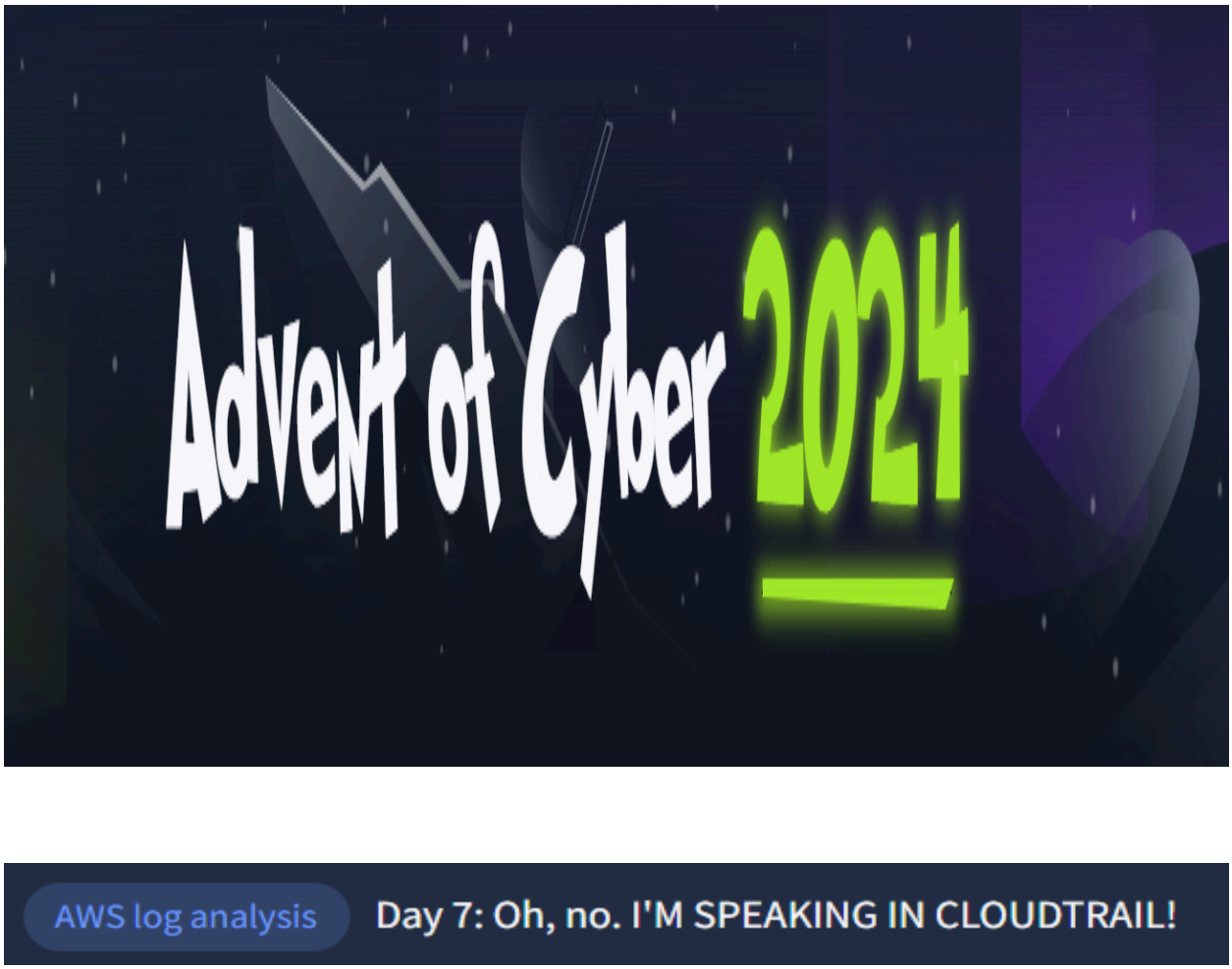


17..12.2024

—

Nikhil Kumar

The Story

A SOC-Mas Mystery: The Curious Case of Care4Wares' Vanishing Donations

As the festive air of SOC-mas filled the streets of Wareville, the Care4Wares team had just one goal in mind—to spread joy through their charity tradition of gifting books to those in need. The warehouse was buzzing, their trolleys piled high with books like *Wares Wally*, *The Princess and the Pcap*, and *Charlie and the 8 Bit Factory*.

