

# Business case relevance

## A new request from the Café leadership team

Martha and Frank are concerned because the website was hacked. They are relying on you to discover who did it and to make sure that it does not happen again.

Faythe, Frank, Martha, and others make frequent changes to the website, and sometimes those changes cause issues. Also, this morning, it looks like the website was hacked. Martha and Frank are asking Sofia if there is a way to track what was changed and who made the changes.

Play the role of Sofia, become a detective, and discover the culprit.

## Task 1: Modifying a security group and observing the website

5. From the **Services** menu, choose the **EC2** service.
6. Choose **Instances**, and then locate and select the **Café Web Server** instance.
7. In the **Security** tab, choose the **sg-xxxxxxxxxx** security group.
8. In the **Inbound rules** tab, notice that only one inbound rule has been defined, which is for HTTP access over TCP port 80.
9. Choose **Edit inbound rules**, and then choose **Add Rule** and configure the rule as follows:
  - **Type:** Select **SSH**
  - **Port Range:** Enter **22**
  - **Source:** Enter **My IP**

**Important:** Confirm that the TCP port 22 access will be open to only your IP address. The entry should show a Classless Inter-Domain Routing (CIDR) block that has a particular IP address followed by /32, not to all IP addresses (which would be shown by 0.0.0.0/0).

10. At the bottom of the page, choose **Save rules**.
11. Observe the Café website:
  - Choose **Instances**, select the **Café Web Server** instance. Click the **Details** tab and copy the **Public IPv4 address** value
  - Open a new browser tab, and navigate to `http://<WebServerIP>/cafe/` (substitute the `<WebServerIP>` value).
  - Notice that the website looks normal. For example, the photos are all appropriate for a bakery café.

## Task 2: Creating a CloudTrail log and observing the hacked website

In this task, you create a CloudTrail trail in your AWS account. You also notice that soon after creating the trail, the Café website is hacked.

### Task 2.1: Create a CloudTrail log

12. In the AWS Management Console, from the **Services** menu, select **CloudTrail**.
13. On the navigation pane on the left, choose **Trails**.

If the navigation pane is not displayed, choose the three icon of the horizontal lines on the top left of the screen.

14. Choose Create trail
15. Configure the trail as follows:
  - For **Trail name**, enter `Monitor` **Important:** Verify that you set the **Trail name** to **Monitor**, or this activity will not work as intended.
  - Select Create a new S3 bucket.
  - For **Trail log bucket and folder**, enter `monitoring####` (the `####` characters are four random digits).
  - For **AWS KMS alias**, enter your initials followed by `-KMS` (for example, `kc-KMS`).
16. Choose Next
17. On the **Choose log events** page, choose Next
18. On the **Review and create** page, choose Create trail
19. Verify that you see your trail on the **Trails** page.

### Task 2.2: Observe the hacked website

20. Return to the browser tab where you have the Café website open, and refresh the page.

**Important:** You might need to wait a full minute before the hack will occur. Also, your browser may be caching the images on this website. Press and hold Shift while you also choose the browser refresh button in order to see the latest changes to the website.

Notice that the website has been hacked. Who put that image there? The image certainly does not look correct.

It is up to you to figure out who hacked the website.

It is good that you enabled CloudTrail before this happened. CloudTrail can give you valuable information about what users have been doing in your account.

21. In the AWS Management Console, browse to the **EC2** service, and observe the **Café Web Server** instance details.

Does anything look suspicious?

22. In the **Security** tab, choose the **sg-xxxxxxxxxx** security group again, and then choose the **Inbound rules** tab.

Where did that extra entry come from?

You still see the entry you created earlier: the rule that opens port 22 to only your IP address. However, you also now see that someone else created an additional inbound rule that allows Secure Shell (SSH) access from anywhere (0.0.0.0/0).

Who added this security hole? You can search the CloudTrail logs to find out.

## Task 3: Analyzing the CloudTrail logs by using grep

In this task, you analyze the CloudTrail logs by using the grep Linux utility to see if you can figure out who hacked the website.

