Oops, I Leaked It Again — How Mitiga Found PII in Exposed Amazon RDS Snapshots

By Tim Boutin

Published: 2026-03-05 · Archived: 2026-04-05 21:24:13 UTC

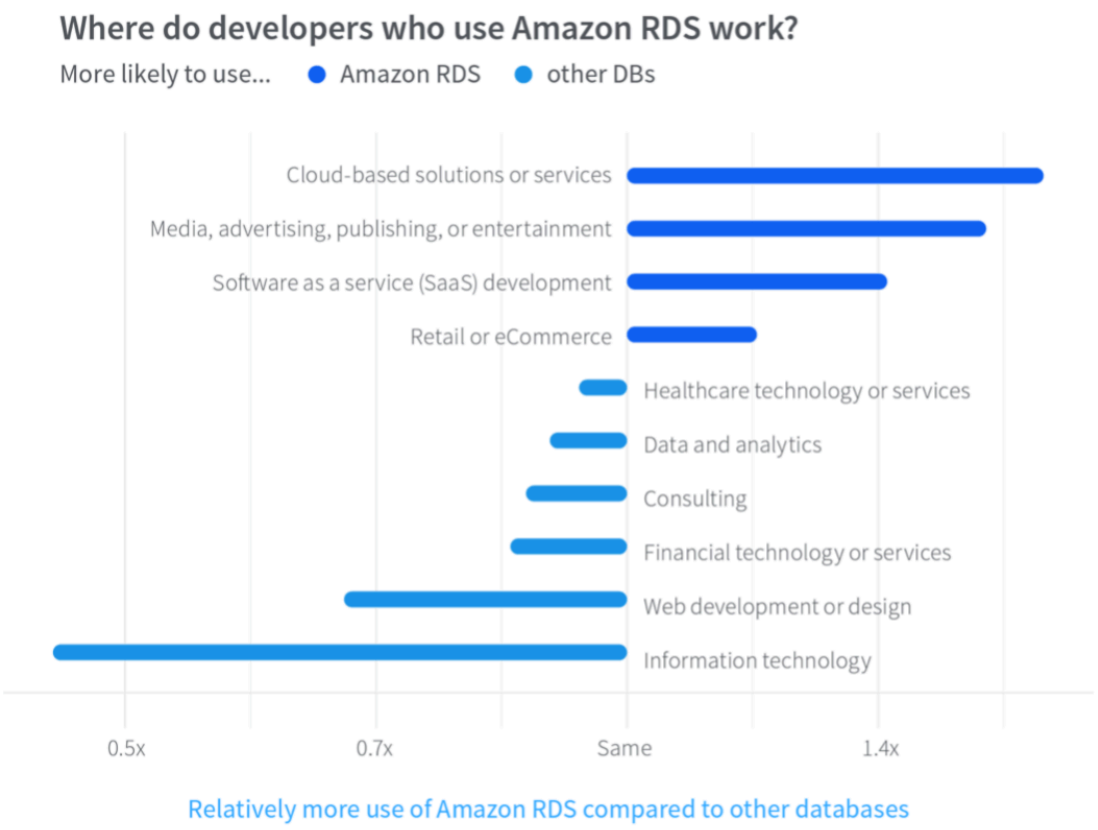
TL; DR:

The Mitiga Research Team recently discovered hundreds of databases being exposed monthly, with extensive Personally Identifiable Information (PII) leakage. Leaking PII in this manner provides a potential treasure trove for threat actors — either during the reconnaissance phase of the cyber kill chain or extortionware/ ransomware campaigns

This Mitiga Research Team blog goes into the tactics used, insights determined, and recommendations on how organizations relying on Amazon RDS can take precautions against this scenario inadvertently happening in their environments.

Background

The Amazon Relational Database Service (Amazon RDS) is a Platform-as-a-Service (PaaS) that provides a database platform based on a few optional engines (e.g., MySQL, PostgreSQL, etc.). Released in October 2009, Amazon RDS is very popular in today's cloud world. In 2018, Stackoverflow's annual survey included a published a report with the non-ambiguous name *The Incredible Growth of Amazon RDS*, which contained some eye-opening figures about RDS distribution. For example, the following report graph shows the popularity of the service (in comparison to other managed databases) across several industry sectors.



When using RDS service in AWS, you can employ RDS snapshots — a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. These snapshots can be shared across different AWS accounts — in or out of the on-premises organization, as well as AWS accounts that make the RDS snapshots publicly available.

With that, one might unintentionally leak sensitive data to the world, even if you use highly secure network configuration.

What is a Public RDS Snapshot, and Why Is it Interesting?

An RDS snapshot is an intuitive feature that helps you to back up your database. A Public RDS snapshot is a useful feature that allows a user to share public data or a template database to an application. Additionally, a Public RDS snapshot is a valuable feature when a user wants to share a snapshot with colleagues, while not having to deal with roles and policies. In this manner, the user can share the snapshot publicly for just a few minutes... What could possibly happen? 🙂

Well… obviously, leaked snapshots might potentially be very valuable asset for a threat actor — either during the reconnaissance phase of the cyber kill chain (databases can include sensitive technical data that can be used for exploitation, like API keys) or for extortion or ransomware campaigns.

Due to the potential of unintentional data leakage of the RDS service, we decided to investigate if this potential really has a foothold in the real-world.

Surprisingly, or not, **we found a lot of snapshots that were shared publicly for few hours, days, and even weeks — either intentionally or by mistake.**

It's important to note that making a snapshot public, even for a very short amount of time, can have unwanted outcomes. Our research shows how a threat actor might take advantage of snapshots that are shared for even a short timeframe.

In this blog post, we describe our research process and associated findings. We will also explain the forensics visibility in the relevant logs and share best-practices to detect and mitigate the potential risk related to public RDS snapshots.

While investigating the relevant log records, we found that the information in the logs was not as intuitive as we would have wanted — both in terms of the visibility and of the events we were expecting to see. We elaborate on this topic in the *Detection and Mitigation* section.