But amidst the cheer came a sudden shock.

The Mystery Unfolds

On the second day of their donation campaign, disaster struck. Donations stopped coming. Confused and worried, Care4Wares checked their account only to find an eerie silence—no new funds. A generous donor reached out, showing proof of a transaction. To their shock, the account number on the transaction was wrong!

How could this happen? McSkidy, the Wareville SOC team lead, jumped into action.

> "The digital flyer hasn't changed. The link hasn't been altered," said McSkidy. "But something suspicious is going on here."

The investigation was on.....

Clues from the Cloud: AWS and CloudTrail

McSkidy remembered that Care4Wares' infrastructure ran on AWS—Amazon Web Services. The digital flyer, `wareville-bank-account-qr.png`, had been stored in an **S3 bucket** called `wareville-care4wares`. AWS CloudTrail and CloudWatch would provide the answers.

1. What is AWS CloudWatch?

CloudWatch is a monitoring service that tracks metrics, application logs, and system performance. For a dynamic cloud environment like Care4Wares', logs come from multiple sources. CloudWatch simplifies monitoring by centralizing all logs into:

- **Log Events:** Individual log entries.
- **Log Streams:** Collection of log events from a source.
- **Log Groups:** Logical grouping of log streams (e.g., for the same service across hosts).

CloudWatch enables **filter patterns**, allowing McSkidy to quickly query logs for specific events or anomalies.

2. What is AWS CloudTrail?

While CloudWatch monitors application performance, CloudTrail focuses on **user and service actions** in the AWS environment. Every action—from accessing an S3 bucket to modifying objects—gets captured as a JSON-formatted event.

Key CloudTrail features:

- **Always On:** Enabled by default.
- **Event History:** Logs actions for up to 90 days.
- **Trails:** Customizable trails to monitor specific events and retain data longer.
- **Integration with CloudWatch:** CloudTrail logs can be sent to CloudWatch for centralized monitoring.

McSkidy suspected foul play in the S3 bucket and dove into the CloudTrail logs.

JSON Logs: A Digital Treasure Map

The logs were JSON-formatted—great for machines, tricky for humans. McSkidy needed something to sift through the data quickly.

What is JQ?

JQ is a lightweight *command-line tool for parsing*, filtering, and transforming JSON data. Think of it as `grep`, but for JSON.

For example:

Given the `book_list.json` file:

```
[
  {"book_title": "Wares Wally", "genre": "children",
"page_count": 20},
  {"book_title": "The Princess and the Pcap", "genre":
"children", "page_count": 48}
]
```

You could extract only the book titles using:

```
jq '.[] | .book_title' book_list.json
```

Output:

```
"Wares Wally"
"The Princess and the Pcap"
```

This ability to filter JSON made JQ invaluable for analyzing CloudTrail logs.

Now Our Case Description!!!

The Peculiar Case of Care4Wares' Dry Funds

Now that we have refreshed our knowledge of AWS Cloudtrail and JQ alongside McSkidy, let's investigate this peculiar case of Care4Wares' dry funds.

The responsible ware for the Care4Wares charity drive gave us the following info regarding this incident:

We sent out a link on the 28th of November to everyone in our network that points to a flyer with the details of our charity. The details include the account number to receive donations. We received many donations the first day after sending out the link, but there were none from the second day on. I talked to multiple people who claimed to have donated a respectable sum. One showed his transaction, and I noticed the account number was wrong. I checked the link, and it was still the same. I opened the link, and the digital flyer was the same except for the account number.

McSkidy recalls putting the digital flyer, **wareville-bank-account-qr.png**, in an Amazon AWS S3 bucket named **wareville-care4wares**. Let's assist McSkidy and start by finding out more about that link. Before that, let's first review the information that we currently have to start the investigation:

