SPL Query Question 1

Question

You're tasked to find the IAM (Identity & Access Management) users that accessed an AWS service in Frothly's AWS environment.

Refer to the following link to get an idea of what source type you need to query and what field in the results will have the answer you're seeking.

Link: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-fileexamples.html

List out the IAM users that accessed an AWS service (successfully or unsuccessfully) in Frothly's AWS environment?

Answer guidance: Comma separated without spaces, in alphabetical order. (Example: ajackson,mjones,tmiller)

Hint: Use aws:cloudtrail as the source type

Understanding Question 1

What Being Asked:
Find all **IAM users** (not root accounts, not service accounts) who made API calls to AWS services in Frothly's environment, regardless of whether those calls succeeded or failed.

Key Requirements:

1. **Source type:** aws:cloudtrail (AWS audit logs)
2. **Field to examine:** userIdentity field (contains user information)
3. **Format:** Comma-separated usernames, alphabetical order, no spaces
4. **Scope:** All AWS services accessed (not just one service)

Why CloudTrail?
CloudTrail is AWS's logging service that records **API calls** made in an AWS account. Every time someone (or something) calls an AWS API, CloudTrail logs it. The link shows examples of what these logs look like.

Step-by-Step Guide to Solve Question 1

Step 1: Examine the CloudTrail Data Structure

First, see what CloudTrail data looks like:

```
index=botsv3 sourcetype=aws:cloudtrail
| head 5
| table _time, eventName, userIdentity.*
```

This shows sample records. Notice the userIdentity field - it's a JSON object containing information about who made the API call.

Step 2: Identify IAM Users Specifically

Not all users in CloudTrail are IAM users. need to filter for **only IAM users**:

```
index=botsv3 sourcetype=aws:cloudtrail
userIdentity.type="IAMUser"
| head 5
| table _time, eventName, userIdentity.userName,
userIdentity.type
```

Why userIdentity.type="IAMUser"?

- type can be: IAMUser, Root, AssumedRole, AWSAccount, AWSService
- We only want IAMUser (human users or service accounts created in IAM)

Step 3: Extract Unique Usernames

Now get all unique usernames:

```
index=botsv3 sourcetype=aws:cloudtrail
userIdentity.type="IAMUser"
| stats values(userIdentity.userName) as usernames
```

This gives you a list of all IAM users who made API calls.

Step 4: Format the Answer Properly

```
index=botsv3 sourcetype=aws:cloudtrail userIdentity.type="IAMUser"
| stats values(userIdentity.userName) as usernames
| eval usernames = mvdedup(usernames)
| eval usernames = mvmap(usernames, lower(usernames))
| eval sorted_usernames = mvsort(usernames)
| eval answer = mvjoin(sorted_usernames, ",")
| table answer
```

Complete Working Solution

Here's the complete query that should give the answer:

```
index=botsv3 sourcetype=aws:cloudtrail userIdentity.type="IAMUser"
| stats dc(userIdentity.userName) as unique_users
| appendcols [
   search index=botsv3 sourcetype=aws:cloudtrail userIdentity.type="IAMUser"
   | stats values(userIdentity.userName) as usernames
   | eval usernames = mvdedup(usernames)
   | eval usernames = mvsort(usernames)
   | eval answer = mvjoin(usernames, ",")
]
| table unique_users, answer
```

SPL Query Question 2

Question

The following links are provided to help you with this question.
Links:
• https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucketpublic-access/
• https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatchalarms-for-cloudtrail-additional-examples.html#cloudwatch-alarms-forcloudtrail-no-mfa-example
Make sure you exclude events related to console logins.
It might be a good idea to do a keyword search query on this one. Don't forget to surround the keyword with asterisks.
What field would you use to alert that AWS API activity has occurred without MFA (multi-factor authentication)? Answer guidance: Provide the full JSON path. (Example: iceCream.flavors.traditional)
Hint: Use aws:cloudtrail as the source type.

Understanding Question 2

What Being Asked:
Find the **specific JSON field path** in CloudTrail logs that indicates whether MFA (Multi-Factor Authentication) was used for an AWS API call. Need to identify the exact field that would trigger an alert when API activity occurs **without** MFA.

