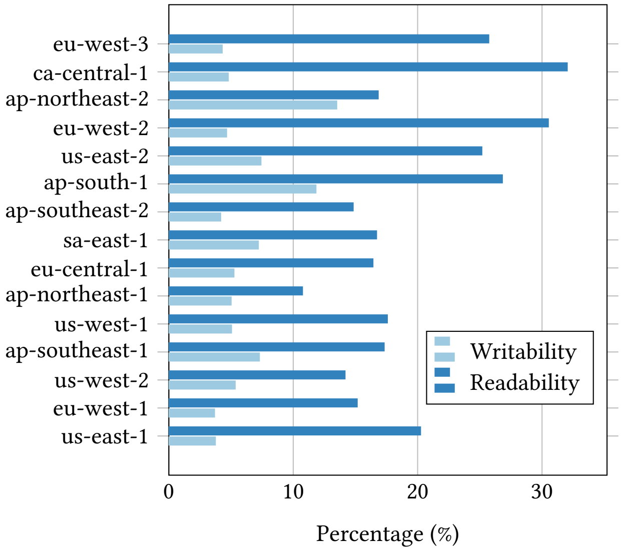

Figure 5: Number of readable/writable buckets per region, normalized by the number of existing buckets of the region. This is an indicator of how much a region is "misconfigured."

Instead, Figure 5 shows the amount of readable and writable buckets per region, normalized by the number of existing buckets of the same region. This represents an indicator of how many security issues a certain region has. Interestingly, some regions (e.g., ca-central-1, eu-west-2) have a higher percentage of readable buckets, and others (e.g., ap-northeast-2, ap-south-1) have a higher percentage of writable ones.

File Types and Data Leakage in Public Buckets. The public buckets we found contain 5,319,033,170 objects in total. To understand which file types are more at risk while preserving owners' privacy, we conservatively relied on file name extensions. Table 2 shows the 20 most common file extensions in public buckets, along with their popularity in readable and writable buckets. Then, we aggregated extensions in higher-level file types (e.g., Images, Code). For instance, we mapped .jpg, .jpeg, .png, and .gif to the Image file type. Figure 6 shows the percentage of buckets containing at least one file of four common file types (Image, Code, Executable, Text-data). In particular, about 19% of writable buckets contain executables or source code. This is particularly dangerous because attackers can tamper with such files, injecting malicious code that users may end up executing. Table 3 instead shows a summary of the potential data leaks that we identified in 27,492 readable buckets.

4.4 Vulnerable Websites

Crawling the home pages of the Alexa top 1 Million visited websites, we found 4,056 websites that load resources from 2,196 distinct S3 buckets. Similarly, querying PublicWWW, we found 1,178 websites that load resources from 353 different S3 buckets. In total we collected 5,196 websites relying on 2,468 buckets. Then, we analyzed

Table 2: Top 20 most common file extensions in public buckets, and relative percentages in readable, and writable buckets.

File Type	Public	Readable	Writable
.jpg	2,910,267,542 (54.71%)	2,558,262,624 (59.95%)	235,351,378 (31.16%)
.gz	287,907,921 (5.41%)	197,515,636 (4.63%)	11,592,423 (1.54%)
.png	269,629,214 (5.07%)	242,195,095 (5.68%)	64,326,347 (8.52%)
.jpeg	120,727,775 (2.27%)	113,709,980 (2.66%)	10,080,934 (1.33%)
.json	76,712,690 (1.44%)	75,143,122 (1.76%)	1,557,601 (0.21%)
.txt	67,364,012 (1.27%)	66,602,965 (1.56%)	2,892,145 (0.38%)
.html	56,107,720 (1.05%)	51,083,841 (1.20%)	33,230,611 (4.40%)
.pdf	55,077,872 (1.04%)	38,526,176 (0.90%)	13,162,004 (1.74%)
.gif	45,673,393 (0.86%)	42,632,503 (1.00%)	4,958,141 (0.66%)
.webp	36,731,079 (0.69%)	36,678,489 (0.86%)	1,921,274 (0.25%)
.hll	35,386,943 (0.67%)	0 (0.00%)	35,386,943 (4.69%)
.mp3	24,339,969 (0.46%)	23,359,348 (0.55%)	2,537,729 (0.34%)
.js	23,753,672 (0.45%)	9,404,205 (0.22%)	2,061,558 (0.27%)
.xml	19,816,767 (0.37%)	18,170,613 (0.43%)	10,423,540 (1.38%)
.ts	18,414,462 (0.35%)	17,457,380 (0.41%)	2,030,012 (0.27%)
.text	18,223,671 (0.34%)	18,223,563 (0.43%)	213 (0.00%)
.mp4	15,310,901 (0.29%)	14,378,514 (0.34%)	1,618,013 (0.21%)
.log	10,499,793 (0.20%)	8,691,642 (0.20%)	5,252,765 (0.70%)
.zip	10,323,009 (0.19%)	5,185,148 (0.12%)	2,821,950 (0.37%)
.docx	8,324,004 (0.16%)	5,831,401 (0.14%)	2,162,751 (0.29%)
No ext.	1,063,128,472 (19.99%)	626,339,218 (14.68%)	272,213,269 (36.04%)
Total	5,319,033,170	4,267,620,813	755,204,687

