The File That Contained the Keys Has Been Removed: An Empirical Analysis of Secret Leaks in Cloud Buckets and Responsible Disclosure Outcomes

Soufian El Yadmani
LIACS, Leiden University
Modat B.V.

Olga Gadyatskaya
LIACS, Leiden University

Yury Zhauniarovich
TU Delft

Abstract—**With the growing reliance on cloud services for storage and deployment, securing cloud environments has become critically important. Cloud storage solutions like AWS S3, Google Cloud Storage, and Azure Blob Storage are widely used to store vast amounts of data, including sensitive configuration files used in software development. These files often contain secrets such as API keys and credentials. Misconfigured cloud buckets can inadvertently expose these secrets, leading to unauthorized access to services and security breaches.**

In this work, we explore the issue of secret leaks in files exposed through misconfigured cloud storage. Our analysis covers a variety of file formats frequently used in development and focuses on different secrets that have diverse types of impact as well as the possibility for a non-intrusive validation. By systematically scanning a large collection of publicly accessible cloud buckets, we identified 215 instances where sensitive credentials were exposed. These secrets provide unauthorized access to services like databases, cloud infrastructure, and third-party APIs, posing significant security risks.

Upon discovering these leaks, we responsibly reported them to the respective organizations and cloud service providers and measured the outcomes of the disclosure process. Our responsible disclosure efforts led to the remediation of 95 issues. Twenty organizations directly communicated their actions back to us, promptly addressing the issues, while the remaining fixes were implemented without direct feedback to the disclosers. Our study highlights the global prevalence of secret leaks in cloud storage and emphasizes the varied responses from organizations in mitigating these critical security risks.

Index Terms—**Misconfigured cloud buckets; secret leaks; responsible disclosure.**

1. Introduction

Modern applications do not typically build all their functionality from scratch. Instead, they employ the functionalities provided by other services. Access to these services is protected with various types of credentials, such as usernames and passwords, tokens, API keys, and certificates.

Unfortunately, application developers sometimes fail to secure these credentials properly. This may lead to unauthorized access to private information or illegitimate use of resources, with potentially disastrous consequences [1], [2], [3]. For instance, unauthorized parties could gain access to the personal information of 296,019 Toyota Motors Corporation's customers due to the database access keys contained in a public GitHub repository [2]. What is more scary is that these keys were available for almost five years, from December 2017 to September 2022. Rahman et al. [4] found that the median lifetime of secrets hardcoded in infrastructure-as-code scrips is 20 months. According to the security company Truffle Security, 74 % of leaked secrets are never revoked [5].

During the last decade, the problem of leaked credentials has become so acute that modern version control system platforms had to take action. For example, GitHub[1] now provides a service for automatic code scanning for secrets, which was recently enabled for all push actions to stop secret leaks [6]. Unfortunately, there are many other channels of secret leaks. Researchers have carried out measurement studies to assess how frequently credentials leak through different channels, such as GitHub [7], [8], [9], [10], mobile applications [11], [12], Docker containers [1], virtual server images [13], [14], etc.

In this work, we explore how the secrets are leaked through improperly configured cloud storage. Recent news suggests that this channel may become a new source of leaked secrets [15], [16], [17]. Security issues and secret leaks related to misconfigured cloud buckets were also reported by researchers [18], [19]. Yet, so far no research has attempted to systematically measure the issue of secret leaks within the misconfigured buckets. Moreover, it is not yet clear, whether the owners of such buckets care about this leakage, or if, perhaps, many of the exposed buckets are made public intentionally, as, e.g., honeybuckets [20].

In recent years, several platforms have emerged that facilitate enumeration and access to open cloud buckets [18], [19]. Examples of such platforms include GrayhatWarfare [21] and Public Cloud Storage Search[2]. Additionally, several open-source tools for generating candidate bucket names and checking their accessibility are available, e.g.,

[1] https://docs.github.com/en/code-security/secret-scanning/secret-scanning-patterns

[2] https://github.com/nightwatchcybersecurity/public-cloud-storage-search

S3Scanner[3] or Slurp[4]. In our study, we leverage the GrayhatWarfare platform to measure the prevalence of secret leaks in open cloud buckets. We focus on different file types often used to store development secrets and scan using regular expressions for several popular types of secrets that have various significant impacts if exposed, e.g., financial loss or access to cloud infrastructure. As our goal is to assess the actual prevalence of secret leaks and their importance to bucket owners, in this work, we aim to find *only valid secrets*. To do so ethically, we selected the secret types that can be validated automatically in a *non-intrusive way*. Thus, these constraints suggest that our findings only estimate the lower bound of the secret leak problem.

In our study, we have discovered 215 instances of secret leaks, some of which could potentially compromise the security of an entire organization. By collaborating with a reputable CSIRT[5] organization, we have responsibly disclosed 160 of the leaks to the identified owners of these buckets, while the rest were shared with the corresponding cloud storage providers. We deployed a monitoring system that allowed us to track the effectiveness of our notification campaign automatically by checking the mitigation measures taken by the organizations. In total, 95 instances (59.37 %) of secret leaks were mitigated upon our notification. Twenty organizations replied to the disclosers' notification emails acknowledging that the secret leak was an actual issue.

Thus, the contributions of this work are the following:

1) We develop an *automated system* that scans the files of interest retrieved from misconfigured cloud buckets, detects candidate secret leaks and validates them in a non-intrusive way.
2) We *measure the prevalence of secret leaks in open cloud buckets*, showing that while the amount of leaked valid secrets is relatively low, their impact can be huge.
3) Secret leaks via misconfigured cloud buckets are a *global issue*. Vulnerable organizations that we have identified come from a wide variety of sectors, including finance, government, and IT technology, and from a variety of countries across the globe. The case studies we examined demonstrate the potentially devastating impact of the identified leaks.
4) We *examine the outcomes of the responsible disclosure process*, demonstrating that reporting to the owners results in a mitigation rate of 59.37 %. Moreover, some owners do not fix the issue properly: e.g., they restrict access to the bucket but do not revoke the secret. Thus, raising awareness about cloud security challenges is an important objective for the community.

2. Examining the Open Bucket Dataset

As a preliminary step in our study, we first examine the publicly available bucket data source to understand what kind of data we are dealing with and to plan an analysis of this data for detecting secret leaks.

TABLE 1: Number of Buckets by Cloud Provider

Update Date	**Cloud Provider**				**Total**
	ABS	**AWS**	**DOS**	**GCP**	
11-07-2023	100,702	375,451	8,096	163,424	**647,673**
24-09-2023	49,883	315,941	6,780	44,089	**416,693**
12-11-2023	50,392	311,960	7,018	72,484	**441,854**
19-01-2024	55,846	316,180	7,138	74,132	**453,296**

The Dataset. In this work, we rely on the GrayhatWarfare platform (Grayhat for short) [21], which was already used in research studies before [18], [19], [22]. This platform scans the Internet and collects links to open storage buckets at four major cloud service providers, namely Azure Blob Storage (*ABS*), Amazon Web Services (*AWS*), Digital Ocean Spaces (*DOS*) and Google Cloud Platform (*GCP*). Grayhat scans and enumerates the files these buckets contain, indexes the file properties, such as a path to or size of a file, and facilitates the search over this data[6]. Table 1 reports the statistics on the number of discovered open buckets on each platform for the last several scans at the time of writing. As we can see, the number of discovered open buckets varies considerably between the scans, and there appears to be no pattern in those changes, even across individual providers. There are several reasons for this. First, it is obvious that some bucket owners realize the issue and restrict access to their buckets between the scans. Second, the Grayhat platform has a takedown option, i.e., they can delete links to open buckets and files per request. This explains the reduction of the number of buckets and files. However, at the same time, new open buckets appear, contributing to an increase in the numbers.

In this research, we rely on the data about the buckets released by Grayhat on January 19, 2024. As we can see in Table 1 for that date, the AWS provider hosts the majority (69.75 %) of all discovered open buckets, GCP and ABS occupy the second (16.35 %) and the third (12.32 %) places correspondingly with numbers of the same order, while DOS tails the list (1.57 %).

Representativeness of Grayhat Data. In 2021, Cable et al. [18] estimated that Grayhat indexed 37 k AWS buckets, while their system Stratosphere discovered 190 k more. The Grayhat snapshot we used in the study contained information about 316 k AWS buckets. This means that while the snapshot in 2021 was not representative, over time Grayhat improved its indexing a lot. Another bucket indexing platform OpenBuckets[7] currently lists 416 k misconfigured AWS buckets. Overall, the total amount of misconfigured buckets is not known; there is no ground truth to check. However, we believe the Grayhat sample to be sufficiently large for the study purposes.

[3] https://github.com/sa7mon/S3Scanner

[4] https://github.com/0xbharath/slurp

[5] https://csirt.global/

[6] https://buckets.grayhatwarfare.com/buckets

[7] https://www.openbuckets.io/

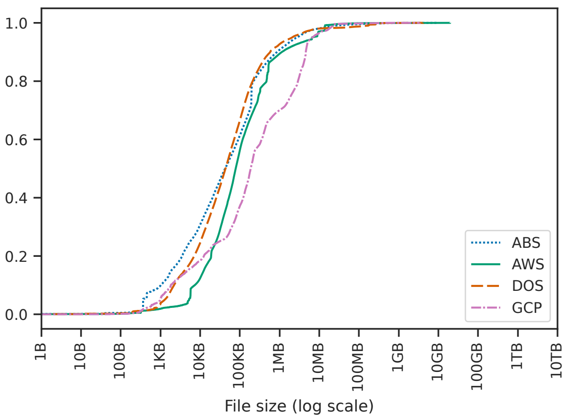

Figure 1: File size eCDF for each provider's buckets sample.

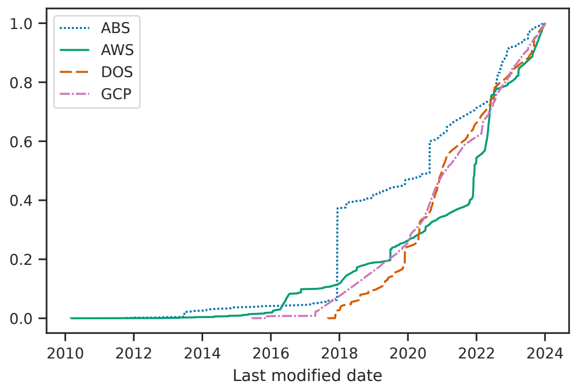

Figure 2: File last modified date eCDF for each provider's buckets.

Examining the Dataset. Table 1 shows that Grayhat provides a large dataset of open buckets. As each bucket is a snapshot of a filesystem, a full analysis of the dataset might be infeasible. We thus first strive to characterize the composition of this dataset. Note that the buckets might fluctuate widely in the storage requirements, amount and distribution of stored files, and other parameters. It is thus very challenging to identify all these different parameters and their distribution in the dataset. To get a preliminary, coarse estimate, we used the standard binary sample size calculation for an unknown proportion (50 %) of the variable of interest, with the 95 % confidence level and 5 % margin of error. For the AWS provider that contains the largest number of buckets (316,180), this results in a representative sample of 384 buckets[8]. Therefore, for a preliminary bucket data analysis, using the Grayhat API [21], on March 27-31, 2024, we retrieved from each cloud provider the metadata for a slightly larger number of 400 non-empty buckets. Table 2 reports the descriptive statistics for this sample (containing 1,600 buckets in total).