Key Requirements:

1. **Source type:** aws:cloudtrail (AWS audit logs)
2. **Exclude console logins:** Focus on API calls only
3. **Format:** Full JSON path (e.g., parent.child.field)
4. **Goal:** Find field that shows MFA was NOT used

Why This Matters:

- AWS CloudTrail logs API calls with detailed metadata
- Some fields indicate authentication method
- to monitor security gaps (API calls without MFA)

Step-by-Step Guide to Solve Question 2

Step 1: Understand the Documentation Links

Link 1: S3 Bucket Public Access

- Amazon S3 Block Public Access provides settings for access points, buckets, organizations, and accounts to help manage public access to Amazon S3 resources.

Link 2: CRITICAL - CloudTrail No-MFA Example

- Shows exactly how to detect API calls without MFA
- Key example: { $.additionalEventData.MFAUsed = "No" }
- This is a CloudWatch alarm pattern that triggers when additionalEventData.MFAUsed equals "No"

Step 2: Search for MFA-Related Fields

explore fields containing "MFA" in CloudTrail:

```
index=botsv3 sourcetype=aws:cloudtrail
| search *MFA*
| fieldsummary
| search field="*MFA*"
```

Analysis of Results:
found TWO MFA-related fields:

- **additionaleventData.MFAUsed** (4 occurrences, all with value "No")
- **userIdentity.sessionContext.attributes.mfaAuthenticated** (2155 occurrences, all with value "false")

Understanding the Difference

Field 1: additionaleventData.MFAUsed

- This is for **API activity** (non-console logins)
- Appears only 4 times in your data
- Value "No" means API call was made without MFA
- This is the field mentioned in the AWS documentation

Field 2: userIdentity.sessionContext.attributes.mfaAuthenticated

- This is for **console logins** (using AWS Management Console)
- Appears 2155 times in your data
- Value "false" means console login was without MFA

Step 3: Exclude Console Logins

Console logins use different MFA fields. We only want API calls:

```
index=botsv3 sourcetype=aws:cloudtrail NOT eventName="ConsoleLogin"
| search *MFA*
| head 5
| table _time, eventName, additionalEventData.*
```

Step 4: Identify the Correct Field

From the documentation and exploring data, the key field is:

- **additionalEventData.MFAUsed** - Contains "Yes" or "No"
- **userIdentity.sessionContext.attributes.mfaAuthenticated** - Contains "True" or "false"

Let's verify it exists:

```
index=botsv3 sourcetype=aws:cloudtrail NOT eventName="ConsoleLogin"
| where isnull(additionalEventData.MFAUsed)
| stats count by additionalEventData.MFAUsed
```

```
index=botsv3 sourcetype=aws:cloudtrail NOT eventName="ConsoleLogin"
| where isnull(userIdentity.sessionContext.attributes.mfaAuthenticated)
| stats count by userIdentity.sessionContext.attributes.mfaAuthenticated
```

Step 5: Alternative Field Check

Check all possible MFA fields:

```
index=botsv3 sourcetype=aws:cloudtrail
| foreach * [eval <<FIELD>>_exists = if(isnotnull('<<FIELD>>'), "<<FIELD>>", null())]
| fields *_exists
| transpose
| search column="*MFA*"
```

Complete Working Solution

Query to Find the Field:

```
index=botsv3 sourcetype=aws:cloudtrail NOT eventName="ConsoleLogin"
| search *MFA*
| head 1
| fieldsummary
| search field="*MFA*"
| table field, count
```

```
index=botsv3 sourcetype=aws:cloudtrail
| fieldsummary
| search field="*mfa*" OR field="*MFA*"
| table field, count
```

SPL Query Question 3

Question

Look at the source types available in the dataset. There might be one in particular that holds information on hardware, such as processors.
What is the processor number used on the web servers? Answer guidance: Include any special characters/punctuation. (Example: The processor number for Intel Core i7-8650U is i7-8650U.)
Hint: Use hardware as the source type in Splunk Search for find hardware information such as CPU statistics, hard drives, network interface cards, memory, and more

Understanding Question 3

What Being Asked:
Find the **processor number** used on the **web servers** in Frothly's environment.
This means need to:

1. Identify which hosts are web servers
2. Find their processor information from hardware data
3. Report the processor number with any special characters/punctuation

Key Requirements:

1. **Source type:** hardware (contains system hardware info)
2. **Focus:** Web servers specifically (not all servers)
3. **Answer format:** Processor number exactly as shown (e.g., i7-8650U, E5-2670 v3)

Step-by-Step Guide to Solve Question 3

Step 1: First, Explore Hardware Data

Use hardware as the source type (per hint)
Hardware logs contain system information including processor details

```
index=botsv3 sourcetype=hardware
| head 10
| table host, processor, os, category, type
```

As per result the the data cant be pull out direcly. Need to take out the data from the raw record.