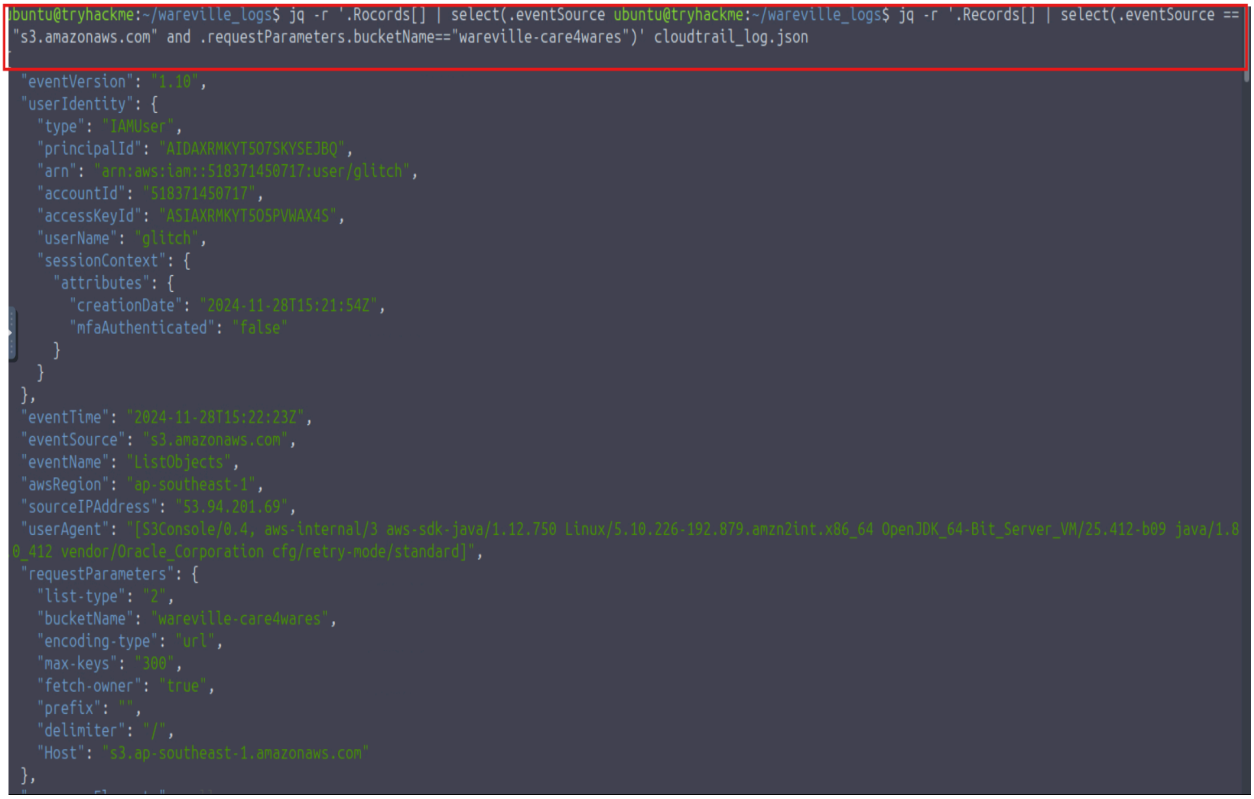
- The day after the link was sent out, several donations were received.
- Since the second day after sending the link, no more donations have been received.
- A donator has shown proof of his transaction. It was made 3 days after he received the link. The account number in the transaction was not correct.
- McSkidy put the digital flyer in the AWS S3 object named **wareville-bank-account-qr.png** under the bucket **wareville-care4wares**.
- The link has not been altered.

Here we go on our machine for analysis !!

- We have the logs let us investigate and see the .json file to gather information about the issue suggested by McSkidy.
- On out machine we look for the investigative file as we press **ls** command all the directories pops out and we can see the file **wareville_logs.**


```
ubuntu@tryhackme:~/wareville_logs$ jq -r '.Records[] |
select(.eventSource == "s3.amazonaws.com" and
.requestParameters.bucketName=="wareville-care4wares")'
cloudtrail_log.json
```

- ***This command lists the raw data from cludtrail_log.json file which includes all the details and parameters in an organised manner.***



More filtering of the commands for better output from json file

```
ubuntu@tryhackme:~/wareville_logs$ jq -r '.Records[] | select(.eventSource == "s3.amazonaws.com" and .requestParameters.bucketName=="wareville-care4ware
s") | [.eventTime, .eventName, .userIdentity.userName // "N/A",.requestParameters.bucketName // "N/A", .requestParameters.key // "N/A", .sourceIPAddress
 // "N/A"]' cloudtrail_log.json
```

- ***The command shown lists the details including the eventname, eventTime, username and the source ip address from the s3 bucketName= wareville-care4wares***

NOTE: ***Note that the string `"N/A"` is included purely for formatting reasons. This means that if the defined key does not have a value, it will display N/A instead.***