Some conclusions can be drawn from Table 2. First, the number of files in each bucket is relatively high. The mean number of files in each bucket is between 6 k and 16 k. Note that the maximum number of files per bucket in ABS, AWS and DOS is close to 1 M. The Grayhat platform imposes this number of results limit; therefore, in reality, the mean and maximum number of files in a bucket can be even larger. Second, the size of the files varies considerably. While the mean file size is about 2.1 MB, the largest files have a size of 2 TB, 19.88 GB, 19.83 GB and 255.65 GB for the ABS, AWS, DOS and GCP providers, correspondingly. Figure 1 shows the empirical cumulative distribution function of file sizes separately for each cloud provider. Although the distributions look pretty similar, as we can see from the figure, the AWS and GCP buckets contain files of smaller sizes, while the DOS and ABS buckets have larger files.

Third, as we can see from Table 2 some files have not been modified by the owners for years. Figure 2 provides a distribution of the last modified dates of files in the buckets.

Table 4 reports the Top 10 most popular file extensions in each provider's buckets. We see that the most popular extension across all the providers is `jpg`. The files with this extension constitute a substantial portion of all files. For the ABS, AWS, and DOS providers, the second most popular extension is `png`. In general, as we see from Table 4, media files (pictures, videos, and audio) are very popular for storing in cloud buckets: around half of the Top 10 most popular extensions correspond to these types of files. Note that `pdf` files are also very popular. They may contain private or sensitive data [23], [24], but we do not analyze them within the scope of this work as they are not likely to contain credentials.

While analyzing the metadata, we spotted that some files have the same name and size. We assumed that they might be the same one. Unfortunately, the Grayhat platform does not report the hashes of files, and we could not validate our assumption through the API. In order to reach our goal, we randomly picked 100 buckets from each provider and computed a hash digest for each file's content. Table 3 reports the results. As we can see, in total, we analyzed 2,844,924 files of more than 7.5 TB, of which 2,173,116 (76.39 %) are unique ones across all the providers. Table 3 shows for each individual provider, the percentage of duplicates varies from 17.72 % to 31.48 %. Thus, *deduplication of the files may lead to considerable storage space savings for cloud providers*.

To conclude, we see that the open bucket dataset varies widely in the number of stored files, their types, and sizes. Certainly, 400 buckets from each cloud provider is not a representative sample for the bucket data, given such a large distribution of parameters. However, as Table 2 shows, the open buckets dataset is indeed very large, with millions of files (some of which very big) available already in the studied sample. *A complete analysis of the open buckets dataset is infeasible* due to the extraordinary requirements for storage and processing capabilities. We thus set out to design a study that will examine some *representative*

[8] https://www.qualtrics.com/blog/calculating-sample-size/

TABLE 2: Buckets Sample Descriptive Statistics

Provider	Total # of Buckets	% of Buckets Analyzed	# of Files in the Sample	# of Files per Bucket			File Size			Last Modified Date	
				Min	Mean	Max	Min	Mean	Max	Earliest	Latest
ABS	55,846	0.72%	6,299,535	1	15,748.84	999,941	0.0B	1.60MB	2.00TB	2010-09-02	2024-02-27
AWS	316,180	0.13%	2,556,454	1	6,391.14	996,620	0.0B	1.55MB	19.88GB	2008-02-29	2024-02-27
DOS	7,138	5.60%	3,996,683	1	9,991.71	994,371	0.0B	1.55MB	19.83GB	2017-09-26	2024-02-26
GCP	74,132	0.54%	2,420,240	1	6,050.60	393,297	0.0B	3.83MB	255.65GB	2012-10-05	2024-02-27

TABLE 3: Files Descriptive Statistics for Downloaded Buckets

Provider	% of Analyzed Buckets	Total Size	# of Files				% of Duplicates		Max # of Duplicates
			All		w/ Searched Ext.		All	w/ Searched Ext.	
			Total	Unique	Total	Unique			
ABS	0.18%	771.99GB	275,513	216,375	513	327	21.46%	36.26%	442
AWS	0.03%	1.75TB	1,086,888	780,680	45,676	17,903	28.17%	60.80%	15,358
DOS	1.40%	4.32TB	1,167,530	960,655	8,056	2,972	17.72%	63.11%	3,502
GCP	0.13%	710.14GB	314,993	215,826	1,398	1,318	31.48%	5.72%	597

TABLE 4: Top 10 File Extensions in 400 Buckets of Each Provider

ABS			AWS			DOS			GCP		
Ext.	**#**	**%**	**Ext.**	**#**	**%**	**Ext.**	**#**	**%**	**Ext.**	**#**	**%**
jpg	3,668,549	58.24%	jpg	793,239	31.03%	jpg	1,842,461	46.10%	jpg	664,601	27.46%
png	1,110,133	17.62%	png	375,024	14.67%	png	576,014	14.41%		402,631	16.64%
pdf	303,271	4.81%	volt	317,562	12.42%	pdf	553,122	13.84%	jpeg	333,276	13.77%
jpeg	245,337	3.89%	gz	288,226	11.27%	jpeg	256,103	6.41%	png	270,649	11.18%
js	212,922	3.38%	js	152,744	5.97%	docx	250,027	6.26%	webp	222,246	9.18%
json	198,596	3.15%	json	123,968	4.85%	doc	60,017	1.50%	js	150,342	6.21%
	173,868	2.76%	pdf	110,872	4.34%	delaye	59,841	1.50%	json	87,020	3.60%
xlsx	60,306	0.96%	mp4	83,253	3.26%	html	59,701	1.49%	mp3	64,518	2.67%
html	43,032	0.68%	jpeg	75,906	2.97%	webp	59,505	1.49%	css	24,221	1.00%
gif	29,509	0.47%	css	69,083	2.70%		49,430	1.24%	svg	23,386	0.97%

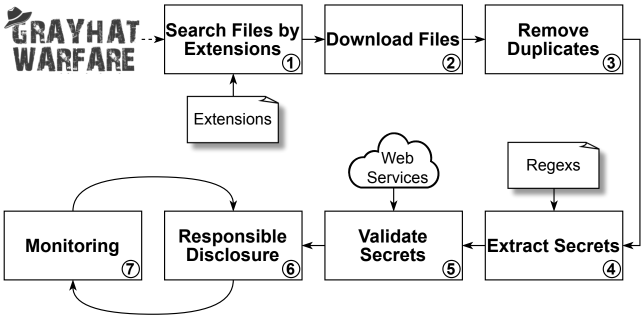

Figure 3: Our methodology.

types of files and some *representative types of secret leaks*, and help to shed light on the magnitude and impact of secret leaks via misconfigured buckets and the responsible disclosure outcomes.

3. Methodology

In this section, we describe the design of our study. Our methodology is summarized in Figure 3. It consists of the following core steps.

Step 1: Files-of-Interest (FoI) Identification. During this step, we collect files in open cloud-storage buckets that potentially contain secret leaks. To execute this step, we use the Grayhat platform to search for the files with particular extensions, namely: `bat`, `conf`, `config`, `env`, `ini`, `json`, `ps1`, `py`, `sh`, and `yml`, that are often used to store access tokens. To select the extensions, we relied on previous studies of secret leaks from the industry [25], [26], [27] and academia [7], [28], [29], [30]. Here, we are looking at a combination of file types that are frequently published as part of code repositories (e.g., `json` or `py`), and also types that should not be published (like `yml`, `env` or `config`) [29]. This latter case helps us to measure the prevalence of secret leaks even in the case if the buckets are not misconfigured, but are intentionally made public (e.g. as a means to share data). We can anticipate that publicly shared files like `json` or `sh` might be less sensitive, while private configuration files such as `env`, `config`, and `ini` might be more likely to contain secrets and, if accessible, are thus expected to be inadvertently exposed rather than intentionally shared.

From Table 4 we see that among the considered extensions only `json` is found among the Top 10 most popular extensions in the ABS, AWS, and GCP buckets. As Table 5 demonstrates, the other considered extensions are not as popular. Indeed, we can expect that many such files are not so prevalent in the studied data, because they should not be shared publicly in the first place.

We executed the search for the files on February 7, 2024, while the last update of the platform data was done on

TABLE 5: Rank of Searched Extensions over the Analyzed Sample of 1,600 Buckets

Ext.	Extension Rank			
	ABS	AWS	DOS	GCP
bat	318	157	429	182
conf	-	146	-	257
config	41	239	117	81
env	-	-	-	-
ini	330	119	102	102
json	6	6	19	7
ps1	103	173	-	274
py	176	61	126	75
sh	99	73	85	55
yml	78	53	105	62

January 19, 2024. Note that the Grayhat platform returns a maximum of 1,000,000 entries for a search through its API. To overcome this limitation and collect all relevant results, we additionally employed a file size filter. Specifically, if the initial search without the filter returned the maximum number of entries, we restricted the search by adding an additional condition on the size of the files and performing the search incrementally in several steps. Thus, we were able to collect all files. Also, due to storage limitations, we only considered files smaller than 100 MB. We do not expect that code-related files are frequently larger than this size limit.

Step 2: Downloading Files. The Grayhat platform provides the URL to the files as a result of the search. We developed a script that downloads the files using the provided link. *It took us 48 days (from February 7, 2024 to March 26, 2024) to download all the files meeting our criteria.*

Step 3: Removing duplicates. This step aims to avoid analyzing the same file multiple times. Before analyzing a file, its content is hashed, producing a unique fingerprint representing the file's content. The generated hash is then compared to a list of the hashes of previously analyzed files. If a match is found, the file is considered a duplicate and skipped. If the hash is unique (no match), the file is analyzed for secrets.This process ensures that only unique files are analyzed, preventing redundant analysis and improving the efficiency of the workflow.

Step 4: Extracting Secrets. To find the secrets in the files, we applied a regular expression search as is usual in such studies [1], [7], [11], [31]. The selection of secrets to look for was guided by their *diverse yet significant potential impact* and the *feasibility of non-intrusive validation*. We focused on high-value secrets such as AWS tokens, Dropbox, Falcon API, Stripe, and others (the full list is in Table 6). These types of secrets were chosen due to the significant security risks they pose if exposed, and they exemplify different types of impact from secret leaks, including unauthorized access to sensitive data, financial transactions, and the ability to modify or delete crucial resources [7]. In choosing the secrets, we also strived for a balance between covering the cloud secrets that could be leveraged for lateral movements and privilege escalation in the cloud and the secrets embodying supply chain risks that would allow the attackers to access other sensitive services.

In our study, we aim to systematically examine the issue of valid secret leaks in misconfigured cloud buckets and show its importance. Thus, due to the scale of the data to analyze, it is important that our secret detection methods do not lead to too many false positives. It is also crucial that our secret validation methods do not compromise the confidentiality or integrity of the affected services. This non-intrusive approach is essential to maintain trust and integrity in the research community, as well as to prevent any unintended consequences that might arise from more invasive validation techniques.

To prepare an initial list of candidate secret patterns, we studied the relevant literature on secret detection that shared regexes, e.g., [1], [7], [11], [31], [32], and experiences in validating secrets, e.g., [12], [33], [34], and investigated several popular repositories containing secret patterns, such as Keyhacks[9], all-about-apikey[10], and Noseyparker[11]. These repositories provide a collection of methods and tools for testing the validity of various API keys as well as regexes to detect them, and test cases and examples based on API providers' documentation. These repositories are used by security researchers and penetration testers who need to verify the validity of API keys they encounter during security assessments. Their approach generally aligns with best practices for non-intrusive validation, ensuring that the verification process does not disrupt services. However, we performed an extensive investigation for all considered patterns to ensure precise detection and non-intrusiveness in validation, and in some cases, we had to alter or improve their methods, as explained below.

To achieve our goals, building upon the foundation described above, we had to develop a new regex (AWS SMTP) and adapt known ones (e.g., for AWS Access Key, Google, Alibaba tokens) to reduce false positives and accommodate the configuration file use case. We also proposed completely new regular expressions that take into account the structure and format of configuration file types (Generic API Key and Generic Secret). For instance, the `env` config file entry has the format `key=value`, so we applied these new generic regexes to find leaks and used the corresponding key name to determine the exact service. This customized approach led to significant discoveries, including the identification of a Crowdstrike Falcon API secret (see Section 6). We also delved deeply into the API documentation to understand which secrets can be validated non-intrusively and automatically; the existing tools do not provide this functionality.