How Attackers Can Abuse It

Attackers are always looking for new ways to put their hands on confidential information of organizations, mostly for financial gain. Some cloud services that allow sharing of cloud resources widely to the world expose a new threat to organizations – unintentional sharing of information through resources like Disk snapshots (EBS), or in our case DB snapshots (RDS).

In adopting a cyber attacker's mindset, the Mitiga Research Team discovered there was a significant problem, and we developed a way to massively exploit that issue – just as attackers would.

As described in this blog, this lack of visibility prevents organizations from knowing whether a shared publicly RDS snapshot was ever accessed by an unauthorized third party and if actions should be taken immediately to mitigate any risk that could be derived from the information being public.

In our research, we developed an AWS-native technique, using AWS Lambda Step Function and boto3, to scan, clone, and extract potentially sensitive information from RDS snapshots in scale, as follows:

- **Scan** — Hourly scan for DB snapshots that have been marked as public from all regions (except the regions not enabled by default: *af-south-1, ap-east-1, ap-southeast-3, eu-south-1, eu-central-2, me-south-1 and me-central-1.)*
- **Clone** — Clone the snapshot to our own AWS account and maintain a state file to make sure we are not cloning duplicates.
- **List** — Create a list of cloned RDS snapshot to extract, making sure we are not creating a DB instance from a snapshot that was examined in the past.
- **Prepare** — Get a list of snapshot ARNs and create a DB instance out of it. To do so we reset the master password so we can create and access the restored DB instance.
- **Extract** — Automatically extract DB schema (particularly the table names and column names) as well as the table's content (limit 10,000 lines per table) to S3 for further analysis.
- **Cleanup** — Delete the DB instances to cut charges.

What We Discovered: Main Insights

We decided to focus on a specific timeframe for our research. We chose one month as a timeframe we think is enough to clean the noise of optional peaks or desert days. Our data is from 21/09/2022 to 20/10/2022.

OMG, Is That Personally Identifiable Information!?

One of our main questions in this research was how much sensitive data was exposed during this month, if at all. Before we show the statistics and insights, we want to share some (anonymized) examples of personally identifiable information (PII) we found.

The first example is a **MySQL database that was exposed all the research month.** This DB was created on 03/03/22, and the snapshot was taken on 31/08/22. This DB looks like a car rental agency database. One of the interesting tables appears to summarize car rental transactions. This table contains 51 columns and approximately 10,000 rows. The data in this table is updated — there is a created date time with dates from 15/06/22 to 31/08/22.

Here are some examples of the most interesting columns:

- Personal Identifiable Information
  - first_name, last_name, phone, email, marital_status (married/single)
  - occasion — such as birthday, special day, or anniversary
  - sales_consultant — full names of company employees, quite likely

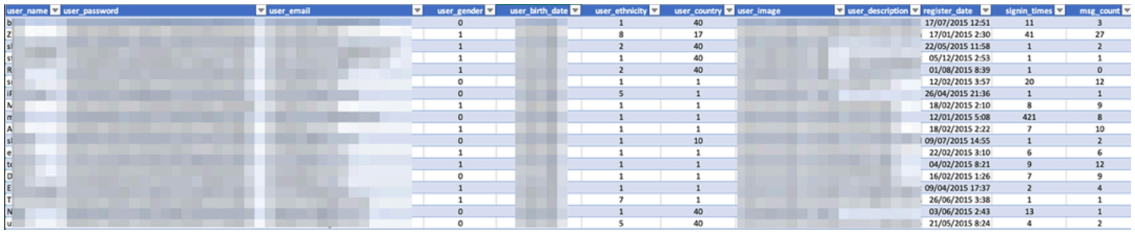
- Business knowledge
  - enquiry_segment — the options we see for this column are: Personal, Company, Commercial, Captive, Fleet Owner, Hire, Individual, Institutional.
  - organization_id — there is another table with the correlation between organizations' IDs and names.
  - customer category type — B2B/B2C

- Rental information
  - model — model of the car
  - expected_delivery_date

sales_consultant	createddatetime	first_name	last_name	phone	email	marital_status	occasion
D	21/07/2022 7:18	A	B		@gmail.com	Single	marriage anniversary
D	23/07/2022 6:54	G	G		@gmail.com	Single	special day
A	23/07/2022 7:35	K	T		@gmail.com	Married	businesses
C	23/07/2022 9:03	S	C		@gmsil.com	Single	birthday
G	23/07/2022 9:44	K	S		@gmail.com	Single	birthday
S	25/07/2022 11:37	B	R		@gmail.com	Married	No occasion just delivery planning
S	26/07/2022 13:19	A	B		@gmail.com	Single	birthday
S	27/07/2022 11:21	M	B		@gmail.com	Single	marriage
S	27/07/2022 13:27	A	K		@gmail.com	Single	sister birthday
N	29/07/2022 4:06	S	V		@gmail.com	Married	no occation
S	29/07/2022 8:03	B	B		@gmail.com	Married	Baby Birthday
S	29/07/2022 11:08	M	A		@gmail.com	Single	birthday
C	29/07/2022 13:18	K	P		@gmail.com	Married	Nothing
S	30/07/2022 10:34	K	S		@gmail.com	Single	gift
K	30/07/2022 10:55	T	G		@gmail.com	Married	festival
S	30/07/2022 11:41	K	C		@gmail.com	Single	birthday
S	03/08/2022 12:34	J	T		@gmail.com	Married	Good day

The second example is a **MySQL database that was exposed for less than four hours**. This DB was created on 14/04/16, and the snapshot was taken more than 6 years later — on 02/10/22. This database appears related to a dating application. It seems that this app is not in use anymore, but it worked until at least 2020. The users table contains approximately 2,200 users. This table contains emails, password hashes, birthdates, links to personal