Task 3.1: Connect to the Café Web Server host EC2 instance by using SSH

Task 3.3: Download and extract the CloudTrail logs

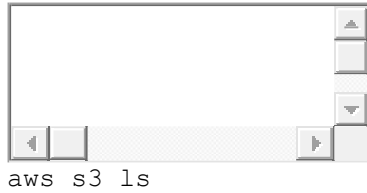
38. Verify that your terminal is connected via SSH to the Café Web Server EC2 instance.
39. Run the following command to create a local directory on the web server to download the CloudTrail log files to:



40. Run the following command to change the directory to the new directory:



41. Run the following command to list the buckets to recall the bucket name:



42. In the command below, replace `<monitoring####>` with the actual bucket name that starts with **monitoring** (the bucket name is part of the output from the `ls` command that you ran). Run the adjusted command to download the CloudTrail logs:



```
aws s3 cp s3://<monitoring####>/ . --recursive
```

If the command is successful, you should see that a few log files are downloaded.

**Important:** If there was no output in the command line when you ran the last command, it likely means that not enough time has passed since you created the CloudWatch trail. CloudWatch posts logs to Amazon Simple Storage Service (Amazon S3) every 5 minutes. You might need to wait and try running the command again. Do not proceed to the next step until you have downloaded at least one log file.

43. Use the `cd` and `ls` commands repeatedly (or enter `cd` and then press Tab multiple times) as necessary to change the directory to the subdirectory where the logs were downloaded. When you run `ls`, all of the downloaded log files should display. They will be located in an **AWSLogs/<account-num>/CloudTrail/<Region>/<yyyy>/<mm>/<dd>** subdirectory.

Notice that the log files end in `.json.gz`, which indicates that they are compressed as GNU zip files.

44. Run the following command to extract the logs:



```
gunzip *.gz
```

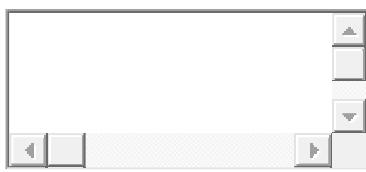
45. Run `ls` again. Notice that all files are now extracted.

### Task 3.4: Analyze the logs by using grep

In this section of the activity, you use the Linux `grep` utility to analyze the CloudTrail logs.

46. To analyze the structure of the logs, do the following:

- Copy one of the file names returned by the `ls` command that you ran.
- Enter `cat` in the terminal window, followed by a space, and then paste the copied file name. Run the command.
- Note that the files are in JavaScript Object Notation (JSON) format. However, it is difficult to read them in this output format.
- Run the `cat` command again, but this time format the output (replace `<filename.json>` with the actual file name):



A terminal window with a light gray background. The command `cat <filename.json> | python -m json.tool` is entered at the prompt. The window has standard Linux terminal window controls (minimize, maximize, close) in the top right corner and navigation buttons in the bottom left corner.

```
cat <filename.json> | python -m json.tool
```

This format is more readable. You can now also see the structure of the log entries. Notice that each entry contains the same standard fields, including `awsRegion`, `eventName`, `eventSource`, `eventTime`, `requestParameters`, `sourceIPAddress`, `userIdentity`, and more.

46. You can now read the log entries. However, the number of entries—even in just this one log file—can be large. You might have downloaded more than one log because new log files are created over time. You need to find a way to search log entries across multiple files and also filter the results.
47. Consider how you want to target the search. You are not interested in everything that is happening in this account. Instead, your interest is in an action that was taken on a particular EC2 instance (that is, the web server that was hacked).

Start by filtering the log results where the **sourceIpAddress** matches the IP address of the Café Web Server instance.