Step 2: Extract Processor Information

The processor field contains the CPU model information:

```
index=botsv3 sourcetype=hardware
| rex field=_raw "CPU_TYPE\s+(?<processor>[^\n]+)"
| table host, processor
```

This query was successfully to pull out the CPU model information.

SPL Query Questions 4

Questions

Questions 4-6:
A common misconfiguration involving AWS is publically
accessible S3 buckets. Read the following resource to understand ACLs
and S3 buckets.
Link: https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutBucketAcl.html
Question 4: Bud accidentally makes an S3 bucket publicly accessible. What is the event ID of the API call that enabled public access? Answer guidance: Include any special characters/punctuation.
Hint: Use aws:cloudtrail as the source type to search for the PutBucketAcl event.

Understanding Question 4

What Being Asked:
Find the **Event ID** of the specific AWS CloudTrail log entry where Bud performed a PutBucketAcl API call that made an S3 bucket publicly accessible.

Key Requirements:

1. **Source type:** aws:cloudtrail (AWS audit logs)
2. **Event name:** PutBucketAcl (the API call that changes bucket permissions)
3. **Goal:** Find the exact Event ID (a unique UUID)
4. **Format:** Include all special characters/hyphens

Why This Matters:

- PutBucketAcl is the API call that modifies S3 bucket Access Control Lists (ACLs)
- Someone (Bud) accidentally set a bucket to public access
- Each CloudTrail event has a unique eventID for tracking

Step-by-Step Guide to Solve Question 4

Step 1: First, Find All PutBucketAcl Events

Let's start by seeing if there are any PutBucketAcl events in the data:

```
index=botsv3 sourcetype=aws:cloudtrail eventName="PutBucketAcl"
| table _time, eventID, userIdentity.userName, requestParameters.bucketName
```

Step 2: Look for Public ACL Settings

The ACL could be set to public in several ways. examine the event details:

```
index=botsv3 sourcetype=aws:cloudtrail eventName="PutBucketAcl"
| table _time, eventID, userIdentity.userName, requestParameters.bucketName,
requestParameters.acl, requestParameters.grantList
```

Step 3: Direct Approach

```
index=botsv3 sourcetype=aws:cloudtrail eventName="PutBucketAcl"
userIdentity.userName="bstoll"
| table eventID, requestParameters.bucketName, requestParameters.acl,
requestParameters.grantList
```

Step 4: Check the Raw Event Details

```
index=botsv3 sourcetype=aws:cloudtrail eventID="ab45689d-69cd-41e7-8705-
5350402cf7ac"
| spath
| table *
```

```
index=botsv3 sourcetype=aws:cloudtrail eventID="9a33d8df-1e16-4d58-b36d-
8e80ce68f8a3"
| spath
| table *
```

SPL Query Question 5

Question

What is Bud's username?

What Being Asked:
Find the name of a Bud's username based on question 4.

Key Requirements:

1. **Source type:** aws:cloudtrail (AWS audit logs)
2. **Event name:** PutBucketAcl (the API call that changes bucket permissions)
3. **Goal:** Find the name of a Bud's username

Step-by-Step Guide to Solve Question 5

Step 1: Verify User Identity

```
index=botsv3 sourcetype=aws:cloudtrail eventName="PutBucketAcl"
| stats values(userIdentity.userName) as user
```

Step 2: Cross-Reference with Other Activities

```
index=botsv3 sourcetype=aws:cloudtrail userIdentity.userName="bstoll"
| stats count by eventName
| sort -count
```

Step 3: Confirm Through Additional Evidence

```
index=botsv3 sourcetype=aws:cloudtrail userIdentity.userName="bstoll"
| head 5
| table _time, eventName, eventSource, sourceIPAddress
```

SPL Query Question 6

Question

What is the name of the S3 bucket that was made publicly accessible? Hint: Use aws:cloudtrail as the source type.

What Being Asked:
Find the name of the S3 bucket that was made publicly accessible.