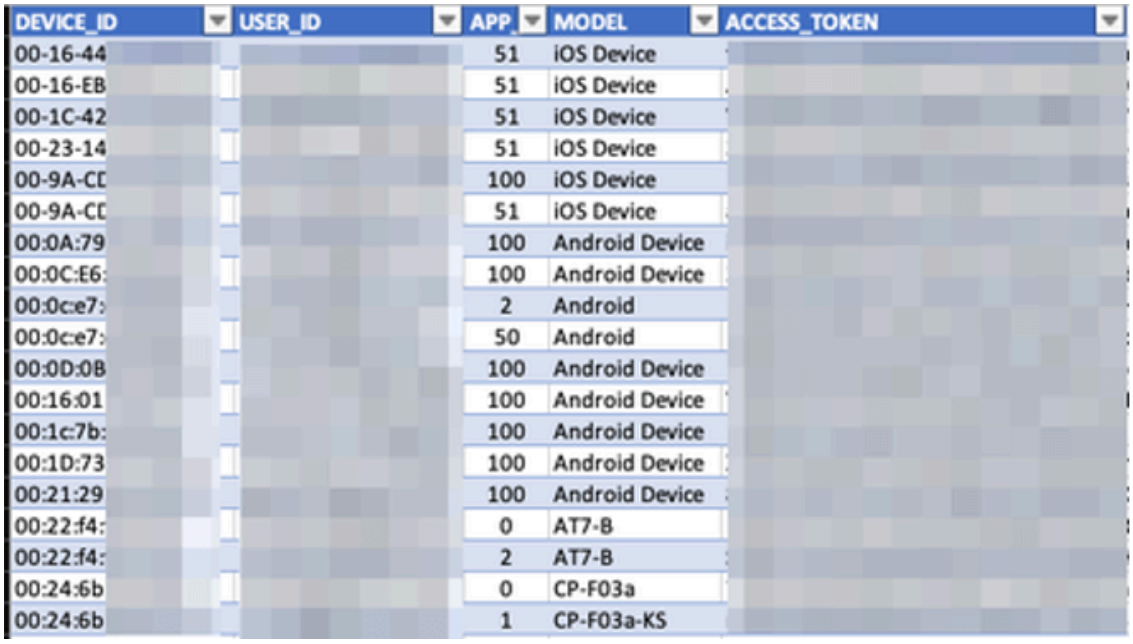
images links, and a lot more. This table contains data between 08/01/15 to 15/07/20. Another table in this DB contains the private messages.



user_name	user_password	user_email	user_gender	user_birth_date	user_ethnicity	user_country	user_image	user_description	register_date	signin_times	msg_count
D			0		1	40			17/07/2015 12:51	11	3
Z			1		8	17			17/01/2015 2:30	41	27
E			1		2	40			22/05/2015 11:58	1	2
st			1		1	40			05/12/2015 2:53	1	1
R			1		2	40			01/08/2015 8:39	1	0
st			0		1	1			12/02/2015 3:57	20	12
ir			0		5	1			26/04/2015 21:36	1	1
N			1		1	1			18/02/2015 2:10	8	9
m			0		1	1			12/01/2015 5:08	421	8
A			1		1	1			18/02/2015 2:22	7	10
sl			0		1	10			09/07/2015 14:55	1	2
e			1		1	1			22/02/2015 3:10	6	6
tx			1		1	1			04/02/2015 8:21	9	12
D			0		1	1			16/02/2015 1:26	7	9
E			1		1	1			09/04/2015 17:37	2	4
T			1		7	1			26/06/2015 3:38	1	1
N			0		1	40			03/06/2015 2:43	13	1
U			0		5	40			21/05/2015 8:24	4	2

The third example is a **MySQL database that was exposed for an entire month**. This DB was created on 22/07/15, and the snapshot was taken more than 7 years later (!) - on 12/09/22. This DB looks like a telephone applications company database. One of the tables summarizes all the logins to the company applications. This table has 17 columns and approximately 7,500 rows. The data in this table is from 10/08/15 to 12/09/22.

In this table, for example, we have user ID (that equal to email addresses), phone device models, mac addresses, client access tokens and application ID of the app this user uses.



DEVICE_ID	USER_ID	APP_	MODEL	ACCESS_TOKEN
00-16-44		51	iOS Device	
00-16-EB		51	iOS Device	
00-1C-42		51	iOS Device	
00-23-14		51	iOS Device	
00-9A-CD		100	iOS Device	
00-9A-CD		51	iOS Device	
00:0A:79		100	Android Device	
00:0C:E6:		100	Android Device	
00:0c:e7:		2	Android	
00:0c:e7:		50	Android	
00:0D:0B		100	Android Device	
00:16:01		100	Android Device	
00:1c:7b:		100	Android Device	
00:1D:73		100	Android Device	
00:21:29		100	Android Device	
00:22:f4:		0	AT7-B	
00:22:f4:		2	AT7-B	
00:24:6b		0	CP-F03a	
00:24:6b		1	CP-F03a-KS	