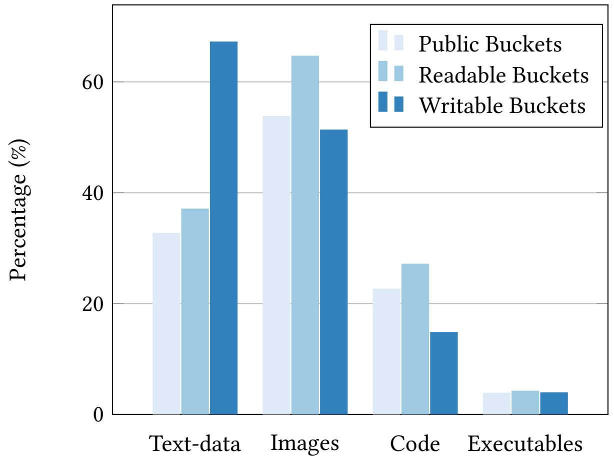

Figure 6: Percentage of buckets containing at least one file of four file types. We aggregated file extensions to higher-level file types.

such buckets to verify whether they are writable, and what the impact for the websites’ users is. The results, summarized in Table 4, are as follows:

- *Defacement.* We found 175 websites vulnerable to defacement. In fact, these websites load, from writable buckets, resources that attackers can manipulate to alter the layout and content of the websites. While this may seem harmless, it can have a dangerous security and reputation impact. Consider the homepage of the website of an important newspaper. If an attacker succeeds in modifying the caption image, she can inject fabricated content or a picture that contains phishing text (e.g., asking users to visit a certain website or to reveal private information). Hence, users would see such content on the homepage, and likely trust its content since it comes from a well-known source.
- *Code Injection.* The scenario is even more dangerous when websites load JavaScript or any other kind of executable

Table 3: Examples of sensitive exposure we identified.

Type	File	No. Buckets	No. Resources
Key Material	.pem,	84	335
	.p12,	17	98
	.pfx,	13	112
	.key (Keys)	17	361
Databases	.sql (Dumps)	249	2,825
Backups	.bak (Generic)	169	8,911
Financial Information	.qdf (Quicken Data)	5	5
Password DB	.kdbx (KeePassX)	4	4
	.kdb (KeePass)	1	1

code from writable buckets. In fact, in this case attackers can inject malicious code into the loaded resources and indirectly deliver it to the users visiting the website. We found 39 vulnerable websites that load JavaScript, HTML, executable, PDF, and GZIP resources from writable buckets.

- *Subdomain Takeover.* In this scenario, attackers can register non-existing, linked buckets, and directly load content into vulnerable websites. This can expose to both defacement and malicious injection, depending on the loaded resource. We found 13 websites loading resources from dangling buckets.

Examples. We found the website of a national South American newspaper loading two images in the homepage from a writable bucket. Hence, an attacker could tamper with these images to inject and display arbitrary content into the newspaper homepage. Another example is the website of a company that provides secured background screening solutions, which we found loading JavaScript and CSS resources in the homepage from a writable bucket, therefore being exposed to defacement and injection attacks.

