ENPM665 - Classwork Lab 3: Investigating an AWS Breach with CloudTrail & IAM

Name - **Kalpesh Bharat Parmar**
UMD Directory ID - [redacted]
Course and section - **ENPM66**[redacted]

Part 1 - Attack Timeline & Entry Point Identification

We use the following command

```
jq -r '
.Records[]?
| select(.userAgent | test("aws-cli/1.27.74"))
| [
.eventTime,
.eventName,
(.userIdentity.arn // .userIdentity.userName // "-"),
.sourceIPAddress,
.eventSource,
 (.requestParameters.bucketName // "-")
]
| @tsv
' *.json | sort | column -ts$'\t'
```

We can see following events in order as followed

Time stamp	IP	Event name	Analysis
2023-08-26T20:29:37Z	84.32.71.19	GetCallerIdentity	By temp-user
2023-08-26T20:54:28Z	84.32.71.33	AssumeRole	temp-user assumes AdminRole
2023-08-26T21:17:16Z	84.32.71.3	GetObject	Using the assumed AdminRole S3 object was accessed and files were downloaded.

```
┌──(kp㉿kali)-[~/Desktop/ENPM665/lab3/IncidentLogs]
└─$ jq -r '
.Records[]?
| select(.userAgent | test("aws-cli/1.27.74"))
| [
    .eventTime,
    .eventName,
    (.userIdentity.arn // .userIdentity.userName // "-"),
    .sourceIPAddress,
    .eventSource,
    (.requestParameters.bucketName // "-")
  ]
| @tsv
' *.json | sort | column -ts$'\t'

2023-08-26T20:29:37Z  GetCallerIdentity  arn:aws:iam::107513503799:user/temp-user                 84.32.71.19   sts.amazonaws.com  -
2023-08-26T20:35:56Z  ListObjects        arn:aws:iam::107513503799:user/temp-user                 84.32.71.33   s3.amazonaws.com   emergency-data-recovery
2023-08-26T20:54:28Z  AssumeRole         arn:aws:iam::107513503799:user/temp-user                 84.32.71.33   sts.amazonaws.com  -
2023-08-26T20:59:54Z  GetCallerIdentity  arn:aws:sts::107513503799:assumed-role/AdminRole/MySession  84.32.71.36   sts.amazonaws.com  -
2023-08-26T21:17:10Z  ListObjects        arn:aws:sts::107513503799:assumed-role/AdminRole/MySession  84.32.71.125  s3.amazonaws.com   emergency-data-recovery
2023-08-26T21:17:16Z  GetObject          arn:aws:sts::107513503799:assumed-role/AdminRole/MySession  84.32.71.3    s3.amazonaws.com   emergency-data-recovery

┌──(kp㉿kali)-[~/Desktop/ENPM665/lab3/IncidentLogs]
└─$
```

Screenshot shows timestamp, event name, arn, IP and resource used.

```
┌──(kp㉿kali)-[~/Desktop/ENPM665/lab3/IncidentLogs]
└─$ jq '.Records[]? | select(.eventName=="GetCallerIdentity" and (.userIdentity.userName=="temp-user" or (.userIdentity.arn? | test("temp-user"))))' *.json
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAR5CCN4A3X2YWZ37ZI",
    "arn": "arn:aws:iam::107513503799:user/temp-user",
    "accountId": "107513503799",
    "accessKeyId": "AKIAR5CCN4A3WD4RO4P4",
    "userName": "temp-user"
  },
  "eventTime": "2023-08-26T20:29:37Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "GetCallerIdentity",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "84.32.71.19",
  "userAgent": "aws-cli/1.27.74 Python/3.10.6 Linux/5.15.90.1-microsoft-standard-WSL2 botocore/1.29.74",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "3db296ab-c531-4b4a-a468-e1b05ec83246",
  "eventID": "ea6ae4b8-aae8-4fca-a495-2df427bdce46",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "107513503799",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "sts.amazonaws.com"
  }
}
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAR5CCN4A3X2YWZ37ZI",
    "arn": "arn:aws:iam::107513503799:user/temp-user",
    "accountId": "107513503799",
    "accessKeyId": "AKIAR5CCN4A3WD4RO4P4",
    "userName": "temp-user"
  },
  "eventTime": "2023-08-26T20:46:42Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "GetCallerIdentity",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "84.32.71.48",
  "userAgent": "aws-sdk-go-v2/1.3.2",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "672c7327-4d6a-4c6a-890c-c9de10f0d3d9",
  "eventID": "de9fc077-301c-4615-890b-0f2ac27803d9",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "107513503799",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "sts.us-east-1.amazonaws.com"
  }
}
```