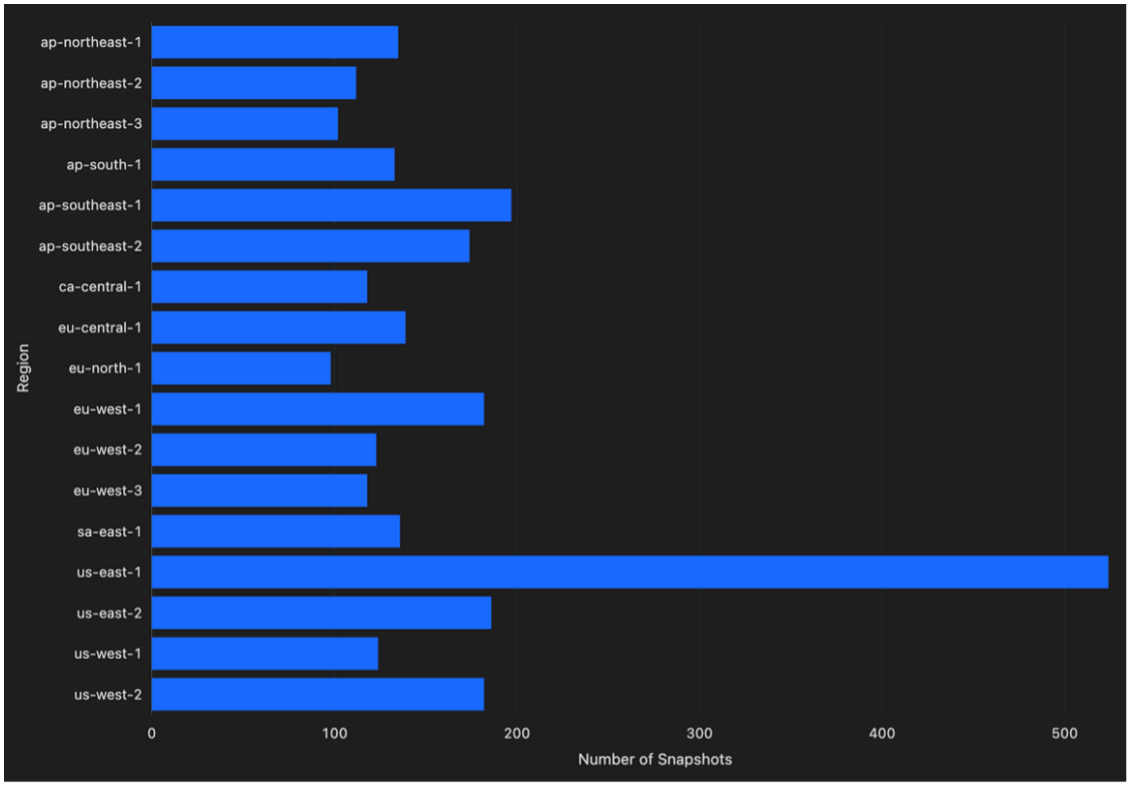
Statistics

We were convinced this was a really interesting situation, but the important follow-up question here was: *"How prevalent is this issue?"*.

Looking for answers, we collected some interesting statistics:

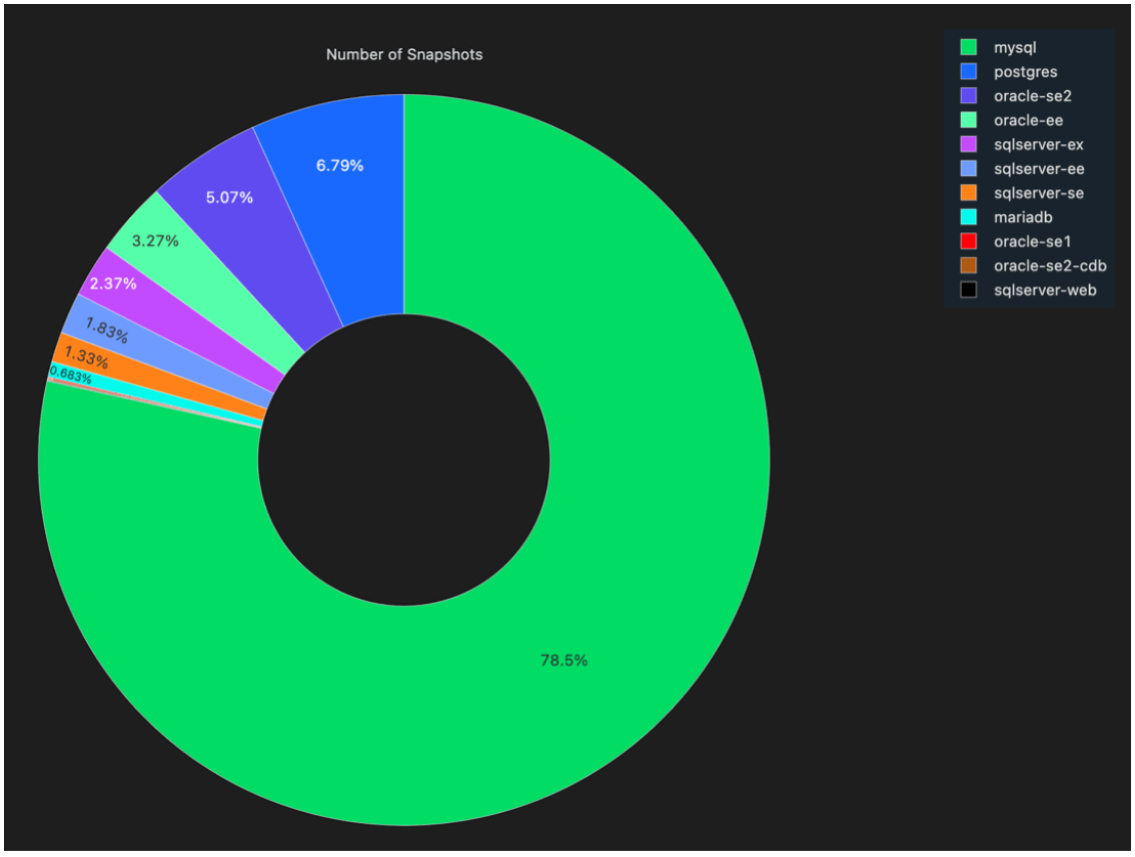
- The total number of snapshots seen in this month was 2,783
- Of those 2,783 snapshots, 810 snapshots were exposed during the analyzed timeframe
- Additionally, 1,859 snapshots of the 2,783 were exposed for just 1-to-2 days

Here are two graphs that illustrate the prevalence of this issue: Public Snapshots Per Region and Public Snapshots Per Database Engine.



Public Snapshots Per Region

In terms of these regional snapshots, it makes sense that in us-east-1 there are the most of them. But it’s interesting to see snapshots created across most regions — **so, as expected, this is a worldwide phenomenon.**



OK, so we saw the amount and the regional distribution of the snapshots. But it’s not a big deal if all of them are supposed to be public, right?