To optimize our search and simplify the process of extracting both secrets and service names, we continually refined our patterns. This iterative process of creating, testing, and adapting regex patterns allowed us to tailor our search methodology to our specific needs, resulting in more comprehensive and accurate secret detection across various configuration files.

[9] https://github.com/streaak/keyhacks

[10] https://github.com/daffainfo/all-about-apikey

[11] https://github.com/praetorian-inc/noseyparker

```
curl -IX POST https://api.dropboxapi.com/2/users/
     get_current_account -H "Authorization: Bearer <
     REDACTED>"
HTTP/2 200
content-type: application/json
cache-control: no-cache
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
x-server-response-time: 248
--------------------------------------------------
curl -IX POST https://api.dropboxapi.com/2/users/
     get_current_account -H "Authorization: Bearer <
     REDACTED>"
HTTP/2 401
content-type: application/json
cache-control: no-cache
content-security-policy: sandbox allow-forms allow-
     scripts
www-authenticate: Bearer realm="Dropbox-API"
x-content-type-options: nosniff
```

Listing 1: Example of valid (status code 200) and invalid (status code 401) responses for the Dropbox API.

The final list of secrets to detect (available in Table 6) was selected from the candidate secret patterns after reviewing the API documentation to ensure that non-intrusive validation was possible. Our validation process is outlined in the next step.

Step 5: Validating Secrets. As mentioned, the validation process relied heavily on the API documentation provided by the corresponding services. This documentation outlines the necessary steps to authenticate and interact with their respective APIs. By following these guidelines, we could verify the validity of the exposed credentials without performing any actions that could have side effects and potentially disrupt the service or cause damage to the resource owners. For these selected API-based services, we relied on status code responses to verify tokens in a non-intrusive and non-disclosing manner. Listing 1 demonstrates how status codes can be interpreted to verify whether a token is valid or not. In this way, for instance, with Dropbox tokens, we could confirm access to files; and with GitHub and GitLab tokens, we could verify repository access levels.

Additionally, we searched for credentials hard-coded in a URL. Using status code responses, we can determine if the credentials are valid without causing any disruption. Another example involved CrowdStrike Falcon API[12] discovered using a generic API pattern, where the key and secret are needed to first generate an access token. If the credentials are valid, an access token would be generated; if not, an error would be raised. This process can be monitored using the status code, allowing us to validate the credentials in an automated and non-intrusive manner.

For cloud keys such as AWS access keys, Google Service accounts, and Alibaba Cloud keys (marked with a * in Table 6), we employed their respective Python SDKs. These SDKs are designed to handle authentication and will raise exceptions if the keys are invalid, providing a straightforward and secure method to validate the credentials. Additionally, for AWS SMTP servers, we attempted to log in using Python's smtplib[13]. A failed login attempt, indicated by a raised exception, was sufficient to determine that the credentials were invalid, ensuring we could stop further attempts and avoid unnecessary interactions with the server. This approach ensured our validation process was both effective and minimally invasive.

This approach ensured that we could accurately validate compromised credentials while adhering to ethical guidelines and minimizing any potential impact on the services or their users. This validation process was fully automated through the implementation of functions designed not only to validate the credentials once but also to monitor their validity over time. This continuous monitoring was crucial during the responsible disclosure phase, allowing us to track whether the credentials remained active or were revoked by the owners following our reports.

Our approach aims to detect and report only valid secrets. Thus, we intentionally ignore all secret patterns discovered in the files of interest that were found to be invalid. This means that our method does not generate false positives, as was reported for other scanning approaches [1], [7], [35], [36]. However, our method will miss the exposed secrets that have already been rotated. We discuss this limitation further in Section 8.

Step 6: Responsible Disclosure. This step in the workflow involves reporting valid secrets found in the files to their respective owners or the cloud providers to ensure proper mitigation. Firstly, the owner of the secret is identified using public information associated with the bucket, file and/or data leak in question, or through Open Source Intelligence (OSINT) techniques, which involve collecting information from publicly available sources. Once identified, we report the discovered secret(s) via email.

Step 7: Monitoring. Subsequently, we followed up with the owners to ensure that the discovered secrets were properly secured and the necessary remediation steps were taken. Additionally, we provided extra details about the issues when requested by the owners to help them understand the risks. This follow-up is an integral part of the monitoring phase, where we continuously assess whether reported issues have been effectively resolved. If not, we proactively follow up with the owners to ensure that appropriate measures are taken. This proactive approach helps to mitigate potential risks associated with these unsecured secrets. At this stage, we also collect information on what mitigation strategies are executed by the owners and when. This allows us to evaluate the effectiveness of the responsible disclosure campaign. More details of our responsible disclosure process and its outcomes are provided further in Section 5.

4. Discovered Secret Leaks

Using the Grayhat API [21], we searched for files with the extensions listed in Section 3 (Step 1) and downloaded

[12] https://www.falconpy.io/Service-Collections/OAuth2.html#usage

[13] https://docs.python.org/3/library/smtplib.html

TABLE 6: Searched Patterns. (* denotes the patterns that were validated using the corresponding Python SDKs provided by the targeted platform; other secrets were validated based on the status codes. +denotes patterns that we created.)

Type	Pattern Regex
AWS Access Key*	(?:A3T[A-Z0-9]—AKIA—ASIA—ABIA—ACCA)[A-Z0-9]{16}
OpenAI API Key	sk-[a-zA-Z0-9]{48}
Google OAuth	(?i)\b([0-9]+-[a-z0-9_]{32})\.apps\.googleusercontent\.com
Google (GCP) Service Account Key*	\\"type\\": \\"service_account\\"
Google Client Secret	(?x)(?i)client.?secret.{0,10}\b([a-z0-9_-]{24})(?:[^a-z0-9_-]—$)
Twilio API Key	SK[0-9a-fA-F]{32}
Sendgrid API Key	(SG\.[0-9A-Za-z_-]{22}\.[0-9A-Za-z_-]{43})
Slack Bot Token	(xoxb-[0-9]{12}-[0-9]{12}-[a-zA-Z0-9]{24})
Slack User Token	(xoxp-[0-9]{12}-[0-9]{12}-[0-9]{12}-[a-f0-9]{32})
Slack App Token	(xapp-[0-9]{12}-[a-zA-Z0-9/+]{24})
Slack Webhoock	(?i)(https://hooks.slack.com/services/T[a-z0-9_]{8}/B[a-z0-9_]{8,12}/[a-z0-9_]{24})
Password in URL	[a-zA-Z]{3,10}://[^/\\\\s:@]{3,20}:[^/\\\\s:@]{3,20}@.{1,100}[\\\"'\\\\s]
Stripe API Key	sk_live_[0-9a-zA-Z]{24}
MailChimp API Key	[0-9a-f]{32}-us[0-9]{1,2}
Facebook Access Token	EAACEdEose0cBA[0-9A-Za-z]+
Mailgun API Key	key-[0-9a-zA-Z]{32}
GitHub APP Token	(ghu—ghs)_[0-9a-zA-Z]{36}
GitHub Fine Grained	github_pat_[0-9a-zA-Z_]{82}
GitLab Pattern	glpat-[0-9a-zA-Z\-_]{20}
AWS SMTP Server*+	[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z—a-z]{2,}—email-smtp\.[a-z0-9-]+\.amazonaws\.com
GitHub Personal Access Token	ghp_[0-9a-zA-Z]{36}
GitHub oAuth Access Token	gho_[0-9a-zA-Z]{36}
Alibaba Cloud*	LTAI[a-zA-Z0-9]{20}
Generic API Key+	[a—A][p—P][i—I][_]?[k—K][e—E][y—Y].*['—\\\"][0-9a-zA-Z]{32,45}['—\\\"]
Generic Secret+	[s—S][e—E][c—C][r—R][e—E][t—T].*['—\\\"][0-9a-zA-Z]{32,45}['—\\\"]

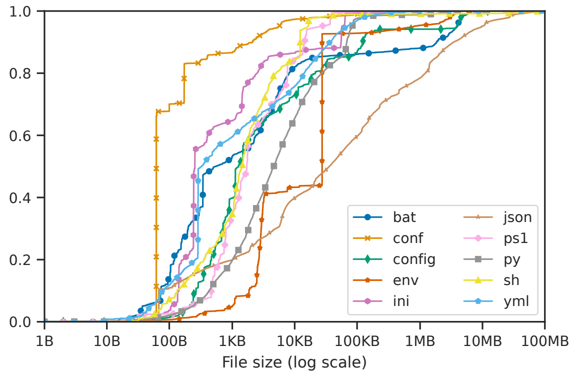

Figure 4: Size eCDF for files with each searched extension.

all available files fitting our criteria for analysis. Table 7 reports the number of analyzed files and the number of buckets where these files are found per each extension.

In Table 8, we provide the statistics of files with the searched extensions that we retrieved from Grayhat. Interestingly, the percentage of duplicate files with searched extensions (68.21 % on average) is much higher than for the sample of downloaded buckets (see Table 3).

Furthermore, Figure 4 shows the distribution of the sizes for files with each searched extension, while Figure 5 reports when the corresponding files were modified last time. We can see that the studied files are relatively small (for all types, 50 % of the retrieved files are smaller than 100 kB), and they are mostly not very recent (for all types, 50 % of the files were last updated in 2023 or earlier).

Within the analyzed files, *we have discovered* 215 *validated secret leaks.* Figure 6 outlines the properties of the

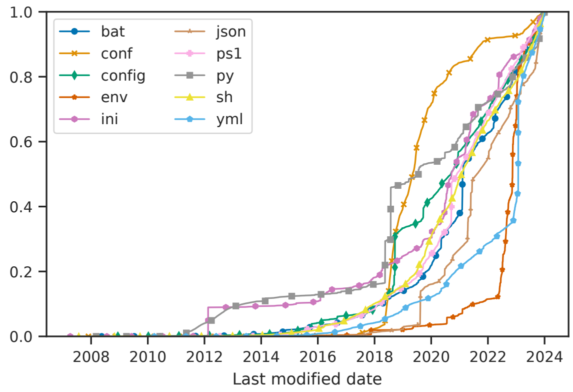

Figure 5: Last modified date eCDF for files with each searched extension.

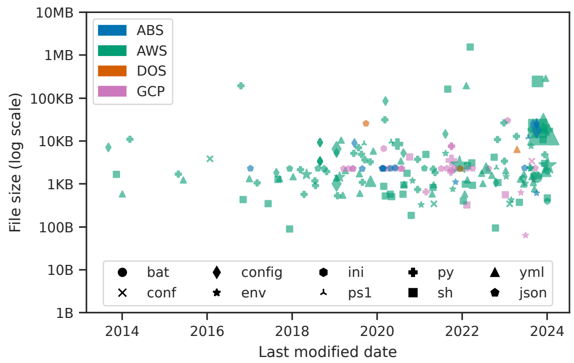

Figure 6: Properties of the files with leaked data.