5 MITIGATION

While the specific countermeasure for misconfigurations in S3 buckets is trivial (e.g., change the access control policy), the results of our experiments highlight a typical situation that arise when developers adopt new technologies and services. As also shown by a recent study [10], our analysis unveils the need for stricter default policies and automated tools that check for common configuration errors and warn users of their security implications. For instance, in July 2017 Amazon AWS sent emails to the owners of public buckets to warn them of the security implications that such access policy may have [6]. Similarly to what has been implemented in major browsers, which mark HTTP pages as *Not Secure* [25], we need mechanisms that notify users when their access policies are potentially misconfigured. In accordance with this rationale, the AWS web console marks as “Public” any bucket that has any ACL permission granted to the anonymous user; furthermore, Amazon recently released for free the AWS Trusted Advisor, a tool to help their customers better secure their data by providing S3 bucket permissions check [3]. This is certainly a good step. However, the AWS Trusted Advisor lacks fine-grain inspection capabilities. In fact, setting bucket permission as publicly readable does not necessarily bring security or privacy issues, as long as no sensitive data is stored on the bucket. Moreover, Amazon AWS recently allowed users to enable files encryption on their buckets [4]. However, this feature cannot be enabled when files are meant to be publicly accessible, for instance when they are loaded by websites.

A Tool for Bucket Owners. As part of this research, we release a web-application[9] through which bucket owners can safely and privately check the access policy of their S3 buckets, verifying whether the buckets are public, readable, or writable. Different from AWS Trusted Advisor, our tool not only checks for bucket permissions, but also verifies whether readable buckets might contain sensitive information by looking at file extensions. For instance, we trigger an alert for `.pem` files, database dumps, backups, and so on. We foresee this approach to be embedded into cloud storage services backends and applied at upload time, therefore preventing users from unintentionally uploading sensitive data. Note that, the web application only accesses the minimum amount of data in order to provide the service to the users.

A Tool for Client-side Protection. While the aforementioned tool can be used by buckets owners to analyze security issues of their own buckets, writable buckets can affect the security of end-users when resources of such buckets are loaded by websites. Hence, we designed a client-side solution to protect end-users. We designed a browser extension that, at runtime, checks whether the rendered page loads resources from S3 buckets (by matching the URL formats) and queries our backend if such buckets are writable. If so, we consider the resources contained in the bucket *untrusted* and prevent loading them. We implemented our browser extension[10] for `Google Chrome`, allowing users to set different aggressiveness levels (e.g., do not load untrusted resources, ask before loading, only show a warning). Specifically, we leverage the `chrome.webRequest.onBeforeRequest.addListener` API to register a callback that gets called before loading resources, and we check whether they are loaded from a S3 bucket. In order to identify untrusted buckets, our Chrome extensions downloads from our backend a list of writable buckets. When providing such blacklist, our backend hashes all the bucket names. In the same way, our Chrome extension hashes the bucket name to be checked and verifies whether the hash is contained in the blacklist. In this way, our backend does not reveal in plaintext the names of the writable buckets. At the same time, since the check is performed client-side, this mechanism prevents privacy issues for users (i.e., sending information about the websites they visit). The list of untrusted buckets can be updated daily. As a result, when a website tries to load a

[9] https://bucketsec.necst.it/

[10] https://github.com/necst/truster

Table 4: Number of vulnerable websites and relative resources loaded from writable buckets.

	Loaded Resources												
Vulnerability	**JPG**	**PNG**	**JS**	**CSS**	**GIF**	**ICO**	**SVG**	**JSON**	**HTML**	**EXE**	**GZIP**	**PDF**	**Tot**
Defacement	130	80	26	12	13	8	6	3	1	-	-	-	175
Injection	-	-	26	-	-	-	-	-	1	1	1	12	39
Dangling	3	6	2	-	1	1	-	-	-	-	-	1	13
Total	130	80	26	12	13	8	6	3	1	1	1	12	191