- **OUTPUT**


```
[
  "2024-11-28T15:22:23Z",
  "ListObjects",
  "glitch",
  "wareville-care4wares",
  "N/A",
  "53.94.201.69"
]
[
  "2024-11-28T15:22:25Z",
  "ListObjects",
  "glitch",
  "wareville-care4wares",
  "N/A",
  "53.94.201.69"
]
[
  "2024-11-28T15:22:39Z",
  "PutObject",
  "glitch",
  "wareville-care4wares",
  "bank-details/wareville-bank-account-qr.png",
  "53.94.201.69"
]
[
  "2024-11-28T15:22:39Z",
  "PreflightRequest",
  "N/A",
  "wareville-care4wares",
  "bank-details/wareville-bank-account-qr.png",
  "53.94.201.69"
]
```

Further narrowing our search as it appears to us in a better column way we type the following command to make it tabular and efficient to read and understand logs.

```
ubuntu@tryhackme:~/wareville_logs$ jq -r '["Event_Time", "Event_Name", "User_Name", "Bucket_Name", "Key", "Source_IP"],(.Records[] | sel
ect(.eventSource == "s3.amazonaws.com" and .requestParameters.bucketName=="wareville-care4wares") | [.eventTime, .eventName, .userIdenti
ty.userName // "N/A", .requestParameters.bucketName // "N/A", .requestParameters.key // "N/A", .sourceIPAddress // "N/A"]) | @tsv' cloud
trail_log.json | column -t
Event_Time           Event_Name        User_Name  Bucket_Name          Key                                        Source_IP
2024-11-28T15:22:23Z  ListObjects       glitch     wareville-care4wares  N/A                                        53.94.201.69
2024-11-28T15:22:25Z  ListObjects       glitch     wareville-care4wares  N/A                                        53.94.201.69
2024-11-28T15:22:39Z  PutObject         glitch     wareville-care4wares  bank-details/wareville-bank-account-qr.png  53.94.201.69
2024-11-28T15:22:39Z  PreflightRequest  N/A        wareville-care4wares  bank-details/wareville-bank-account-qr.png  53.94.201.69
2024-11-28T15:22:44Z  ListObjects       glitch     wareville-care4wares  N/A                                        53.94.201.69
ubuntu@tryhackme:~/wareville_logs$
```

`| @tsv'`	• ***Sets each array element, the output processed after the filters, as a line of tab-separated values***.
`| column -t -s $'\t'`	• *It takes the output of the **jq** command, now resulting in tab-separated values, and beautifies its result by processing all tabs and aligning the columns.*

—>From this we found a username ***glitch ….. But…..****in the investigation McSkidy* was sure there was no user ***glitch*** in the system before. There is no one in the city hall with that name, either. The only person that McSkidy knows with that name is the hacker who keeps to himself. McSkidy suggests that we look into this anomalous user.

Now let us further go deep in investigation to know the culprit !!

- *Now we need to look for the information of the username **glitch** we found as it should not be a part of this aws account mentioned by the **McSkidy.***
- *So with this we found that this user glitch is performing suspeicious tasks like **ConsoleLogin** and **GetCostAndUsage***
- ***We find Ip-address : 53.94.201.69***

```
ubuntu@tryhackme:~/wareville_logs$ jq -r '["Event_Time", "Event_Source", "Event_Name", "User_Name", "Source_IP"],(.Recor
serIdentity.userName == "glitch") | [.eventTime, .eventSource, .eventName, .userIdentity.userName // "N/A", .sourceIPAdd
| @tsv' cloudtrail_log.json | column -t -s $'    '
Event_Time            Event_Source                        Event_Name                          User_Name  Source_IP
?024-11-28T15:22:12Z  s3.amazonaws.com                    HeadBucket                          glitch     53.94.201.69
•024-11-28T15:22:23Z  s3.amazonaws.com                    ListObjects                         glitch     53.94.201.69
2024-11-28T15:22:25Z  s3.amazonaws.com                    ListObjects                         glitch     53.94.201.69
2024-11-28T15:22:39Z  s3.amazonaws.com                    PutObject                           glitch     53.94.201.69
2024-11-28T15:22:44Z  s3.amazonaws.com                    ListObjects                         glitch     53.94.201.69
2024-11-28T15:21:54Z  signin.amazonaws.com                ConsoleLogin                        glitch     53.94.201.69
2024-11-28T15:21:57Z  ce.amazonaws.com                    GetCostAndUsage                     glitch     53.94.201.69
2024-11-28T15:21:57Z  cost-optimization-hub.amazonaws.com  ListEnrollmentStatuses              glitch     53.94.201.69
2024-11-28T15:21:57Z  health.amazonaws.com                DescribeEventAggregates             glitch     53.94.201.69
2024-11-28T15:22:12Z  s3.amazonaws.com                    ListBuckets                         glitch     53.94.201.69
2024-11-28T15:22:14Z  s3.amazonaws.com                    GetStorageLensConfiguration         glitch     AWS Internal
2024-11-28T15:22:14Z  s3.amazonaws.com                    GetStorageLensDashboardDataInternal glitch     AWS Internal
2024-11-28T15:22:13Z  s3.amazonaws.com                    GetStorageLensDashboardDataInternal glitch     AWS Internal
2024-11-28T15:21:57Z  health.amazonaws.com                DescribeEventAggregates             glitch     53.94.201.69
2024-11-28T15:21:57Z  ce.amazonaws.com                    GetCostAndUsage                     glitch     53.94.201.69
```

