

MASARYK UNIVERSITY  
FACULTY OF INFORMATICS



# Automating Bug Bounty

MASTER'S THESIS

**Ján Masarik**

Brno, Spring 2019

*Replace this page with a copy of the official signed thesis assignment and the copy of the Statement of an Author.*

## **Declaration**

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Ján Masarik

**Advisor:** prof. RNDr. Václav Matyáš M.Sc. Ph.D.

## Acknowledgement

First, I would like to express huge gratitude to my advisor prof. RNDr. Václav Matyáš M.Sc. Ph.D. for the continuous guidance, personal approach, open-mindedness and for his interesting lectures that were one of the reasons I decided to study cybersecurity.

My thanks also goes to Anshuman Bhartiya, creator of the Bounty-Machine framework, for the idea and consultations that guided me through the initial steps of the implementation.

I would also like to thank my colleagues from Kiwi.com, especially to Stanislav Komanec for the encouragement and to Alex Viscreanu for his support and knowledge.

Last but not least, I would like to thank my family and girlfriend for their support and patience.

## **Abstract**

This thesis introduces the reasoning behind the increasing popularity of the bug bounty programs and analyzes the most common, impactful bugs whose discovery can be automated at scale. As there was no suitable existing solution for similar automation identified, the thesis introduces a new framework, specifically designed for the bug bounty automation use case. Particular attention of the thesis is dedicated the usable security of the object storage console on six major cloud providers, and a large scale quantitative research that revealed dozens of misconfigured Azure Blob Storage buckets containing highly sensitive data. A set of recommendations, based on the results of the research, was formulated and delivered to each of the tested cloud providers.

## **Keywords**

automation, bug bounty, Kubernetes, penetration testing, security

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Bug bounty</b>	<b>3</b>
2.1	<i>History</i>	3
2.2	<i>Benefits for the organization</i>	4
2.3	<i>Comparison with traditional approaches</i>	4
2.4	<i>Self-hosted bug bounty program</i>	6
2.5	<i>Bug bounty platforms</i>	7
2.5.1	HackerOne	7
2.5.2	Bugcrowd	9
2.5.3	Others	9
<b>3</b>	<b>Common bugs in bug bounty</b>	<b>11</b>
3.1	<i>Object storage misconfiguration</i>	11
3.1.1	Amazon Web Services S3	13
3.1.2	DigitalOcean Spaces	15
3.1.3	Azure Blob Storage	17
3.1.4	Google Cloud Storage	18
3.1.5	Alibaba Object Storage Service	20
3.1.6	Oracle Cloud Object Storage	21
3.2	<i>Leaked sensitive API keys</i>	23
3.3	<i>Broken Authentication</i>	24
3.4	<i>Security Misconfiguration</i>	25
3.5	<i>Subdomain takeover</i>	26
3.6	<i>Server Side Request Forgery</i>	27
<b>4</b>	<b>Bug bounty automation</b>	<b>29</b>
4.1	<i>Motivation</i>	29
4.2	<i>Existing solutions</i>	30
4.3	<i>Technology used</i>	31
4.3.1	Docker	32
4.3.2	Kubernetes	32
4.3.3	Argoproj	33
4.3.4	GitLab	36
<b>5</b>	<b>Proposed solution</b>	<b>37</b>

5.1	<i>Architecture</i> . . . . .	37
5.2	<i>Triggers</i> . . . . .	39
5.2.1	Manual . . . . .	39
5.2.2	Bugtab . . . . .	40
5.3	<i>Workflows</i> . . . . .	43
5.3.1	Subdomain enumeration . . . . .	43
5.3.2	Bucket enumeration . . . . .	44
5.3.3	Subdomain takeover . . . . .	46
5.3.4	Git secrets . . . . .	46
5.4	<i>Storage</i> . . . . .	47
5.5	<i>Alerting</i> . . . . .	47
5.6	<i>Evaluation</i> . . . . .	49
<b>6</b>	<b>Misconfigured Azure Blob Storage</b>	<b>50</b>
6.1	<i>Dataset</i> . . . . .	50
6.1.1	Forward DNS . . . . .	50
6.1.2	Alexa top 1 million . . . . .	51
6.2	<i>Enumeration workflow</i> . . . . .	51
6.3	<i>Results</i> . . . . .	51
6.4	<i>Recommendations</i> . . . . .	53
<b>7</b>	<b>Conclusions</b>	<b>55</b>
	<b>Appendices</b>	<b>69</b>
<b>A</b>	<b>Archive structure</b>	<b>70</b>
<b>B</b>	<b>Vulnerability disclosure email</b>	<b>72</b>
<b>C</b>	<b>DigitalOcean Spaces enumeration workflow</b>	<b>73</b>



# 1 Introduction

The bug bounty industry has experienced a significant growth in the past few years. Now, more and more companies are running their bug bounty program and accepting reports from hackers<sup>1</sup> around the world [1]. In Chapter 2, the bug bounty phenomenon will be introduced and discussed from the perspective of a hacker and a company running a program, along with a comparison with traditional application security approaches such as penetration testing.

Widespread, high impact bugs in bug bounty, along with the options of their automated discovery are analyzed in Chapter 3. A significant part of this chapter is focused on object storage misconfiguration bugs, that were responsible for a substantial amount of data breaches that affected companies such as Verizon, Viacom or Accenture [2, 4, 3], along with U.S. agencies NSA and NGA<sup>2</sup> [5, 6] in the past two years. The usable security of object storage console was analyzed on six different cloud providers, and mechanisms for enumerating bucket privileges by an unauthenticated attacker are introduced for each provider.

Worldwide competition, along with a rapidly growing amount of programs requires hackers to be either very creative in finding unique bugs or to automate part of their workflow, as only the first hacker reporting a bug gets paid. Numerous hackers, such as Eric Head, better known as *todayisnow*<sup>3</sup>, have proven that the financial gain from a large scale bug bounty automation may be in hundreds of thousands of dollars per year [9]. However, most of those tools are kept private, and there are only very few attempts to release a unified, open-source automation framework, similar to Metasploit [97] in the penetration testing field. Further motivation, already existing solutions, and the technological basis of the proposed framework is discussed in Chapter 4. In Chapter 5, the thesis introduces an attempt to fill this gap and

---

1. Term “hacker” in this thesis is always used in the meaning of an ethical computer hacker, also referred to as “white-hat hacker.” It is not referring to a person supporting or doing any illegal activities.

2. National Geospatial-Intelligence Agency (NGA) is a combat support and intelligence agency housed within the U.S. Department of Defense.

3. <https://bugcrowd.com/todayisnew>

provide a modular bug bounty automation framework, built on top of modern DevOps systems such as Kubernetes or GitLab.

Results from quantitative research finding Azure Blob Storage misconfigurations in the wild that revealed numerous potential data breaches are presented in the Chapter 6. Moreover, based on the knowledge obtained from both qualitative, and quantitative research performed, a set of recommendations for object storage providers is formed and discussed.

## 2 Bug bounty

From a high-level perspective, a bug bounty program is a simple process – the company publishes a page with assets to get tested, conditions that hackers need to follow and the rewards for the discovery of valid bugs.

Once a hacker finds a bug, he reports it to the company with a proof of concept exploit and the possible impact. The company or a dedicated triage team then validates the report, forwards it to the relevant team for remediation and rewards the researcher accordingly. The form of a reward may vary. It may be monetary, in the form of recognition (public hall of fame) or in the form of branded clothing (referred to as a “SWAG”).

In the past few years, this concept has proven to be beneficial for both companies and hackers. Companies receive continuous feedback on the security of their products, and hackers can earn money legally by doing what they enjoy, with a company of their choice.

### 2.1 History

The first known bug bounty program was initiated by Hunter & Ready in 1983 for their Versatile Real-Time Executive operating system. As a reward, Volkswagen Beetle (also known as “bug”) or an equivalent cash reward was offered.

The first modern-day bug bounty program was launched by Netscape in 1995, offering monetary rewards for the Netscape Navigator 2.0 Beta. Years after, in 2004, Mozilla Foundation started offering bounties for bugs found in their core software. Technological giants like Google and Facebook followed in 2010 and 2011, respectively. In April 2016, the first Federal bug bounty program, Hack the Pentagon was launched as a program of a first major, government-backed agency and the number of programs has substantially increased over the past few years [1].

Significant growth of programs has opened a vast amount of opportunities for hackers around the world. A good example is Santiago Lopez, who at the age of 19 has earned more than \$1,000,000 on the bug bounty platform HackerOne. By March 2019, he had over 1,683 valid

unique vulnerabilities discovered with companies such as Verizon Media, Twitter or HackerOne itself [9].

## 2.2 Benefits for the organization

Whether formalized or not, every mature technology company has a software development lifecycle (SDL). An SDL covers training, requirements, design, development, deployment, testing and response. While central to the testing and response phases, bug bounty programs provide valuable insights across all parts of the SDL. A bug bounty program may point out areas or applications where the company needs to put more focus or improve overall design practices [10]. For example, it may include severe authentication bugs that would point out to a missing language-independent authentication across services in the organization.

However, running a healthy bug bounty program also brings other benefits like building a strong, security-aware culture across the organization:

- Developers get access to reports from the world best hackers, which feeds the curiosity about the security industry.
- Non-technological departments like business development acknowledge the risks connected with having a public feedback form, which hackers utilize to submit Blind Cross-Site Scripting (XSS)<sup>1</sup> payloads or similar content injection attacks.
- Mid-level management may have an easier time recognizing the real security risks before an actual breach occurs.

## 2.3 Comparison with traditional approaches

There are currently two widespread security testing practices – penetration testing and dynamic application security testing (DAST).

---

1. Blind XSS occurs when the attacker input is saved and then executed in a different application. This fact makes it extremely dangerous, as the payload may get triggered in an internal application with privileged accesses.

Penetration testing is an authorized process done to find vulnerabilities in a given scope and exploit those, in order to make the tested system more secure [129]. It has been an essential part of the secure development lifecycle for years, and it is required by certifications such as The Payment Card Industry Data Security Standard (PCI DSS) [109]. However, as the current trend in development switched, from the waterfall model, where one release cycle could take even six months, to DevOps that is based on shipping features even multiple times a day, penetration tests twice a year are not sufficient anymore.

DAST is a program that tries to find vulnerabilities in web applications, while the application is already running. It performs a black box test<sup>2</sup> by actually performing attacks and analyzing responses of the application. This automated process can be done very frequently, but it is unable to cover the application in depth. Penetration testing, on the other hand, offers a depth of coverage, but is resource intensive and therefore applied only with a limited frequency. Bug bounties stand in the middle, covering far more of the application than DAST and occurring far more frequently than penetration testing [130].

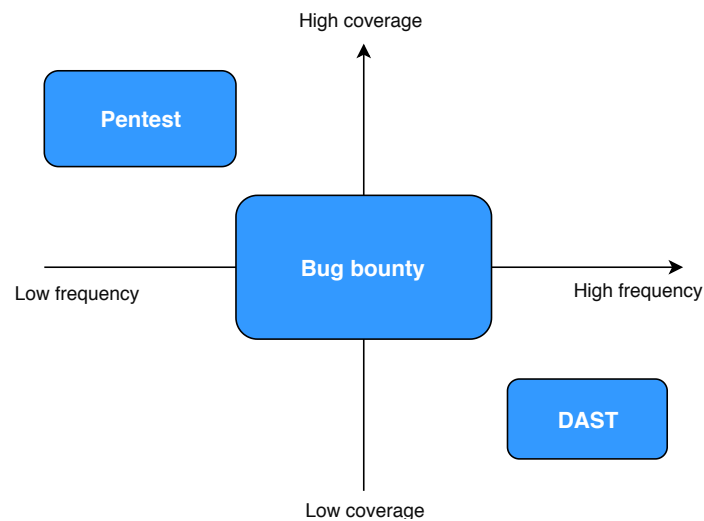


Figure 2.1: Coverage and the frequency continuance of the different security testing approaches [130].

2. Black box in this context means that the application does not have access to the source code.

Bug bounty programs are not a replacement for a penetration testing, as the coverage may not be as in-depth as in a penetration test and the compliance requirements still need to be met. However, because of their ongoing nature, bug bounty programs provide a continuous feedback loop. Combining penetration tests with a breadth of coverage of bug bounties enables penetration testers to be focused on the high risk, new, or recently changed application functionality [11].

Penetration testing	Bug bounty
Point-in-time – usually one to two weeks	Ongoing
Single researcher or a small team	High amount of researchers with different skillsets
Tightly scoped – single application or a specific feature	Typically broader scope
Mostly performed in staging environment	Performed on production with all defenses enabled
Standards/methodology driven – focused on coverage	Results-driven – opens doors for creative bugs
Paid by time spent	Paid per valid bug
Usually compliance driven	Security driven – not a compliance focus

Table 2.1: Comparison of penetration testing and bug bounty [128].

## 2.4 Self-hosted bug bounty program

Running a bug bounty program can be simplified as having a vulnerability disclosure policy (VDP) with details about rewards. A VDP is intended to give ethical hackers clear guidelines for submitting found

security vulnerabilities to organizations [108], but it does not necessarily have to include any mention of a bounty or any commitment on fixing reported vulnerabilities, only a way how security researchers can disclose details to the affected company.

Even though the guidelines for writing a VDP got standardized by the freely accessible ISO/IEC 29147 in 2014 [80], the adoption is still quite poor. In the middle of 2016, 94% of the Forbes Global 2000 list did not have the policy to receive, respond, and resolve critical bug reports submitted by the outside world [87]. At the end of 2017, this number barely decreased to 93% [1], which shows that this negligence is more or less stable.

Once a company has a VDP, it takes only a few modifications to turn it into a full-fledged bug bounty program. This modifications usually include adding a scope, rules, and rewards. However, companies need to understand that running a bug bounty program requires a significant amount of resources, as reports are usually far more frequent than in case of a VDP. Examples of reputable self-hosted bug bounty programs are the ones from technological giants like Facebook [51], or Google [50] that is offering rewards up to \$200,000.

## 2.5 Bug bounty platforms

Another option is to publish a bug bounty program on a Software-as-a-Service (SaaS) platform, that offers a web application tailored to the needs of vulnerability disclosure. Some of the benefits that these platforms offer include an easier payment system, already registered researchers waiting for new programs and an option for having a team dedicated to validate and verify the reports, usually known as triagers. The specific business model of a platform varies from always online programs to time-restricted tests only lasting a few weeks. As the total amount of such platforms is now more than 15 [79], only the most popular platforms will be reviewed.

### 2.5.1 HackerOne

HackerOne, currently the largest bug bounty platform was founded in 2012 by Dutch hackers Jobert Abma and Michiel Prins, Head of

Product Security at Facebook Alex Rice, and a startup founder Merijn Terheggen, after frustrating efforts of reporting vulnerabilities to 100 prominent high-tech companies during their challenge called “Hack 100” [82]. HackerOne quickly gained traction, and in January 2019 it was hosting a bug bounty program for 1,200 companies such as GitHub, with more than 300,000 hackers registered on the platform [48].

For companies, HackerOne is offering both private and public programs, in combination with self-managed or fully-managed triage of reports. In the case of a fully-managed option, HackerOne triage team is filtering and validating submissions of hackers before they get to the security team of the customer. Companies usually start with a private program and few dozen of invited researchers to not overload the security team or of the application itself. As the program matures, companies gradually invite more hackers and then potentially make their program completely public [101].

HackerOne is also very active in community work. At the beginning of 2018, HackerOne started an open-source educational series aimed mostly at newcomers to the bug bounty called Hacker101. This series became a very valuable starting point to new hackers, as it enabled them to receive invites to private programs just by finishing challenges from the Hacker101 Capture The Flag (CTF)<sup>3</sup> competition [81].