To pass this barrier of noise and skepticism, we decided to err on the side of caution and sanitize the data in two ways:

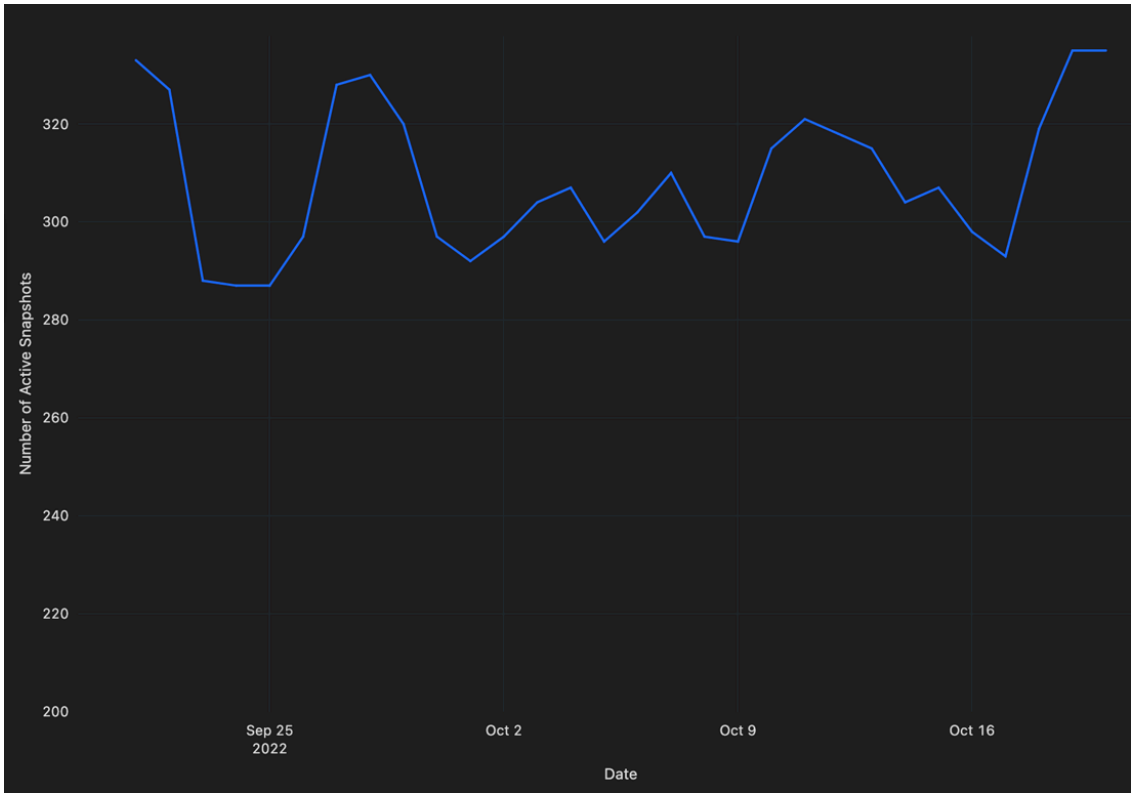
- We saw some account IDs that published a lot of snapshots, changing daily or remaining consistent through the month, probably as part of their product. According to that, we checked how many snapshots there were per account and filtered out the big ones to clean the few accounts that published a lot.
  * We filtered out any account that had more than five monthly snapshots. There were 2,059 snapshots of 36 accounts. That is a lot of snapshots created by just a few accounts.

- After that, we filtered out snapshots that contained in their name one of these keywords: “test”, “tst”, “public”. We assumed stringently that if snapshot name contained one of those keywords, it may not contain interesting data.
  * We found 74 snapshots with these keywords after the previous filter — not much.

Now, we have 650 snapshots that were published by accounts that published a few more public snapshots (if at all), and without a keyword in their name that hints on the possibility it may be just a test and doesn't contain interesting information. We call the remaining snapshots — the interesting snapshots. Let's take a deep dive into the interesting snapshots.

Insights Based on Metadata Only

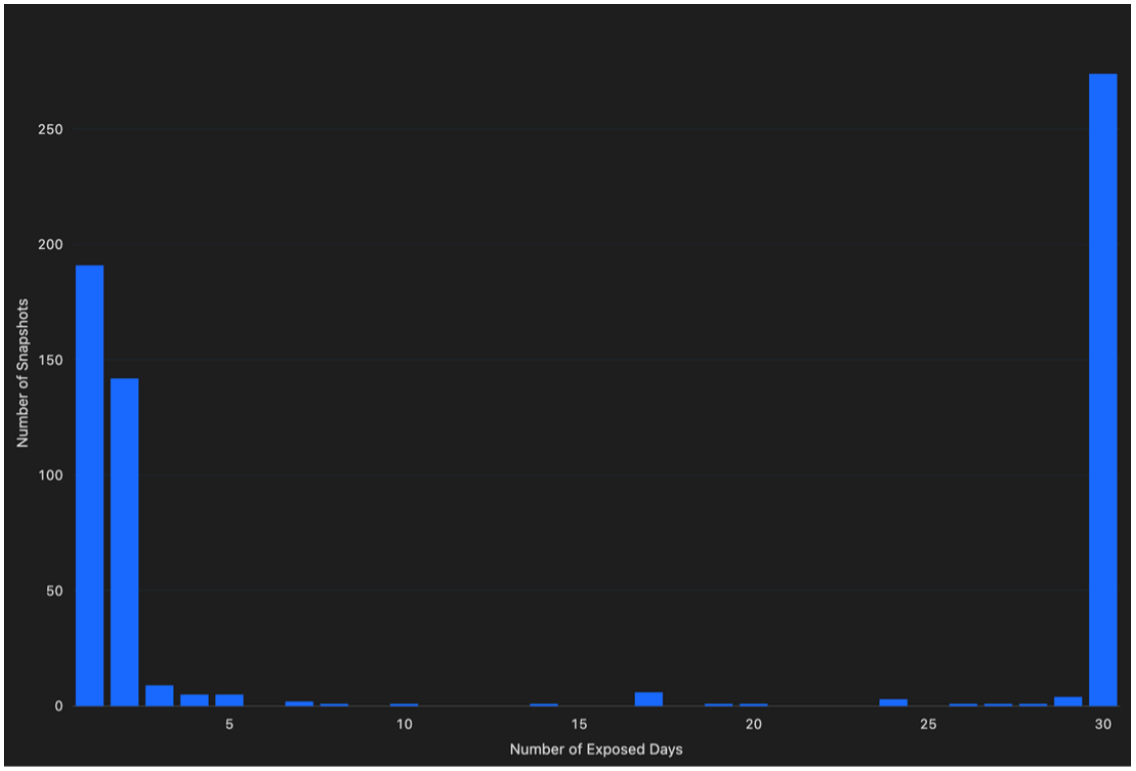
This graph shows how many different interesting snapshots were active every day in our month:



Active Snapshots by Day

We can see a change, of course, but the range is 287-335 (i.e., it’s not a unique peak but a stable phenomenon).

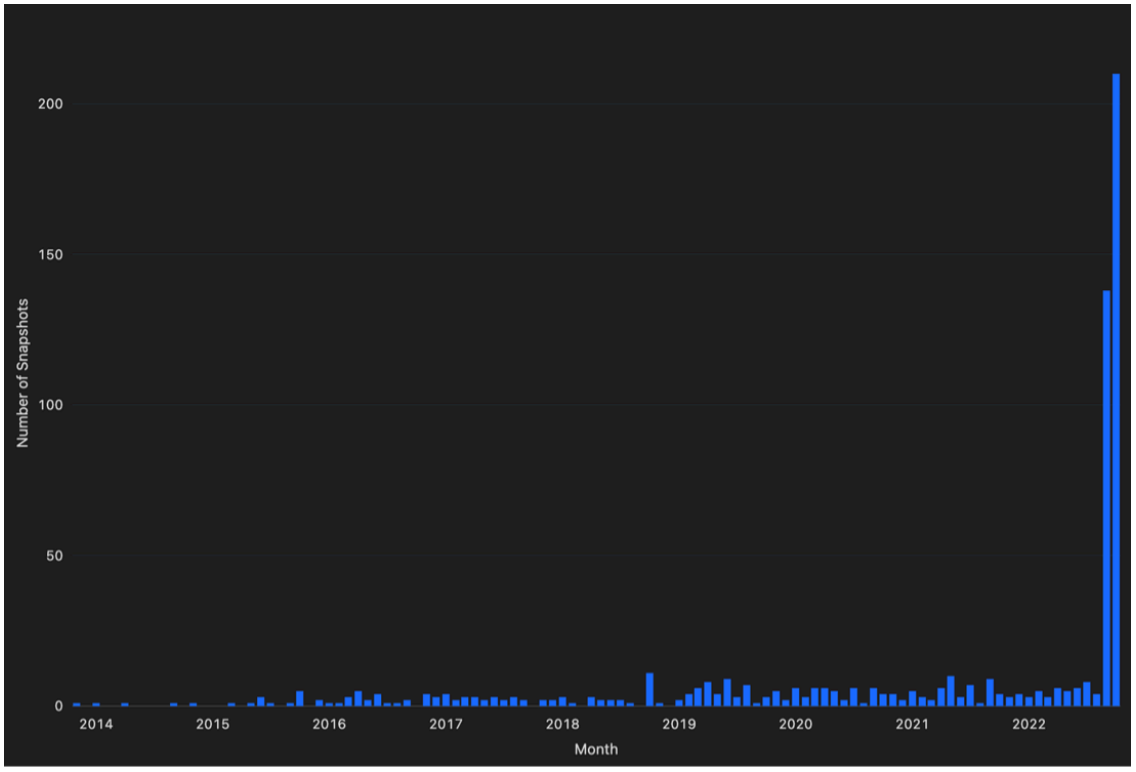
This graph shows the distribution of how many days in the analyzed timeframe the interesting snapshots were exposed:



Exposed Snapshots by Day

It's a great graph — we can see that most of the snapshots were exposed just for a few hours or were forgotten / supposed to be public. Snapshots left exposed for just a few hours are most likely exposed unintentionally, and may contain data that should not be available, even for a short time, to the public.

Every snapshot was taken from an RDS instance. This graph shows the instances creation time per month:



Instances Creation Time Per Month

Most of them appeared last September and October, of course — 348 snapshots. But it’s exciting to see how many snapshots based on RDS instances were basically launched before that timeframe! It means there are many public snapshots that aren't based on a new database for a check, test, template, or just empty database, but they are based on exist-for-a-long-time database.

Insights Based on the Content

Another approach to explore the cloned DB instances is by analyzing the content of the tables. To do so we have built an automated process to highlight tables that contain, in high probability, confidential PII or other interesting pieces of information.

We decided to focus on MySQL snapshots as the most common engine. Throughout the analyzed timeframe, we saw 469 MySQL snapshots from the 650 special snapshots, those who may include real data (i.e., not test or intently public). To spot DB tables that had more potential of containing the juicy stuff, we extracted all columns names from the tables and searched for a list of indicative keys:

> 'address', 'admin', 'password', 'card', 'credit', 'secret', 'fax', 'ip', 'mac', 'account', 'hash', 'pass', 'token', 'phone', 'billing', 'contact', 'document', 'tax', 'passport'.

After filtering for the keys, we found 5,766 columns that are more likely to have interesting information by their name. We then filtered for tables that contained content by the number of rows and table size, which left 3,491 columns to examine. That comprised 171 MySQL DB instances that potentially contained sensitive information which should not be exposed publicly.

