Cloud DFIR Project

glue_privesc

Mentor	Niko
Date	2024.08.13 (Sat)
Track	Digital forensic
Name	Kim Gyu Jin(김규진)

--

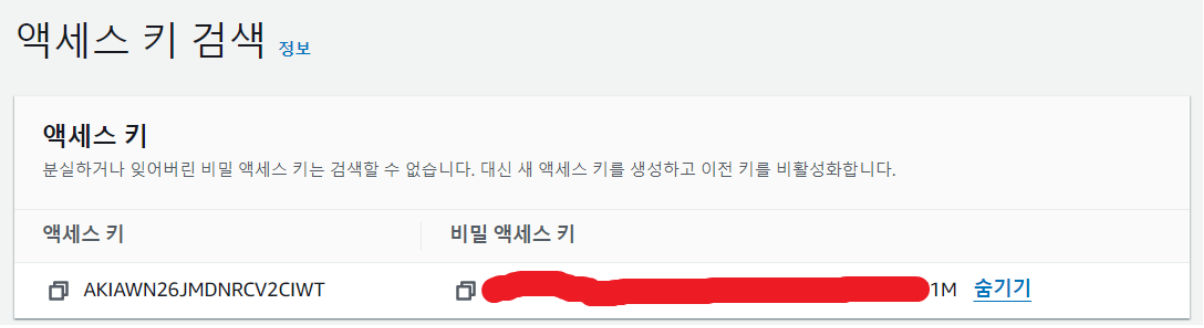
Index

1. Scenario Environment

1.1. AWS Account

After creating a personal AWS account, you need to access the AWS CLI and create the IAM account that will be used for running CloudGoat. In the dashboard, select the "IAM" tab and choose to configure permissions.

Create an IAM account named "BOB" and assign the "AdministratorAccess" permission to it.



Also, generate an access key and store the information separately.

1.2. CloudGoat Setting

```
aws_instance.ec2-vulnerable-proxy-server: Still creating... [10s elapsed]
aws_instance.ec2-vulnerable-proxy-server: Still creating... [20s elapsed]
aws_instance.ec2-vulnerable-proxy-server: Still creating... [30s elapsed]
aws_instance.ec2-vulnerable-proxy-server: Provisioning with 'file'...
aws_instance.ec2-vulnerable-proxy-server: Still creating... [40s elapsed]
aws_instance.ec2-vulnerable-proxy-server: Creation complete after 48s [id=i-
04580fba605c14b8b]

Apply complete! Resources: 18 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = "442042507483"
cloudgoat_output_target_ec2_server_ip = "34.226.211.130"

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.
cloudgoat_output_aws_account_id = 442042507483
cloudgoat_output_target_ec2_server_ip = 34.226.211.130

[cloudgoat] Output file written to:

    /home/user/awstest/cloudgoat/cloud_breach_s3_cgidbdjan9whwt/start.txt

(.venv) user@BOOK-PR1OF313PJ:~/awstest/cloudgoat$
```

You need to create a breach specifically for use with CloudGoat. In the previously set up CloudGoat environment from the last lesson, execute the command `./cloudgoat.py create cloud_breach_s3` to create the branch.

Afterward, I attempted to execute `./cloudgoat.py create glue_privesc` to automatically build the specified scenario. However, an error occurred because the `aws_db_instance` does not support postgreSQL version 13.7.

```
(.venv) user@BOOK-PR1OF313PJ:~/awstest/cloudgoat$ aws rds describe-db-engine
-versions --engine postgres --query "DBEngineVersions[].EngineVersion" --reg
ion us-east-1
[
    "11.22",
    "11.22-rds.20240418",
    "11.22-rds.20240509",
    "12.15",
    "12.16",
    "12.17",
    "12.18",
    "12.19",
    "12.20",
    "13.11",
    "13.12",
    "13.13",
    "13.14",
    "13.15",
    "13.16",
    "14.9",
    "14.10",
    "14.11",
    "14.12",
    "14.13",
    "15.4",
    "15.5",
    "15.6",
    "15.7",
    "15.8",
    "16.1",
    "16.2",
    "16.3",
    "16.4"
]
```

I executed the command `aws rds describe-db-engine-versions --engine postgres --query "DBEngineVersions[].EngineVersion` to check which versions of PostgreSQL are supported in the current region of the AWS account.

```
user@BOOK-PR1OF313PJ: ~/a
 1 resource "aws_db_instance" "cg-rds" {
 2   allocated_storage    = 20
 3   storage_type         = "gp2"
 4   engine               = "postgres"
 5   engine_version       = "13.11"
 6   instance_class       = "db.t3.micro"
 7   db_subnet_group_name = aws_db_subnet_group.cg-rds-subnet-group.id
 8   db_name              = var.rds-database-name
 9   username             = var.rds_username
10   password             = var.rds_password
11   parameter_group_name = "default.postgres13"
12   publicly_accessible  = false
13   skip_final_snapshot  = true
```

I needed to modify the Terraform configuration file for the scenario located at cloudgoat/scenarios/glue_privesc/terraform/rts.tf. I set the engine_version to PostgreSQL 13.11, the supported version I confirmed earlier.