HackerOne also encourages public disclosure of reports after resolution with the feature called Hacktivity [102], that proved to be an invaluable resource of knowledge for bug bounty hunters and security teams around the world. Hacktivity, along with a book called Web Hacking 101 [86] that is based solely on public HackerOne submissions, was one of the main resources for the research on common bugs in bug bounty done in Chapter 3.

HackerOne ranking system is based on reputation, that the hacker gains or loses after each report gets closed. The amount of gained reputation is based on two factors – closing status of the report, such as resolved or not applicable, and the bounty amount. Total reputation is then being split into two metrics:

---

3. Hacker101 CTF is an always online training ground with various intentionally vulnerable web applications that contain hidden flags for which points are awarded.



- **Signal** – average reputation per report. As reputation is gained or lost each time a report is closed, the signal is an aggregate representation of report validity. It corresponds to reputation changes for triage states ranging from “spam” (-10) to “resolved.” (+7)
- **Impact** – average reputation per bounty. As reputation is gained based on the relative size of the awarded bounty, Impact is an aggregate representation of report severity. It corresponds to reputation gains calculated by bounty levels awarded [103].

### 2.5.2 Bugcrowd

The business offering of Bugcrowd is roughly the same as HackerOne except an option for self-managed programs [84]. From the community perspective, Bugcrowd also introduced an educational series aimed at beginners called Bugcrowd University. However, it has much less coverage than Hacker101 at the time of writing, and it does not offer any option of receiving private invites [83].

The ranking system for hackers is based on “kudos points,” that are assigned based on the severity of submitted reports. Private invites are then determined based on the number of points and additional metrics, such as average severity. Monetary rewards are excluded in the ranking, which highlights the main difference compared to the HackerOne ranking system [85].

### 2.5.3 Others

As previously mentioned, there are many more platforms emerging in this industry with a similar offering. Examples include:

- **Hacktrophy** – smaller Slovak platform, offering public bug bounty programs, aimed at the local business. This platform currently hosts bug bounty program of companies such as ESET, Mall or Moneta bank [88].
- **Intigrity** – a platform that hosts a majority of programs launched for open-source software such as Drupal or KeyPass, which received sponsorship from the European Union [89].

### 3 Common bugs in bug bounty

Bug bounty hunters follow a similar methodology as usual penetration testers, such as OWASP testing guide [104]. One of the most prevalent vulnerabilities in the world – Cross Site Scripting (XSS) [12, 13, 14], is also the most popular vulnerability among hackers on the HackerOne [1, 48].

This class of injection bugs has, however, already received substantial coverage by various researches before [41, 42] and it is also very rarely present as a root cause for breaches. On the other hand, two other vulnerability types covered in OWASP Top 10 2017<sup>1</sup> – *Using Components with Known Vulnerabilities* and *Security Misconfiguration* were together responsible as a root cause for 44% out of 50 analyzed breaches in the recent study of Snyk [91].

The focus of this chapter is mostly on similarly impactful, but easily automatable vulnerability types like security misconfiguration. These vulnerability types shortly follow XSS in its prevalence, but the discoverability is usually significantly easier to automate [1]. A growing amount of programs with a broad scope, along with an always-online nature of a bug bounty programs, make these automatable vulnerability types a perfect candidate for further research.

#### 3.1 Object storage misconfiguration

Object storage is a widely adopted cloud service, mostly used to store unstructured data like logs, backups, attachments or a static content of a website. It was designed with high durability and availability in mind, but the usable security aspect has been considerably overlooked in the past. Consoles had missing safety nets for less experienced administrators, that resulted in hundreds of leaks caused by a simple misconfiguration [132].

All newly created buckets on the tested providers had secure defaults, but each cloud provider is taking a slightly different approach

---

1. The OWASP Top 10 is a report outlining ten most impactful web application security issues. This report is updated roughly every three years, with 2017 being the most recent one at the time of writing.

on how to make its buckets more difficult to misconfigure. I would argue that Oracle Cloud Object Storage has the least expressive warnings in UI, along with some less secure default configurations. If not addressed, these issues could potentially lead to a considerable amount of future misconfigurations. Recommendations based on the quantitative research performed in the Chapter 6 are further discussed in Section 6.4 on page 53.

**The following areas were analyzed on each provider:**

- **Possible public permissions in the bucket ACLs** – a set of possible permissions which may be assigned to the bucket Access Control List (ACL) accessible either to an unauthenticated attacker or to an attacker in possession of a dummy account with the given provider. Each provider allows a different set of permissions assignable to the bucket ACL, but it can be generalized as a subset of the following permissions:
  - *READ* – permission that allows reading all objects by their full path. Rarely impactful on its own as it requires knowledge of an object name. In AWS S3 terminology, *READ* is used for the permission describing *LIST* in the thesis terminology.
  - *LIST* – permission that lists the objects in the bucket. If combined with public *READ* permission, it effectively allows to dump the content of the bucket. This may have a critical impact in case the bucket is used to store sensitive data.
  - *WRITE* – permission that allows to create, overwrite, and delete any object in the bucket. If misconfigured, it often has a high impact on its own due to the possibility of hosting malicious files or overwriting valid files, possibly leading to website takeovers or even remote code execution [47].
  - *READ\_ACP* – permission that allows reading the bucket ACL. The impact is usually low as it only leaks account names with access to the bucket along with their permissions, but it may be potentially escalated and misused in some targeted attacks.

- *WRITE\_ACP* – permission that gives the ability to modify the bucket ACL. Overwriting the ACL effectively leads to a bucket takeover and gives the full control over the bucket to the attacker. This is the most dangerous misconfiguration, commonly directly leading to a critical impact.
- **Ease of misconfiguration** – how difficult is it to set public access on bucket ACL and how well is the impact of the user actions expressed in the web console. For each provider, screenshot during or shortly after the bucket creation is included.
- **Default object ACL** – default permissions while uploading a new object to the bucket.
- **Enumeration flow of a bucket ACL for an unauthorized attacker** – detailed steps which an unauthorized attacker needs to make to enumerate the bucket ACL. As no previous research was done on some of the providers, and no modular tool for object storage misconfiguration analysis was found, I have developed and open sourced a tool called BucketsPerm [110]. This tool implements listed workflows for each provider and allows for easy analysis of bucket permissions at a larger scale.

#### 3.1.1 Amazon Web Services S3

The most popular, and also the most researched object storage service from Amazon Web Services (AWS) is a well-known source of income in bug bounty [47]. In 2017, there were 38 times more insecure storage vulnerabilities reported on the biggest bug bounty platform HackerOne, compared to 2016 [1]. AWS S3 has been a noteworthy source of data breaches for a vast amount of companies worldwide [132, 133].

In the end of 2017 (11 years after the launch of S3), AWS finally acknowledged the voice of security researchers worldwide, and they improved the visibility to bucket misconfigurations in the console. They have added warnings explaining the impact of set permissions, and also added an orange label for publicly accessible buckets in the main console [134]. In November 2018, they continued making improvements with global account bucket controls [135]. Both of those



Figure 3.1: Screenshot from the AWS S3 console during the bucket creation.

features helped to improve usable security of S3 dramatically, but there are likely still many misconfigured S3 buckets left in the wild.

- The administrator can assign any set of permissions to an unauthenticated user. Each of those misconfigurations except public *READ* is possible via a checkbox in the console.
- Since November 2018, four different checkboxes (Figure 3.1) need to be unchecked during bucket creation to allow any future misconfigurations with a public access [135].
- Setting a public bucket policy will trigger a warning with a message stating “Everyone has access to one or all of the following: list objects, write objects, read and write permissions.” I would argue that this warning is not sufficient and should state the impact more clearly.
- The default ACL for uploaded objects is private regardless of the bucket policy. This can be either specified during upload or overwritten by assigning a hand-written *READ* policy to the bucket.

Interestingly, it is possible to enumerate existing bucket names only based on DNS response. This allows for significant optimization of valid bucket names enumeration and reduces the number of requests sent to AWS infrastructure to a minimum.

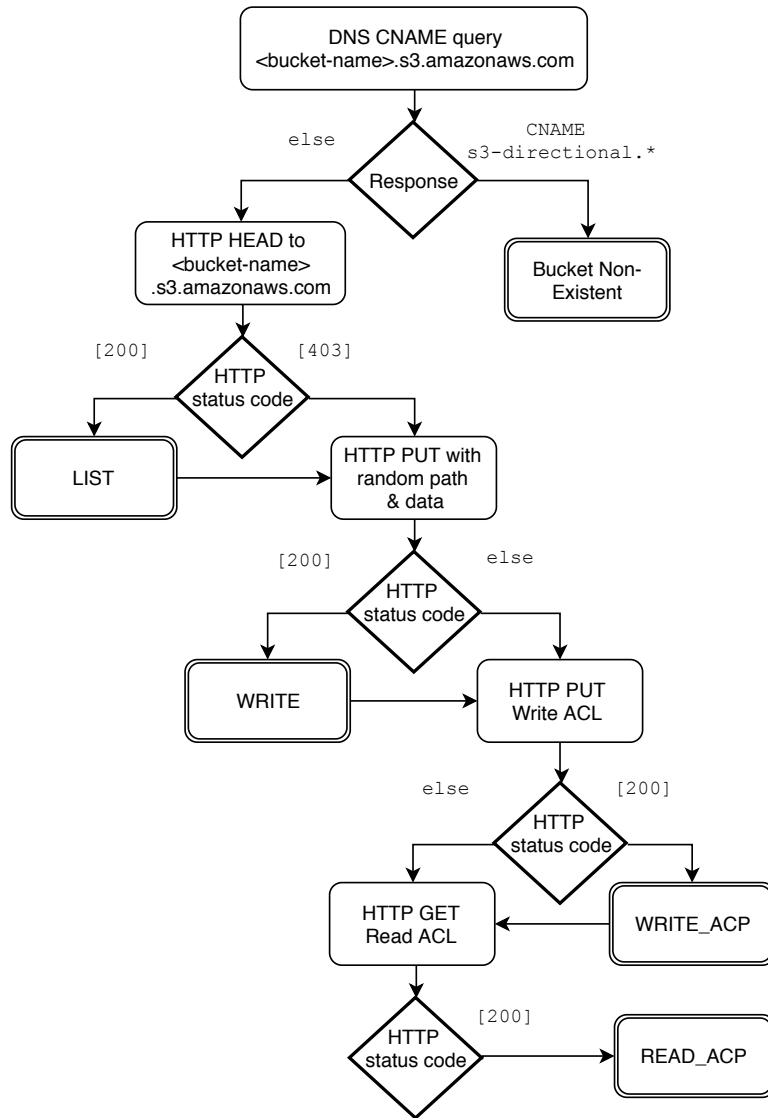


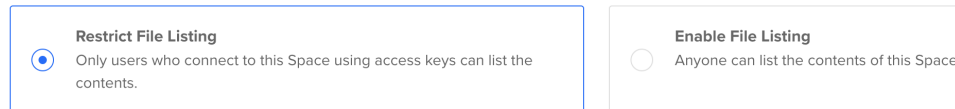
Figure 3.2: Enumeration of the S3 buckets.

### 3.1.2 DigitalOcean Spaces

DigitalOcean, one of the smaller, yet well-known cloud providers offers its object storage under the name DigitalOcean Spaces. The API attempts to be fully compatible with AWS S3, so the flow of enumeration is very similar except for the bucket identification based on

DNS records. Please see Appendix C for the complete enumeration workflow that was omitted for brevity.

Allow file listing?



☒ **Restrict File Listing**  
Only users who connect to this Space using access keys can list the contents.

☐ **Enable File Listing**  
Anyone can list the contents of this Space.

**Important:** This setting has no effect on whether individual files are visible. It only determines if anonymous users can list the name, size, and other metadata for the files in this Space.

Figure 3.3: Screenshot from the DigitalOcean Spaces console during the bucket creation.

- As DigitalOcean attempts to be fully compatible with the AWS S3 API, there are the same options of bucket ACL as in the AWS S3. However, there is an important difference; that only public *LIST* permission is possible via the DigitalOcean console. Other misconfigurations, such as public *WRITE* or *WRITE\_ACP* are possible only through the API [93]. Having these settings mutable only via API adds friction to the process, that arguably makes it a bit less prone to the misconfiguration.
- However, if the administrator assigns policy such as *WRITE\_ACP* to the bucket ACL through the DigitalOcean API, there is no indication of this configuration in the console. For example, an administrator can set a publicly listable policy through the API, but the file listing is still marked as “restricted” in the console. As this does not seem as a desired behaviour, it was reported as a bug to the DigitalOcean through their bug bounty program<sup>2</sup> and now awaits their response.
- The consequences of having a bucket ACL set as listable are well explained as seen in Figure C.1, but additional confirmation or visual elements are missing.
- The default object ACL is, similarly as the enumeration workflow the same as in the case of AWS S3.

2. <https://bugcrowd.com/digitalocean>

### 3.1.3 Azure Blob Storage

The object storage service from Microsoft Azure, called Azure Blob Storage, is offering quite a complex set of additional features such as monitoring or Advanced Threat Protection attempting to detect malicious activity on buckets. Tests of this feature were not conducted, but similar anomaly detection algorithms might be useful additional layer of protection against data leaks caused by misconfigured buckets.

\* Name

test-bugshop ✓

Public access level ⓘ

Container (anonymous read access for containers and blobs) ▼



 All container and blob data can be read by anonymous request. Clients can enumerate blobs within the container by anonymous request, but cannot enumerate containers within the storage account.

Figure 3.4: Screenshot from the Azure Blob Storage console during the bucket creation.

- It is possible to assign public *READ*, or *READ + LIST* policies in the console, but both are accompanied by a clear warning. Any other misconfigurations, such as public *WRITE* policy, are not possible [92]. There a column dedicated to public access level in the buckets overview, but further clarification of the impact or some warning elements are missing.
- The uploaded objects are strictly following the bucket ACL [111].
- The bucket names are namespaced to the account name, not global as in case of AWS S3 or Google Cloud Storage. However, it is possible to enumerate account names based on DNS responses, as DNS request containing a valid account name responds with CNAME to blob.<microsoft-id>.store.core.windows.net. This can be used for effective large scale enumeration of Azure Storage account names.



- Shared Access Signature (SAS), an equivalent of S3 pre-signed URLs, is the only option to open a bucket or an object to writing for unauthenticated users, where the documentation is very clear about the risks connected with it [112]. There are a few attack vectors involving SAS URL with writable permissions remain, but as the discovery is not easily automatable, it is out of the scope of this research.

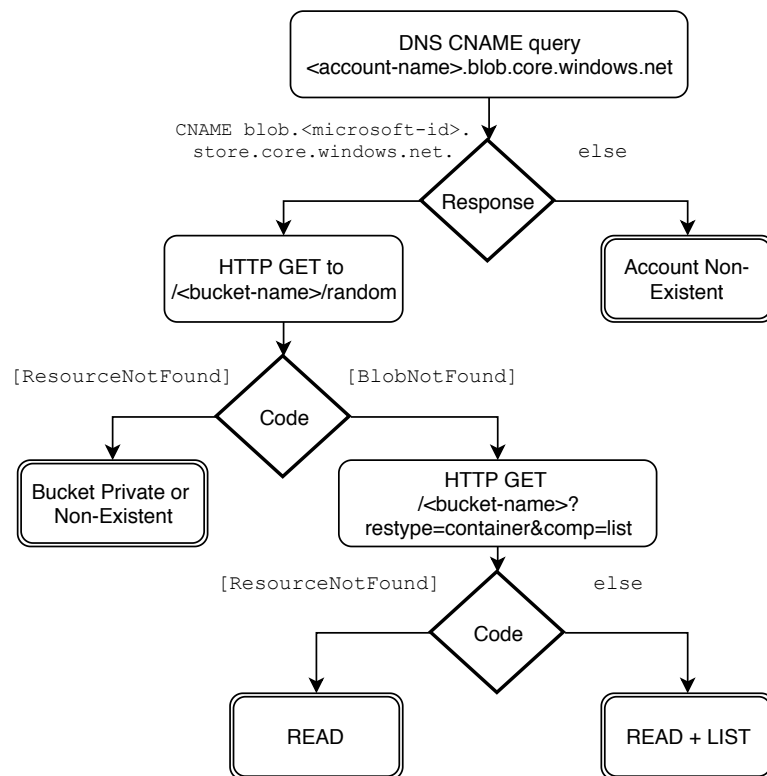


Figure 3.5: Enumeration of the Azure Blob Storage buckets.