*We still need information about which tool and OS were used in the requests. Let's view the **userAgent** value related to these events using the following command.*

```
ubuntu@tryhackme:~/wareville_logs$ jq -r '["Event_Time", "Event_type", "Event_Name", "User_Name", "Source_IP", "User_Agent"],(.Records[] | select(.userIdentit
y.userName == "glitch") | [.eventTime,.eventType, .eventName, .userIdentity.userName // "N/A",.sourceIPAddress // "N/A", .userAgent // "N/A"]) | @tsv' cloudtr
ail_log.json | column -t -s $'    '
```

OUTPUT -

```
Event_Time           Event_type       Event_Name                          User_Name  Source_IP     User_Agent
2024-11-28T15:22:12Z  AwsApiCall       HeadBucket                          glitch     53.94.201.69  [S3Console/0.4, aws-internal/3 aws-sdk-java/1.12.750 Lin
ux/5.10.226-192.879.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.412-b09 java/1.8.0_412 vendor/Oracle_Corporation cfg/retry-mode/standard]
2024-11-28T15:22:23Z  AwsApiCall       ListObjects                         glitch     53.94.201.69  [S3Console/0.4, aws-internal/3 aws-sdk-java/1.12.750 Lin
ux/5.10.226-192.879.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.412-b09 java/1.8.0_412 vendor/Oracle_Corporation cfg/retry-mode/standard]
2024-11-28T15:22:25Z  AwsApiCall       ListObjects                         glitch     53.94.201.69  [S3Console/0.4, aws-internal/3 aws-sdk-java/1.12.750 Lin
ux/5.10.226-192.879.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.412-b09 java/1.8.0_412 vendor/Oracle_Corporation cfg/retry-mode/standard]
2024-11-28T15:22:39Z  AwsApiCall       PutObject                           glitch     53.94.201.69  [Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWe
bKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36]
2024-11-28T15:22:44Z  AwsApiCall       ListObjects                         glitch     53.94.201.69  [S3Console/0.4, aws-internal/3 aws-sdk-java/1.12.750 Lin
ux/5.10.226-193.880.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.412-b09 java/1.8.0_412 vendor/Oracle_Corporation cfg/retry-mode/standard]
2024-11-28T15:21:54Z  AwsConsoleSignIn  ConsoleLogin                        glitch     53.94.201.69  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
2024-11-28T15:21:57Z  AwsApiCall       GetCostAndUsage                     glitch     53.94.201.69  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
•024-11-28T15:21:57Z  AwsApiCall       ListEnrollmentStatuses              glitch     53.94.201.69  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
2024-11-28T15:21:57Z  AwsApiCall       DescribeEventAggregates             glitch     53.94.201.69  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWeb
Kit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
2024-11-28T15:22:12Z  AwsApiCall       ListBuckets                         glitch     53.94.201.69  [S3Console/0.4, aws-internal/3 aws-sdk-java/1.12.750 Lin
ux/5.10.226-193.880.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.412-b09 java/1.8.0_412 vendor/Oracle_Corporation cfg/retry-mode/standard]
```