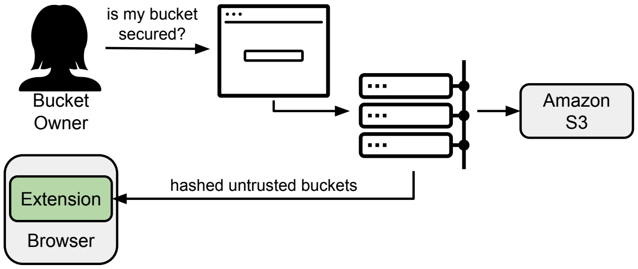

Figure 7: Overview of our mitigation. Bucket owners can check their access policies through a web-application. Client-side users avoid loading resources hosted in untrusted, writable buckets through a browser extension.

resource from an untrusted bucket (i.e., writable), the extension prevents it from being loaded.

Alternatively, the Chrome extension can query our backend for unknown buckets, which are analyzed in real-time. However, this option raises the aforementioned privacy issues for users. In fact, doing so our backend would collect data about the users browsing. Hence, we let the users choose whether to activate this option.

Figure 7 shows a high-level overview of our mitigation mechanism, within the two scenarios: buckets owners that want to check the issues in their buckets, and end-users that want to be protected from loading untrusted resources.

6 DISCUSSION

Ethical Consideration. By conducting a large scan, our work raises important ethical considerations, similarly to other previous measurements [26]. In fact, the only way to test whether a bucket is writable is to write a file and check if such operation has succeeded. Therefore, we carefully considered the impact of our experiments and we took several precautions to prevent potential harms.

We never attempted to read users data. In fact, to test whether a certain file is readable we perform a HTTP `HEAD` request, which returns only the HTTP response headers without the body (i.e., file content), and we only check the HTTP response status code. Indeed, our analysis of sensitive exposure relies only on the file extensions, identifying those ones that characterize typical sensitive data, and never on the data (that we do not access nor store).

We only accessed S3 buckets using HTTPS, hence avoiding exposing users to man in the middle risks. When testing whether buckets are writable, we only wrote a new, plaintext file containing text that discloses the security issue and informs on how to fix it. The file's name is a 128-byte long randomly generated string. Also, to avoid overwriting any existing file, we checked via a `HEAD` that our filename did not exist on the remote bucket. Therefore, we did not modify any user data.

Finally, we do not publish the list of affected buckets as this could be abused, putting users at risk. Instead, as discussed in the following paragraph, we notified responsible entities to timely and properly fix access policies.

Disclosure. First, we contacted webmasters of the vulnerable websites that we identified. We visited each website to obtain a contact email, and, in the case we did not find any email address, we sent the notification to both `info@domain.com` and `webmaster@domain.com`. Regarding the misconfiguration of S3 buckets, we were not able to contact the owners as the bucket name is not publicly linked to any personal information (owner name or email). To address this, we disclosed our findings to Amazon AWS, which acknowledged the issues and proceeded to notify affected users. We also anonymized references to misconfigured buckets and we do not release any information that can be abused by malicious attackers.

Limitations. Our crawler only looked at the home pages of the top 1 Million domains. We plan to extend our evaluation by scraping the linked pages of each website in future work. While analyzing passive DNS data, we did not study whether further domains have a CNAME record that indirectly points to a S3 bucket. We plan to extend this study in future work.

7 RELATED WORK

Amazon S3. The problem of open S3 buckets was first raised by Robin Wood in a 2011 blogpost [31], which showed the results of a scan based on a small word-list. More recently, information security companies performed similar analyses [24, 28], and various projects are available on open-source code hosting platforms (e.g., GitHub) to verify the settings of S3 buckets or to help discovering open S3 buckets, e.g., for penetration testing engagements. However, such studies lack deep data analysis and scientific methodology. We filled this gap by performing the first comprehensive, large-scale analysis of the usage of cloud storage in the wild.

Security-oriented Web Measurement. Large-scale web measurement is a common tool in the security community to study phenomena such as the extent of common (mis)configurations with an impact on the security, as well as to understand the shape and relationships of various ecosystems.