TABLE 7: Number of Buckets and Files with Searched Extensions

Ext.	# of Buckets					# of Files				
	ABS	AWS	DOS	GCP	Total	ABS	AWS	DOS	GCP	Total
bat	232	1,570	47	233	2,082	7,837	56,924	323	7,408	72,492
conf	80	1,121	21	154	1,376	4,089	1,086,445	450	56,604	1,147,588
config	623	1,868	50	315	2,856	38,919	65,512	2,513	11,910	118,854
env	25	224	3	63	315	319	23,288	50	947	24,604
ini	427	2,998	93	499	4,017	80,816	630,257	4,724	18,471	734,268
json	581	1,486	363	919	3,349	1,725,623	1,176,889	967,744	1,143,380	5,013,636
ps1	483	995	14	117	1,609	11,184	36,304	279	22,115	69,882
py	183	2,824	54	549	3,610	148,873	354,255	1,652	118,817	623,597
sh	359	4,621	96	883	5,959	6,920	161,787	669	163,484	332,860
yml	300	6,033	131	891	7,355	25,127	804,708	7,788	93,268	930,891
Unique	**2,327**	**15,545**	**603**	**3,281**	**21,756**	**2,049,707**	**4,396,369**	**986,192**	**1,636,404**	**9,068,672**

TABLE 8: Statistics of Files with Searched Extensions

Ext.	Files			File Sizes				Last Modified Date	
	#	Unique	% Duplicates	Min	Mean	Median	Max	Earliest	Latest
bat	72,492	32,037	55.81%	2.0B	438.14KB	553.0B	29.00MB	2008-05-16	2024-01-04
conf	1,147,588	215,246	81.24%	1.0B	4.05KB	62.0B	9.86MB	2007-11-27	2024-01-04
config	118,854	19,365	83.71%	1.0B	283.95KB	1.26KB	44.50MB	2008-12-10	2024-01-04
env	24,604	4,106	83.31%	6.0B	164.54KB	27.51KB	19.56MB	2013-09-23	2024-01-04
ini	734,268	255,086	65.26%	1.0B	10.79KB	258.0B	29.68MB	2007-04-07	2024-01-04
json	5,013,636	4,190,292	16.42%	2.0B	1.89MB	35.89KB	99.98MB	2012-04-03	2024-01-04
ps1	69,882	13,699	80.40%	2.0B	7.87KB	1.71KB	24.07MB	2012-07-17	2024-01-04
py	623,597	198,278	68.20%	1.0B	45.01KB	4.62KB	28.56MB	2008-03-06	2024-01-04
sh	332,860	39,603	88.10%	1.0B	401.52KB	1.46KB	99.90MB	2007-07-26	2024-01-04
yml	930,891	375,545	59.66%	1.0B	18.42KB	329.0B	88.56MB	2010-03-24	2024-01-04

TABLE 9: Leak Statistics per Provider

Provider	Detected	Reported	Fixed
ABS	18	8	6
AWS	162	129	82
DOS	3	2	0
GCP	32	21	7
Total	**215**	**160 (71,42%)**	**95 (59,37%)**

TABLE 10: Secret Leaks Found by File Extension

Ext.	Files w/ Secrets		# Secrets	
	#	% of Unique	Detected	Reported
bat	2	0.0062%	3	3
conf	4	0.0019%	4	2
config	19	0.0981%	22	20
env	18	0.4384%	24	22
ini	4	0.0016%	4	2
json	33	0.0008%	33	14
ps1	4	0.0292%	4	4
py	37	0.0187%	40	26
sh	34	0.0859%	37	29
yml	37	0.0099%	44	38
Total	**192**	**0.0036%**	**215**	**160**

files with discovered secrets. Each point on this figure corresponds to a file with secrets. The shape of the marker represents the extension, while the size is proportional to the number of leaks per file. Finally, the color relates to the provider. We can see that, overall, we have files with leaks going way back and the leaks are widely distributed temporally.

Table 9 shows the breakdown of the found leaks for each provider. Not surprisingly, the number of leaked secrets correlates with the total number of open buckets (see Table 2). Table 10 reports on the statistics of leaked secrets for each file extension. The table shows that files with the `yml`, `py`, `sh` and `json` extensions contain the largest number of leaked secrets. However, if an attacker would like to maximize the chances of discovering the secrets then the files with `env`, `config`, `sh` and `ps1` look the most promising (see the "% of Unique" column in Table 10).

On average, *secrets are found in* $0.0036\,\%$ *of files*. Note that this is a lower estimation, as in this work, we concentrated only on secrets that can be verified. Considering the large number of exposed buckets and the even greater number of files within them, the threat level of this source is significantly high. Interestingly, we can see from Table 10 that the number of detected secrets is usually higher than the number of files with secrets (on average, 1.12 leaks per file with secrets). This means that *the same file is often used to store multiple secrets*.

Table 11 presents the breakdown of the secrets by service type. As we can see, the most common leaked secrets also provide access to cloud services. An attacker who gains access to AWS access keys or Google Service accounts can cause significant damage and escalate their privileges within a cloud environment. With AWS access keys, for example, an attacker can access various AWS services, such as S3 for data exfiltration, EC2 for launching additional instances, and IAM for modifying user permissions to elevate their privileges. They could also create backdoors or leverage AWS services for further malicious activities, like deploying crypto-mining operations or launching attacks from within

TABLE 11: Detected Leaks

Service	Count
AWS access tokens	100
GCP Service accounts	32
Slack Webhook	15
Sendgrid	14
AWS SES SMTP	13
OpenAI, Facebook, Stripe	6
Twitter	4
GitHub Personal Access Token, GitLab Runner Registration Token	3
Dropbox, Telegram Bot, Twilio, Mailgun	2
Password in URL, CrowdStrike, Github Client and Secret, HubSpot, Slack User Token	1
Total	**215**

the compromised cloud infrastructure [18], [19], [34]. Note that Table 10, Table 9 and Table 11 also show the details of our responsible disclosure results, which we discuss in the next section.

Analysis of Missed Secrets. Since we only analyzed selected file extensions, we now aim to evaluate the number of missed secrets that might be leaked in other file types. To estimate this, we downloaded all files in the 400 randomly selected buckets (100 buckets for each provider) that were previously analyzed for examining the cloud buckets dataset as a part of the larger set of 1,600 buckets (Section 2). While this number of buckets might seem small compared to the total number of open buckets, it is important to understand the scale and complexity involved in analyzing cloud storage buckets. Many of these buckets contain millions of files, making a full analysis of the entire dataset impractical. However, the results on this small set of buckets allow estimating how many secrets our pipeline might have missed.

We scanned each file within these buckets, using the same list of secret patterns from Table 6, to identify any leaks we may have missed by only focusing on the file extensions in Table 5. During our analysis, we found 1 *valid* Slack Webhook in a `7z` file that was not password-protected and fully readable. In our set of downloaded buckets, there are 2,026 files with this extension, meaning that a leak appeared in 0.0494 % of unique files with this extension. This percent is on par with the numbers reported in Table 10; therefore this file extension is also worth investigating for future studies. According to Grayhat, there are currently 483,770 `7z` files in all open buckets.

Additionally, we discovered an *invalid* AWS token (Key, Secret) in a `cscfg` file. Although this was a true negative, we decided to further scrutinize this file type given the high probability of discovering secrets in them in our sample – a secret, although invalid, was discovered after analyzing only one file with this extension. At Grayhat, there were 661 references of files with this extension. Out of them, we managed to download 260 and analyzed them using our pipeline (the rest are no longer accessible publicly at the time of writing). In these files, we found 3 *valid* secret leaks: 1 token for Sendgrid, 1 for Stripe, and 1 for Mailchimp[14]. Thus, the `cscfg` file type appears to be a very relevant extension for secret leak monitoring and needs to be included in future studies.

5. Responsible Disclosure

In our responsible disclosure process, we partnered with an established volunteer-led, not-for-profit CSIRT organization `CSIRT.global`[15] based in The Netherlands and experienced in vulnerability research and responsible disclosure. By leveraging their extensive global community of security experts and trustful relations with different organizations, we were able to locate the owners of the exposed data more effectively. Moreover, receiving a vulnerability notification from a reputable (and thus more trustworthy) organization can also be more conducive for vulnerable organizations to react to the message and fix the issue [37], [38], [39]. One of the objectives of `CSIRT.global` is to notify only about high-confidence vulnerabilities, reducing the number of false negative notifications as much as possible. This allows the organization to maintain good connections with the notification recipients. The validation process ensures that this objective is met.

Note that we chose to report to the owners directly, whenever it was possible to identify them in a responsible manner, instead of only bulk-reporting to the cloud provider. Cloud providers manage a vast number of resources, making it challenging and time-consuming to identify and alert the correct owner quickly. Also, cloud providers often have numerous reports to process, which can delay the notification to the actual owner of the compromised data. Contacting bucket owners directly ensures that the notification reaches the precise owner of the exposed data, who can respond and mitigate the issue more swiftly. Owners can also provide feedback on the disclosure process, helping to improve future security practices. This is something we saw during this process: we got multiple responses explaining the cause of the issue. All responses we received from the owners were very positive and thankful for the efforts spent on doing this responsible disclosure. Furthermore, our participation in the responsible disclosure process ensured that any additional details requested by organizations regarding leaked secrets could be provided more quickly and comprehensively. This kind of interaction cannot be done if a third party (cloud provider) handles the communications. As a side benefit, this also allowed us to estimate the effectiveness of a global direct notification campaign.

5.1. Responsible Disclosure Process

The disclosure process involved the following steps:

[14] Note that the leaks discovered during the false negative analysis are not included in the responsible disclosure results reported in Section 5, as they were found at a later stage. We shared the details about these secrets with our partner `CSIRT.global` organization to report them to the corresponding bucket owners.

[15] https://csirt.global/

Step A: *Identify contact details*. We looked for relevant details in the files that contained the leaks, which sometimes included email addresses, URLs, organization names, or developer names. These details were subsequently analyzed and enriched using OSINT (open source intelligence) techniques, such as LinkedIn searches and email format detection to identify the appropriate contact details of the right individuals. Our primary target for contact was technical personnel, typically starting with members of the security team. If no suitable contacts were found within the security team, we expanded our search to include DevOps teams, followed by developers. In some cases, particularly with startups, we reached out directly to founders. Additionally, we contacted organizations through their responsible disclosure programs, if available, or via information provided on their privacy policy pages. This proved to be highly effective in establishing communication channels: 9 out of 11 organizations contacted using emails from their privacy policy pages resolved the reported issues. `CSIRT.global`'s expertise was essential in this step, as finding the right channel to securely report a vulnerability is critical.

Step B: *Categorize critical data exposure.* The data leaks at organizations whose contacts were identified (160 in total) were categorized based on their severity using the Traffic Light Protocol (TLP) system. Critical exposures were prioritized for immediate action (71 leaks). Less critical exposures were noted but may not require urgent action (89 leaks). We prioritized based on the organization's size, sector, and the specific nature and impact of the leaks. For instance, larger organizations were prioritized due to the higher volume of sensitive data and potential impact. Critical sectors such as finance, healthcare, and government were also prioritized given the severe consequences of breaches in these areas. The nature of the exposed data, such as AWS tokens, Google service accounts, Dropbox, and Crowdstrike API, was carefully evaluated to determine the potential risk. Leaks with the highest impact, such as those granting unauthorized access to essential services or sensitive information, were addressed first. This structured approach ensured that the most significant vulnerabilities were mitigated swiftly, reducing the overall risk and protecting sensitive data.

Step C: *Send a notification email.* Using a standard `CSIRT.global` template, a disclosure email was prepared and sent to the vulnerable organization, detailing the discovered data exposure and how it can be located.

Step D: *Tracking and follow-up.* The interaction with the vulnerable organization and the status of the leaked secret was monitored. Some organizations requested to be provided with more information about the leak or to prove that it was exploitable. If no response or an inadequate action was taken within 30 days, the `CSIRT.global` team initiated follow-up actions to ensure the data exposure was addressed, by reaching out to the organization one more time and eventually reporting the issue to the cloud provider as well.