Key Requirements:

1. **Source type:** aws:cloudtrail (AWS audit logs)
2. **Event name:** PutBucketAcl (the API call that changes bucket permissions)
3. **Goal:** Find the name of the S3 bucket that was made publicly accessible

Step-by-Step Guide to Solve Question 6

Step 1: Extract Bucket Name from PutBucketAcl Event

```
index=botsv3 sourcetype=aws:cloudtrail eventName="PutBucketAcl"
| stats values(requestParameters.bucketName) as bucketName
```

Step 2: Check Bucket Configuration

```
index=botsv3 sourcetype=aws:cloudtrail
 requestParameters.bucketName="frothlywebcode"
| stats count by eventName
```

Step 3: Analyze Bucket Purpose
Based on the name frothlywebcode:

- Likely contains web application code
- Public exposure could lead to application compromise
- Potential for supply chain attacks

SPL Query Question 7

Question

You're tasked with identifying a text file uploaded to the S3 bucket. Here is a link for more information related to this topic.
Link: https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html
Since you know the name of the S3 bucket, you should easily find the answer to this question.
You will need to query a different AWS-related source type. HTTP status code might be helpful as well.
What is the name of the text file that was successfully uploaded into the S3 bucket while it was publicly accessible? Answer guidance: Provide just the file name and extension, not the full path. (Example: filename.docx instead of /mylogs/web/filename.docx)
Hint: Use aws:s3:accesslogs.

Understanding Question 7

What Being Asked:
Find the name of a **text file (.txt)** that was successfully uploaded (via PUT request) to the **public S3 bucket** (from Question 6) while it was publicly accessible.

Key Requirements:

1. **Source type:** aws:s3:accesslogs (different from aws:cloudtrail)
2. **Bucket:** Use the bucket name from Question 6
3. **HTTP method:** PUT (uploading a file)
4. **HTTP status:** 200 (successful upload)
5. **File type:** .txt extension
6. **Answer format:** Only filename and extension (no path)

Important Context:

- aws:s3:accesslogs are S3 server access logs (different from CloudTrail)
- These logs track every request made to an S3 bucket
- After making a bucket public, attackers might upload files to it
- We're looking for evidence of such an upload

Step-by-Step Guide to Solve Question 7

Step 1: First, Verify You Have S3 Access Logs Data

```
index=botsv3 sourcetype=aws:s3:accesslogs
| head 1
| table _time, bucket, key, http_method, http_status
```

Step 2: Run this diagnostic query first

```
index=botsv3 sourcetype=aws:s3:accesslogs
| head 1
| fieldsummary
| table field
```

As been understanding the log was raw.

Step 3: Let's Parse the Raw Logs

```
index=botsv3 sourcetype=aws:s3:accesslogs
| rex field=_raw
"^(?<bucket_owner>[^\s]+)\s+(?<bucket>[^\s]+)\s+\[(?<timestamp>[^\]]+)\]\s+(?<remote_ip>[^\s]+)\s+(?<requester>[^\s]+)\s+(?<request_id>[^\s]+)\s+(?<operation>[^\s]+)\s+(?<key>[^\s]+)\s+\"(?<request_uri>[^\"]+)\"\s+(?<http_status>\d+)"
| table _time, bucket, operation, key, http_status, request_uri
```

Step 4: Filter for the Correct Bucket

Based on your Q4-6, the bucket is frothlywebcode:

```
index=botsv3 sourcetype=aws:s3:accesslogs
| rex field=_raw
"^(?<bucket_owner>[^\s]+)\s+(?<bucket>[^\s]+)\s+\[(?<timestamp>[^\]]+)\]\s+(?<remote_ip>[^\s]+)\s+(?<requester>[^\s]+)\s+(?<request_id>[^\s]+)\s+(?<operation>[^\s]+)\s+(?<key>[^\s]+)\s+\"(?<request_uri>[^\"]+)\"\s+(?<http_status>\d+)"
| search bucket="frothlywebcode" http_status="200" key!="-"
| table _time, bucket, operation, key, http_status
```

Step 5: Look for PUT Operations

```
index=botsv3 sourcetype=aws:s3:accesslogs
| rex field=_raw
"^(?<bucket_owner>[^\s]+)\s+(?<bucket>[^\s]+)\s+\[(?<timestamp>[^\]]+)\]\s+(?<re
mote_ip>[^\s]+)\s+(?<requester>[^\s]+)\s+(?<request_id>[^\s]+)\s+(?<operation>[^\
s]+)\s+(?<key>[^\s]+)\s+\"(?<request_uri>[^\"]+)\"\s+(?<http_status>\d+)"
| search bucket="frothlywebcode" http_status="200" operation="REST.PUT.OBJECT"
| table _time, bucket, operation, key, http_status
```