### 3.1.4 Google Cloud Storage

Google Cloud Platform (GCP) offers its Google Cloud Storage (GCS) service since 2010, yet until recently it was overlooked during bug hunting efforts. As a critical misconfiguration of GCS is possible in the console, it may be arguably more prone to misconfiguration. This

was recently confirmed by the recently concluded research that discovered 34 critically misconfigured buckets among Alexa Top 10,000 websites [136].

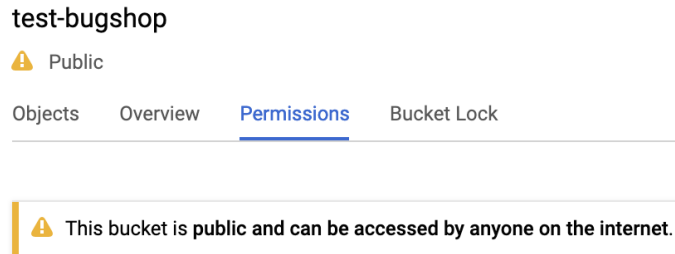


Figure 3.6: Screenshot from the Google Cloud Storage console after the creation of a public bucket.

- All misconfiguration options are possible, but GCS console does not provide a simple checkbox in the console to make the bucket publicly accessible. The administrator needs to write the name of a special group *allAuthenticatedUsers* (everyone with Google account) or *allUsers* (unauthenticated users) while assigning a policy to the bucket.
- During the misconfiguration, there is no warning explaining the possible impact displayed to the user. However, the GCS console starts displaying a permanent warning explaining the risks right after the bucket is misconfigured.
- The objects can be forced to inherit the bucket ACL by enabling a new beta feature called Bucket Policy Only (similarly to AWS S3).
- Privilege enumeration is possible via the *testPermissions* endpoint<sup>3</sup> instead of having to check each permission individually like in the rest of tested providers. Two requests should be made: one unauthenticated (group *allUsers*), and the second one authenticated with any Google account (group *allAuthenticatedUsers*).

3. <https://www.googleapis.com/storage/v1/b/<bucket-name>/iam/testPermissions>

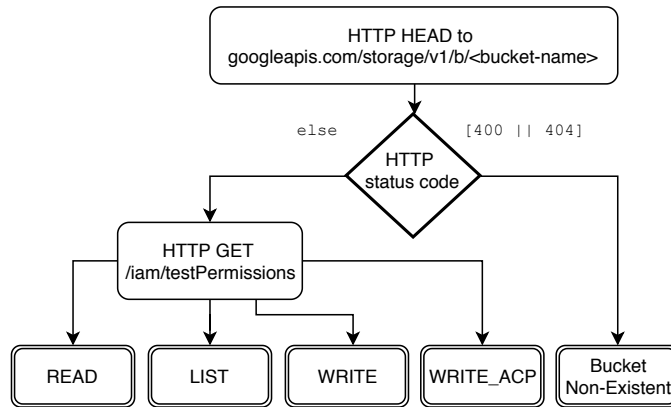


Figure 3.7: Enumeration of the Google Cloud Storage buckets.

### 3.1.5 Alibaba Object Storage Service

Alibaba Cloud, also known as Aliyun ([aliyun.com](http://aliyun.com)), is a Chinese cloud computing company that offers its object storage service under the name Alibaba Object Storage Service (OSS).

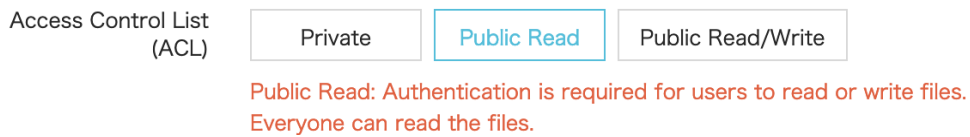


Figure 3.8: Screenshot from the Alibaba Object Storage Service console during the bucket creation.

- There are only two options for public misconfiguration – *READ* (Public Read) and *READ + WRITE* (Public Read/Write) policy, both accessible directly from the console. No option to assign public *LIST* permission to the bucket was found, which is unique among the tested providers and greatly limits the exploitability of the remaining misconfigurations.
- Both, Public Read and Public Read/Write, are accompanied by a clear warning and a prompt asking for confirmation.
- The default setting for a new object is set to inherit the ACL from the bucket, but it can be changed on an object level, similarly to AWS S3.

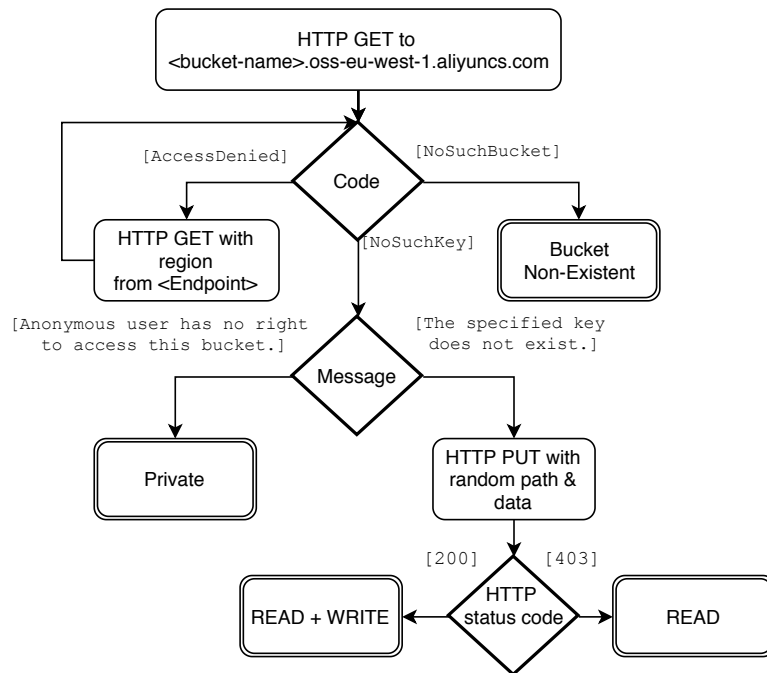


Figure 3.9: Enumeration of the Alibaba OSS buckets.

### 3.1.6 Oracle Cloud Object Storage

Oracle is one of the major players on the market, differentiating mostly with its data-as-a-service offering. From the usability perspective, the console feels the least user-friendly from all of the tested cloud providers.

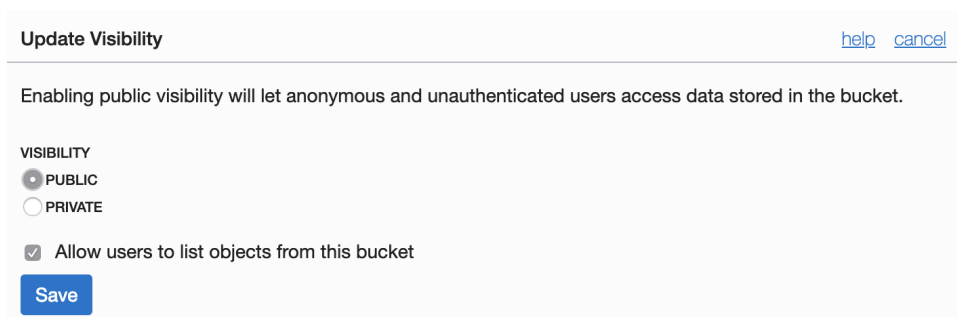


Figure 3.10: Screenshot from the Oracle Cloud Object Storage console while updating the visibility.

- Oracle allows only a limited set of public permissions – *READ* or *READ + LIST* policy. Both options are accessible through the console.
- Compared to the rest of the tested cloud providers, Oracle provides the least expressive warning about the consequences of public listable access. It is missing visual elements and, more importantly, a prompt with clearly explained impact.
- Insecure defaults are present in case the user decides to change the bucket from private to public. The checkbox “Allow users to list objects from this bucket” is turned on by default, that effectively makes a rarely suitable, and dangerous *LIST* policy, the default option for the public buckets.
- On the other hand, the enumeration of Oracle Cloud Object Storage ACL is the most difficult among the tested providers due to the very generic error messages. There is no way to enumerate valid namespaces or to know if a correct region was found, that greatly increases the difficulty of the enumeration process itself.
- Default setting for new objects is to inherit ACL from the bucket.

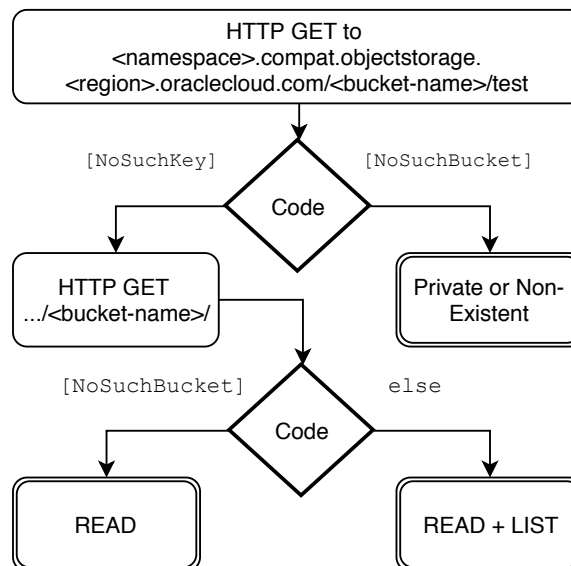


Figure 3.11: Enumeration of the Oracle Cloud Object Storage buckets.

### 3.2 Leaked sensitive API keys

Leaked credentials were always a critical issue that with the rise of cloud services only increased in its prevalence and severity. API keys to sensitive environments like AWS, Github, Slack or GCP are being leaked and exploited every day [16, 17, 15]. The most common sources are:

- **Git repositories** – leaked credentials are sometimes present in the organization repositories, or more frequently in one of the personal repositories of an employee [17, 106]. Extensive academic research on secret keys leaked in GitHub repositories by researchers from North Carolina State University yielded alarming results. Every day, thousands of new API keys are being pushed into public GitHub repositories, with more than 100,000 currently affected repositories overall [7].
- **Logs of continuous integration (CI) services** – sometimes companies either misconfigure their CI/CD service (such as Jenkins) to allow access from unauthorized users that then get access to the logs, or private tokens leak in intentionally public CI logs of various open-source projects [18, 107].
- **Embedded in application** – mobile or electron application developers tend to forget that everything they embed in the client-side application will be accessible by anyone in possession of the binary [15]. Sometimes, developers are also relying on client-side obfuscation, not realizing the fact that any client-side obfuscation can be deobfuscated.
- **Forgotten files on web servers** – it may happen that during debugging or doing some maintenance work, some file gets included on the web server. Impact of it may vary from mostly harmless *.DS\_Store* files<sup>4</sup>, up to the full access to the organization repository. A great example from this category includes a cloned private repository of Slack being temporarily available on one

---

4. *.DS\_Store* is Apple OS X specific file used to store attributes of folders with paths as described in the report <https://hackerone.com/reports/142549>

of the quality assurance servers under the `/.git/config` path for a short period of time [128].

Continuous monitoring is vital in case of similar “ephemeral” vulnerabilities, especially nowadays when both GitHub [20] and GitLab is searching for secrets in the code. This monitoring can be achieved by periodically running numerous open-source tools like *TruffleHog* [21] or *gitleaks* [22], searching for high-entropy strings or strings matching specific regular expressions. However, the research “How Bad Can it Git?” mentioned above, showed that even the best open-source tools such as *TruffleHog* are detecting only 29% of secrets compared to their methods [7].

### 3.3 Broken Authentication

Missing authentication or insufficient authorization is a widespread critical problem. In the OWASP Top 10 – 2017 list, Broken Authentication vulnerability type placed second, and Broken Access Control was on fifth place [14]. In bug bounties, it is a similarly prevalent issue. With the DevOps culture and growing amount of assets, companies sometimes deploy services publicly, and they forget to add an authentication layer on top of it, as open-source tools frequently do not have a built-in authentication. This leads to unauthenticated services revealing sensitive information or providing destructive functionality.

Examples of commonly exposed services:

- **Jenkins** – the impact of misconfigured authentication on Jenkins CI servers may lead from the unintentionally visible CI logs revealing sensitive data, up to Remote Code Execution using its plugins [18, 24, 23].
- **Kibana / Elasticsearch** – Elasticsearch, Logstash, Kibana stack (ELK) is widely deployed in various companies. However, the user interface (Kibana) does not have an authentication mechanism built-in in the basic version [105]. This frequently leads to publicly accessible Kibana dashboards, whose impact heavily varies based on the nature of data stored inside of the Elasticsearch cluster [28, 25, 27].

- **MongoDB** – insecure default configuration of MongoDB connected with its popularity caused havoc among the noticeable amount of companies hit by ransomware attacks, that encrypted their database. Even after several years, the number of misconfigured MongoDB instances in the wild is still alarming, and it continues to be a common source of data breaches [29].
- **Django Admin** – admin login typically present on the `/admin` endpoint on the web server is often missing any password brute-force protection, that many times allows to guess the correct admin credentials. Once the attacker gains access to the Django admin site, he can usually effectively compromise the website or the underlying database [34].

The automated discovery of similar bugs is usually achieved by subdomain enumeration tools discovering new assets. These assets are then periodically fingerprinted and, in the case of web services, a screenshot is taken. In case of any change, a notification is sent for further manual analysis, as some custom services cannot be possibly covered only by fingerprinting.

## 3.4 Security Misconfiguration

This type of vulnerability, representing the sixth place in OWASP Top 10 – 2017 [14], covers similar issues as previously discussed authentication errors. Some services still suffer from insecure defaults, or administrators enable debug features for testing and then forget to disable them on production. Severity is usually not as high as with previously mentioned categories, but its prevalence is compensating this fact in average bounties. Examples include:

- **Exposed phpinfo** – very common misconfiguration where PHP application leaves this debug functionality enabled on production. This page then allows anyone to request detailed information about the system on which the application is running [31, 32].
- **Django debug mode** – leaving debug mode turned on Python Django framework may result in various problems, the most



evident is that in case of exception in program flow, Django will display a detailed traceback, including a lot of metadata about application environment, such as all the currently defined Django settings. These settings usually contain secret tokens and configuration details, that, if exposed, may have devastating consequences. Documentation is unambiguous, stating “Never deploy a site into production with DEBUG turned on” [33]. This specific vulnerability was once a root cause of Remote Code Execution on Facebook servers that led to a reward of \$5,000 in their bug bounty program [35].

- **Directory listing** – it is common to leave directory listing enabled on a web server such as Apache which has this feature enabled by default. This problem is prevalent in various WordPress installations, potentially leading to the exposure of some sensitive files [36].

Similarly, as in the previous category, the discovery of security misconfiguration may be done by various vulnerability scanners with fingerprints for known vulnerabilities. Examples of open-source tools are Snallygaster [45] or Inception [44] that attempts to spread the load evenly across a vast amount of hosts. As both tools lacked necessary flexibility for my needs, I wrote and open-sourced a tool called *Low-hanging* [46].