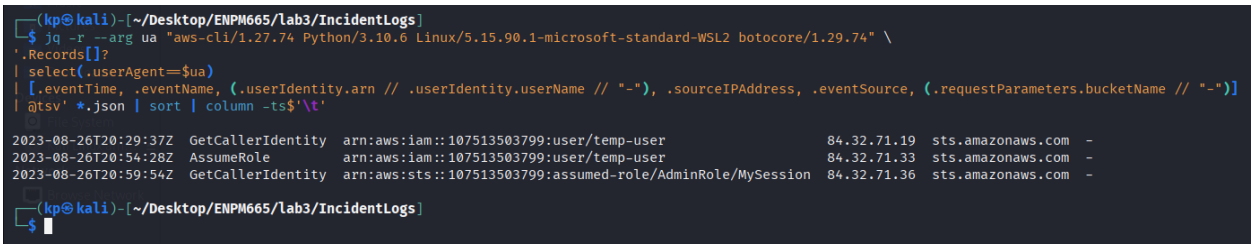
Above screenshot shows temp-user event and potential entry point.

Analysis - The earliest recorded action by **temp-user** is a GetCallerIdentity event at **2023-08-26T20:29:37Z**, suggesting that compromised credentials found on dark web leak were likely used as the initial access point. The following AssumeRole activity indicates that the attacker attempted privilege escalation after verifying the credentials.

Part 2 Privilege Escalation or Role Abuse

Using following command

```
jq -r --arg ua "aws-cli/1.27.74 Python/3.10.6 Linux/5.15.90.1-microsoft-standard-WSL2 botocore/1.29.74" \
'.Records[]?
| select(.userAgent==$ua)
| [.eventTime, .eventName, (.userIdentity.arn // .userIdentity.userName // "-"), .sourceIPAddress, .eventSource, (.requestParameters.bucketName // "-")]
| @tsv' *.json | sort | column -ts$'\t'
```



Screenshot showing privilege escalation sequence

sequence is as followed

1. GetCallerIdentity event by temp-user
2. AssumeRole event by temp-user
3. GetCallerIdentity event by the assumed AdminRole by the original temp-user

GetCallerIdentity event by temp-user immediately assumes AdminRole. Following the AssumeRole, subsequent actions occur under the assume-role / AdminRole /... principal, performing S3 operations. This illustrates the change in identity context from the original user ARN to the assumed-role ARN.

From the above screenshot and explanation, we can see that this is privilege escalation where temp-user assumes the role of AdminRole and then uses the GetCallerIdentity event. This proves that the temp-user assumed higher privilege role and the screenshot proves before and after roles of the attacker.

Part 3 - Resource Access & Data Handling

Using the following command

```
jq -r '
.Records[]?
| select(
 .eventSource=="s3.amazonaws.com"
```

```
  and (.eventName | test("GetObject|ListObjects|ListBucket"))
  and ((.requestParameters.bucketName // "") | test("backup|recovery|data|sensitive|logs"; "i"))
 )
| [
  .eventTime,
  .eventName,
  (.userIdentity.arn // .userIdentity.userName // "-"),
  (.requestParameters.bucketName // "-"),
  (.requestParameters.key // "-"),
  .sourceIPAddress
 ]
| @tsv
' *.json | sort | column -ts$'\t'
```

Screenshot showing ListObjects and GetObject by temp-user and AdminRole

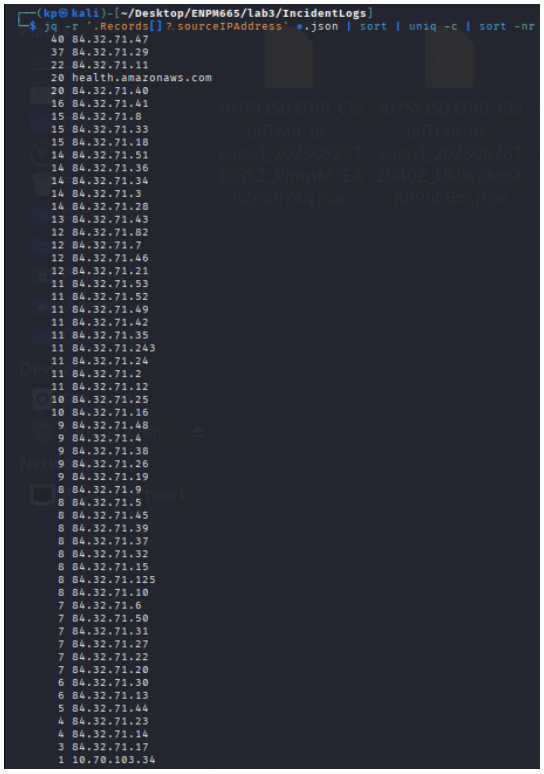
1. S3 activity indicates bucket enumeration and object-level access, such as ListObjects and GetObject. Evidence from **Filedownload.png** shows both GetObject (download) and ListObjects events performed by **temp-user** as well as the **assumed AdminRole** principal.