- ***So we found that the username glitch is using the browser "Mozilla/5.0 and GoogleChrome in MAC OS rather than the s3 console which increases our suspition.***

Now we will look at for who created this anomalous user account. We will filter for all IAM-related events, and this can be done by using the select filter `.eventSource == "iam.amazonaws.com"`

```
ubuntu@tryhackme:~/wareville_logs$ jq -r '["Event_Time", "Event_Source", "Event_Name", "User_Name", "Source_IP"], (.Records[] | select(.
azonaws.com") | [.eventTime, .eventSource, .eventName, .userIdentity.userName // "N/A", .sourceIPAddress // "N/A"]) | @tsv' cloudtrail_l
$'     '
```

- ***OUTPUT***

```
Event_Time            Event_Source         Event_Name          User_Name  Source_IP
2024-11-28T15:21:26Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:29Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:30Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:30Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:30Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:31Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:31Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:30Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:25Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:31Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:31Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:31Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:31Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:33Z  iam.amazonaws.com    GetPolicy           mcskidy    53.94.201.69
2024-11-28T15:21:33Z  iam.amazonaws.com    GetPolicy           mcskidy    53.94.201.69
2024-11-28T15:21:31Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:33Z  iam.amazonaws.com    GetPolicy           mcskidy    53.94.201.69
2024-11-28T15:21:33Z  iam.amazonaws.com    GetPolicy           mcskidy    53.94.201.69
2024-11-28T15:21:32Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:32Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:36Z  iam.amazonaws.com    CreateLoginProfile  mcskidy    53.94.201.69
2024-11-28T15:21:36Z  iam.amazonaws.com    AttachUserPolicy    mcskidy    53.94.201.69
2024-11-28T15:21:32Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:32Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:32Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:32Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:32Z  iam.amazonaws.com    ListPolicies        mcskidy    53.94.201.69
2024-11-28T15:21:44Z  iam.amazonaws.com    ListUsers           mcskidy    53.94.201.69
2024-11-28T15:21:35Z  iam.amazonaws.com    CreateUser          mcskidy    53.94.201.69
ubuntu@tryhackme:~/wareville_logs$
```

Based on the results, there are many ListPolicies events. By ignoring these events, it seems that the most significant IAM activity is about the user mcskidy invoking the CreateUser action and consequently invoking the

***AttachUserPolicy* action. The source IP where the requests were made is *53.94.201.69*.**

- ***Remember that it is the same IP the anomalous user "glitch" used.***

Now I need to know what permissions the anomalous user has. It could be devastating if it has access to our whole environment. We need to filter for the AttachUserPolicy event to uncover the permissions set for the newly created

```
ubuntu@tryhackme:~/wareville_logs$ jq '.Records[] | select(.eventSource=="iam.amazonaws.com" and .eventName== "AttachUserPolicy")' cloudtrail_log.json
```

- ***OUTPUT***

Q. When we asked about this to McSkidy??

McSkidy is baffled by these results. She knows that she did not create the anomalous user and did not assign the privileged access. She also doesn't recognise the IP address involved in the events and does not use a Mac OS; she only uses a Windows

machine. All this information is different to the typical IP address and machine used by McSkidy, so she wants to prove her innocence and asks to continue the investigation.

Logs Don't Lie

McSkidy suggests looking closely at the IP address and operating system related to all these anomalous events. Let's use the following command below to continue with the investigation:

```
File  Edit  View  Search  Terminal  Help
ubuntu@tryhackme:~/wareville_logs$ jq -r '["Event_Time", "Event_Source", "Event_Name", "User_Name", "Source_IP"]
, (.Records[] | select(.sourceIPAddress=="53.94.201.69") | [.eventTime, .eventSource, .eventName, .userIdentity.
userName // "N/A", .sourceIPAddress // "N/A"]) | @tsv' cloudtrail_log.json | column -t -s $'    '
Event_Time            Event_Source                Event_Name              User_Name       Source_IP
2024-11-28T15:20:38Z  s3.amazonaws.com            HeadBucket              mayor_malware   53.94.201.69
2024-11-28T15:22:12Z  s3.amazonaws.com            HeadBucket              glitch          53.94.201.69
2024-11-28T15:22:23Z  s3.amazonaws.com            ListObjects             glitch          53.94.201.69
2024-11-28T15:22:25Z  s3.amazonaws.com            ListObjects             glitch          53.94.201.69
2024-11-28T15:22:39Z  s3.amazonaws.com            PutObject               glitch          53.94.201.69
2024-11-28T15:22:39Z  s3.amazonaws.com            PreflightRequest        N/A             53.94.201.69
2024-11-28T15:22:44Z  s3.amazonaws.com            ListObjects             glitch          53.94.201.69
2024-11-28T15:18:37Z  signin.amazonaws.com        ConsoleLogin            mayor_malware   53.94.201.69
2024-11-28T15:20:54Z  signin.amazonaws.com        ConsoleLogin            mcskidy         53.94.201.69
2024-11-28T15:21:54Z  signin.amazonaws.com        ConsoleLogin            glitch          53.94.201.69
2024-11-28T15:21:26Z  iam.amazonaws.com           ListPolicies            mcskidy         53.94.201.69
2024-11-28T15:21:29Z  iam.amazonaws.com           ListPolicies            mcskidy         53.94.201.69
2024-11-28T15:21:30Z  iam.amazonaws.com           ListPolicies            mcskidy         53.94.201.69
```