### 3.5 Subdomain takeover

With the adoption of DevOps approach, digital footprint<sup>5</sup> of numerous companies IS currently increasing. Previously internal applications are often getting a public DNS record, that, along with an inefficient management of domain names, may frequently cause *dangling DNS records*<sup>6</sup> pointing to various cloud resources such as an AWS S3 bucket or Heroku application [126, 127]. Once claimed on the side of the resource provider, attacker can get full control over the subdomain

---

5. Cloud assets, DNS records, third-party services, and so forth

6. DNS record is considered dangling if the resource to which it is pointing to can be manipulated by a third party other than the one who controls the record.

and misuse it to launch phishing attacks, or gain access to the cookies of the parent domain.

Automation of this vulnerability class can be achieved by a continuous monitoring of DNS records, periodically sending an HTTP request and comparing the output against a list of fingerprints with applications susceptible to subdomain takeover. There are numerous open source tools helping with discovery, such as *Subjack* [122] or *Tko-sub*s [121]. Furthermore, there is a repository which attempts to centralize knowledge about subdomain takeovers, along with its fingerprints and exploitability named *can-i-take-over-xyz* [120].

### 3.6 Server Side Request Forgery

Even though Server Side Request Forgery (SSRF) is not easily detectable by automated scanners, the severity is rising dramatically with the growing adoption of cloud services. In connection with instance metadata server on several cloud providers like AWS or Google Cloud Platform, it sometimes allows for escalating up to the full compromise of the company account [37]. The instance metadata server is a special server present on 169.254.169.254, used to store details such as hostname, zone, tags, associated SSH keys and cloud provider keys. For reads and writes from within an instance, the metadata server provides automatic instance-level authentication and authorization. Each instance can read or write only to its own metadata server [38, 40].

In the SSRF attack, the attacker can abuse functionality on the server to request internal resources originating from the host itself. Example of such internal resource is the instance metadata server on AWS, that may then reveal AWS API tokens. In case the IAM permissions<sup>7</sup> are lax, and the attacker can retrieve keys belonging to the instance, he may completely take over the AWS account of the victim.

Google Cloud Platform recently added protections to its metadata server to address this issue. Every request to metadata server now must contain *Metadata-Flavor: Google* header and every request

---

7. AWS Identity and Access Management (IAM) is a web service that helps securely control access to AWS resources. It may be used to assign granular permissions to each instance or service, but it is commonly left with an unnecessarily permissive policy by AWS administrators.

with *X-Forwarded-For* header is discarded as it generally indicates that the requests are proxied. Both of these protections are, however, still bypassable through legacy *v1beta1* metadata endpoint [38], so the functionality. AWS, as the most popular cloud provider [131] does not take similar precautions yet and leaves securing of instance metadata server as a responsibility of the administrator.

The intriguing fact is that SSRF is currently on a relatively low rank in the OWASP Top 10 project on the 28th place as based on the dataset used in 2017, only 1.69% out of 114,000 applications contain this vulnerability [39]. In the most recent HackerOne Top 15 list, this vulnerability was, however, ranked on 15th spot [67], that may eventually forecast its possible rise in the upcoming years.

## 4 Bug bounty automation

Bug bounty automation is currently a subject undergoing intense study in the bug bounty community. In this chapter, the motivation for working on such automation, along with the requirements for a potential bug bounty automation framework will be discussed. Some existing solutions that emerged during the past two years will be discussed, but as none of the presented solutions fulfills the set of requirements, the technological basis of the proposed solution will be introduced.

### 4.1 Motivation

Bug bounty, being a very young and competitive scene, lacks a common platform for automation of tedious tasks. Various open-source command line tools are released frequently, but there is currently no common framework that would allow combining them easily. Even if such a framework is created, it is either kept private or shared only in small groups with the indisputable motivation of receiving a bounty. A robust, scalable and easily extensible solution is missing, and many talented bug hunters around the globe are forced to reimplement it over and over again. This goes against a known open-source principle that “no problem should ever have to be solved twice,” also mentioned in the *How To Become A Hacker* article by Eric Steven Raymond [99].

Existing penetration testing frameworks like Metasploit [97] may be useful, but they were initially made for a different use-case. These tools are only semi-automatic, and they mostly lack continuous monitoring functionality. In the end, they were done a different use case, as a penetration testing is usually done in depth on a narrow scope during a restricted timeframe.

Conversely, DAST, that is a fully automatic solution capable of continuous monitoring, was not built to provide a base ground for further manual testing and only focuses on finding vulnerabilities based on specific fingerprints. Another disadvantage of DAST is that it usually causes a considerable amount of traffic that may potentially overwhelm the servers of the target.

Sitting in the middle, an ideal bug bounty tool should arguably satisfy the following requirements:

- **Scalable** – the number of targets should be able to scale up to hundreds of wildcard domains, to be able to monitor all public bug bounty programs simultaneously.
- **Modular** – bug bounty techniques and tools are advancing quickly, so adding new functionality has to be straightforward.
- **Segregated duties and resources** – every step should have clearly defined input, output, separated logs, and resource usage. A single misbehaving step should not be able to impact the rest of the running tasks negatively.
- **Provide a base for manual testing** – results from the tool should be able to serve as a good resource for further manual analysis or testing.
- **Open-source** – as the framework is mostly using already created open-source tools; it should also be open-source.

### 4.2 Existing solutions

I would argue that providing a robust framework should be in the interest of bug bounty platforms. This would benefit both sides as hackers would not have to spend less time on automation that could consequently result in an increased amount of valuable, manual findings that benefit the platform. However, none of the platforms currently attempts to solve this issue. Some of the tools from bug bounty community emerged, but as the community is quite decentralized, they are mostly made by a single contributor.

- **Bounty Machine** – a tool that inspired the core infrastructure of later proposed solution. Built on top of Argo, with queues and proper database, this tool seems like a complete product. The original plan of this research was to contribute and help develop this already existing tool, but unfortunately, authors have decided to keep this tool closed source [94].

- **Intrigue Core** – fairly sophisticated tool written in Ruby with dozens of tests already included. Intrigue Core abstracts the attack surface into graphs where nodes represent an asset and edge are a task. This tool also comes with a built-in web interface, from which tasks can be initialized and results analyzed. A disadvantage of this solution is that the application itself is a monolithic ruby app, and thus, a single misbehaving tool can take down the whole application. Because of its monolithic nature, extensibility of this solution outside of native Ruby code may also be a problem as system changes could potentially affect dependencies of other already implemented tools [95]. The proposed solution may end up integrating Intrigue Core as one of the modules in the future.
- **Lazy Recon** – series of bash scripts by various authors such as Ben Sadeghipour that offers a quick and dirty solution for bug bounty automation. However, due to its bash nature, requirements such as modularity, compartmentalization or future scalability are not realistic [96].

As no existing solution matching all specified requirements was found, the rest of the thesis will introduce the proposed solution that was developed as an attempt to fill this gap. This proposed tool will be further referred to as *Bugshop*.

### 4.3 Technology used

All technologies used in the proposed framework are currently being utilized in the world leading technology companies such as Google. Interestingly, the utilization of modern DevOps technologies such as Docker is still rare in the security community. During the development of Bugshop, I have contributed Dockerfiles with an example of docker usage to some popular offensive security tools, such as S3Scanner [113] or Altdns [114], that positively impacted awareness about the benefits of Docker in the security community.

### 4.3.1 Docker

Docker is a tool released in 2013, designed to make it easier to create, deploy, and run applications by using containers. Containers allow developers to package up an application with all underlying parts it needs, such as libraries and other dependencies and ship it out as a single package. By doing so, it is assured that the application will run in the same environment on any other machine.

In a way, Docker is a bit like a virtual machine. However, unlike a virtual machine, Docker allows applications to use the same Linux kernel as the system that they are running on and only requires applications be shipped with things not running on the host computer. This gives docker a significant performance boost while reducing the size of the application [59]. Docker itself is, by now, a quite well-researched tool with various academic studies on its usability and security [60, 61], where further recommendations can be found.

Bugshop requires dockerized applications due to its underlying Kubernetes native infrastructure. However, “dockerization” also effectively abstracts the language and package requirements and enables to create an optimal environment for any given tool. The nature of containers also helps with debugging, as the production environment effectively matches the one on the local machine and real-time metrics about CPU usage, memory, consumed disk and network bandwidth, are available using *docker stats* [116]. Furthermore, Docker assures that a single misbehaving tool cannot crash the whole cluster, as it is possible to apply CPU and memory limits on each container easily.

### 4.3.2 Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications (usually, but not limited to Docker). Kubernetes is decoupling the application containers from the details of the system, and it also resolves networking within the cluster. It was built upon 15 years of experience of running production workloads at Google, and it allows scaling of a single cluster up to 5000 nodes (hosts) and 300,000 containers [60, 62, 63]. Choice of this technology was constrained the by choice of Argo, but

Kubernetes has lately established itself as the de facto standard for container orchestration [64].

Kubernetes is providing or supporting every functionality of Argo and provides easy scalability and deployment. During my development, I have used managed Kubernetes on Google Cloud Platform (GCP) called Google Kubernetes Engine (GKE). GCP offers a trial period for new accounts with free \$300 credits can be spent over one year [65]. Thanks to this, I was able to test and develop my solution without spending anything on the infrastructure. Their managed solution also effectively reduced the deployment time of Kubernetes to zero as the GKE comes preconfigured with everything required by Argo. Google also offers some convenient functionality like real-time monitoring of resource usage per container (as seen in Figure 4.1) and Cloud Identity-Aware Proxy [66] that was utilized to provide authentication to the web interface of Argo.

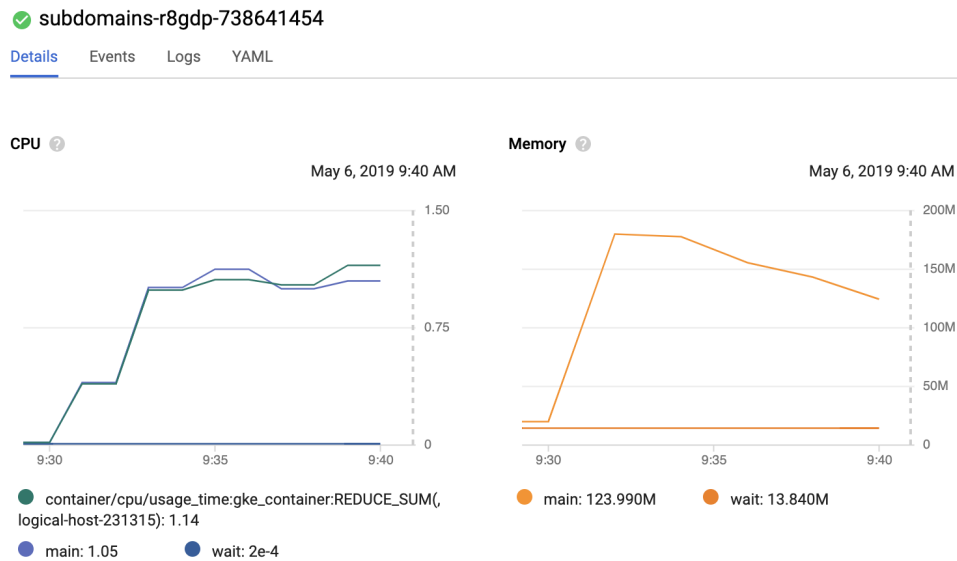


Figure 4.1: Resource usage graph of a container in the GKE console.

### 4.3.3 Argoproj

Argoproj is a collection of open-source tools for getting work done with Kubernetes. While being a very young project where the first



stable version 1.0.0 was released in August 2017, it is already being used by companies such as Google, NVIDIA, Datadog or Adobe for a part of their workload [68]. In addition to that, some popular open-source tools for Machine Learning, such as Kubeflow Pipelines were already built on top of Argo [69].

The original project of Argoproj is called Argo Workflows (also referred to as Argo), but by now, it consists of several parts:

- **Argo Workflows** – container-native Workflow Engine (utilized in Bugshop).
- **Argo Events** – event-based Dependency Manager (utilized in Bugshop).
- **Argo CD** – declarative GitOps Continuous Delivery.
- **Argo CI** – simple CI based on GitHub and Argo Workflows (currently inactive).

Alternatives of Argo include Apache Airflow [72] and Brigade made by Microsoft [73]. After a short evaluation, Argo was chosen due to its simplicity and active, helpful community built around it.

#### Argo Workflows

Argo Workflows is an open-source container-native workflow engine for orchestrating parallel jobs on Kubernetes. Argo Workflows are implemented as a Kubernetes Custom Resource Definition (CRD). It offers to define workflows where each step in the workflow is a separate Docker container with clearly defined responsibilities.

These steps can be then chained into multi-step workflows, where each level consists of one or more steps (containers). An alternative approach can capture the dependencies between tasks using a directed acyclic graph (DAG), similar to Apache Airflow. The multi-step workflow declaration is more straightforward to define and allows an option of dynamic workflows, where DAG based declaration offers much more granularity needed for the optimization of more complex workflows, such as buckets enumeration [71]. For the screenshot of bucket workflow, please see Figure 5.4 on page 46.

Workflows are represented by a yaml configuration file, where steps with clearly defined input, output, docker image, and artifacts. Example snippet of a single step can be seen on listing 4.1. This enables easy debugging and high level of modularity – if a new workflow needs to be developed, separate workflow file is created and deployed without potentially affecting any of the existing workflows. A significant advantage of this segregation is also a separation of logs and monitoring of resource usage, thanks to which the failure point is easily identified and debugged. In Bugshop, this is an essential feature as it mostly utilizes third-party open-source tools with questionable reliability.

```
– name: amass
  container:
    image: s14ve/amass:2.9.9
    volumeMounts:
      – name: config
        mountPath: "/config"
    resources:
      requests:
        memory: 4Gi
    args:
      [
        "-d", "{{ workflow.parameters.domain }}",
        "-c", "/config/amass.ini",
        "-o", "/tmp/amass.txt"
      ]
  outputs:
    artifacts:
      – name: results
        path: /tmp/amass.txt
```

Listing 4.1: Configuration snippet of a single step in subdomain enumeration workflow.

### Argo Events

Argo Events is an event-based dependency manager for Kubernetes that allows defining multiple dependencies from a variety of event

sources like webhooks, S3 notifications or calendar schedules. After the successful resolution of specified event dependencies, it triggers specified Argo Workflows. Events also offer additional functionality such as defining arbitrary logic after the resolution of dependencies or filtering of inputs, but it was not utilized for a simple use case of Bugshop.

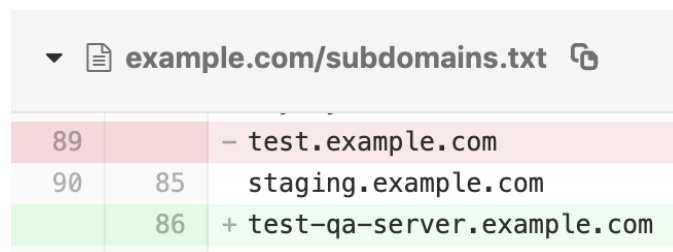
Argo Events is made up of two main components, both implemented as a Kubernetes Custom Resource Definition:

- **Gateway** that processes events from the event sources, such as HTTP webhook or S3 event notifications.
- **Sensor** that defines a set of event dependencies and after its fulfillment triggers Kubernetes resources, such as Argo Workflow.

#### 4.3.4 GitLab

GitLab is a web application that provides a Git-repository manager with wiki, issue-tracker and a native CI/CD pipelines for free [75]. In Bugshop, GitLab with its CI/CD is utilized not only as a repository of the source code but also as a crontab scheduler and storage of workflow results.