Run the following command to set the `WebServerIP` address as a variable that you can use in future commands (replace `<WebServerIP>` with the actual IP address that displays to the left of these instructions):



A terminal window with a light gray background. The command `ip=<WebServerIP>` is entered at the prompt. The window has standard Linux terminal window controls in the top right corner and navigation buttons in the bottom left corner.

```
ip=<WebServerIP>
```

48. Run the following command:



```
for i in $(ls); do echo $i && cat $i | python -m json.tool | grep
sourceIPAddress ; done
```

The command you ran does the following:

- It creates a **for** loop that includes the names of the files in the current directory.
- During each iteration of the for loop, it echoes the file name and then prints the contents of the file in JSON format.
- Only the lines of JSON that contain the `sourceIPAddress` tag are printed.

Note that there are several log entries in the trail where the **sourceIPAddress** was the Café Web Server instance.

49. Run a similarly structured command but where the command returns the **eventName** of every captured event:



```
for i in $(ls); do echo $i && cat $i | python -m json.tool | grep
eventName ; done
```

The command you ran follows the same logic as the command you ran before, but this time, it filters log entries for the **eventName**.

The results of the previous command contain different details. Many **describe** and **list** actions were recorded, and they look relatively harmless. However, if you scroll through the list, you notice that occasional **update** actions were also recorded. You could use a text editor like `vi` to open a log that contains a recorded event that you want to know more about. You can then search for that `eventName` and look at the details.

However, you might benefit from using a different tool other than `grep` to locate these log entries more easily.

### Task 3.5: Analyze the logs by using AWS CLI CloudTrail commands

Another approach you can use to analyze CloudTrail logs is to use AWS CLI CloudTrail commands.

50. Open the [AWS CLI Reference page for CloudTrail](#).
51. Choose the **lookup-events** command to see details about the command.

- Notice that you can look up events based on one of eight different attributes, including AWS access key, event name, user name, and others.
- In the AWS CLI Command Reference page, scroll to the **Example**, which shows how to filter the trail for console logins. Run that command in your terminal window:



```
aws cloudtrail lookup-events --lookup-attributes
AttributeKey=EventName,AttributeValue=ConsoleLogin
```

The results indicate that the only user who has logged into the console is the same user that you are logged into the console as

However, there are other ways to modify resources on AWS instead of using the console. The hacker might have used a different approach.

52. Run the following command to find any actions that were taken on security groups in the AWS account:



```
aws cloudtrail lookup-events --lookup-attributes
AttributeKey=ResourceType,AttributeValue=AWS::EC2::SecurityGroup --
output text
```

Something in this result set might contain some information that would help you discover what happened, but there might be too many results for you to easily identify the issue.

Perhaps you can narrow the search results further so that you get only the results related to the security group that is used by the web server instance.

53. Run the following commands to find the security group ID that is used by the Café Web Server instance, and then echo the result to the terminal:



```
region=$(curl http://169.254.169.254/latest/dynamic/instance-
identity/document|grep region | cut -d '"' -f4)
sgId=$(aws ec2 describe-instances --filters "Name=tag:Name,Values='Cafe
Web Server'" --query
```

```
'Reservations[*].Instances[*].SecurityGroups[*].[GroupId]' --region  
$region --output text)  
echo $sgId
```

Notice that a single security group ID was found.

54. Now use the security group ID that the previous command returned to further filter your AWS CLI CloudTrail command results:



```
aws cloudtrail lookup-events --lookup-attributes  
AttributeKey=ResourceType,AttributeValue=AWS::EC2::SecurityGroup --  
region $region --output text | grep $sgId
```

You could keep experimenting with different commands to filter the log results. However, you might wonder whether there is a better tool or solution for reading these logs. AWS has the AWS Partner Network (APN), where companies specialize in helping AWS customers with this challenge. See <https://aws.amazon.com/cloudtrail/partners/> for a listing of APN Partner solutions.

The APN Partner solutions suit the needs of many AWS customers. However, for the purposes of this activity, there is one additional approach to examining CloudTrail log files that you might use, and it uses another AWS service. In the next task, you explore CloudTrail logs by using Athena.

## Task 4: Analyzing the CloudTrail logs by using Athena

As you experienced in the previous task, it can be difficult to find specific information within a very large dataset. CloudTrail logs are verbose for a reason: you might want to know every relevant detail about a particular action that was taken in your AWS account. However, using command line tools to filter the logs can be tedious.