- *Based on the command output, three user accounts (**mcskidy**, **glitch**, and **mayor_malware**) were accessed from the same IP address.*

Confirming my suspicion on Mayor malware

```
ubuntu@tryhackme:~/wareville_logs$ jq -r '["Event_Time","Event_Source","Event_Name", "User_Name","User_Agent","S
ource_IP"],(.Records[] | select(.userIdentity.userName=="mayor_malware") | [.eventTime, .eventSource, .eventName
, .userIdentity.userName // "N/A",.userAgent // "N/A",.sourceIPAddress // "N/A"]) | @tsv' cloudtrail_log.json |
column -t -s $' '
Event_Time            Event_Source          Event_Name    User_Name      User_Agent

                                                                                                  Source_IP
2024-11-28T15:20:38Z  s3.amazonaws.com      HeadBucket    mayor_malware  [S3Console/0.4, aws-internal/3 aws-sdk-
java/1.12.750 Linux/5.10.226-193.880.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.412-b09 java/1.8.0_412 vendor/O
racle_Corporation cfg/retry-mode/standard]  53.94.201.69
2024-11-28T15:18:37Z  signin.amazonaws.com  ConsoleLogin  mayor_malware  Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
                                                                                                  53.94.201.69
2024-11-22T11:08:03Z  signin.amazonaws.com  ConsoleLogin  mayor_malware  Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
                                                                                                  53.94.201.69
```

Now from this we have confirmed that mayor_malware username is the one doing malicious activities it used the ip address ***53.94.201.69*** **.**

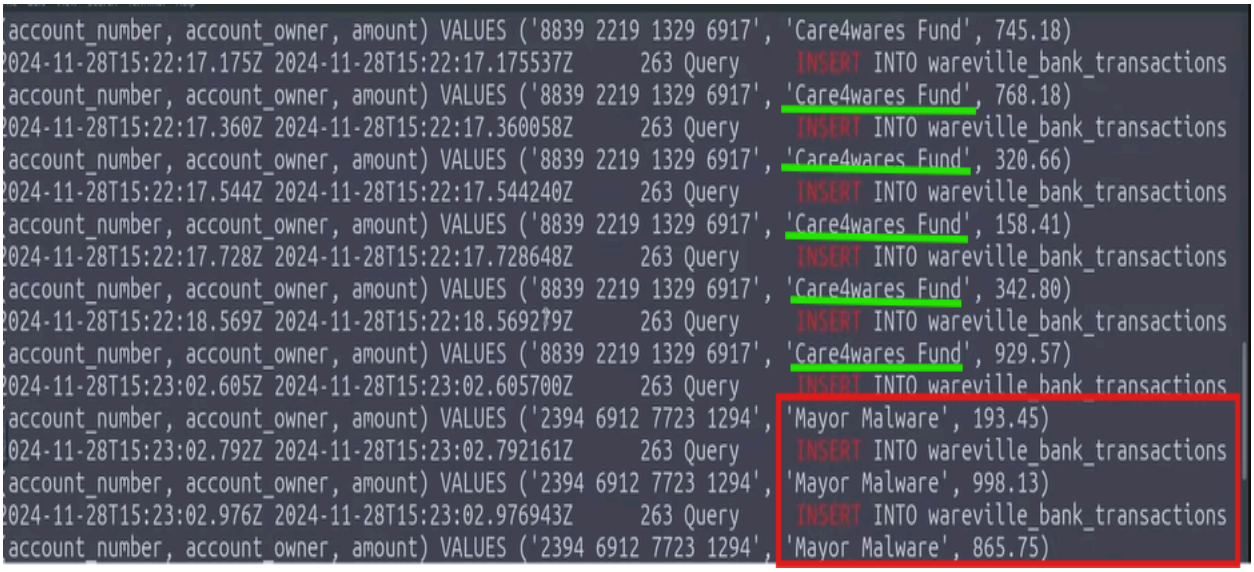
As glitch was using mac and mozilla 5.0 and here username mayor_malware also uses the same . confirms our suspition …….