Commit history allows for easy tracking of changes (see Figure 4.2) and various integrations such as built-in alerting on changes can be further utilized. Using Git repository as a database in bug bounty was inspired by a blog post about the bug bounty monitoring [76] by the author of security.txt IETF draft [77] – Edwin Foudil. GitLab may get replaced by a full-fledged database in the future, but it is sufficient for the current requirements of Bugshop.



▼ example.com/subdomains.txt		
89		- test.example.com
90	85	staging.example.com
	86	+ test-qa-server.example.com

Figure 4.2: View of the changes on a commit in GitLab.

## 5 Proposed solution

As none of the existing solutions matched all specified requirements, a new automation framework was implemented with emphasis on easy extendibility and high scalability. The aim of this solution was to abstract the underlying technology and provide a reliable, modular framework for any bug bounty hunter with an intermediate technical background. The proposed solution can be deployed only with an elementary knowledge of Kubernetes and extended easily by adding or editing existing workflows via definitions written in YAML serialization language. The framework is currently split into two standalone components:

- **Bugshop** – the core of the tool that includes workflow definitions, configuration files, and Kubernetes manifests used to deploy Argo and Argo Events in the desired state.
- **Bugtab** – short python program and set of configuration files used to schedule tasks via GitLab CI pipeline schedules.

Majority of the Bugshop functionality comes from the utilization of already existing tools such as Argo or GitLab. Connecting existing tools to create something bigger instead of reimplementing it from the grounds up is ultimately the philosophy that Bugshop tries to embrace, similarly to UNIX philosophy of making each program do one thing well [115]. During the implementation of specific workflows, some gaps in open-source tooling have been identified, for which python tools following UNIX philosophy were developed and published under my GitHub account<sup>1</sup>. Some of them will be introduced in Section 5.3, dedicated to the implementation of workflows.

### 5.1 Architecture

Firstly, the terminology used in further sections needs to be defined. **Trigger** is used to describe an initiator of the task. It has a form of an HTTP request. **Notification** is a trigger initiated by some action, such

---

1. <https://github.com/janmasarik>

as detection of a new public program or a successfully finished task, where **task** is an instance of workflow with defined input parameters, such as domain name. **Workflow** is a set of tools, characterized as docker images in a specific order. For a more detailed explanation, please refer to Section 4.3.3. **Record** is a file with a specific structure that usually represents an output of a single workflow.

From a high-level perspective, the architecture of Bugshop can be split into four main phases:

1. **Trigger** – each new task is initiated by a trigger. Sources of triggers vary, but they can be generally split into three categories – invoked periodically, by notification, or manually by an operator that engages a new target. Triggering a new task is currently possible either via Argo Events HTTP gateway or via Kubernetes API.
2. **Process** – once a task is triggered, it starts the selected workflow and parses the results into some machine-readable output. Optionally, a task may trigger a notification to start new, normally dependent tasks. Processing of tasks is handled by the Argo Workflows engine in the Kubernetes cluster.
3. **Store** – parsed results of task are being stored in a persistent database with a log of past changes for further analysis. Bugshop uses GitLab repositories with a defined file structure as a storage for the results of tasks, and raw output of each step of every task is saved in a dedicated Google Cloud Storage bucket.
4. **Alert** – a change in some of the records usually means a new asset or a newly detected vulnerability. Bugshop alerts the operator on a change in selected files so he can evaluate the change in real time. Alerting is provided by GitLab's Slack or email integrations that can be set up to send a message on each push along with an overview of changes. (see Figure 5.6). In the future, this step will be partially replaced by an automatic submission of selected findings.

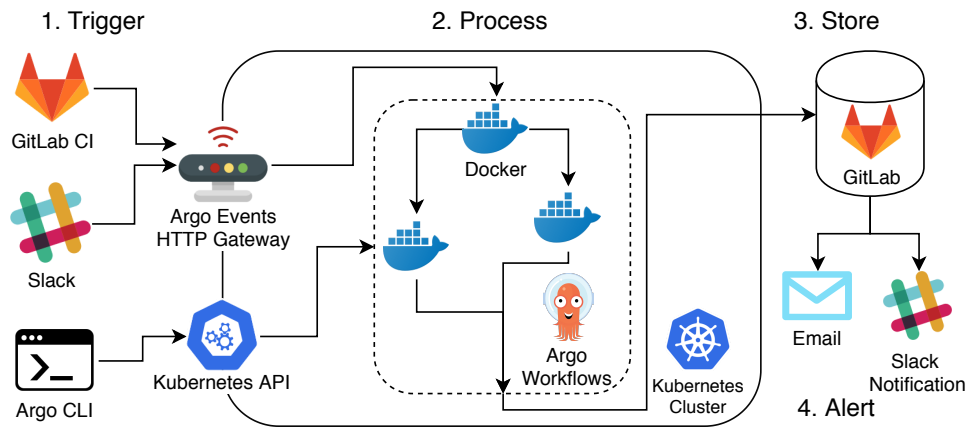


Figure 5.1: High level overview of the Bugshop.

## 5.2 Triggers

As shown in Figure 5.1, there are currently two trigger listeners implemented in BugShop; Argo Events gateway and Kubernetes API. Argo Events HTTP gateway is utilized by Bugtab, Slack webhook or as a listener for notifications from the onExit handler of workflows. New triggers can be added easily, as the gateway is essentially just a web server listening for POST requests with the specified JSON payload. Listener for Kubernetes API is used only via the Argo command line interface (Argo CLI), as it requires authentication to the Kubernetes cluster.

### 5.2.1 Manual

The simplest way how to trigger a workflow is by the Argo CLI. This command line interface from Argoproj offers commands for workflow initialization, termination, status check or logs. Argo CLI is essentially just a wrapper around the Kubernetes command line tool called *kubectl*, that allows you to run commands against Kubernetes clusters [52]. The additional value of Argo CLI is in features like syntax validation of workflow manifests, more readable output, and more intuitive interface. For example, the following command would be used to trigger a new workflow for subdomain enumeration:

```
argo submit subdomains-workflow.yml -p domain=example.com
```

### 5.2.2 Bugtab

Continuous monitoring is a fundamental feature of Bugshop that was originally planned to be implemented via the Argo Events calendar gateway. However, implementing it this way would require too many lines of duplicate code for each new target, which was a significant drawback considering the need for an easy extendibility of the scope. Furthermore, other use-cases requiring custom code emerged, such as automatically starting a full monitoring string on both, new public programs and new invites to private programs on HackerOne.

Because of this, a python utility called *Bugtab* was implemented, where GitLab CI schedules are then being used to execute it periodically and keep track of time. The architecture of Bugtab, represented as Gitlab CI in Figure 5.1 will be further described in this subsection and can be seen in detail in Figure 5.2.

Thanks to Bugtab, it was possible to achieve an easy extendibility of scope, as starting a continuous monitoring process on a new target requires only a few added lines into the scope, as illustrated in the Listing 5.1 below, and it also saves time by automatic monitoring of new bug bounty programs. Bugtab is designed in a modular way to allow others to potentially make use of it without having to use Bugshop or Gitlab CI.

Listing 5.1: Example entry in scope.yaml

```
starbucks :  
  domains :  
    - starbucks .com  
    - teavana .com  
  schedule : aggressive
```

#### Schedule

Bugtab currently consists of two configuration files *config.yaml* and *scope.yaml*. These two files are being parsed and linked together by Bugtab that initiates a series of HTTP requests containing the name of workflow along with required parameters to the HTTP gateway of Argo Events. To keep resulting configuration short, it groups the sets of modules into so-called “schedules” that contain details on how

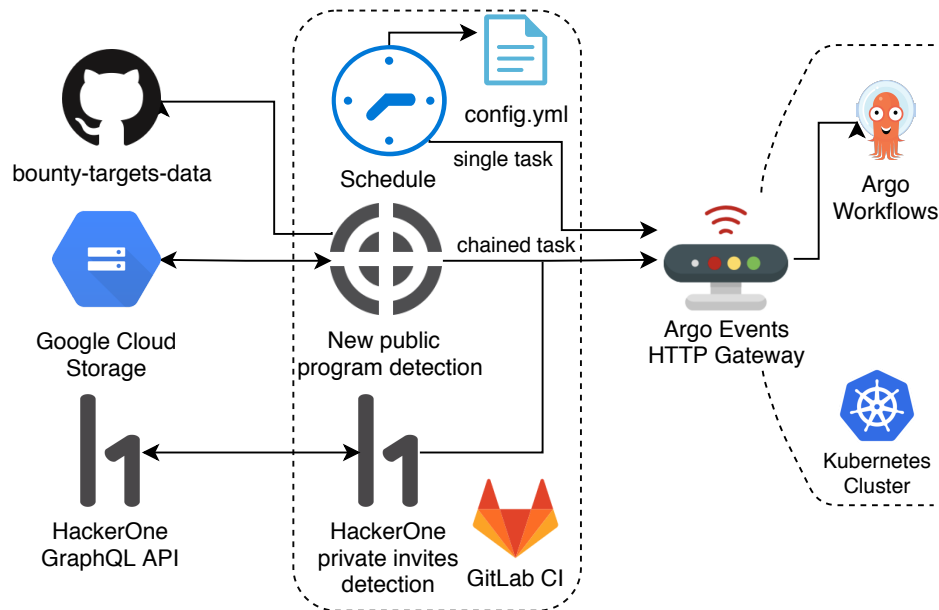


Figure 5.2: Detailed overview of the Bugtab.

often should each module run. Schedules are then assigned to each target, as seen on Listing 5.1. Currently, there are only two schedules implemented – *aggressive* and *light*, however, adding a new schedule takes just a few lines of configuration, as seen on Listing 5.2.

Listing 5.2: Example configuration of schedule in config.yaml

```
aggressive :
  - module: subdomains.yml
    frequency: 4h
  - module: subtakeover.yml
    frequency: 30m
  - module: buckets.yml
    frequency: 1d
```

To achieve a straightforward implementation, new GitLab CI pipeline schedule is set up for each desired interval, such as 30 minutes. It then passes the frequency identifier to the script that takes all modules that are supposed to run, parses the *scope.yaml* file and triggers the tasks.



### New public program

Besides schedules, Bugtab also implements a convenient feature that monitors a list of public programs of the two major bug bounty platforms – HackerOne and Bugcrowd. Once it detects a new wildcard domain to scan, it starts a special chained task that will trigger all available workflows in the relevant order.

This is implemented by hourly monitoring of GitHub repository *bounty-targets-data* that contains all wildcard domains from the scopes of public bug bounty programs. Each run, the program fetches the current *wildcards.txt* file, fetches the *wildcard.txt* file from the previous run saved in GCS bucket, and performs a set subtraction. If any changes were detected, it triggers an API request to Argo Events gateway. Afterward, it saves the current *wildcards.txt* file in the GCS bucket to prevent redundant executions in the next run.

### Private program invite

To add an additional level of automation, Bugtab is also triggering a scan upon receiving a private program invitation. In 2017, 79% of bug bounty programs on HackerOne platform were private, invite-only programs with only dozens of hackers participating [1]. It is usually easier to find bugs in private programs, as there is generally fewer hackers actively participating in it that makes private program invitations a valuable resource.

To get an invitation, a hacker needs to have a certain reputation, non-negative signal, and some activity in recent three months. The number of invites one hacker can get is limited, but if a hacker declines an invite on HackerOne and provides a reason why he did so, he is automatically assigned to the priority queue for new invitations. Priority queue effectively means that a new invite for the different program is sent the day after [117]. This causes hackers to choose programs in which they are going to participate very carefully.

As the new invitations are on HackerOne usually sent around noon, Gitlab CI schedule set for 13:00 checks for invites received in past 24 hours and potentially triggers a scan. Triggering a fully fledged scan on received program invitation allows the operator to make an informed decision faster, as additional information is available during

the decision process. This functionality is not finished in the Bugtab as of the time writing this thesis (May 19).

### 5.3 Workflows

The need for some workflow engine emerged while trying to do more complex, chained tasks, such as general bucket enumeration. There are numerous open-source, offensive bug bounty tools available and they are being grouped to various directories such as the one on Bug Bounty Forum<sup>2</sup>. However, new tools are often being created instead of chaining already existing tools together. A good example is *mass3*<sup>3</sup> that was created for enumeration of S3 buckets on the DNS level. There is, however, already an existing tool for brute forcing DNS called *MassDNS*<sup>4</sup>. Instead of implementing a new tool, it is easy to chain *MassDNS* to brute-force DNS lookups, *sed* to filter only valid buckets names, and then optionally run *BucketsPerm* to audit the permissions of the buckets found.

#### 5.3.1 Subdomain enumeration

When the bug bounty hunter first approaches a target with wildcard domain in scope, the first thing that he usually does is to run some subdomain enumeration tool. Various tools for this use case has emerged in the past five years. Examples of well maintained tools from this category are *Subfinder* or *Amass*. Both of these tools utilize various passive or active techniques for gathering subdomains, such as Shodan, Censys, Webarchive or Certificate Transparency.

These tools have some unique data sources and usually provide a slightly different set of results. Instead of choosing a single tool and relying on the maintainers to keep the sources up to date, we can instead just use them both, add *MassDNS* for brute-forcing and then merge the results. Merge process is done by a tool call *Resolvable*, during which the results are sorted, validated, and enriched by some additional information such as AAAA DNS records. This decreases

---

2. <https://bugbountyforum.com/tools>

3. <https://github.com/smiegles/mass3>

4. <https://github.com/blechschmidt/massdns>

the maintenance burden as even if one tool starts malfunctioning, the workflow will continue to work.

In the implemented workflow, this process is further extended by *AltDNS* that performs alternations of the existing subdomains in order to find similarly named domains, such as `subdomain-dev.example.com` from `subdomain.example.com`. This list is then passed to *MassDNS*, and the resulting list is again merged, resolved, enriched and stored.

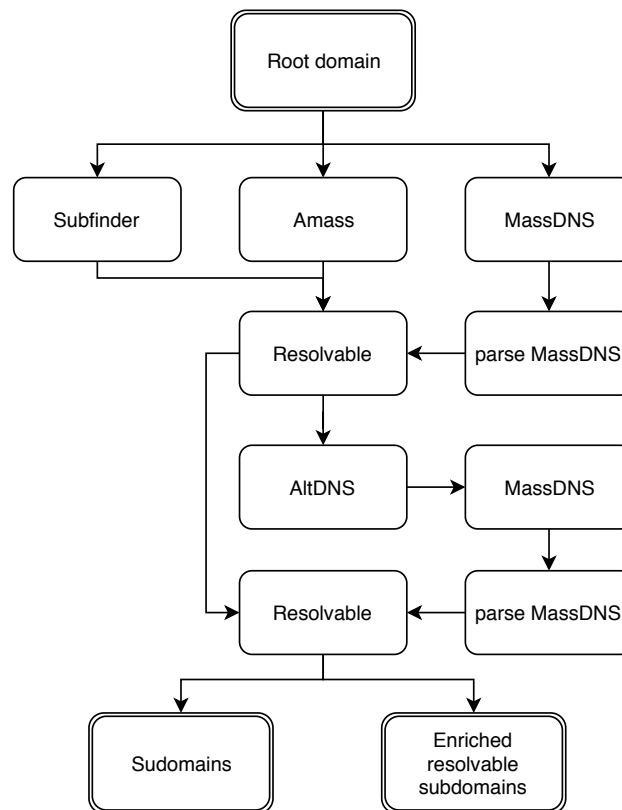


Figure 5.3: Subdomain enumeration workflow implemented in the Bugshop.

### 5.3.2 Bucket enumeration

Insecure buckets which got great coverage in the Section 3.1, are still a prevalent problem. This workflow tried to address it and brings some additional steps to the usual bucket enumeration workflows.