In 2014, Nikiforakis et al. [21] investigated the ad-based URL shortening ecosystem, highlighting the dangers of this type of services due to unpredictable—and often malicious—advertisements. More recently, Dell'Amico et al. [8], motivated by recent attacks to SaaS and IaaS cloud service providers and by the increasing adoption of such services, investigated the dependencies between websites and cloud services, establishing a set of metrics to assist website and service operators in making informed choices about their Internet/cloud service dependencies and to minimize their risk exposure.

The use of JavaScript in web applications—often hosted in third-party services, such as CDNs or even S3 buckets—has been studied through various lenses: On one hand, Nikiforakis et al. [20] analyzed the implications of the inclusions of remote JavaScript code, while, on the other hand, Lauinger et al. [15] studied the security implications of the usage of (client-side) JavaScript libraries and how the attack surface of websites is affected by vulnerabilities in included libraries.

With respect to misconfiugurations, Eshete et al. [12] audited the security configuration settings of server environments in web application deployment and development; Springall et al. [26] presented a comprehensive analysis of how anonymous FTP has been deployed in the real world, discovering that there are more than

13.8M FTP servers on the IPv4 address space, of which 1.1M (8%) allow anonymous access and often expose sensitive data, with implications similar to those of misconfigured Amazon S3 buckets. Liu et al [16] performed a large-scale measurement to quantify the security threats posed by dangling DNS records, and, more recently, Borgolte et al. [5] exploited (temporarily) dangling DNS entries pointing to cloud IP addresses to deceive domain validation and obtain TLS certificates.

8 CONCLUSIONS

In this paper, we investigated security implications of using the Amazon S3 service, one of the most popular service offering cloud storage. We focused on identifying misconfigurations that affect users' privacy and security, with the goal of raising the awareness of a real-world security problem and warn users of its security implications. To do so, we developed an automated tool that discovers public S3 buckets and verifies their access policies. We performed a comprehensive, large-scale analysis of more than 240,000 buckets discovering that about 14% of them are public, about 11% are also readable, and about 2% writable. We then studied the consequences of such misconfigurations. We found more than 200 readable buckets leaking sensitive data, such as private keys and database dumps. Then, we studied how attackers can exploit writable buckets to harm end-users. We found that 19% of the writable buckets contain executables or source code, allowing attackers to tamper with such files and inject malicious content (e.g., backdoors). Even more dangerous, we found 191 vulnerable websites that load resources from writable buckets, and that could potentially be defaced or deliver malicious content. We concluded our study by proposing a mitigation to protect end-users from this threat. Specifically, we developed a browser extension that, communicating with our backend, verifies if the resources requested from a visited website come from an untrusted, writable, bucket, and prevents loading such insecure resources.

ACKNOWLEDGMENTS

We would like to thank our reviewers for their valuable comments and input to improve our paper. We would also like to thank the Amazon AWS security team for their prompt reply.

Politecnico di Milano received funding for this project from the European Union's Horizon 2020 research and innovation programmender the Marie Skłodowska-Curie grant agreement nr. 690972.

REFERENCES

[1] Alexa. The top 500 sites on the web. https://www.alexa.com/topsites.

[2] Amazon. Amazon s3. https://aws.amazon.com/s3/.

[3] Amazon. Aws trusted advisor's s3 bucket permissions check is now free. https://aws.amazon.com/about-aws/whats-new/2018/02/aws-trusted-advisors-s3-bucket-permissions-check-is-now-free/.

[4] Jeff Barr. New amazon s3 encryption & security features. https://aws.amazon.com/blogs/aws/new-amazon-s3-encryption-security-features/, Nov 2016.

[5] Kevin Borgolte, Tobias Fiebig, Shuang Hao, Christopher Kruegel, and Giovanni Vigna. Cloud strife: mitigating the security risks of domain-validated certificates. In *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS)*, 2018.

[6] AWS For Business. Aws sends warning emails to s3 bucket users. https://www.awsforbusiness.com/aws-sends-warning-emails-s3-bucket-users/, Jul 2017.

[7] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. ShieldFS: A self-healing, ransomware-aware filesystem. In *Proceedings of the ACM Annual Computer Security Applications Conference (ACSAC)*, 2016.