LAST PART OF INVESTIGATION ……

McSkidy suggests gathering stronger proof that that person was behind this incident. Luckily, Wareville Bank cooperated with us and provided their database logs from their Amazon Relational Database Service (RDS). They also mentioned that these are captured through their CloudWatch, which differs from the CloudTrail logs as they are not stored in JSON format. For now, let's look at the bank transactions stored in the `~/wareville_logs/rds.log` *file.*

Since the log entries are different from the logs we previously investigated, McSkidy provided some guidance on how to analyse them. According to her, we can use the following command to show all the bank transactions.

```
ubuntu@tryhackme:~/wareville_logs$ ls
cloudtrail_log.json  rds.log
ubuntu@tryhackme:~/wareville_logs$ grep INSERT rds.log
2024-11-28T14:28:37.962Z 2024-11-28T14:28:37.962985Z      263 Query     INSERT INTO wareville_bank_transactions
(account_number, account_owner, amount) VALUES ('8839 2219 1329 6917', 'Care4wares Fund', 638.99)
2024-11-28T14:30:13.383Z 2024-11-28T14:30:13.383504Z      263 Query     INSERT INTO wareville_bank_transactions
(account_number, account_owner, amount) VALUES ('8839 2219 1329 6917', 'Care4wares Fund', 406.22)
```



*So this transaction made by **Mayor Malware** from care4wares Funds gets our culprit what was informed initially by the McSkidy.*

Answer the questions below

What is the other activity made by the user glitch aside from the ListObject action?

PutObject — ✓ Correct Answer

What is the source IP related to the S3 bucket activities of the user glitch?

53.94.201.69 — ✓ Correct Answer

Based on the eventSource field, what AWS service generates the ConsoleLogin event?

signin.amazonaws.com — ✓ Correct Answer

When did the anomalous user trigger the ConsoleLogin event?

2024-11-28T15:21:54Z — ✓ Correct Answer

What was the name of the user that was created by the mcskidy user?

glitch — ✓ Correct Answer — Hint

What type of access was assigned to the anomalous user?

AdministratorAccess — ✓ Correct Answer — Hint

Which IP does Mayor Malware typically use to log into AWS?

53.94.201.69 — ✓ Correct Answer

```
ubuntu@tryhackme:~/wareville_logs$ jq -r '["Event_Time", "Event_Source", "Event_Name", "User_Name", "Source_IP"]
,(.Records[] | select(.userIdentity.userName == "glitch") | [.eventTime, .eventSource, .eventName, .userIdentity
.userName // "N/A", .sourceIPAddress // "N/A"]) | @tsv' cloudtrail_log.json | column -t -s $'    '
Event_Time           Event_Source                Event_Name                User_Name  Sourc
e_IP
2024-11-28T15:22:12Z  s3.amazonaws.com           HeadBucket                glitch     53.94
.201.69
2024-11-28T15:22:23Z  s3.amazonaws.com           ListObjects               glitch     53.94
.201.69
2024-11-28T15:22:25Z  s3.amazonaws.com           ListObjects               glitch     53.94
.201.69
2024-11-28T15:22:39Z  s3.amazonaws.com           PutObject                 glitch     53.94
 201.69
2024-11-28T15:22:44Z  s3.amazonaws.com           ListObjects               glitch     53.94
.201.69
2024-11-28T15:21:54Z  signin.amazonaws.com        ConsoleLogin              glitch     53.94
.201.69
```

What is McSkidy's actual IP address?

31.210.15.79 ✓ Correct Answer

What is the bank account number owned by Mayor Malware?

2394 6912 7723 1294 ✓ Correct Answer