It would be convenient if all the log data were in a database and you could use structured query language (SQL) queries to search for the log entries that you are most interested in. Athena provides such a solution. Athena is an interactive query service that makes it easy to analyze data in Amazon S3 by using standard SQL.

In this task, you use Athena to analyze your CloudTrail logs.



#### Task 4.1: Create the Athena table

55. From the AWS Management Console **Services** menu, choose **CloudTrail** to open the CloudTrail console.

56. In the navigation pane, choose **Event history**.

Notice that CloudTrail provides this event history interface where you can apply filters and conduct a basic search based on parameters, such as **Event name** or **Resource type**. The **Event history** page can be a useful tool, and you are free to explore it. However, in this activity, you use Athena.

57. From the **Event history** page, click **Create Athena table**

- **Storage location:** Choose the **monitoring####** S3 bucket where you configured CloudTrail to store log files.

58. Take a moment to analyze how the Athena **CREATE TABLE** statement is formed.

- It creates a database column for each of the standard name-value pairs in each JSON-formatted CloudTrail log entry. Refer to the image of the JSON format of a typical log entry in Task 3.4 to confirm this information.
- At the bottom of the **CREATE TABLE SQL** statement, notice the **LOCATION** statement. This indicates the Amazon S3 location where the table data will be stored. In this case, the data is already there. You are defining the table schema that will be used to parse existing JSON-structured data.
- For details on CloudTrail record structure, see <https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-event-reference.html>.
- For details on how this Athena table was created, see the **CREATE EXTERNAL TABLE** document at <https://docs.aws.amazon.com/athena/latest/ug/cloudtrail-logs.html>.

59. After you are done analyzing the **CREATE TABLE** details, choose **Create table**.

The table is created with a default name that includes the name of the S3 bucket.

60. From the **Services** menu, choose the **Athena** service.

#### Task 4.2: Analyze logs using Athena

The advantage of using Athena is that you can now run SQL queries over your log data.

61. If you do not already see the **Athena Query Editor**, choose **Get Started** and it should then display.

If a **Tutorial** screen appears, choose the **X** in the top corner to exit out of it.

62. In the left panel of the **Athena Query Editor**, you should see the **cloudtrail\_logs\_monitoring####** table.

Choose the table to reveal the column names.

**Analysis:** Notice how each standard child element that exists in a CloudTrail log record in JSON format has a corresponding column name in this database. The **useridentity** database column is a struct type, because it contains more than a single name-value pair. Similarly, the **resources** database column is an array.

63. Start by setting up a query results location and then running a simple query to get an idea of the data that is available in the logs.

On the menu bar at the upper right of the page, choose **Settings**.

- Set **Query result location** to `s3://monitoring####/results/` and replace `monitoring####` with the name of the bucket you created earlier.
- Choose **Save**.
- Paste the following SQL query into the **New query 1** panel. Replace `####` with the numbers in your actual table, and choose **Run query**.



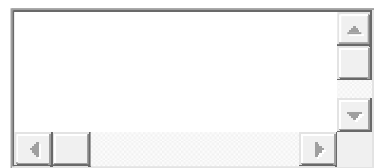
```
SELECT *
FROM cloudtrail_logs_monitoring####
LIMIT 5
```

This query returns five rows of data. Look at the result set (scroll to the right in the **Results** panel to see additional column data).

Focus on the columns **useridentity**, **eventtime**, **eventsources**, **eventname**, and **requestparameters**, which contain the most valuable information to help you find the origin of the hack.

The **useridentity** column has many details that make it more difficult to read though. You now return only the user name for that column.

64. Run a new query that selects only those columns that were previously mentioned. This time, limit the results to 30 rows:



```
SELECT useridentity.userName, eventtime, eventsources, eventname,
requestparameters
FROM cloudtrail_logs_monitoring####
LIMIT 30
```

You should now be able to find out who modified the security group that is associated with the Café Web Server instance.