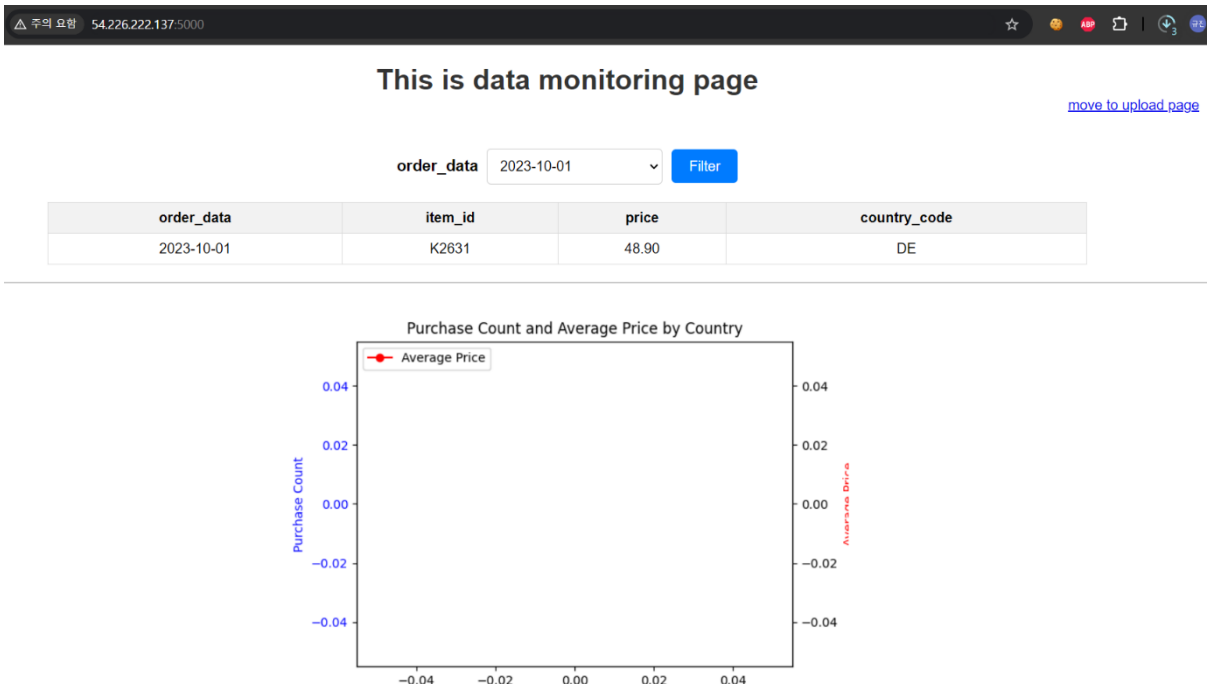
1.3. Build and deploy

```
Apply complete! Resources: 59 added, 0 changed, 0 destroyed.

Outputs:

cg_web_site_ip = "54.226.222.137"
cg_web_site_port = 5000

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.
cg_web_site_ip = 54.226.222.137
cg_web_site_port = 5000

[cloudgoat] Output file written to:

    /home/user/awstest/cloudgoat/glue_privesc_cgidrs3ilxf47z/start.txt

(.venv) user@BOOK-PR1OF313PJ:~/awstest/cloudgoat$
```

After changing the engine version setting, the build was successful, and I was provided with the link: `http://54.226.222.137:5000/`.

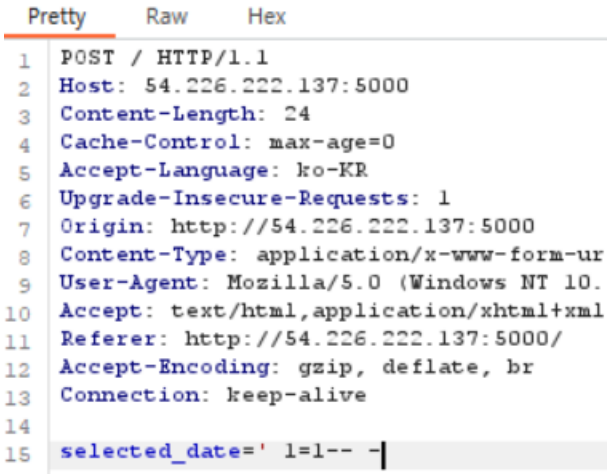
2. Attack Execution

2.1. Site Access



The site accessed through a regular Chrome browser shows an interface where there is a section labeled "order_data" that allows user input to be sent to the server.

2.2. SQL injection



I intercepted the page using Burp Suite and performed an SQL injection by sending the command `selected_data=' 1=1-- -`.

2.3. troubleshotting

However, the account information of the Glue administrator, as described in the scenario, did not appear. Instead, an error page listing Python code was displayed, but I couldn't retrieve the internal account information from the server.

This issue seems to stem from the arbitrary change in the PostgreSQL version during the build process. In the previous version, the SQL injection syntax should have exposed internal server information as described in the scenario. However, due to the update, the syntax does not function as expected.

To resolve the issue, I attempted to find an AWS region that still supports version 13.7, but I couldn't find one. I then tried to downgrade to version 12.11 to proceed with the scenario, but another error occurred during the build process, causing the attempt to fail.

2.4. Future Plan

I will find a solution to this issue before the final report deadline. For example, I could identify an SQL injection syntax that works with version 13.11 to achieve the desired results, or I could find a method to build the scenario using an older version without encountering errors.