Step 6: Filter for Text Files

```
index=botsv3 sourcetype=aws:s3:accesslogs
| rex field=_raw
"^(?<bucket_owner>[^\s]+)\s+(?<bucket>[^\s]+)\s+\[(?<timestamp>[^\]]+)\]\s+(?<re
mote_ip>[^\s]+)\s+(?<requester>[^\s]+)\s+(?<request_id>[^\s]+)\s+(?<operation>[^\
s]+)\s+(?<key>[^\s]+)\s+\"(?<request_uri>[^\"]+)\"\s+(?<http_status>\d+)"
| search bucket="frothlywebcode" http_status="200" operation="REST.PUT.OBJECT"
key="*.txt"
| table _time, bucket, operation, key, http_status
```

SPL Query Question 8

Question

What keywords can you start your search with to help identify what data sources can help you with this?
One of the fields within this source type clearly has the answer, but which is it?
Perhaps expanding upon your search to count on the operating systems and hosts will be helpful.
What is the FQDN of the endpoint that is running a different Windows operating system edition than the others?
Hint: Start with winhostmon as the source type.

Understanding Question 8

What Being Asked:
Find the **Fully Qualified Domain Name (FQDN)** of a Windows endpoint that has a **different operating system edition** compared to all other Windows endpoints in the environment.

Key Requirements:

1. **Source type:** winhostmon (Windows host monitoring data)
2. **Goal:** Find the outlier - one host with a different Windows edition
3. **Answer format:** FQDN (e.g., WIN-SERVER01.domain.local)
4. **Method:** Count operating systems and find the unique one

Step-by-Step Investigation:

Step 1: Start with winhostmon (as per hint)

```
index=botsv3 sourcetype=winhostmon
| stats count by host
```

winhostmon only contains driver information, not OS details.

Step 2: Identify Alternative Data Sources

```
| metadata type=sourcetypes index=botsv3
| search sourcetype="*windows*" OR sourcetype="*win*" OR sourcetype="*os*"
```

Keywords to use: windows, os, version, edition, operating system

Step 3: Use Windows Event Logs

Based on dataset, Windows OS information is in:

1. WinEventLog:* - Windows Event Logs
2. osquery:results - System inventory data

```
index=botsv3 sourcetype=WinEventLog:*
| rex field=_raw "ComputerName=(?<fqdn>[^\s]+)"
| stats count by fqdn
```

Step 4: Analyze Naming Patterns

```
index=botsv3 sourcetype=WinEventLog:*
| rex field=_raw "ComputerName=(?<computer>[^\s]+)"
| eval pattern = if(match(computer, "-L\.froth\.ly$"), "Client Pattern", "Different")
| stats count by computer, pattern
| sort -count
```

Step 5: Investigate Anomalous Host

```
index=botsv3 sourcetype=WinEventLog:*
| rex field=_raw "ComputerName=(?<computer>[^\s]+)"
| eval User=lower(mvindex(split(computer, "-"), 0))
| eval Role=case(
   match(computer, "BSTOLL-L\.froth\.ly"), "Suspect Workstation",
   match(computer, "SEPM"), "Security Server",
   match(computer, "-L\.froth\.ly$"), "Standard Workstation",
   true(), "Other"
)
| stats count as TotalEvents, values(User) as AssociatedUser by computer, Role
| sort -TotalEvents
```

Step 6: Verify Activity Levels

```
index=botsv3 sourcetype=WinEventLog:*
| rex field=_raw "ComputerName=(?<computer>[^\s]+)"
| stats count as TotalEvents by computer
| eval RelativeActivity = round(TotalEvents*100/sum(TotalEvents), 2)
| sort -TotalEvents
```

Complete Working Query:

```
index=botsv3 sourcetype=WinEventLog:*
| rex field=_raw "ComputerName=(?<computer>[^\s]+)"
| eval User=lower(mvindex(split(computer, "-"), 0))
| eval Role=case(
   match(computer, "BSTOLL-L\.froth\.ly"), "Suspect Workstation",
   match(computer, "SEPM"), "Security Server",
   match(computer, "-L\.froth\.ly$"), "Standard Workstation",
   true(), "Other"
)
| stats count as TotalEvents, values(User) as AssociatedUser by computer, Role
| sort -TotalEvents
| eval SuspectDevice=if(match(computer, "BSTOLL-L"), "Suspect", "Normal")
| eval RelativeActivity=round(TotalEvents*100/sum(TotalEvents), 2)
| table computer, AssociatedUser, Role, TotalEvents, RelativeActivity, SuspectDevice
```