2. STS activity includes GetCallerIdentity and AssumeRole actions, which were used to verify credentials and switch identity context.
3. IAM (implicit), the AssumeRole action indicates that an IAM role trust relationship or policy was utilized (or misconfigured) to enable privilege escalation.

4. The logs indicate bucket reconnaissance, including ACL/policy inspections and list operations, followed by at least one object retrieval (GetObject) after privilege escalation — consistent with data-access or exfiltration activity. Where possible, each GetObject event should be associated with the responsible principal and source IP.

Part 4 - Indicators of Compromise (IoCs)

1. Attacker User-Agent (Execution Environment)
    - **Observed Value:** aws-cli/1.27.74 Python/3.10.6 Linux/5.15.90.1-microsoft-standard-WSL2 botocore/1.29.74
    - **Reason for Suspicion:**
      This identifies a scripted AWS CLI session running inside WSL2, appearing repeatedly across GetCallerIdentity, AssumeRole, and subsequent S3 activity.
    - Command to Verify: jq -r --arg ua "aws-cli/1.27.74 Python/3.10.6 Linux/5.15.90.1-microsoft-standard-WSL2 botocore/1.29.74" \
    - '.Records[]? | select(.userAgent==$ua) | [.eventTime, .eventName, (.userIdentity.arn // .userIdentity.userName // "-"), .sourceIPAddress] | @tsv' *.json | column -ts$'\t'
2. Malicious Source IPs / Subnet
    - **IPs Observed:**
      Examples include 84.32.71.19, 84.32.71.33, and additional addresses in the 84.32.71.0/24 range.
    - **Reason for Suspicion:**
      These are external, non-corporate IPs performing STS and S3 actions. Multiple principals are mapped to the same or similar suspicious IPs.
    - Command to Verify: jq -r '.Records[]?.sourceIPAddress' *.json | sort | uniq -c | sort –nr
    - Evidence:



3. Unauthorized Privilege Escalation (Role Assumption)
    - **Observed Action:**
      temp-user assumes AdminRole, after which subsequent activity is performed under the assumed-role/AdminRole/... context.

- **Reason for Suspicion:**
  A low-privilege user switching to an admin-level role and performing S3 object operations is a clear indicator of a compromise.
- Command to Verify: jq -r '.Records[]?
  | select(.eventName=="AssumeRole" and (.userIdentity.userName=="temp-user" or (.userIdentity.arn | test("temp-user"))))' *.json | less –R
- Evidence:
  From the screenshot of assume role and privilege escalation sequence.

Part 5 - Recommendations & Reflection

1. **Eliminate long-term root credentials** — remove or rotate any existing programmatic root keys and enforce multi-factor authentication (MFA) for all root account logins.
2. **Apply least-privilege access controls** — tighten AssumeRole permissions by requiring MFA, enforcing source IP conditions, and reviewing trust relationships, especially for administrative roles.
3. **Enhance logging and monitoring** — enable S3 data event logging in CloudTrail and integrate with a centralized SIEM (or AWS GuardDuty/AWS Config) to trigger alerts on sensitive object-level actions like GetObject.
4. **Adopt short-lived credentials** — enforce rotation policies and use temporary STS tokens for both human and service identities; avoid persistent IAM access keys whenever possible.
5. **Detect anomalies proactively** — establish monitoring rules for unusual user-agents (e.g., AWS CLI from WSL2 environments) and external IP addresses performing STS or S3 activity.

Reflection
This incident demonstrates a classic cloud attack chain: initial credential verification (GetCallerIdentity), privilege escalation through role assumption (AssumeRole), and subsequent data access (ListObjects / GetObject). The exercise highlights the importance of enforcing short-lived credentials, strict role assumption policies, and granular object-level logging to effectively prevent and detect such breaches in cloud environments.