This workflow is implemented using the DAG model, as because of the complexity of the workflow, specifying granular dependancies moderately improved the performance. Part of this workflow can be seen in Figure 5.4.

The usual process of bucket enumeration takes a company name and generates a list of possible candidates by doing permutations with some wordlist. For example, if a company would be named `acme.com`, the usual approach would generate bucket names such as `acme-dev`, `acme-assets`, or `acme-attachments`. This approach proved to be quite effective, but as misconfiguration of S3 buckets is now regularly tested by a vast amount of researchers, most of the similarly named misconfigured buckets were already found and fixed. As AWS S3 considerably improved the usable security of their console, it is believed<sup>5</sup> that the number of new misconfigurations is not increasing as quickly as in the past.

Buckets workflow picks up the list of valid subdomains for the target, runs *AltDNS* with a customized wordlist of permutations on it, and then runs *AltDNS* on it again, but using a wordlist with a single entry – the name of the company. This leads to unique combinations that are directly dependant on the subdomains of the company, such as `test-subdomain-acme`. This is, however, done only for AWS S3 buckets, as the resulting permutation list is usually very long<sup>6</sup>, and only AWS S3 allows efficient brute-forcing via DNS. Specifically, *MassDNS* is used to find the names of S3 buckets and *BucketsPerm* used to check the permissions of the found buckets.

For the rest of the providers, usual permutations using domain and permutation wordlist are created using a simple tool written as a part of this thesis: *generate-bucketnames* [123]. The wordlist is then passed to *BucketsPerm* and once finished, the results are merged with AWS S3 results, and stored. List of all found buckets with related permissions is archived, and list of buckets with misconfigured permissions is saved to *target-alerts* repository.

---

5. No large scale research that could confirm this assumption was done in the past two years.

6. With list of only 100 subdomains and 1000 permutation words, this list takes at minimum 1 million entries (100 subdomains \* 1000 permutation words \* 10 positions/delimiters).

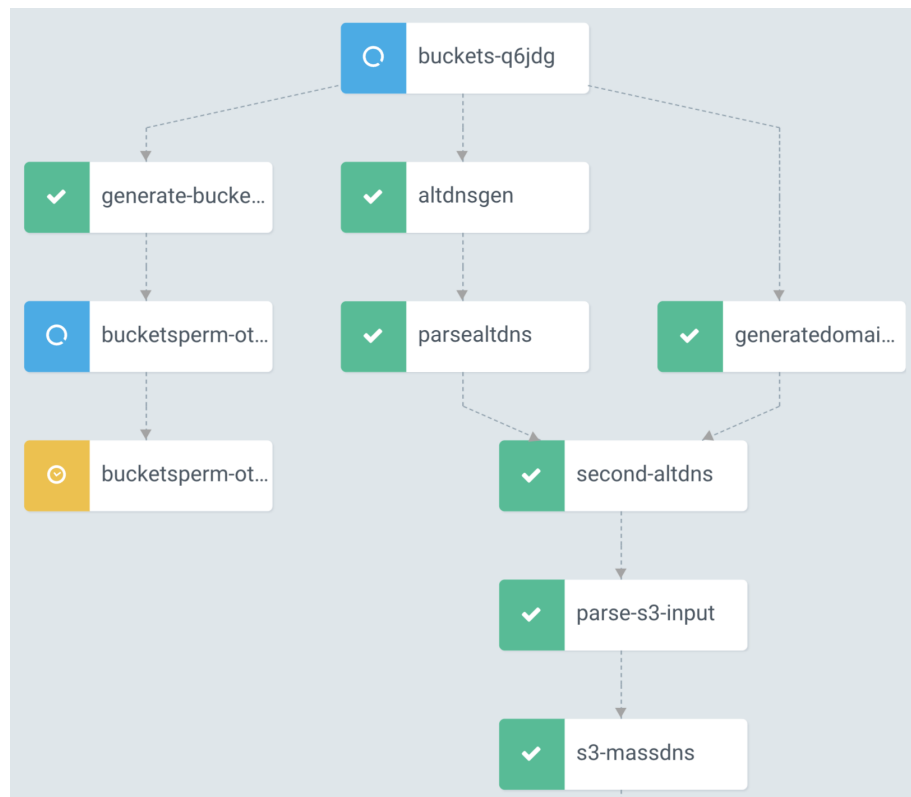


Figure 5.4: Screenshot of the buckets workflow from the Argo UI.

### 5.3.3 Subdomain takeover

Subdomain takeover workflow is implemented by running *Subjack* [122] and *Tko-subs* [121] in parallel and then merging the results using inline python script. This assures that even if one of the tools would start misbehaving, the workflow would still detect the vulnerable subdomain. *Tko-subs* has the functionality to take over the GitHub pages or Heroku application automatically, but it is currently not utilized in this workflow. I plan to include and extend this functionality in the future, to be used together with automatic reporting to HackerOne.

### 5.3.4 Git secrets

As mentioned in Section 3.2 on page 23, credentials pushed to a public Git repository are a quite easily discoverable issue with rewards rang-

ing up to \$15,000 [17]. Scanning of Git repositories for secrets is done using gitleaks [22] with custom config file combining entropy and regular expressions, inspired by rules published in the recent “How Bad Can it Git?” research [7].

Each rule has specified severity, which is a combination of confidence and impact. New results above a certain threshold then trigger an alert to the operator so he can validate the secret and submit the report. In the future, both of these steps can be automated by validating the key using a method from *keyhacks*<sup>7</sup> and creating a template for automatic submissions.

## 5.4 Storage

Most of the workflows are running periodically based on the schedule. This brought a need to store the results in some persistent storage. As the changes usually contain the most interesting information, a requirement for easily viewable changes along with an easy way of alerting on change was set. GitLab was then chosen as it fulfills both requirements out of the box, as described in Section 4.3.4.

Two different GitLab repositories are used with similar directory structure where the second-level domain represents the directory containing results from workflows, as shown in Figure 5.5. This offers an easy way how to have alerts only for specified, low false positive issues, with the rest of the data nicely viewable on one place. As all results are saved in a simple file structure, it is accessible for further manual ad hoc analysis using familiar tools such as *grep* or *sed*.

## 5.5 Alerting

As described in the previous Section 4.3.4, alerting is implemented using built-in GitLab integrations for email or Slack [118]. Both of these integrations can send a notification on every push to the repository, but email has an advantage of showing the diff directly in the content of the message even for private repositories. The operator then sets

---

7. <https://github.com/streaak/keyhacks>

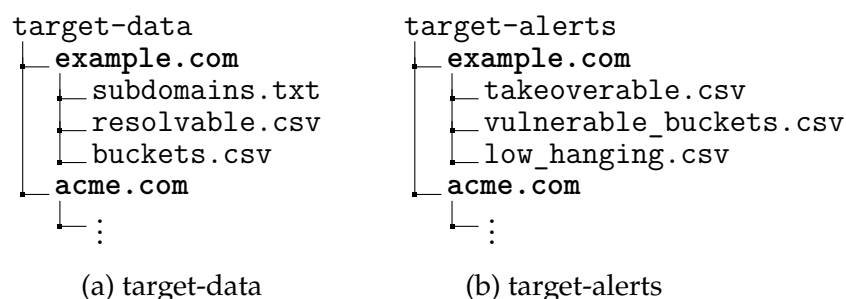


Figure 5.5: Structure of the GitLab storage repositories

up his phone to receive a notification every time he receives an email from the selected repository, which concludes the alerting process.

This integration is usually set up for the *target-alerts* repository where only results from selected, low false positive tools are being pushed not to overwhelm the operator. Alerting only for selected files is not possible out of the box in GitLab yet, but it is a tracked feature request<sup>8</sup> that would further increase the usefulness of this alerting system.

Ján Masarik pushed to branch master at [Ján Masarik / targets-alerts](#)

#### Commits:

- [22554163](#)  
by buckets-zu3np at 2019-05-05T13:37:42Z  
buckets-zu3np | [example.com](#)

#### Changes:

- [example.com/vulnerable\\_buckets.txt](#)

1	+ <a href="https://dev-example.s3.amazonaws.com">https://dev-example.s3.amazonaws.com</a> LIST
2	+ <a href="https://example-assets.s3.amazonaws.com">https://example-assets.s3.amazonaws.com</a> LIST,READ_ACP

Figure 5.6: Example of an email alert received from GitLab.

8. GitLab issue for notification subscription only to individual files is present at <https://gitlab.com/gitlab-org/gitlab-ee/issues/1817>.

## 5.6 Evaluation

Bugshop is a very young framework which is, however, already able to serve the purpose for which it was built. During the implementation, it has found three misconfigured buckets in scope, for which I was rewarded a few hundreds of dollars in private HackerOne programs. There is certainly some work left on the Bugshop, but it is already possible to partially evaluate fulfillment of requirements set in the Section 4.1:

- **Scalable** – the Kubernetes cluster on which Argo is running can scale up to up to 5000 hosts [63]. This means that the current bottleneck of Bugshop is not computing power, but the GitLab storage. GitLab was not built as a database, it does not allow high upload speeds, and uploading can fail due to a possible race condition during the push operation. However, even with a single GitLab repository effectively used as a database, Bugshop can currently scale up to scanning of hundreds of targets in parallel.
- **Modular** – adding a new module requires to add a separate workflow file that cannot affect the functionality of the already existing workflows.
- **Segregated duties and resources** – input and output of each task are clearly defined in the workflow definition. As every tool is running in a separate container, Docker resource usage limits can be applied to each step of the workflow.
- **Provide a base for manual testing** – this point heavily depends on the nature of the running workflows, but even with the initial set of workflows, it provides a considerable amount of detail about the target.
- **Open-source** – source code and configuration guide will be published on my GitHub page<sup>9</sup> under Apache-2.0 license.

---

9. <https://github.com/janmasarik>



## 6 Misconfigured Azure Blob Storage

Object storage service misconfiguration was analyzed on six major cloud providers in Section 3.1, where the usable security of the console was evaluated and the workflows for enumeration of the bucket privileges introduced. To demonstrate the severity of this vulnerability class, and to partially demonstrate potential of the workflows implementable in Argo, a large scale research on the misconfigured Azure Blob Storage buckets was concluded. Azure Blob Storage was chosen because it is possible to enumerate valid account names based on DNS records (see Section 3.1.3) and misconfiguration allowing to effectively dump a bucket is possible via a dropdown in the console. Moreover, there was no large scale public research concluded on Azure Blob Storage up to this date (May 2019). Two large datasets were utilized to find valid Azure Storage account names, that were then further brute-forced to find the valid bucket names along with its permissions.

### 6.1 Dataset

The initial dataset was a combination of data from the CNAME FDNS dataset and domains from Alexa top 1 million. Due to the slightly different nature of the datasets, the initial steps were slightly different, as described further in this section.

#### 6.1.1 Forward DNS

Forward DNS (FDNS) is a dataset that contains the responses to DNS requests for all forward DNS names known by Rapid7's Project Sonar. An up to date version of this dataset is freely available at the Project Sonar website<sup>1</sup>, and this has already proven to be quite effective in various researches related to DNS [125, 127]. As the Azure Storage account name can be identified based on <account-name>.blob.core.windows.net DNS CNAME records, an FDNS dataset containing CNAME records was used, and all CNAME records containing blob.core.windows.net substring were parsed into a newline delimited file for further analysis.

---

1. <https://opendata.rapid7.com/about/>

### 6.1.2 Alexa top 1 million

As choosing an account name that matches a company name is a commonly seen pattern, domain names from Alexa top 1 million dataset were chosen to further extend the size and variety of the dataset. From this dataset, the second-level domain<sup>2</sup> was taken and `.blob.core.windows.net` appended to the end of each line, that effectively constructed 1,000,000 unique candidates for valid account names, out of which 23,210 were identified as a valid Azure Storage account names.

## 6.2 Enumeration workflow

An Argo workflow implementing Figure 3.4 was developed using *MassDNS* chained with two iterations of *wfuzz*<sup>3</sup>. The file containing the merged datasets was used as the input to the workflow which configuration file can be found in the Bugshop repository, file *azure-brute-perf.yml*. As the name of a bucket is namespaced to the given account, a static wordlist with 700 entries was used to find existing buckets for each of the accounts. The top list of the most common bucket names was then created, analyzed, and the 100 most common occurrences split into two groups (see Figure 6.1). The first group contained names which suggested that the bucket does not contain any sensitive data, such as *images* or *public*. The second group consisted of names which suggested that the bucket holds potentially sensitive data, such as *backups*, *logs* or *invoices*. For each bucket name, samples with the highest amount of objects were chosen, and then manually analyzed for potentially sensitive data.

## 6.3 Results

The results of the research showed a great amount of misconfigured Azure Blob Storage buckets in the wild, where a noticeable amount

---

2. In the DNS hierarchy, a second-level domain is a domain directly below the top-level domain (TLD). For example, in `subexample.example.com`, `example` is the second-level domain of the `.com` TLD.

3. <https://github.com/xmendez/wfuzz>

## 6. MISCONFIGURED AZURE BLOB STORAGE

(a) Group 1		(b) Group 2	
bucket name	occurrences	bucket name	occurrences
images	662	assets	210
media	295	files	174
public	214	documents	97
videos	168	uploads	93
content	149	docs	81
static	104	backup	43
video	94	data	41
test	93	user-assets	39
temp	91	backups	30
logos	79	logs	29

Table 6.1: Top 10 bucket names found in both groups.

contained highly sensitive data, such as health records, invoices or backups. In total, 13,705 publicly readable buckets were identified, out of which 5,546 (~ 40.5%) were also publicly listable. As objects saved in Azure Blob Storage strictly adhere to the permissions set on the bucket level, each object from the 5,546 buckets could have been effectively downloaded by an unauthenticated attacker. It is necessary to note that all misconfigurations were due to human error, as Azure Blob Storage has safe defaults and warnings for the administrator configuring the bucket.

After performing a detailed manual analysis of the buckets belonging to the second group, 40 buckets full of personally identifiable information (PII), passwords, logs, financial information, or other sensitive information was identified. The companies that owned the vulnerable buckets included non-profit organizations, clinics, large medical school, technical start-ups, contractor of a major telecommunications provider, large marketing agency, dating sites, government election agency, airlines, e-shops, or ironically, companies offering security solutions.

I have attempted to identify all the account owners through the contact information found on the websites mentioned in the downloaded samples. A notification email with details and recommendations was then sent to all the affected parties (see Appendix B). In a few hours

from reporting, I have received emails from two C-level executives that expressed their gratitude. It is also important to mention that no large scale download of the objects was concluded to not violate the privacy of the affected parties. Only a few samples were downloaded, the minimum necessary to help with an identification of the bucket owner. Everything was permanently erased once the owner was identified and notified.

## 6.4 Recommendations

After the analysis of six different providers and the quantitative research done on Azure Blob Storage, some patterns that may help to increase usable security of the object storage console were observed. As we have seen in Section 3.1, each provider is taking a slightly different approach on how to prevent misconfiguration of the buckets permissions, but there is a set of recommendations applicable to most of them:

1. Have a **blacklist of the keywords** with a low false positive ratio, such as *invoice*, *backup* or *logs*, against which the bucket name is validated in the event of creation or reconfiguration. In case the name of the bucket will contain any of the blacklisted keywords and gets misconfigured to allow public access in the same time, an email notification clearly stating the possible impact will be sent to all administrators of the account.
2. Use more **visual elements** such as warnings with a red or orange color in the console, drawing the attention of the administrator to further investigate the possible impact.
3. Permanently display a **warning label** with the impact next to the bucket name in the console, and not just during the configuration process. Half of the analyzed providers (AWS, Aliyun, and Google) are already doing this, along with Azure, that has a column for publicly accessible buckets, but there are no visual elements which would hint a possible misconfiguration to an inexperienced administrator.