Below you can find the breakdown of matching keys:

Match	Count
ip	1405
address	231
tax	226
pass	218
account	191
phone	184
password	170
contact	165
token	150
admin	113
hash	76
mac	65
billing	61
card	57
credit	54
document	46
fax	41
secret	32
passport	6

My Snapshot Doesn’t Contain Any PII — Why Should I Worry?

That is a good question. Let's assume you are very responsible, and there is no way you go and expose database with PII publicly, not even for one second. Is there anything else you need to worry about when you expose your snapshot to anyone?

The answer is YES!

Today, there is no native way to correlate between account ID and the company that owns this account. It's not an official secret, but AWS doesn't publish an API or a big table with this correlation, and not without a reason — threat actors in many situations will pay a lot for this. There are attack vectors that a successful correlation means a successful attack. Let's take an example case of exposed RDS snapshot with PII. Unfortunately for the attacker, there is no way to understand from the content of the DB which company this data belongs to. If the attacker had a way to find out the company based on account ID, they could commit their precious blackmail and get a lot of money.

Back to reality: You are a professional employee, so your public snapshot doesn't contain any PII and a way to attribute the content to your company. But you have to ask yourself — *"Is there a way to find which company this account ID belongs to, based on metadata only?"*.

Actually, in our research, we succeeded in understanding who owned account IDs for many snapshots. The simple way is to look at the snapshot name and see the company name in it. Thank you, nice man who created this snapshot and chose the name. But there is a more cool and fun way. Every snapshot metadata contains a field called *MasterUsername* — the main DB username. In most cases, this field contains something boring, such as: *admin, root, master, prod, db_user, myrds*, and so on.

But, in many cases, we saw interesting things in this field — company names fully spelled, in acronyms, or shortcuts. But the crowning glory — **names of people**. We searched for these people on LinkedIn and ascertained where they work. A little creepy, but useful and widely employed by threat actors.

Detection and Mitigation

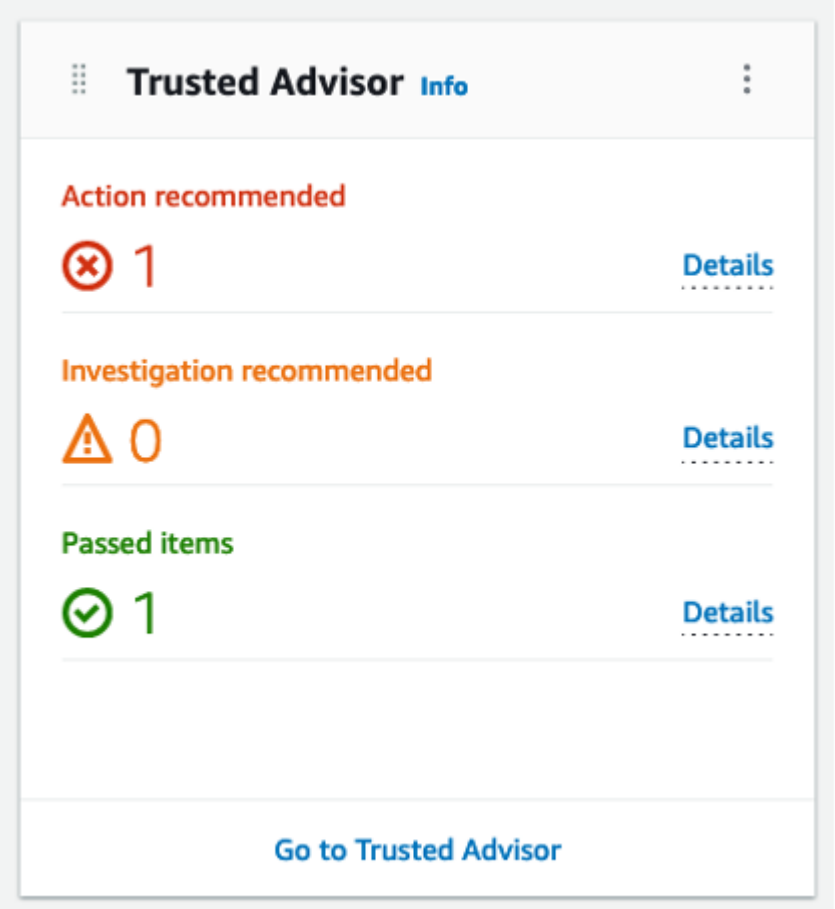
It is important to note that after you share a snapshot publicly, you should get an email from AWS that warns you about a public snapshot in your account — Amazon sends this notification to ensure that the public snapshot in your account was indeed intended to be publicly available.
In our lab environment, we received the following email notification 23 minutes after sharing the snapshot. The email subject was:

> *"RDS Identified Public Snapshots In AWS Account ************"*

Great feature from Amazon. Nevertheless, the email might be overlooked. Even if it doesn't, it might be already too late, because the leak might have been exploited by someone.

In addition, there is a tool called AWS Trusted Advisor that advises recommended steps to improve the environment in aspects of costs, performance, and security. In the dashboard, there is a "Trusted Advisor" widget with an overview of recommendations state. When you have public snapshots, this widget presents an "Action recommended" that warns you about your public snapshots with exact snapshots (in the details). Same feedback

here: it is a nice feature, but there is no guarantee you will notice it, especially in a short time. For more information, please see additional context provided in the fifth hyperlink in the references section of this blog.



AWS Trusted Advisor Tool

How Do I Know If I Have Any Public Snapshots?

To check for public snapshots in your environment, we suggest taking the following steps:

First Step: Historical Check Using CloudTrail Logs — Did I Create a Public Snapshot or Share an Existing Snapshot?

There is no way to create a public snapshot — you should create a snapshot and then share it. It makes sense — this separation helps to avoid configuration mistakes.

When you share a snapshot publicly, there is an event with the event name:

ModifyDBSnapshotAttribute.