[8] Matteo Dell'Amico, Leyla Bilge, Ashwin Kayyoor, Petros Efstathopoulos, and Pierre-Antoine Vervier. Lean on me: Mining internet service dependencies from large-scale dns data. In *Proceedings of the ACM Annual Computer Security Applications Conference (ACSAC)*, 2017.

[9] Bob Diachenko. Global communication software and service provider left massive amount of data online. potentially exposing millions of subscribers. https://mackeepersecurity.com/post/global-communication-software-left-massive-amount-of-data-online, Sep 2017.

[10] Constanze Dietrich, Katharina Krombholz, Kevin Borgolte, and Tobias Fiebig. Investigating System Operators' Perspective on Security Misconfigurations. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2018.

[11] DigitalOcean. Spaces: Beautifully simple object storage. https://www.digitalocean.com/products/spaces/.

[12] Birhanu Eshete, Adolfo Villafiorita, and Komminist Weldemariam. Early detection of security misconfiguration vulnerabilities in web applications. In *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, 2011.

[13] Google. Google cloud storage. https://cloud.google.com/storage/.

[14] Radhesh Krishnan Konoth, Emanuele Vineti, Veelasha Moonsamy, Martina Lindorfer, Christopher Kruegel, Herbert Bos, and Giovanni Vigna. MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2018.

[15] Tobias Lauinger, Abdelberi Chaabane, Sajjad Arshad, William Robertson, Christo Wilson, and Engin Kirda. Thou shalt not depend on me: Analysing the use of outdated javascript libraries on the web. In *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS)*, 2017.

[16] Daiping Liu, Shuai Hao, and Haining Wang. All your dns records point to us: Understanding the security threats of dangling dns records. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2016.

[17] Federico Maggi, Marco Balduzzi, Ryan Flores, Lion Gu, and Vincenzo Ciancaglini. Investigating web defacement campaigns at large. In *Proceedings of the ACM ASIA Conference on Computer and Communications Security (ASIACCS)*, 2018.

[18] Microsoft. Azure storage: Secure cloud storage. https://azure.microsoft.com/en-us/services/storage/.

[19] Shaun Nichols. Guys, you're killing us! la times homicide site hacked to mine crypto-coins on netizens' pcs. https://www.theregister.co.uk/2018/02/22/la_times_amazon_aws_s3/, Feb 2018.

[20] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2012.

[21] Nick Nikiforakis, Federico Maggi, Gianluca Stringhini, M Zubair Rafique, Wouter Joosen, Christopher Kruegel, Frank Piessens, Giovanni Vigna, and Stefano Zanero. Stranger danger: exploring the ecosystem of ad-based url shortening services. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2014.

[22] Dan O'Sullivan. System shock: How a cloud leak exposed accenture's business. https://www.upguard.com/breaches/cloud-leak-accenture, Nov 2017.

[23] PublicWWW. PublicWWW: Source Code Search Engine. https://publicwww.com/.

[24] Rapid7. There's a hole in 1,951 amazon s3 buckets. https://blog.rapid7.com/2013/03/27/open-s3-buckets/, Mar 2013.

[25] Emily Schechter. A secure web is here to stay. https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html, Feb 2018.

[26] D. Springall, Z. Durumeric, and J. A. Halderman. Ftp: The forgotten cloud. In *Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.

[27] Liam Tung. Mongodb ransacking starts again: Hackers ransom 26,000 unsecured instances. https://www.zdnet.com/article/mongodb-ransacking-starts-again-hackers-ransom-26000-unsecured-instances/.

[28] Cyril Vallicari. Impact study - amazon s3 aws buckets configuration. https://www.httpcs.com/fr/actualites-cybersecurite/impact-study-aws-buckets-amazon-s3-configuration, Feb 2018.

[29] VirusTotal. VirusTotal Documentation: Searching. https://support.virustotal.com/hc/en-us/articles/115002739245-Searching.

[30] Mark Ward. Exposed amazon cloud storage clients get tip-off alerts. http://www.bbc.com/news/technology-42839462, Feb 2018.

[31] Robin Wood. Analysing amazon's buckets. https://digi.ninja/blog/analysing_amazons_buckets.php, May 2011.