4. **Minimize the space for misconfigurations.** This approach was taken by half of the providers that allow to apply only a limited set of permissions to a public group. For example, Aliyun does not allow the public listing of the buckets, so there is no space for misconfiguration. Public listing can potentially be a valid use-case for some applications, even though in reality it is not necessary for the majority of the observed cases. However, during my research, no use-case for having publicly overwritable bucket ACL (*WRITE\_ACP*<sup>4</sup>) was found, yet Google, AWS and DigitalOcean (only through the API) still allow it.

This list of recommendations was sent to all tested providers, and now awaits their response. Especially the blacklist approach brings a novel idea on how to reduce the amount of misconfigured buckets in the wild. Even though the providers have much better access to similar data, I have included the composed blacklist of low false positive keywords based on the research in the notice.

---

4. As described in Section 3.1, public *WRITE\_ACP* effectively allows to take over the bucket for an unauthenticated attacker.

## 7 Conclusions

The thesis analyzed widespread, high severity vulnerabilities commonly discovered in bug bounty programs and introduced a novel attack methods targeting one of the most prominent vulnerability over the last few years – *object storage misconfiguration* on six different cloud providers.

To demonstrate the severity of this issue, a large scale research on misconfigured Azure Blob Storage was concluded. This was the first large scale research focusing on the Azure Blob Storage in the wild, and the results of it confirmed the assumption about the severity of this issue. A substantial amount (5,546) of the buckets with full public access was found, and after a tedious manual analysis, 40 buckets containing highly sensitive data were identified. All affected companies were notified by an email explaining the risks and a recommended mitigation. Furthermore, the list of recommendations created in Section 6.4 was communicated to the six tested cloud providers and now awaits their response regarding a possible implementation.

This research showed that while the discoverability of misconfigured buckets is a fully automated process, the identification of the owner can be a very tedious and manual task, as there is no easy way on how to reach the bucket owner. Due to the time-consuming nature of this process, the motivation for similar large scale researches may be arguably more tempting for malicious actors. No vulnerable bucket belonging to a company with a public bug bounty program was identified, but due to the small sample size and the fact that around 70% of the bug bounty programs is private, no reliable conclusion on this correlation can be drawn.

The second added value of the thesis was the introduction of a new open-source framework for the automation of bug bounty. This framework was built in a modular way that allows easy extensibility, which is necessary for the quickly advancing bug bounty industry. While the added value has not been proven by numerous users yet, it is actively being used in my bug bounty flow, and it is currently adjusted internally for the defensive purposes of the technology company Kiwi.com.

Both the framework and the research about object storage misconfiguration has several open-ended problems and a need for further quantitative research. This thesis may hopefully serve as a trigger for further research in this high impact area that shows that whenever there is human-factor involved in a simple, yet impactful security decisions, there is an urgent need to keep usable security in mind. Even though clear warnings could arguably prevent the majority of the misconfigurations, it will not protect against negligent or potentially malicious actors that brings the need for additional counter-measures, such as the proposed automatic notification for the remaining administrators.

## Bibliography

- [1] *The Hacker-Powered Security Report 2018* [Online]. Available: <https://www.hackerone.com/sites/default/files/2018-07/The%20Hacker-Powered%20Security%20Report%202018.pdf>
- [2] *Millions of Verizon customer records exposed in security lapse* [Online]. Available: <https://www.zdnet.com/article/millions-verizon-customer-records-israeli-data/>
- [3] *Accenture left a huge trove of highly sensitive data on exposed servers* [Online]. Available: <https://www.zdnet.com/article/accenture-left-a-huge-trove-of-client-passwords-on-exposed-servers/>
- [4] *Viacom Leak May Have Exposed Hundreds of Digital Properties* [Online]. Available: <https://gizmodo.com/viacom-leak-may-have-exposed-hundreds-of-digital-proper-1818504796>
- [5] *NSA leak exposes Red Disk, the Army's failed intelligence system* [Online]. Available: <https://www.zdnet.com/article/nsa-leak-inscom-exposes-red-disk-intelligence-system/>
- [6] *Security company finds unsecured bucket of US military images on AWS* [Online]. Available: [https://www.theregister.co.uk/2017/06/01/us\\_national\\_geospatial\\_intelligence\\_agency\\_leak/](https://www.theregister.co.uk/2017/06/01/us_national_geospatial_intelligence_agency_leak/)
- [7] Michael Meli, Matthew R. McNiece, Bradley Reaves *How Bad Can It Get? Characterizing Secret Leakage in Public GitHub Repositories* North Carolina State University, 2019.
- [8] Ravi Sen and Sharad Borle *Estimating the Contextual Risk of Data Breach: An Empirical Approach*, Journal of Management Information Systems, page 314-341, Routledge 2015. Available: <https://doi.org/10.1080/07421222.2015.1063315>
- [9] *@try\_to\_hack makes history as first bug bounty hacker to earn over \$1 Million* [Online]. Available: <https://www.hackerone.com/blog/trytohack-Makes-History-First-Bug-Bounty-Hacker-Earn-over-1-Million>



## BIBLIOGRAPHY

---

- [10] *Bug bounty program basics for companies* [Online]. Available: <https://www.hackerone.com/resources/bug-bounty-basics>
- [11] *Bug Bounty Programs: Enterprise Implementation* [Online]. Available: <https://www.sans.org/reading-room/whitepapers/application/bug-bounty-programs-enterprise-implementation-38250>
- [12] *Edgescan vulnerability statistics report* [Online]. Available: <https://www.edgescan.com/wp-content/uploads/2019/02/edgescan-Vulnerability-Stats-Report-2019.pdf>
- [13] *XSS Attacks: The Next Wave* [Online]. Available: <https://snyk.io/blog/xss-attacks-the-next-wave/>
- [14] *OWASP Top 10 – 2017* [Online]. Available: [https://www.owasp.org/images/7/72/OWASP\\_Top\\_10-2017\\_%28en%29.pdf.pdf](https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf)
- [15] *Leaking sensitive information on Github lead full access to all Grab Slack channels* [Online]. Available: <https://hackerone.com/reports/397527>
- [16] *AWS urges developers to scrub GitHub of secret keys* [Online]. Available: <https://www.itnews.com.au/news/aws-urges-developers-to-scrub-github-of-secret-keys-375785>
- [17] *Github Token Leaked publicly for https://github.sc-corp.net* [Online]. Available: <https://hackerone.com/reports/396467>
- [18] *Open prod Jenkins instance* [Online]. Available: <https://hackerone.com/reports/231460>
- [19] *A HackerOne employee's GitHub personal access token exposed in Travis CI build logs* [Online]. Available: <https://hackerone.com/reports/215625>
- [20] *Github About token scanning* [Online]. Available: <https://help.github.com/en/articles/about-token-scanning>
- [21] *TruffleHog* [Online]. Available: <https://github.com/dxa4481/truffleHog>

## BIBLIOGRAPHY

---

- [22] *gitleaks* [Online]. Available: <https://github.com/zricethezav/gitleaks>
- [23] *RCE/LFI on test Jenkins instance due to improper authentication flow* [Online]. Available: <https://hackerone.com/reports/258117>
- [24] *Public Jenkins instance with /script enabled* [Online]. Available: <https://hackerone.com/reports/403402>
- [25] *Unsecured Elasticsearch instance* [Online]. Available: <https://hackerone.com/reports/267161>
- [26] *Unsecured Kibana/Elasticsearch instance* [Online]. Available: <https://hackerone.com/reports/188482>
- [27] *Protecting Against Attacks that Hold Your Data for Ransom* [Online]. Available: <https://www.elastic.co/blog/protecting-against-attacks-that-hold-your-data-for-ransom>
- [28] *ElasticSearch server exposed the personal data of over 57 million US citizens* [Online]. Available: <https://www.zdnet.com/article/elasticsearch-server-exposed-the-personal-data-of-over-57-million-us-c>
- [29] *MongoDB databases still being held for ransom, two years after attacks started* [Online]. Available: <https://www.zdnet.com/article/mongodb-databases-still-being-held-for-ransom-two-years-after-attacks->
- [30] *aquatone* [Online]. Available: <https://github.com/michenriksen/aquatone>
- [31] *PHP info page disclosure on www.day.dk* [Online]. Available: <https://hackerone.com/reports/165930>
- [32] *Rapid7 phpinfo() Information Leakage* [Online]. Available: <https://www.rapid7.com/db/vulnerabilities/http-php-phpinfo-leak>
- [33] *Django – Official Documentation* [Online]. Available: <https://docs.djangoproject.com/en/dev/ref/settings/#debug>

- 
- [34] *Administrator access to a Django Administration Panel on \*.sc-corp.net via bruteforced credentials* [Online]. Available: <https://hackerone.com/reports/128114>
- [35] *Remote Code Execution on a Facebook server* [Online]. Available: <https://blog.scr.t.ch/2018/08/24/remote-code-execution-on-a-facebook-server/>
- [36] *Wordpress directories/files visible to internet on Ubiquiti Networks* [Online]. Available: <https://hackerone.com/reports/201984>
- [37] *How To: Server-Side Request Forgery (ssrf)* [Online]. Available: <https://www.hackerone.com/blog-How-To-Server-Side-Request-Forgery-SSRF>
- [38] *Storing and Retrieving Instance Metadata* [Online]. Available: <https://cloud.google.com/compute/docs/storing-retrieving-metadata>
- [39] Github OWASP/Top10 *What are the stats on SSRF and others?* [Online]. Available: <https://github.com/OWASP/Top10/issues/28>
- [40] *SSRF in Exchange leads to ROOT access in all instances* [Online]. Available: <https://hackerone.com/reports/341876>
- [41] Bakare K. Ayeni , Junaidu B. Sahalu, and Kolawole R. Adeyanju *Detecting Cross-Site Scripting in Web Applications Using Fuzzy Inference System*, 2018
- [42] Isatou Hydara, Abu Bakar Md.Sultan, Hazura Zulzalil, Novia Admodisastro *Current state of research on cross-site scripting (XSS) – A systematic literature review*, 2014
- [43] Net-security.org *Cross-Site Scripting worms and viruses : The impending threat and the best defense*, 2011
- [44] *Inception* [Online]. Available: <https://github.com/proabiral/inception>
- [45] *Snallygaster* [Online]. Available: <https://github.com/hannob/snallygaster>

## BIBLIOGRAPHY

---

- [46] *Low-hanging* [Online]. Available: <https://github.com/janmasarik/low-hanging>
- [47] *A deep dive into AWS S3 access controls – taking full control over your assets* [Online]. Available: <https://labs.detectify.com/2017/07/13/a-deep-dive-into-aws-s3-access-controls-taking-full-control-over-your->
- [48] *HackerOne The Hacker Report 2019* [Online]. Available: <https://www.hackerone.com/sites/default/files/2019-03/the-2019-hacker-report.pdf>
- [49] *Top Ten Bug Bounty Payouts of 2018* [Online]. Available: <https://www.htbridge.com/blog/top-ten-bug-bounty-payouts-of-2018.html>
- [50] *Google Vulnerability Reward Program (VRP) Rules* [Online]. Available: <https://www.google.com/about/appsecurity/reward-program/>
- [51] *Facebook Whitehat Program* [Online]. Available: <https://www.facebook.com/whitehat>
- [52] *Kubernetes Overview of kubectl* [Online]. Available: <https://kubernetes.io/docs/reference/kubectl/overview/>
- [53] *Motherboard Why Did Slack Win Out Over IRC, Anyway?* [Online]. Available: [https://motherboard.vice.com/en\\_us/article/xw5wvj/why-did-slack-win-out-over-irc-anyway](https://motherboard.vice.com/en_us/article/xw5wvj/why-did-slack-win-out-over-irc-anyway)
- [54] *Bug Bounty Forum* [Online]. Available: <https://bugbountyforum.com/>
- [55] *Bug Bounty World* [Online]. Available: <https://bugbountyworld.com/>
- [56] *Slack Slash Commands* [Online]. Available: [https://api.slack.com/slash-commands#responding\\_response\\_url](https://api.slack.com/slash-commands#responding_response_url)
- [57] *Slack Incoming Webhooks* [Online]. Available: <https://api.slack.com/incoming-webhooks>

## BIBLIOGRAPHY

---

- [58] *Argo Gateway Guide* [Online]. Available: <https://github.com/argoproj/argo-events/blob/f2cf46f6157ae37912fb1ed897771cef914644c3/docs/gateway-guide.md>
- [59] *What is Docker?* [Online]. Available: <https://opensource.com/resources/what-docker>
- [60] *Containers and Cloud: From LXC to Docker to Kubernetes*, 2014
- [61] Theo Combe, Antony Martin. Roberto Di Pietro *To Docker or not to Docker: a security perspective*, 2016
- [62] *Kubernetes Production-Grade Container Orchestration* [Online]. Available: <https://kubernetes.io/>
- [63] *Kubernetes Documentation Building Large Clusters* [Online]. Available: <https://kubernetes.io/docs/setup/cluster-large/>
- [64] *What Is Container Orchestration?* [Online]. Available: <https://blog.newrelic.com/engineering/container-orchestration-explained/>
- [65] *Google Cloud Platform Free Tier* [Online]. Available: <https://cloud.google.com/free/>
- [66] *Cloud Identity-Aware Proxy* [Online]. Available: <https://cloud.google.com/iap/>
- [67] *Exploring the Top 15 Most Common Vulnerabilities with HackerOne and GitHub* [Online]. Available: <https://register.gotowebinar.com/register/6499105876772027139>
- [68] *Argoproj – Get stuff done with Kubernetes* [Online]. Available: <http://argoproj.io>
- [69] *Github Kubeflow/Pipelines* [Online]. Available: <https://github.com/kubeflow/pipelines#acknowledgments>
- [70] *Argoproj – Official Readme* [Online]. Available: <https://argoproj.github.io/docs/argo/readme.html>

## BIBLIOGRAPHY

---

- [71] GithubArgoproj – DAG-based (dependency) model of workflow execution [Online]. Available: <https://github.com/argoproj/argo/issues/625>
- [72] Apache Airflow Documentation [Online]. Available: <https://airflow.apache.org/>
- [73] Brigade – Event-driven scripting for Kubernetes [Online]. Available: <https://brigade.sh/>
- [74] Github Argo Events [Online]. Available: <https://github.com/argoproj/argo-events>
- [75] GitLab SaaS pricing [Online]. Available: <https://about.gitlab.com/pricing/>
- [76] The poor man's bug bounty monitoring setup [Online]. Available: <https://edoverflow.com/2018/the-poor-mans-monitoring-setup/>
- [77] A Method for Web Security Policies draft-foudil-securitytxt-05 [Online]. Available: <https://tools.ietf.org/html/draft-foudil-securitytxt-05>
- [78] OWASP AppSecCali Pose a Threat: How Perceptual Analysis Helps Bug Hunters, 2019 [Online]. Available: <https://www.youtube.com/watch?v=Gwv29depW9o>
- [79] Github Edoverflow: Bug Bounty Platforms [Online]. Available: <https://github.com/EdOverflow/bugbounty-cheatsheet/blob/master/cheatsheets/bugbountyplatforms.md>
- [80] ISO/IEC 29147:2018 Information technology – Security techniques – Vulnerability disclosure [Online]. Available: <https://www.iso.org/standard/72311.html>
- [81] Hacker101 [Online]. Available: <https://www.hacker101.com/>
- [82] HackerOne Connects Hackers With Companies, and Hopes for a Win-Win [Online]. Available: <https://www.nytimes.com/2015/06/08/technology/>

