

# Yet another S3 leak? Why there are still so many victims?

More and more companies decide to store their files in the Amazon cloud, using S3 service. Unlimited storage space and a relatively cheap price makes it very attractive. Why then are there so many sensitive data leaks from S3 and how to secure your data not to become another victim? In this post, I'll try to answer this question.



Pawel Rzepa [Follow](#)

Aug 17, 2018 · 5 min read

## Yet another S3 leak

Since the era of a browser of files in publicly available S3 buckets, data leaks are even more common than before. With this online browser, you even

don't have to install the AWS CLI to go through available files. That makes the whole situation even scarier, because a kid with no tools and no knowledge can look into your public bucket.

As an example, let's take the newest leak from a Polish company InJobs.pl. The article is written in Polish, so to cut a long story short: they run a job portal publishing offers in many European countries. But using improper access control for their S3 service... they also published 769 CVs containing private data of their applicants.


Files in [injobs.s3.eu-central-1.amazonaws.com](#)

1 - 20 of 769 results

#	Bucket	Filename	Size
1	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1003/CV-[REDACTED]-51a6.pdf</a>	379175
2	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1004/CV-[REDACTED]-bea8.pdf</a>	144073
3	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1005/CV-[REDACTED]-f21e.pdf</a>	126032
4	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1014/CV-[REDACTED]-075b.pdf</a>	281025
5	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1018/CV-[REDACTED]-ab99.docx</a>	34521
6	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1018/CV-[REDACTED]-ab99.pdf</a>	29441
7	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1019/CV-[REDACTED]-211a.pdf</a>	43738
8	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1020/CV-[REDACTED]-2e7b.docx</a>	67887
9	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1020/CV-[REDACTED]-3e7b.pdf</a>	77250
10	<a href="#">injobs.s3.eu-central-1.amazonaws.com</a>	<a href="#">uploads/resume/file/1021/CV-[REDACTED]-c654.pdf</a>	147916

The "Grayhatwarfare" browser unveils multiple CVs stored by InJobs.pl.

Here's one of them:



**Curriculum Vitae**

**Tel.** +48 [redacted]

**Email:** [redacted]@gmail.com

**WYKSZTAŁCENIE**

Zespół Szkół Technicznych [redacted]  
kierunek-ślusarz narzędziowy

**KURSY**

Spawacz gazowy i elektryczny  
Kierowca kat B (przewożący osoby)  
Uprawnienia na wózki widłowe

The disclosed CV reveals private data like the applicant's phone number, e-mail and his career resume.

Now, let's think about the consequences. The leak itself doesn't seem to be a dramatic one (at least when compared to leaks of access keys, passwords, etc.). However, in the GDPR era, such a leak can put the company into big trouble. What is more the PR of InJobs will be much worse, what may mean significantly less customers = much less money. Are people ready to trust such a company again, even if it puts a lot of money and effort into securing client data? Maybe. But obviously this story is a lesson for all other companies — secure your data before anyone can find it publicly available on the Internet.

## Why are there still so many buckets offering public access?

Access control issues are the case with AWS S3 service from its very beginning, which was about 2006. According to my latest research up to **21% of buckets allow public access to stored data.**

First of all, it should be noted that badly configured S3 access control is always the administrator's fault. The default settings of this service give private access only to its creator. To assign public permissions either to read

stored data or list bucket's content, a specific option has to be enabled, which is explicitly emphasized by the vendor with alerts in the AWS console interface as well as via e-mail. Actually, granting public read access to all of the bucket's contents is justified only if our bucket is hosting a static WWW page. In every other case we should avoid this kind of access.

On the other hand, assigning public read permissions to all of the objects in the bucket is the fastest and the simplest way of providing access to data for other components, e.g. applications using S3 for storing user uploads or other AWS services. This is often so tempting for administrators that they completely forget about the dramatic consequences, which is a possibility of confidential data leak. It is important to be aware that configuring access control according to the principle of least privilege in a complex architecture might be problematic, as in S3 you can do it in at least 4 ways, which include:

1. Identity and Access Management policies,
2. Bucket policies,

3. Access Control List,

4. Pre-signed URLs and query string authentication.

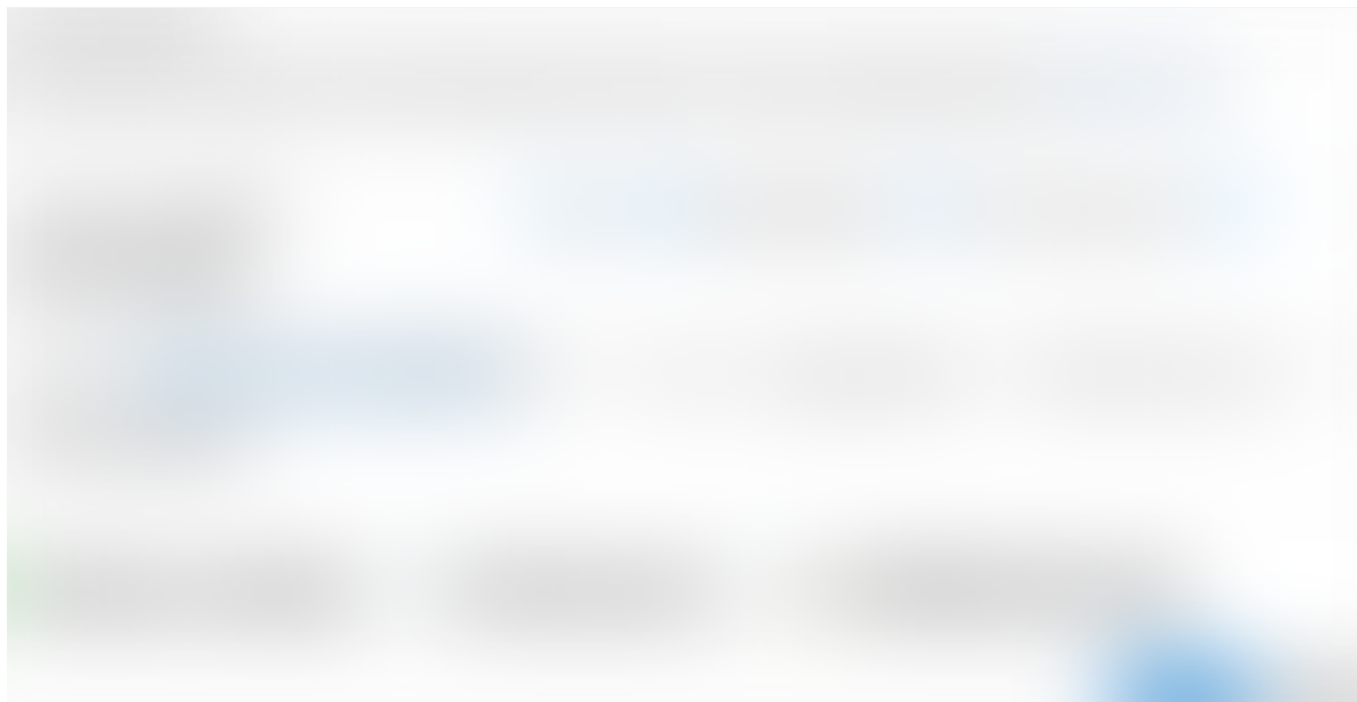
These policies can be mutually exclusive, meaning that an incorrectly managed service can easily lead to mistakes.



Bucket policies and ACLs can be mutually exclusive.

Moreover, if one would like to blame Amazon for the leaks, the unnecessarily created group “Any authenticated AWS user” should be taken into consideration. In years 2006–2017 S3 administrators could grant access to the bucket to users of the group “Any authenticated AWS user” via

an AWS console. The name turned out to be misleading, as many administrators assumed that this group only includes authenticated users that are assigned to their root account. However, this option gave access permissions to every AWS account. In the new AWS console interface (introduced in mid-2017) the possibility of granting access to that group is no longer present, but we can still do it via AWS CLI or external libraries that cooperate with AWS like Python's boto. Indeed, according to Murphy's Law anything that can go wrong will go wrong .



In years 2006–2017 S3 administrators could grant access to the bucket to users of the group “Any authenticated AWS user” via AWS console.

## What should I do to avoid becoming another victim of a data leak?

While configuring an S3 service, first ask yourself what kind of data you want to store, keeping in mind the principle that one bucket should serve only one purpose. For example if you have a bucket with application users’ files, you shouldn’t upload your HR files there. Next, you should specify who will have access to these files and define access policies using an adequate method (IAM policies, bucket policies, Access Control Lists, pre-signed URLs, or query string authentication). It is also worth considering to encrypt the data stored in the bucket. Amazon offers several ways of doing it:

- Encryption with a key provided by the administrator (SSE-C),
- Encryption with keys managed by Amazon (SSE-S3),



- Encryption with keys managed by the administrator via AWS Key Management Service (AWS-KMS).

In addition, your cloud environment should be regularly audited. If you want to quickly and easily check whether your buckets have any access control issues, you can use one of our tools — the [BucketScanner](#).

*Looking for more? Check out our [Seven-Step Guide to Securing your AWS Kingdom](#).*

*Subscribe to our [newsletter](#) and get the updates directly to your inbox.*

AWS

SaaS

Application Security

Cloud Security

Cybersecurity



2 claps



WRITTEN BY

**Pawel Rzepa**

Interested in pentesting and cloud security | OSCP | eMAPT |  
AWS SAA

Follow



**SecuRing**

We help to achieve appropriate level of application security

Follow

Write the first response

## More From Medium

Related reads

### The \$12,000 Intersection between Clickjacking, XSS, and Denial of Service



ashish mathur

Jul 21 · 6 min read ★



175



Related reads

## XS-Searching Google's bug tracker to find out vulnerable source code

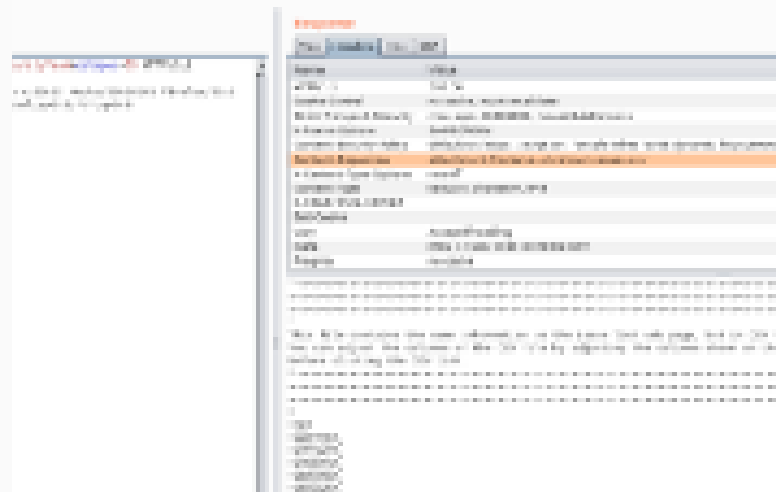


Luan Herrera

Nov 19, 2018 · 6 min read



1K



Related reads

## Security testing for REST API with w3af



Artem Smotrakov in Quick Code

Dec 19, 2018 · 7 min read



38