Step E: *Documentation.* The entire process was documented in a secure file, where the leading researcher kept track of the secret leak status and the responsible disclosure outcomes. We note that the first author is also a member of `CSIRT.global`, experienced in vulnerability research and reporting. He/she was the only person who had access to the files with the leaks and the leak details, and he/she coordinated with the rest of the `CSIRT.global` team regarding the search for the identities of the vulnerable organizations and the right contact details within them.

TABLE 12: Summary of the Identification and Disclosure Process (as of September 15, 2024)

Reported	Dropped	Unknown	Total
160	5	50	**215**

This structured approach ensured that all discovered leaks were handled promptly and responsibly. The notification process involved two `CSIRT.global` teams (the research and the incident response teams) and was led by the first author. Each `CSIRT.global` team had specific responsibilities to ensure effective communication and follow-up. The `CSIRT.global` research team was responsible for steps A, B, and C. The incident response team was copied to the notification email and was responsible for tracking and follow-up (step D). The first author was responsible for step E.

5.2. Disclosure Outcomes

The notification process started on April 15, 2024. Table 12 reports on the status of this process as of September 15, 2024. In this table, the term "Reported" designates instances where we successfully identified the owners of the buckets and subsequently reported the issues to them. "Dropped" refers to issues that were independently resolved by the owners without our intervention. Conversely, "Unknown" denotes buckets for which we were unable to establish a direct link to a specific organization. These unidentified buckets have been reported to the respective cloud providers, with the hope that they will assist in identifying and reporting the issue to the rightful owners.

For these "unknown" buckets, as of paper submission, only Amazon has replied to our responsible disclosure, acknowledging the receipt but reiterating in their reply that it is the responsibility of the customer to fix such issues. Other cloud providers have not responded to our disclosure. *This emphasizes that reporting to the owner is essential in fixing the secret leakage promptly.*

5.2.1. Owners' Response Actions. Three primary security measures are appropriate to fix the issues: restricting access to buckets, restricting access to files with secrets, and revoking access tokens. As not all owners would reply and inform us about their actions, we developed our monitoring system tracking the implementation of these security measures to understand how organizations mitigate the problem. It would be expected that all three of these measures are applied.

65 *out of* 160 *instances (*40.62 %*) resulted in no actions being taken.* This indicates a significant gap in addressing the disclosed security issues. We remark that these numbers

TABLE 13: Security Measures Taken by Owners after Responsible Disclosure

Restrict Bucket	Restrict File	Revoke token	Count
✓	✓	✓	19
✗	✓	✓	18
✗	✗	✓	33
✗	✓	✗	19
✓	✓	✗	6
✓	✗	✗	0

are better than the results observed in other responsible disclosure studies [38], [39], [40], [41], [42][16]. For instance, Maass et al. [40] observed that in their study of notifications about privacy misconfiguration, across all groups representing different means of communication, the average survival rate is 58.8 %. Interestingly, in their study, notifications through the email channel had lower effectiveness, with the survival rate constituting 66.3 %. In our work, the survival rate is lower (40.62 %), suggesting that the choice of involving a reputable `CSIRT.global` organization in the notification process was right [38], [39], [43].

95 *out of* 160 (59.37 %) *cases have resulted in actions upon notification.* Table 13 provides insights into how bucket owners acted upon our responsible disclosure. It shows that different combinations of the three primary security measures were actually applied. Among those, 19 (11.88 %) instances saw the implementation of all three measures, showcasing a comprehensive approach to mitigating risks and protecting sensitive information. Another 18 (11.25 %) cases involved restricting access to the files with secrets and revoking tokens but not restricting access to the buckets, which still leaves a security gap of potential data leakage. Similarly, in 33 (20.62 %) cases, only access tokens were revoked, leaving buckets and files potentially exposed.

There were 19 (11.87 %) cases where only file restrictions were applied, and in 6 (3.75 %) cases, both buckets and files were restricted without revoking tokens. This is clearly insufficient without additional measures, especially if an attacker has already got access to the content of the file. Notably, there were no cases where only the bucket was restricted without additional measures on files or tokens, suggesting that when bucket restrictions were considered, other steps were also taken to secure the data.

We should remark that organizations might respond by restricting access to the exposed files or the entire bucket rather than revoking the exposed keys because these keys can be embedded across multiple production systems, making their immediate revocation complex and risky. Revoking such keys without a thorough understanding of all their dependencies can disrupt critical business operations, causing service outages and operational downtime. Consequently, organizations might opt for quick fixes like access restrictions to prevent further leaks while avoiding the operational risk and disruption that comes with immediate key rotation. Addressing the deeper issue of key leakage requires a more extensive and careful process, involving tracking all uses of the keys and systematically updating or replacing them to maintain continuous system functionality. We keep monitoring if eventually these secrets will be rotated.

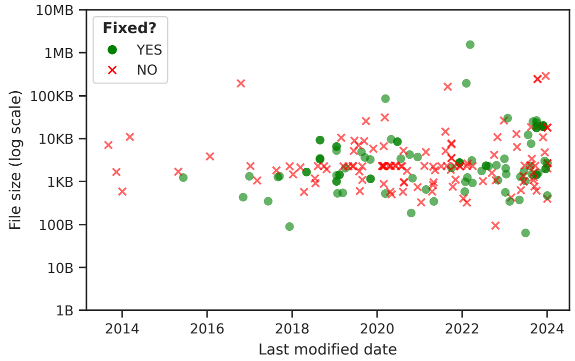

Figure 7: Properties of the files with leaks (for all 215 discovered leaks).

Table 13 underscores the variability in responses to security disclosures. This likely reflects the maturity of the security team. The best approach is to restrict access to the bucket and the files as well as revoke the credentials that have been leaked. It would also be beneficial to investigate internally whether these credentials have been previously misused. However, our external monitoring system is not able to detect this mitigation.

Figure 7 illustrates the distribution of secret leaks over several years, highlighting their last modification timeline from 2014 to 2024. It shows that both fixed and unfixed leaks originate in files that were last modified throughout this period, indicating that exposed secrets remain accessible to attackers for extended periods. We can also see that many fixed secrets were in very old files, indicating that they were used in an active cloud infrastructure. The longer a secret remains exposed, the higher the risk of it being accessed and exploited by malicious actors. This emphasizes the critical need for timely identification and remediation of such leaks.

5.2.2. Incident Response Speed. The statistics in Figure 8 illustrate a wide variation in how organizations respond to responsible disclosures of security issues, particularly in terms of communication and speed of resolution. This figure shows only organizations that acted upon our notification. As of September 15, 2024 (after 5 months of monitoring), 95 cases were fixed in total. We see that some organizations acted swiftly, with 26 (16.25 %) instances of issues being addressed in less than a day. Interestingly, for these quickly fixed issues, in 11 (6.88 % of the total) cases, the notified organizations also informed the disclosers about their actions, showing a high level of responsiveness and transparency. However, many organizations do not communicate their actions, even when they fix issues promptly, as evidenced by 15 (9.38 %) fixes without notification in less than a day. A day after the notification, 28 (17.5 %) more cases were fixed, however, the communication dropped considerably, related

[16] However, it is not fully fair to directly compare with these studies, because they examined different systems.

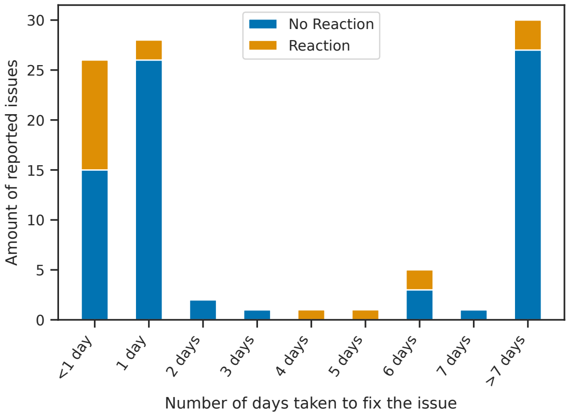

Figure 8: Incident response speed for organizations that took action to fix the issue. We count separately organizations that reacted or not to our notification email.

to only 2 issues. Thus, 33.75 % *cases were fixed within one day after the notification.* As the time to fix issues increases, the number of reactions and communications drops sharply, with minimal responses and few fixes recorded beyond two days. Still, the issues were remediated even long after the notification: between 20 days and the end of the monitoring period, another 28 (17.5 %) cases were fixed. In total, 20 organizations replied to the disclosers in the study period.

Notably, the majority of organizations fail to maintain communication with the disclosers, leading to uncertainty about whether and how the issues have been addressed. This inconsistency highlights the need for more standardized and timely responses, ensuring both swift resolution and transparent communication to better manage security vulnerabilities, as underscored by, for example, the NIS2 Directive in the European Union[17].

5.3. Vulnerable Organizations

Based on the collected contact details for responsible disclosure, we can make an overview of the vulnerable organizations that we reported to. First, Table 14 represents the amount of reported issues by country. Notably, the US is the most prevalent case, significantly higher than other countries, which may reflect the larger number of cloud service users and organizations in the US. In total, 160 issues were reported, highlighting the widespread nature of secret leaks across the globe. The geographical spread of the issues underscores that this is not confined to any one region but is a pervasive challenge for cloud users worldwide. Overall, the table paints a picture of how misconfigured cloud buckets and the resulting secret leaks are a global issue, affecting countries with diverse levels of technological development and cloud service usage.

TABLE 14: Reported Issues per Country

Countries	Count per Country
US	52
IN	13
AU, GB	10
BR	9
KR	7
FR, IT, SG, TW, VN	4
CA, AE, GB-SCT	3
CN, CO, DK, IL, NL, ES, CH	2
BE, KY, CL, CG, EC, DE, ID, JP, MY, PK, PH, PL, SK, ZA, LK, SE	1
Total	**160**

TABLE 15: Reported Issues by Sector

Sector	Count
Computer and Information Technology	81
Retail Stores, Wholesale, and E-commerce Sites	18
Finance and Insurance	12
Education and Research	11
Media, Publishing, and Broadcasting	8
Health Care Services	7
Government and Public Administration	6
Personal	5
Construction and Real Estate	4
Museums, Libraries, and Entertainment	4
Building Security Materials	1
Manufacturing	1
Community Groups and Nonprofits	1
Travel and Accommodation	1

Table 15 shows the reported issues by industry sector. For the categories in this table, we used the classification according to the North American Industry Classification System (NAICSlite)[18], developed to classify Autonomous Systems [44]. The table sheds light on the prevalence of such incidents, with Computer and Information Technology emerging as the sector with the highest number of reported issues. This sector encompasses a wide array of companies, including software development firms, AI companies, and SaaS providers, all of which rely heavily on cloud infrastructure to store and manage vast amounts of data.

However, it is essential to recognize that the risk of data leaks extends beyond the tech sector alone. Other sectors, such as Finance, Healthcare, Government, and Academia, were also found to be vulnerable to these issues during our research. Moreover, these organizations vary widely in terms of size, from small startups to multinational corporations with extensive resources and security teams. Yet, regardless of their scale or industry, they all face similar challenges when it comes to securing their cloud infrastructure effectively. Indeed, the interconnected nature of cloud environments means that this type of data leak in one organization's cloud bucket can have far-reaching consequences. Data breaches can cascade across supply chains, impacting partners, customers, and stakeholders[19]. In Section 6, we report on several illustrative case studies concerning the secret leaks we have discovered.

[17] https://eur-lex.europa.eu/eli/dir/2022/2555

[18] https://asdb.stanford.edu/

[19] https://github.com/nagwww/s3-leaks

6. Case Studies

We now examine several particularly interesting cases found during this study. These examples illustrate that secret leaks in the cloud-hosted infrastructure can have a potentially disastrous impact on organizations and their security.