## BIBLIOGRAPHY

---

- hackerone-connects-hackers-with-companies-and-hopes-for-a-win-win.html?\_r=0
- [83] *Bugcrowd University* [Online]. Available: <https://www.bugcrowd.com/university/>
- [84] *Bugcrowd Program Types* [Online]. Available: <https://www.bugcrowd.com/product/program-types/>
- [85] *How We Measure Crowd Performance* [Online]. Available: <https://www.bugcrowd.com/how-we-measure-crowd-performance/>
- [86] Peter Yaworski, *Web Hacking 101* [Online]. Available: <https://leanpub.com/web-hacking-101>
- [87] *The Time Has Come to Hack the Planet* [Online]. Available: <https://threatpost.com/the-time-has-come-to-hack-the-planet/117419/>
- [88] *Hacktrophy* [Online]. Available: <https://hacktrophy.com>
- [89] *In January, the EU starts running Bug Bounties on Free and Open Source Software* [Online]. Available: <https://juliareda.eu/2018/12/eu-fossa-bug-bounties/>
- [90] *Synack* [Online]. Available: <https://www.synack.com/>
- [91] *Which of the OWASP Top 10 Caused the World's Biggest Data Breaches?* [Online]. Available: <https://snyk.io/blog/owasp-top-10-breaches/>
- [92] Microsoft *Azure Manage anonymous read access to containers and blobs* [Online]. Available: <https://docs.microsoft.com/en-us/azure/storage/blobs/storage-manage-access-to-resources>
- [93] *DigitalOcean Spaces API* [Online]. Available: <https://developers.digitalocean.com/documentation/spaces/#acls>

## BIBLIOGRAPHY

---

- [94] *Introducing BountyMachine* [Online]. Available: <https://medium.com/@bountymachine/introducing-bountymachine-234cad93b5d2>
- [95] Github: *Intrigue-core* [Online]. Available: <https://github.com/intrigueio/intrigue-core>
- [96] Github: *capt-meelo/LazyRecon* [Online]. Available: <https://github.com/capt-meelo/LazyRecon>
- [97] *Metasploit Official Page* [Online]. Available: <https://www.metasploit.com/>
- [98] Github: *datasploit* [Online]. Available: <https://github.com/DataSploit/datasploit>
- [99] Eric Steven Raymond, *How To Become A Hacker* [Online]. Available: <http://www.catb.org/esr/faqs/hacker-howto.html>
- [100] *End of Life clock for Python2* [Online]. Available: <https://pythonclock.org/>
- [101] HackerOne Official Documentation *Program Overview* [Online]. Available: <https://docs.hackerone.com/programs/overview.html>
- [102] HackerOne: *Hacktivity* [Online]. Available: <https://hackerone.com/hacktivity>
- [103] HackerOne Official Documentation *Reputation* [Online]. Available: <https://docs.hackerone.com/hackers/reputation.html>
- [104] *OWASP Testing Guide v4* [Online]. Available: [https://www.owasp.org/index.php/OWASP\\_Testing\\_Guide\\_v4\\_Table\\_of\\_Contents](https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents)
- [105] *Authentication in Kibana* [Online]. Available: <https://www.elastic.co/guide/en/kibana/current/kibana-authentication.html>



## BIBLIOGRAPHY

- [106] *Assetnote: Getting access to Zendesk's Google Cloud and Artifactory from GitHub dotfile repos* [Online]. Available: <https://blog.assetnote.io/bug-bounty/2019/04/23/getting-access-zendesk-gcp/>
- [107] *"CI Knew There Would Be Bugs Here" — Exploring Continuous Integration Services as a Bug Bounty Hunter* [Online]. Available: <https://edoverflow.com/2019/ci-knew-there-would-be-bugs-here/>
- [108] *HackerOne: Vulnerability disclosure policy basics: 5 critical components* [Online]. Available: <https://www.hackerone.com/blog/Vulnerability-Disclosure-Policy-Basics-5-Critical-Components>
- [109] *PCI Data Security Standard (PCI DSS) Penetration Testing Guidance* [Online]. Available: [https://www.pcisecuritystandards.org/documents/Penetration\\_Testing\\_Guidance\\_March\\_2015.pdf](https://www.pcisecuritystandards.org/documents/Penetration_Testing_Guidance_March_2015.pdf)
- [110] *GitHub Bucketsperm* [Online]. Available: <https://github.com/janmasarik/bucketsperm>
- [111] *Azure Storage security guide* [Online]. Available: <https://docs.microsoft.com/en-us/azure/storage/common/storage-security-guide>
- [112] *Using shared access signatures (SAS)* [Online]. Available: <https://docs.microsoft.com/en-us/azure/storage/common/storage-dotnet-shared-access-signature-part-1#controlling-a-sas-with-a-stored-access-policy>
- [113] *GitHub: S3Scanner* [Online]. Available: <https://github.com/sa7mon/S3Scanner#contributors>
- [114] *GitHub Altdns* [Online]. Available: <https://github.com/infosec-au/altdns/pull/21>
- [115] *Basics of the Unix Philosophy* [Online]. Available: [https://homepage.cs.uri.edu/~thenry/resources/unix\\_art/ch01s06.html](https://homepage.cs.uri.edu/~thenry/resources/unix_art/ch01s06.html)

- 
- [116] Official Docker documentation *docker stats* [Online]. Available: <https://docs.docker.com/engine/reference/commandline/stats/>
- [117] HackerOne documentation *Invitations Priority Queue* [Online]. Available: <https://docs.hackerone.com/hackers/invitations-priority-queue.html>
- [118] GitLab documentation *Project services* [Online]. Available: [https://docs.gitlab.com/ee/user/project/integrations/project\\_services.html](https://docs.gitlab.com/ee/user/project/integrations/project_services.html)
- [119] GitLab documentation *Custom server-side Git hooks* [Online]. Available: [https://docs.gitlab.com/ee/administration/custom\\_hooks.html](https://docs.gitlab.com/ee/administration/custom_hooks.html)
- [120] GitHub *EdOverflow/can-i-take-over-xyz* [Online]. Available: <https://github.com/EdOverflow/can-i-take-over-xyz>
- [121] GitHub *anshumanbh/tko-sub* [Online]. Available: <https://github.com/anshumanbh/tko-sub>
- [122] *haccer/subjack* [Online]. Available: <https://github.com/haccer/subjack>
- [123] GitHub *janmasarik/generate-bucketnames* [Online]. Available: <https://github.com/janmasarik/generate-bucketnames>
- [124] Medium *The History of Bug Bounty Programs* [Online]. Available: <https://blog.cobalt.io/the-history-of-bug-bounty-programs-50def4dcaab3>
- [125] Ovidiu Dan, Vaibhav Parikh, Brian D. Davison *IP Geolocation through Reverse DNS*, CoRR, 2018. Available: <http://arxiv.org/abs/1811.04288>
- [126] D. Liu, S. Hao, and H. Wang, *All Your DNS Records Point to Us: Understanding the Security Threats of Dangling DNS Records*, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, pp. 1414–1425.

- 
- [127] HUDÁK, Patrik. *Analysis of DNS in cybersecurity* [online]. Brno, 2018. Available: <https://is.muni.cz/th/byrzn/>. Master thesis. Masaryk University, Faculty of Informatics. Advisor Vít Bukač.
- [128] *Catch Me If You Can – Shubham Shah & Michael Gianarakis at 44CON 2018* [Online]. Available: <https://www.youtube.com/watch?v=C85Z0Jgufuw>
- [129] *Patrick Engebretson, The Basics of Hacking and Penetration Testing, Second Edition: Ethical Hacking and Penetration Testing Made Easy* ISBN-13: 978-0124116443, 2013.
- [130] *AppSecUSA, Zane Lackey – Practical tips for web application security in the age of agile and DevOps, 2016* [Online]. Available: <https://www.youtube.com/watch?v=Hmu21p9ybWs>
- [131] *Top cloud providers 2019* [Online]. Available: <https://www.zdnet.com/article/top-cloud-providers-2019-aws-microsoft-azure-google-cloud-ibm-makes-hy/>
- [132] *S3 Leaks* [Online]. Available: <https://github.com/nagwww/s3-leaks>
- [133] *HackerOne AWS S3 bucket writeable for any authenticated AWS users* [Online]. Available: <https://hackerone.com/reports/128088>
- [134] *New Amazon S3 Encryption & Security Features* [Online]. Available: <https://aws.amazon.com/blogs/aws/new-amazon-s3-encryption-security-features/>
- [135] *AWS rolls out new security feature to prevent accidental S3 data leaks* [Online]. Available: <https://www.zdnet.com/article/aws-rolls-out-new-security-feature-to-prevent-accidental-s3-data-leaks/>
- [136] *Cloud Platform (GCP), Bucket Enumeration and Privilege Escalation* [Online]. Available: <https://rhinosecuritylabs.com/gcp/google-cloud-platform-gcp-bucket-enumeration/>

# Appendices



## A Archive structure

```
bugtab:
├─ bugtab.py
├─ config.yaml
├─ scope.yaml
└─ bugshop
  └─ README.md
    └─ config
      └─ amass.ini
      └─ config.json
      └─ gitleaks.toml
      └─ resolvers.txt
    └─ gateways
      └─ webhook-event-source.yml
      └─ webhook-http.yml
    └─ manifests
      └─ argo-events-cluster-roles.yaml
      └─ argo-events-sa.yaml
      └─ gateway-controller-configmap.yaml
      └─ gateway-controller-deployment.yaml
      └─ gateway-crd.yaml
      └─ iap-backend-config.yaml
      └─ ingress.yaml
      └─ install.yml
      └─ sensor-controller-configmap.yaml
      └─ sensor-controller-deployment.yaml
      └─ sensor-crd.yaml
    └─ sensors
      └─ webhook-http.yml
    └─ wordlists
      └─ bucket-directories.txt
      └─ fingerprints.json
      └─ raft-large-directories-lowercase.txt
      └─ resolvers.txt
      └─ words.txt
    └─ workflows
      └─ azure-brute-perf.yml
      └─ buckets.yml
      └─ gitsecrets.yml
      └─ subdomains.yml
      └─ subtakeover.yml
```

Figure A.1: Structure of the attached archive containing repositories of Bugshop and Bugtab

## B Vulnerability disclosure email

**From:** Ján Masarik [mailto:433634@mail.muni.cz]

**Sent:** [REDACTED] May [REDACTED]

**Subject:** [Security issue] Misconfigured Azure Blob Storage

Hello,

there is a potentially serious security issue that needs your immediate action. Please forward this email to your information security team or the IT department.

Summary:

One of potentially yours Azure Blob Storage containers with potentially highly sensitive data is misconfigured and allows full public access to all stored data for an unauthenticated attacker.

You can verify the public accessibility in the Azure console, or by visiting this URL:

[REDACTED]?restype=container&comp=list

This container seems to belong to [http://\[REDACTED\]](#) and seems to contain quite sensitive data, so please let me know if you would not be the right contact as soon as possible.

Details:

The vulnerability is called object storage misconfiguration and is most commonly (but not exclusively) affecting AWS S3 buckets.

This vulnerability was a root cause of severe data breaches of many companies, such as Viacom, Accenture or National Credit Federation. You can read more about it here:

- <https://blog.detectify.com/2017/07/13/aws-s3-misconfiguration-explained-fix/>
- [https://www.theregister.co.uk/2017/09/19/viacom\\_exposure\\_in\\_aws3\\_bucket\\_blunder/](https://www.theregister.co.uk/2017/09/19/viacom_exposure_in_aws3_bucket_blunder/)
- <https://www.zdnet.com/article/security-lapse-exposes-198-million-united-states-voter-records/>
- <https://www.infosecurity-magazine.com/news/accenture-leaked-data-another-aws/>

Mitigation:

Log in to Azure portal and change "public access level" of the affected container to "Private" or to "Blob" in the special case where you need to keep public accessibility of objects. Right now, your setting is "Container" that allows to list and download all objects within the whole container.

Please review public access of all Azure Blob Storage containers belonging to you, as it is plausible that your account has more misconfigured buckets containing sensitive data.

Background:

I am graduating cyber-security student from Masaryk University, Czech Republic. As part of my master's thesis, I've concluded quantitative research on misconfigured Azure Blob Storage buckets in the wild. During this research, I've encountered some misconfigured buckets, which, if not fixed, could potentially lead to a data breach.

Please let me know if you would have any questions.

Kind regards,  
Jan Masarik

Figure B.1: Notification email sent to the owners of the misconfigured Azure Blob Storage buckets holding sensitive data.

## C DigitalOcean Spaces enumeration workflow

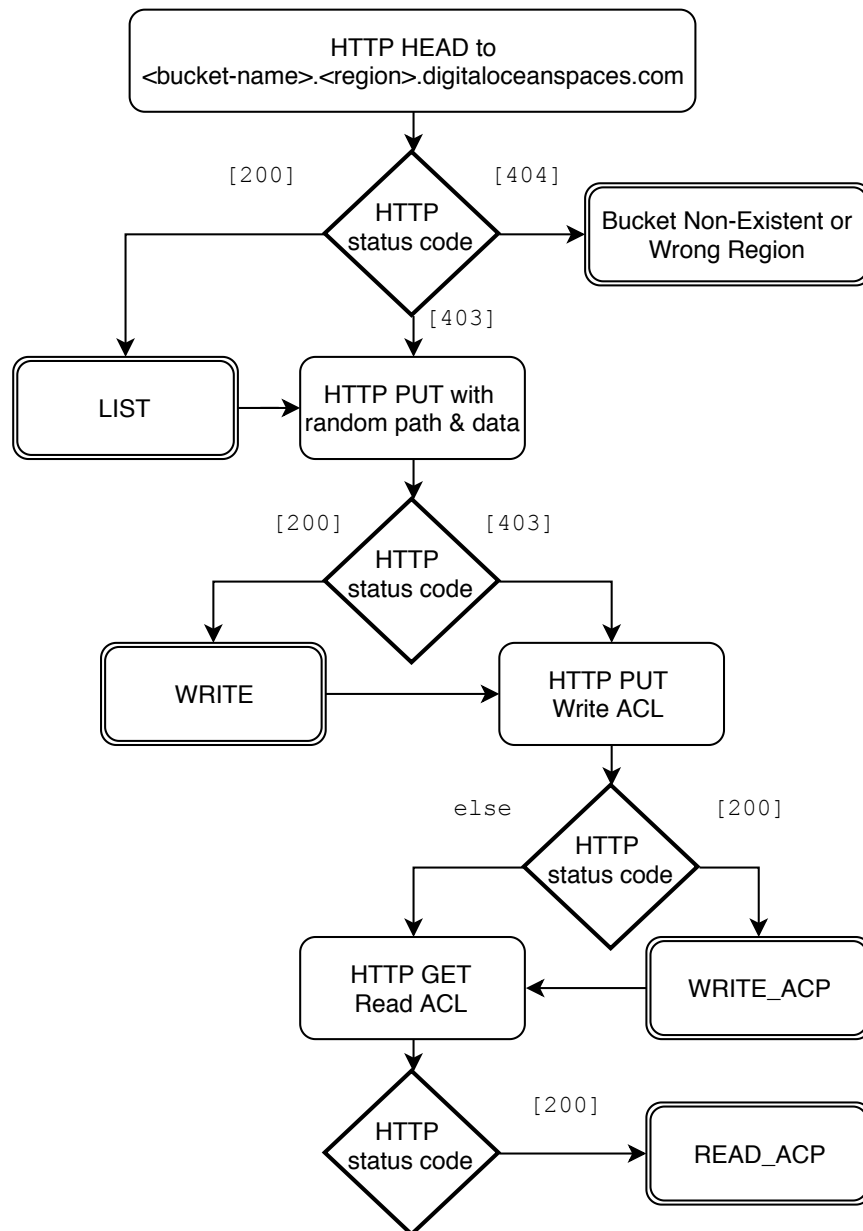


Figure C.1: Enumeration of the DigitalOcean Spaces.