The *requestParameters* field looks like this:

{

```
    "dBSnapshotIdentifier": "our-super-duper-sensitive-snapshot",
    "attributeName": "restore",
    "valuesToAdd": [
      "all"
    ],
    "valuesToRemove": []
  }
```

(If you configure sharing to an account, you will see the account IDs you add / remove *under valuesToAdd / valuesToRemove*, respectively.)

It's nice to see in AWS *ModifyDBSnapshotAttribute* API call documentation the following recommendation:

> ***Do not add the all value for any manual DB snapshots that contain private information that you don't want available to all Amazon Web Services accounts.***

Second step: Current-State Check of Your Organizational RDS Snapshots

Ad-hoc Check

1. Use describe-db-snapshots API call to get a list of all available RDS DB snapshots:
   aws rds describe-db-snapshots

   --region us-east-1
   --snapshot-type manual
   --query 'DBSnapshots[*].DBSnapshotIdentifier'

2. Use **describe-db-snapshot-attributes** against the list of *DBsnapshotIdentifier* returned from the last call, to get the DB snapshot attribute *DBSnapshotAttributes*
   aws rds describe-db-snapshot-attributes

   --region us-east-1
   --db-snapshot-identifier <>
   --query 'DBSnapshotAttributesResult.DBSnapshotAttributes'

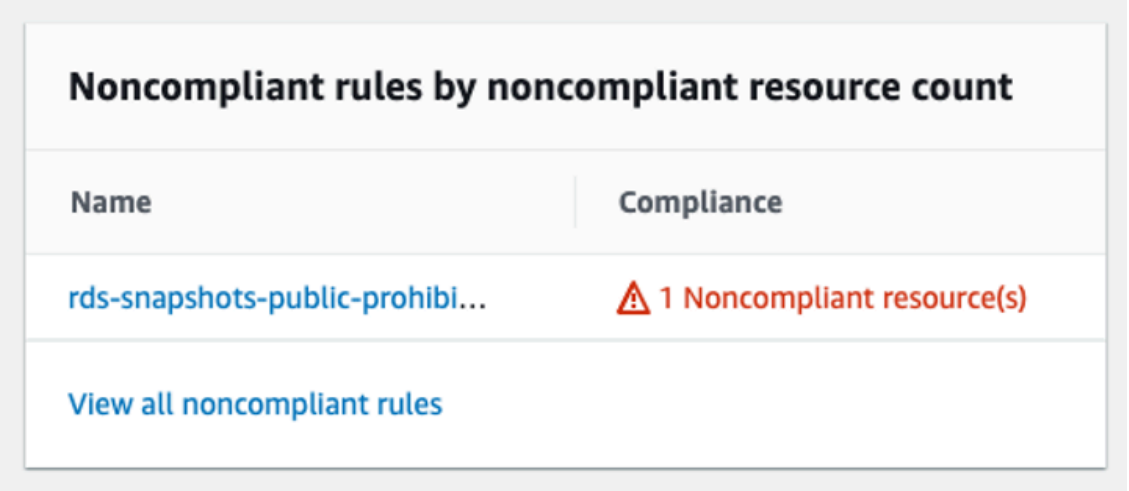
   This call would return the DBSnapshotAttributes for the given DBSnapshotIdentifier, if the AttributeValues field is set to all, the selected RDS snapshot is publicly available for any AWS account to restore.

Continuous Check — *AWS Config*

You can use AWS Config to get visibility of your environment. This tool allows you to manage rules and get your environment state based on these rules. In our case, we recommend adding *rds-snapshots-public-prohibited* rule

which checks if there are RDS snapshots that are public. The rule is *NON_COMPLIANT* if any snapshot is public. For more information, please see the *AWS Config Developer Guide* extract hyperlink provided in the References section of this blog.

In this example we have one public snapshot in our environment, so we see this in AWS Config dashboard:



AWS Config Dashboard

How Can I See if Someone Copied My Public Snapshot?

Unfortunately, you can't.

We were surprised to find out there is no log event on copying public snapshot to another account or restoring a DB instance from another account, in the snapshot's owner account.

In the foreign account, we did saw ***CopyDBSnapshot*** and ***RestoreDBInstanceFromDBSnapshot***, respectively.

In EC2 service, unlike RDS, there are events indicating the copy and restore operation on an EC2 snapshot:

> *SharedSnapshotCopyInitiated — "A shared snapshot is being copied" in AWS documentation words, and SharedSnapshotVolumeCreated — "A shared snapshot is being used to create a volume".*

To complete this gap and create a greater visibility into the operation taken on RDS snapshots, we approached AWS. According to AWS, logging on RDS copy and restore operation is currently unavailable, and a feature request was created to support that.

How Do I Prevent Sharing Snapshots Publicly?

Manage Permissions Well

The basic but the most important recommendation – Don't give unnecessary permissions. This best-practice is known as "least-privilege permissions."

Encrypt Your Snapshot

AWS enables you to encrypt a snapshot with a KMS key. During our research we wanted to investigate the behavior of a public encrypted snapshot with a shared KMS key, and we determined it’s not possible to share publicly an encrypted snapshot.

Summary

In the last decade, it’s obvious that when you publish a new service that is exposed to the internet, your IP and port scanned in a few minutes. These days, it’s naive to think the situations with public RDS snapshots are any different.

We think it’s not an overstatement to assume the worst-case scenario — when you are making a snapshot public for a short time, someone might get that snapshot’s metadata and content. So, for your company and, more importantly, your customers' privacy — don’t do that if you are not 100% percent sure there is no sensitive data in the content or in the metadata of your snapshot.

References

1. https://aws.amazon.com/rds/stack-overflow-incredible-growth-amazon-rds-report/
2. https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html
3. https://docs.aws.amazon.com/config/latest/developerguide/rds-snapshots-public-prohibited.html
4. https://www.techtarget.com/searchaws/tip/Public-snapshots-pose-significant-Amazon-EBS-security-risks
5. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.LoggingAndMonitoring.html
6. https://sysdig.com/blog/aws-rds-security-events-sysdig/