CrowdStrike's Falcon Platform. One of the most impactful discoveries in our study is an API[20] token securing access to the organization's Falcon Platform[21] configuration panel. The Falcon Platform, developed and provided by the security service provider CrowdStrike[22], is a suite of security solutions designed to ensure the protection of organization's IT resources. These solutions are employed by security teams for threat detection (e.g., identity threat detection), incident response (e.g., sandboxes or a Security Information and Event Management platform), and endpoint protection (e.g., Endpoint Detection and Response, firewall, forensic data collector). The found API token provided access to the configuration panel of all security solutions and could be used by a malicious actor to circumvent completely the security protection of the organization. For instance, using this token it would be possible to modify threat detection and firewall rules, add or delete user accounts and manage their privileges, change response policies, add detection exceptions, etc. Thus, an attacker having this token would be able to gain complete control over resources and easily avoid being detected. Both parties took immediate action to address the leak within *one day* by deleting the file in question and revoking the keys, which was confirmed by the CrowdStrike analyst who handled the case.

AWS Credentials Exposing 230 Buckets. In another case, we discovered AWS credentials in a bucket belonging to the government sector. These credentials facilitated seamless access to various AWS services and resources, empowering efficient data management, infrastructure deployment, and security enforcement across the organization's cloud environment. They allowed one to authenticate, obtain temporary security credentials, retrieve account information, and access DynamoDB databases and S3 bucket data.

Within the scope of our study, we discovered that these credentials secured access to the data in a vast array of S3 buckets, totaling 230 in number. From the metadata information, it became clear that these buckets contain a variety of data, each serving different purposes. For instance, there were multiple buckets likely holding logs and event data, while other buckets possibly contained financial information or IT infrastructure data related to financial systems. Additionally, some buckets were used for backups. Furthermore, there were buckets indicating their usage for deploying infrastructure using Terraform[23]. Overall, to an adversary, access to these buckets would provide insights into the organization's operations, data management practices, infrastructure deployment strategies, as well as critical and confidential data related to the organization.

This case is also interesting because it confirms the correctness of our choice for prolonged monitoring. The issue was reported on April 15, 2024. The next day, April 16, 2024, the owner admitted it and partially resolved it by restricting access to the file containing the credentials. However, on April 22, 2024, our monitoring system discovered it being live again. We can assume that some organization's systems or applications deploy the file automatically with the wrong permissions. The issue was reported to the owner again. As of May 24, 2024, the issue was partly fixed again by restricting the file and not revoking the credentials.

AWS SES SMTP. Amazon Simple Email Service (SES)[24] is a cloud-based Simple Mail Transfer Protocol (SMTP) service provided by Amazon. It provides facilities to send emails and manage all SMTP-related issues such as authentication, bounce and complaint handling, and measuring email campaigns. During this study, we discovered 13 valid SMTP SES sets of credentials. All these credentials could be used to send emails on behalf of the corresponding organizations. This can be exploited for various malicious purposes, mainly to spread phishing and spam messages. However, they can also be used for Business Email Compromise (BEC) attacks or to distribute malicious files, which might result in the sender being blocklisted.

Interestingly, during the responsible disclosure process, one organization has requested the `CSIRT.global` team to prove that the credentials indeed can be used to send emails on behalf of the security team account. After verifying that the security team indeed controls this account, a benign demonstration of the attack, consisting in sending a test email on behalf of the organization's security team, was carried out. As the result of the responsible disclosure, 12 out of the 13 reported AWS SES SMTP leaks were fixed by revoking the credentials.

7. Ethical Considerations

Ethical considerations are paramount for this research, as we are dealing with sensitive data (credentials) that can be used to do harm and create substantial damage not only to the owners but also to third parties (e.g., via sending phishing messages). Our study design has been reviewed and approved by the Ethics Review Committee of the Science Faculty at Leiden University (The Netherlands)[25].

There are several important ethical considerations that we took into account when designing and executing this research, trying to understand all possible negative consequences from our work, and minimize them or balance them against the benefits. First, in order to locate leaked credentials, we downloaded and scanned third-party files. To secure this process, we set up a dedicated secure infrastructure, and the downloaded data has been stored encrypted, with only the first author having access to them. After

[20] https://www.falconpy.io/Home.html

[21] https://www.crowdstrike.com/falcon-platform/

[22] https://www.crowdstrike.com/

[23] https://www.terraform.io/

[24] https://aws.amazon.com/ses/

[25] Ref. numbers 2023–024 and 2024–005.

finishing the research process, the data are deleted. We responsibly disclosed the leaked secrets to the owners or the platform providers (as Section 5 details). To reach the owners directly, we have examined some publicly available information in buckets and relied additionally on OSINT, whenever possible. As mentioned in Section 3 and Section 5, reporting to the owners is more effective as it ensures faster issue mitigation. This is recommended as the most effective by previous research on large-scale vulnerability disclosure [38], [39]. These considerations, in our opinion, balance the potential negative impact of looking for owner details in the leaking files and using OSINT techniques, such as LinkedIn search.

Second, to ensure that the leaks are *exploitable*, we have validated them automatically. To minimize the impact of this process, we selected only the secrets that can be validated in a non-intrusive way (this is discussed in Section 3). Secret validation has been explored in previous studies, e.g. [12], [34] sought to validate access tokens by exploiting the difference in transaction outcomes. Our approach is similar, as we are automatically processing the results of the interaction with a remote system (i.e., the status codes), and we are not exploring the obtained privileges further than that.

Finally, our research used data provided by GrayhatWarfare, a platform whose provenance cannot be established, but which was previously used in other research, e.g., [18], [19], [45]. To scale up the data processing using their API, we acquired a license. This is a potential ethical concern. However, we have extensively reviewed all publicly available information about this platform, and we did not find reasons to believe that there is a criminal organization behind this data source.

Specifically, before starting the research, we investigated the reputation of this platform and whether there are any known cybercriminal connections behind it. Furthermore, we have not encountered any complaints or negative feedback about this source from security researchers or the public. Importantly, this platform is a legally registered company in Europe.

Furthermore, the platform indicates on its website that its purpose is to raise awareness about the misconfigured bucket issues[26], as such open buckets often leak confidential or private data. The platform invites all bucket owners to contact them if they observe that their data ended up listed, and we have observed that they do act on reports and remove data. We also have sent them an inquiry email regarding their data collection process. They explained that they follow URLs, analyze content, and extract any publicly accessible bucket links.

To summarize, we investigated this data source prior to deciding to use it for our research and acquiring the license. Our considerations were reported to the Ethics Review Committee, which reviewed our study design and approved it. We note that our study has led to many organizations securing their cloud storage infrastructure; thus, it was beneficial to society.

Still, like any other responsible vulnerability research and disclosure, we acknowledge that we cannot remove all possible harms altogether. A malicious actor can repeat the same process and potentially discover secret leaks and other opportunities to do harm. We believe that the benefits of this research, the responsible disclosure, and the awareness this work creates, overall, outweigh the possible harms.

8. Discussion and Limitations

Discussion. Our study has confirmed that very dangerous and potentially impactful secret leaks occur via misconfigured cloud infrastructures, affecting organizations from diverse sectors and countries. We therefore believe that our study significantly contributes to public benefit by helping organizations to recognize important security issues and fix them. Moreover, we bring additional attention to this problem, which may drive the development and implementation of the standards and best practices for all cloud providers.

There are already initial steps in this direction that can potentially mitigate the issue. For instance, starting from April 2023, Amazon blocked the default public access to S3 buckets[27]. Our research confirms that this measure is likely insufficient, as a lot of buckets with private information are still publicly available. Moreover, in June 2024, Google plans by default to disable access to the compromised Google Cloud keys discovered by all means[28]. This policy may reduce the potential harm of keys being exposed for a considerable amount of time. For instance, in our study, we used the Grayhat snapshot data collected as of January 19, 2024. Due to the vast amount of data to analyze, we managed to finish our research process and start the notification campaign only on April 15, 2024. This shows that the discovered leaked secrets were publicly available for a period of at least 3 months. Given the speed with which the attackers discover and access misconfigured buckets as reported by the honeybuckets study [20], this gives a considerable advantage to adversaries. Thus, raising awareness and supporting organizations and cloud providers in implementing best practices is crucially important.

Preventing Secret Exposure in the Cloud. Secret leaks can pose significant security risks, potentially leading to new attacks and lateral movements in cloud environments[29][30]. To mitigate these risks, organizations should implement a multi-layered approach focused on secure data and credential management. First, they need to employ robust access controls and encryption for all storage buckets, ensuring that only authorized personnel can access sensitive files. Least privilege principles should be robustly implemented across all cloud

[26] https://buckets.grayhatwarfare.com/top_keywords

[27] https://aws.amazon.com/blogs/aws/heads-up-amazon-s3-security-changes-are-coming-in-april-of-2023/

[28] https://cloud.google.com/resource-manager/docs/organization-policy/restricting-service-accounts#disable-exposed-keys

[29] https://unit42.paloaltonetworks.com/large-scale-cloud-extortion-operation/

[30] https://www.paloaltonetworks.com/blog/security-operations/playbook-of-the-week-cloud-token-theft-response/

resources and services. Second, organizations should utilize secret management systems,e.g., AWS Secrets Manager[31], to store and retrieve sensitive information dynamically, rather than hard-coding it in configuration files. They should also implement automated scanning tools to detect and flag potential secret leaks in code repositories and cloud storage.

Additionally, enforcing strict version control practices, including git-secrets[32] or similar tools, helps to prevent accidental commits of sensitive data. Credentials should be temporary and rotated regularly, scoped access tokens must be used where possible. Finally, organizations should conduct regular security audits and penetration testing to identify and address any vulnerabilities in the cloud infrastructure. By combining these strategies, organizations can significantly reduce the risk of secret leaks and their potential for enabling unauthorized access or lateral movement within cloud environments.

If an organization has discovered that a secret has leaked, it is advised, as mentioned in Section 5.2.1, to revoke it and disable access to the corresponding files and cloud storage buckets. It is recommended that the organization then tightly monitors the corresponding infrastructure, performing detailed log analysis and resource usage metering. Finally, it must be assumed that all data in the corresponding bucket has leaked, and based on this assumption, the organization must perform the impact assessment and remediate as many consequences as possible.

Limitations. We acknowledge that our study has certain limitations. First, our work focuses on the sample collected by the Grayhat platform, which might not be representative of the whole set of publicly available cloud buckets. However, this sample is substantial enough and we were able to find serious secret leaks within it. Second, our work has important ethical considerations. We have tried to conduct our research in an ethical and responsible way, minimizing the possible harm as much as possible and balancing the potential harm against the benefits to society and the affected organizations. Our study has been reviewed and approved by our institutional Ethics Review Committee.

We note that $40.63\,\%$ of the reported instances resulted in no actions being taken by the owners. There might be several reasons behind this. One possible reason is the disclosure process itself: there might be a lack of trust in the disclosing email or a spam filter precluding the delivery of the notification, as reported by previous studies on the effectiveness of responsible disclosure [40], [42], [46]. Some buckets might have been abandoned due to staff rotation or business closure. Moreover, there might be some honeybuckets [20] and honeytokens [47], [48] in our sample. Yet, previously Cable et al. [18] reported that in their studied sample of misconfigured buckets, $46\,\%$ of owners did not notice exploitation of their cloud storage. Overall, our responsible disclosure success rate is in line with the previous studies examining the effectiveness of vulnerability disclosure via email [38], [39], [41], and we have not received any negative reaction to the disclosed issues.

Our analysis and responsible disclosure only covered the validated secrets. This is why our results do not include false positives (besides potential honeybuckets [20] data), which have been identified in other relevant studies [1], [7], [35], [36]. This decision is partially driven by our partner `CSIRT.global`, which prioritizes notifying organizations about real incidents. However, as our research only focused on a limited amount of secret types and a limited amount of file extensions of interest, our process has likely resulted in false negatives, indicating that the actual scale of the problem might be much larger in reality. Still, we believe our automated approach achieves a good balance of scalability, non-intrusiveness, and reliability of the results. Our results highlight the importance of the secret leak problem in misconfigured cloud systems. Future studies can expand on our results, looking into other file types and other relevant security issues.

9. Related Work

Secret leakage detection is an active area of research, with many approaches proposed for improved secret detection [22], [33], [35], [49], [50], [51], [52], and studies comparing the performance of such approaches [36]. Researchers also conducted measurement studies to evaluate the prevalence of credential leaks on different platforms, such as GitHub [4], [7], [8], [9], [10], [28], [30] and large language models trained on code sharing platforms [31], [53], mobile applications and mini programs [11], [12], [34], [54], [55], Docker containers [1], virtual server images [13], [14], and FTP servers [56]. At the same time, some studies focused on developers and their perceptions and needs when dealing with protecting secrets [57], [58], [59]. As mentioned in Section 3, we studied this literature on detecting secret leaks on different platforms and examined the shared regexes as our starting point. However, these works did not analyze detecting secrets in cloud systems, which is the focus of our study.

Leaks in Cloud Infrastructures. Most closely related to our research are studies of vulnerable cloud infrastructures and data leaks in such systems. Continella et al. [19] scanned and assessed more than $240\,\mathrm{k}$ S3 buckets, discovering that $11\,\%$ of them were publicly accessible. They did not scan for secrets leaked within files but reported on several discovered examples of sensitive key material leakage identified by file extension. Subsequently, Cable et al. [18] developed Statosphere: a bucket name-guessing system to discover public buckets exposed to the internet. In their study, $173\,\mathrm{k}$ generated names corresponded to publicly accessible buckets. An estimated $10.6\,\%$ of these public buckets exposed sensitive data, ranging from SQL database dumps to backups, and including $14.4\,\mathrm{k}$ private keys. However, this study did not delve into the different types of secret leaks in the exposed buckets, besides reporting on 10 case studies showcasing substantial data leaks discovered. Izhikevich et al. [20]

[31] https://aws.amazon.com/secrets-manager/

[32] https://github.com/awslabs/git-secrets

experimented with honeybuckets, i.e., buckets intentionally made vulnerable and accessible to attackers. They found that attackers discover open buckets often within hours and that the confidentiality and integrity of data hosted in the buckets are at risk. This underscores the importance of our research.

Our work differs from the studies in this area in the following ways. Our research presents a comprehensive analysis of cloud storage security across four major cloud providers AWS, Azure, Google Cloud, and DigitalOcean, while the closest work to us focused only on AWS [19] and AWS, Google Cloud and Alibaba [18]. By comprehensively analyzing the GrayhatWarfare dataset, our study is also the first one, to the best of our knowledge, that systematically evaluates secret leaks in misconfigured cloud storage. While Continella et al. [19] and Cable et al. [18] report some leaks through this channel, a systematic evaluation of this attack vector has never been done before.

Secret Validation Methods. As mentioned, it is not enough to detect a secret being leaked, because it might be revoked or used for testing. TruffleHog implements a pipeline for secret validation for diverse platforms, and they rely on interacting with the target APIs in a limited way, e.g., issuing a `GET` request [60]. GitHub's Secret Scanner also performs token validation for GitHub secrets [61] and, since October 2023, for the major cloud services with Amazon, Microsoft, Google, and Slack [62].

The studies by Wang et al. [34] and Zuo et al. [12] relied on differences in error manifestations to understand whether the tested credential is valid. This process is semi-automated, as it requires first extensively exploring the cloud APIs to be able to infer cases when the access control state can be captured in a non-intrusive way. As an alternative approach, Basak et al. [32] have manually inspected secrets collected in the SecretBench dataset in their context, and have relied on human expert opinions of the authors and project developers to label secrets as valid or not.

Our own work builds on the validation methods shared in the popular repositories Keyhacks, all-about-apikey, and Noseyparker, but we had to extensively test and redevelop their validation scripts to guarantee non-intrusiveness and absence of bugs. We also developed our own validation scripts for the new regexes we created.

Outcomes of Responsible Disclosure. The effectiveness and feasibility of responsible vulnerability disclosure and abuse reporting have been widely studied in the literature [37], [38], [39], [40], [41], [42], [43], [46], [63], [64], [65], [66], [67], [68], [69], [70], [71]. For example, Stock et al. [38], [42] and Li et al. [39] examined the feasibility and effectiveness of large-scale vulnerability notifications to owners (direct) and intermediaries and trusted third parties such as CERTs (indirect). They found the direct reporting to the owners to be the most effective, yet the overall remediation rates were much lower than $100\,\%$. Similarly, van Hove et al. [41] studied responses to a coordinated email vulnerability disclosure. They found that many organizations were difficult to reach for disclosure, and the reaction and fixing rate was lower than $50\,\%$. Our results of the responsible disclosure outcomes are similar to the findings from the literature, as we also observed the remediation levels being far from $100\,\%$, and underline the challenges related to the lack of established vulnerability disclosure channels. However, we have achieved a response rate of $59\,\%$ that is higher than in most of the studies, underlining the effectiveness of disclosures via a trusted third party.

Our study investigated the responsible disclosure process outcomes when done via a reputable third party (`CSIRT.global`) and for specific vulnerabilities (cloud data leaks). The responsible disclosure outcomes for this vulnerability, to the best of our knowledge, have not been examined before. Another new result of our study is its comprehensive analysis of organizational responsiveness to security notifications. By monitoring responses over an extended period (5 months) and categorizing them based on both timing and communication, we gained insights into how organizations handle and prioritize security vulnerabilities. This examination of response patterns – from swift actions within a day to long-term fixes occurring weeks after notification, and from a comprehensive to only partial mitigation – offers valuable data for understanding and potentially improving the disclosure and remediation processes in cybersecurity.

10. Conclusion

Our empirical study has discovered that, unsurprisingly, secret leaks with potentially huge impact exist in misconfigured cloud buckets. By performing large-scale analysis of files exposed in vulnerable buckets accessible via the GrayhatWarfare platform, we were able to automatically and non-intrusively discover 215 valid secret leaks. These leaked secrets belong to a variety of organizations from different sectors, sizes and countries, emphasizing that the issue of misconfigured buckets and development pipeline errors can affect any company. The case study examples we examined show that the impact of leaking these secrets can be immense, ranging from full control of organizations' security infrastructure to impersonation and infiltration into protected cloud infrastructure.

We have responsibly disclosed the discovered secret leaks to the bucket owners, whenever it was possible to discover the organization behind the bucket in a non-intrusive way, and to the cloud providers when this was not feasible. The outcomes of our responsible disclosure show that $59\,\%$ of the owners fixed the issue at least partially, thereby underscoring that such secret leaks are largely not intentional but indeed results of security misconfigurations. Future research should look into other types of files and data leaks exposed in the buckets, whenever the issues can be reliably identified in a non-intrusive way.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their helpful feedback. We are grateful to our partner organization CSIRT.global for their effort in the responsible disclosure of the discovered leaks.

This research has been partially supported by the Dutch Research Council (NWO) under the projects "Cyber Security by Integrated Design (C-SIDe)" (NWA.1215.18.008) and "THESEUS" (NWA.1215.18.006).

References

[1] M. Dahlmanns, C. Sander, R. Decker, and K. Wehrle, "Secrets revealed in container images: An internet-wide study on occurrence and impact," in *Proc. of ASIACCS*, 2023, pp. 797–811.

[2] B. Toulas, "Toyota discloses data leak after access key exposed on github," Bleeping Computer, October 2022. [Online]. Available: https://www.bleepingcomputer.com/news/security/toyota-discloses-data-leak-after-access-key-exposed-on-github/

[3] Ayoub Fathi, "Pwning cloud contexts, the endgame," 2023. [Online]. Available: https://blackhatmea.com/session/pwning-cloud-contexts-endgame

[4] A. Rahman, C. Parnin, and L. Williams, "The seven sins: Security smells in infrastructure as code scripts," in *Proc. of ICSE*. IEEE, 2019, pp. 164–175.

[5] J. Leon, "74% of publicly leaked keys are never revoked," Truffle Security Company, January 2024. [Online]. Available: https://trufflesecurity.com/blog/most-publicly-leaked-keys-are-never-revoked

[6] S. Gatlan, "Github enables push protection by default to stop secrets leak," Bleeping Computer, February 2024. [Online]. Available: https://www.bleepingcomputer.com/news/security/github-enables-push-protection-by-default-to-stop-secrets-leak/

[7] M. Meli, M. R. McNiece, and B. Reaves, "How bad can it git? Characterizing secret leakage in public GitHub repositories." in *Proc. of NDSS*, 2019.

[8] R. Feng, Z. Yan, S. Peng, and Y. Zhang, "Automated detection of password leakage from public GitHub repositories," in *Proc. of ICSE*, 2022, pp. 175–186.

[9] Y. Gu, L. Ying, H. Chai, C. Qiao, H. Duan, and X. Gao, "Continuous intrusion: Characterizing the security of continuous integration services," in *Proc. of IEEE S&P*. IEEE, 2023, pp. 1561–1577.

[10] G. Jungwirth, A. Saha, M. Schröder, T. Fiebig, M. Lindorfer, and J. Cito, "Connecting the. dotfiles: Checked-in secret exposure with extra (lateral movement) steps," in *Proc. of MSR*. IEEE, 2023, pp. 322–333.

[11] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of Google Play," in *Proc. of SIGMETRICS*, 2014, pp. 221–233.

[12] C. Zuo, Z. Lin, and Y. Zhang, "Why does your data leak? Uncovering the data leakage in cloud from mobile apps," in *Proc. of IEEE S&P*. IEEE, 2019, pp. 1296–1310.

[13] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda, and S. Loureiro, "A security analysis of Amazon's Elastic Compute cloud service," in *Proc. of SAC*, 2012, pp. 1427–1434.

[14] S. Bugiel, S. Nürnberger, T. Pöppelmann, A.-R. Sadeghi, and T. Schneider, "AmazonIA: When elasticity snaps back," in *Proc. of CCS*, 2011, pp. 389–400.

[15] Z. Whittaker and C. Page, "Microsoft employees exposed internal passwords in security lapse," TechCrunch, April 2024. [Online]. Available: https://techcrunch.com/2024/04/09/microsoft-employees-exposed-internal-passwords-security-lapse/

[16] "Sensitive information belonging to BMW exposed due to misconfigured cloud bucket," SOCRadar, February 2024. [Online]. Available: https://socradar.io/sensitive-information-belonging-to-bmw-exposed-due-to-misconfigured-cloud-bucket/

[17] J. Singh, "India's national logistics portal exposed sensitive personal data, trade records," TechCrunch, October 2023. [Online]. Available: https://techcrunch.com/2023/10/02/india-national-logistics-portal-marine-data-expose/

[18] J. Cable, D. Gregory, L. Izhikevich, and Z. Durumeric, "Stratosphere: Finding vulnerable cloud storage buckets," in *Proc. of RAID*, 2021, pp. 399–411.

[19] 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 ACSAC*, 2018, pp. 702–711.

[20] K. Izhikevich, G. M. Voelker, S. Savage, and L. Izhikevich, "Using honeybuckets to characterize cloud storage scanning in the wild," in *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2024, pp. 95–113.

[21] "Grayhatwarfare," Grayhatwarfare, 2024. [Online]. Available: https://www.grayhatwarfare.com/

[22] E. Wen, J. Wang, and J. Dietrich, "Secrethunter: A large-scale secret scanner for public git repositories," in *Proc. of TrustCom*. IEEE, 2022, pp. 123–130.

[23] S. Adhatarao and C. Lauradoux, "Exploitation and sanitization of hidden data in pdf files: Do security agencies sanitize their pdf files?" in *Proc. of IHMMSec*. ACM, 2021, pp. 35–44.

[24] Y. Feng, B. Liu, X. Cui, C. Liu, X. Kang, and J. Su, "A systematic method on pdf privacy leakage issues," in *Proc. of TrustCom/BigDataSE*. IEEE, 2018, pp. 1020–1029.

[25] M. Jackson, "GitGuardian. File types that most commonly contain sensitive information," 2021. [Online]. Available: https://blog.gitguardian.com/top-10-file-extensions/

[26] A. Polkovnychenko and S. Menash, "JFrog's security scanners discovered thousands of publicly exposed API tokens — and they're active! The full report," 2022. [Online]. Available: https://jfrog.com/blog/jfrogs-security-scanners-discovered-thousands-of-publicly-exposed-api-tokens-and-theyre-active/

[27] U. Shamay, "Spectral. where your code secrets hide: risky filetypes to know," 2021. [Online]. Available: https://spectralops.io/blog/where-your-code-secrets-hide-filetypes/

[28] N. Lykousas and C. Patsakis, "Tales from the Git: Automating the detection of secrets on code and assessing developers' passwords choices," in *Proc. of EuroS&P Workshops)*. IEEE, 2023, pp. 68–75.

[29] S. K. Basak, L. Neil, B. Reaves, and L. Williams, "What are the practices for secret management in software artifacts?" in *Proc. of SecDev*, 2022.

[30] I. Koishybayev, A. Nahapetyan, R. Zachariah, S. Muralee, B. Reaves, A. Kapravelos, and A. Machiry, "Characterizing the security of GitHub CI workflows," in *Proc. of USENIX Security*, 2022, pp. 2747–2763.

[31] Y. Huang, Y. Li, W. Wu, J. Zhang, and M. R. Lyu, "Your code secret belongs to me: Neural code completion tools can memorize hard-coded credentials," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 2515–2537, 2024.

[32] S. K. Basak, L. Neil, B. Reaves, and L. Williams, "SecretBench: A dataset of software secrets," in *Proc. of MSR*. IEEE, 2023, pp. 347–351.

[33] Truffle Security, "Trufflehog," 2024. [Online]. Available: https://trufflesecurity.com/trufflehog

[34] X. Wang, Y. Sun, S. Nanda, and X. Wang, "Credit karma: Understanding security implications of exposed cloud services through automated capability inference," in *Proc. of USENIX Security*, 2023, pp. 6007–6024.

[35] V. S. Sinha, D. Saha, P. Dhoolia, R. Padhye, and S. Mani, "Detecting and mitigating secret-key leaks in source code repositories," in *Proc. of MSR*. IEEE, 2015, pp. 396–400.

[36] S. K. Basak, J. Cox, B. Reaves, and L. Williams, "A comparative study of software secrets reporting by secret detection tools," in *Proc. of ESEM*. IEEE, 2023, pp. 1–12.

[37] M. Maaß, H. Pridöhl, D. Herrmann, and M. Hollick, "Best practices for notification studies for security and privacy issues on the internet," in *Proc. of ARES*, 2021.

[38] B. Stock, G. Pellegrino, C. Rossow, M. Johns, and M. Backes, "Hey, you have a problem: On the feasibility of large-scale Web vulnerability notification," in *Proc. of USENIX Security Symposium*, 2016.

[39] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson, "You've got vulnerability: Exploring effective vulnerability notifications," in *Proc. of USENIX Security*, 2016, pp. 1033–1050.

[40] M. Maass, A. Stöver, H. Pridöhl, S. Bretthauer, D. Herrmann, M. Hollick, and I. Spiecker, "Effective notification campaigns on the Web: A matter of trust, framing, and support," in *Proc. of USENIX Security Symposium*, 2021.

[41] K. van Hove, J. van der Ham-de Vos, and R. van Rijswijk-Deij, "Your vulnerability disclosure is important to us: An analysis of coordinated vulnerability disclosure responses using a real security issue," *arXiv preprint arXiv:2312.07284*, 2023.

[42] B. Stock, G. Pellegrino, F. Li, M. Backes, and C. Rossow, "Didn't you hear me?-towards more successful Web vulnerability notifications," in *Proc. of NDSS*, 2018.

[43] Y. Nosyk, M. Korczyński, C. H. Gañán, M. Król, Q. Lone, and A. Duda, "Don't get hijacked: Prevalence, mitigation, and impact of non-secure DNS dynamic updates," in *Proc. of TrustCom*, 2023.

[44] M. Ziv, L. Izhikevich, K. Ruth, K. Izhikevich, and Z. Durumeric, "ASdb: A system for classifying owners of autonomous systems," in *Internet Measurement Conference*, 2021, pp. 703–719.

[45] S. J. Witt, "An internet-wide investigation of publicly accessible databases," Master's thesis, University of Twente, 2023.

[46] A. Hennig, F. Neusser, A. A. Pawelek, D. Herrmann, and P. Mayer, "Standing out among the daily spam: How to catch website owners' attention by means of vulnerability notifications," in *CHI Extended Abstracts*, 2022.

[47] D. Bourke and D. Grzelak, "Breach detection at scale with AWS honey tokens," *Blackhat Asia, https://www. blackhat. com/asia-18/briefings. htmlbreach-detection-at-scale-withaws-honey-tokens*, pp. 20–23, 2018.

[48] M. Mouw, M. Cox, and F. Mijnen, "Profiling abuse of exposed secrets in public repositories," 2020.

[49] C. Farinella, A. Ahmed, and C. Watterson, "Git leaks: Boosting detection effectiveness through endpoint visibility," in *Proc. of TrustCom*. IEEE, 2021, pp. 701–709.

[50] A. Saha, T. Denning, V. Srikumar, and S. K. Kasera, "Secrets in source code: Reducing false positives using machine learning," in *Proc. of COMSNETS*. IEEE, 2020, pp. 168–175.

[51] S. Lounici, M. Rosa, C. M. Negri, S. Trabelsi, and M. Önen, "Optimizing leak detection in open-source platforms with machine learning techniques." in *Proc. of ICISSP*. SCITEPRESS, 2021, pp. 145–159.

[52] Y. Guo, J. Liu, W. Tang, and C. Huang, "ExSense: Extract sensitive information from unstructured data," *Computers & Security*, vol. 102, p. 102156, 2021.

[53] L. Niu, S. Mirza, Z. Maradni, and C. Pöpper, "CodexLeaks: Privacy leaks from code generation language models in GitHub Copilot," in *Proc. of USENIX Security*, 2023, pp. 2133–2150.

[54] Y. Zhang, Y. Yang, and Z. Lin, "Don't leak your keys: Understanding, measuring, and exploiting the appsecret leaks in mini-programs," in *Proc. of CCS*, 2023, pp. 2411–2425.

[55] S. Baskaran, L. Zhao, M. Mannan, and A. Youssef, "Measuring the leakage and exploitability of authentication secrets in super-apps: The WeChat case," in *Proc. of RAID*, 2023, pp. 727–743.

[56] D. Springall, Z. Durumeric, and J. A. Halderman, "FTP: The forgotten cloud," in *Proc. of DSN*. IEEE, 2016, pp. 503–513.

[57] A. Krause, J. H. Klemmer, N. Huaman, D. Wermke, Y. Acar, and S. Fahl, "Pushed by accident: A Mixed-Methods study on strategies of handling secret information in source code repositories," in *Proc. of USENIX Security*, 2023, pp. 2527–2544.

[58] S. K. Basak, L. Neil, B. Reaves, and L. Williams, "What challenges do developers face about checked-in secrets in software artifacts?" in *Proc. of ICSE*. IEEE, 2023, pp. 1635–1647.

[59] M. R. Rahman, N. Imtiaz, M.-A. Storey, and L. Williams, "Why secret detection tools are not enough: It's not just about false positives-an industrial case study," *Empirical Software Engineering*, vol. 27, no. 3, p. 59, 2022.

[60] J. Leon, "How TruffleHog verifies secrets," Truffle Security Company, February 2024. [Online]. Available: https://trufflesecurity.com/blog/how-trufflehog-verifies-secrets

[61] M. Sulakian, "Remediation made simple: Introducing new validity checks for GitHub tokens," GitHub, January 2023. [Online]. Available: https://github.blog/2023-01-19-remediation-made-simple-introducing-new-validity-checks-for-github-tokens/

[62] Z. Malik and C. Claessens, "Introducing secret scanning validity checks for major cloud services," GitHub, October 2023. [Online]. Available: https://github.blog/2023-10-04-introducing-secret-scanning-validity-checks-for-major-cloud-services/

[63] M. H. Jhaveri, O. Cetin, C. Gañán, T. Moore, and M. V. Eeten, "Abuse reporting and the fight against cybercrime," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, pp. 1–27, 2017.

[64] M. Vasek and T. Moore, "Do malware reports expedite cleanup? An experimental study," in *Proc. of CSET*, 2012.

[65] M. Vasek, M. Weeden, and T. Moore, "Measuring the impact of sharing abuse data with Web hosting providers," in *Proc. of WISCS*, 2016.

[66] O. Çetin, C. Ganán, L. Altena, T. Kasama, D. Inoue, K. Tamiya, Y. Tie, K. Yoshioka, and M. Van Eeten, "Cleaning up the internet of evil things: Real-world evidence on ISP and consumer efforts to remove Mirai," in *Proc. of NDSS*, 2019.

[67] C. Utz, M. Michels, M. Degeling, N. Marnau, and B. Stock, "Comparing large-scale privacy and security notifications," *Proc. on Privacy Enhancing Technologies*, 2023.

[68] M. Maaß, M.-P. Clement, and M. Hollick, "Snail mail beats email any day: On effective operator security notifications in the internet," in *Proc. of ARES*, 2021.

[69] W. Bai and Q. Wu, "Towards more effective responsible disclosure for vulnerability research," in *Proc. of EthiCS*, 2023.

[70] O. Çetin, C. Gañán, L. Altena, S. Tajalizadehkhoob, and M. Van Eeten, "Tell me you fixed it: Evaluating vulnerability notifications via quarantine networks," in *Proc. of EuroS&P*. IEEE, 2019.

[71] O. Cetin, C. Ganan, M. Korczynski, and M. Van Eeten, "Make notifications great again: Learning how to notify in the age of large-scale vulnerability scanning," in *Workshop on the Economics of Information Security (WEIS)*, vol. 23, 2017.

Meta-Review

The following meta-review was prepared by the program committee for the 2025 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

Summary

This paper systemically discovers numerous instances of keys and credentials in publicly accessible cloud storage buckets. The paper performs a comprehensive analysis of this sensitive data, and evaluates the outcomes of responsibly disclosing these findings to the data owners.

Scientific Contributions

- Provides a new data set for public use.
- Creates a new tool to enable future science.
- Identifies an impactful vulnerability.
- Provides a valuable step forward in an established field.

Reasons for Acceptance

1) The paper develops an automated system for scanning files in misconfigured cloud buckets to detect secret leaks.
2) The paper conducts a large-scale analysis of publicly accessible cloud buckets, identifying numerous instances of secret leaks.
3) The paper validates the exposed secrets using non-intrusive techniques.
4) The paper demonstrates that secret leaks in misconfigured cloud buckets occur across diverse organizations and cloud providers.
5) The paper provides case studies that highlight potential real-world impacts of secret leaks.
6) The paper evaluates the effectiveness of responsible disclosure by tracking the data owners' response.

Noteworthy Concerns

1) The paper analyzes a single snapshot of misconfigured cloud buckets, rather than a longitudinal data set. As such, the paper does not capture potential changes in cloud bucket security practices over time.