

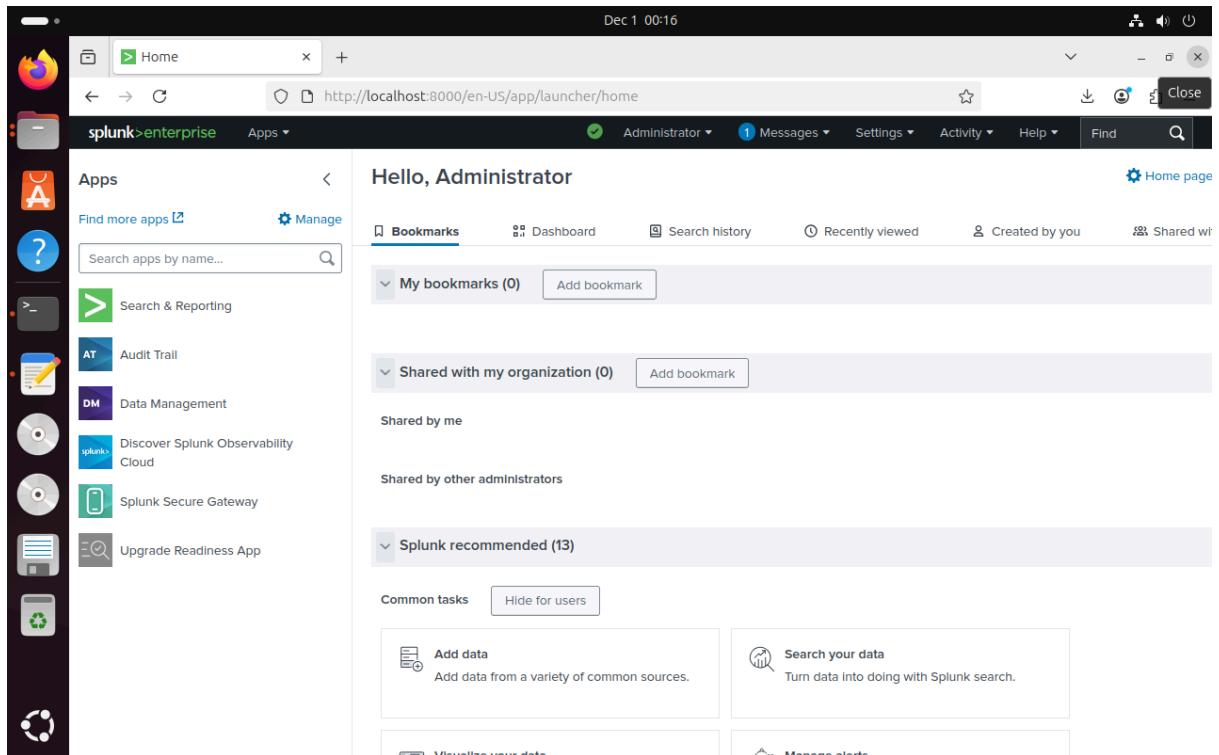
Introduction

This report presents the analysis of the Splunk Boss of the SOC version 3 (BOTSv3) dataset, with an investigation focused on a simulated compromise at Frothly, an organisation with AWS cloud infrastructure and Windows endpoints. Using Splunk Enterprise, I ingested and analysed logs including AWS CloudTrail, S3 access logs, Windows host monitoring, and hardware inventory to answer the set of guided questions and produce this report.

The following will detail the installation of software as well as the loading of the BOTSv3 dataset into Splunk before going over the investigation questions provided on the DLE. This information will be used to surmise an incident narrative, leading to conclusions and recommendations based on my findings.

Installation and Data Preparation

Splunk Enterprise was installed on an Ubuntu virtual machine running in VMware Workstation Pro. The VM was configured with 4 GB RAM and a 70 GB disk to accommodate the large indexed dataset and prevent performance issues during long-running searches.



The official BOTSV3 dataset was downloaded from the Splunk GitHub repository and ingested using the provided `ingest_botsv3.py` script. This created the index `botsv3` containing AWS CloudTrail, S3 access logs, Windows host monitoring (`winhostmon`), hardware inventory, and other relevant sourcetypes.

The screenshot shows the Splunk Enterprise search interface. The search bar contains the query `index="botsv3" earliest=0`. The results section displays 2,083,056 events found before 1/5/2019 7:32:58.000 PM. The Events tab is selected, showing a list of log entries. One entry is expanded to show detailed fields like host, source, sourcetype, bytes, bytes_in, bytes_out, dest_ip, dest_mac, endtime, flow_id, fragment_count, and packet_id. The interface includes a sidebar with various icons and a top navigation bar with tabs like Search, Analytics, Datasets, Reports, Alerts, and Dashboards.

Post-ingestion verification confirmed that the data set's events were searchable so that queries could be done without sampling limitations.

Guided Questions

The following eight questions were answered using targeted Splunk searches.

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

Answer: bstoll, btun, splunk_access, web_admin

The screenshot shows the Splunk Enterprise search interface. The search bar contains the query `index="botsv3" earliest=0 sourcetype="aws:cloudtrail" userIdentity.type="IAMUser" | stats values(userIdentity.userName) as userName | mvexpand userName | sort(userName) | table(userName)`. The results section displays 5,425 events found before 1/6/2019 10:16:39.000 PM. The Statistics tab is selected, showing a table with four rows corresponding to the IAM users bstoll, btun, splunk_access, and web_admin. The interface includes a sidebar with various icons and a top navigation bar with tabs like Search, Analytics, Datasets, Reports, Alerts, and Dashboards.

2. What field would you use to alert that AWS API activity has occurred without MFA (multi-factor authentication)?

Answer: userIdentity.sessionContext.attributes.mfaAuthenticated

Any CloudTrail event can be inspected to reveal this field. When set to "false", it indicates non-MFA-authenticated activity.

The top screenshot shows a 'New Search' page with the search query: `index="botsv3" earliest=0 sourcetype="aws:cloudtrail" *mfa* | head 1 | fieldssummary`. The results table shows 109 events across various date and time dimensions, with the first few rows being:

field	count	distinct_count	is_exact	max	mean	min	numeric_count	stdev	values
apiVersion	0	0	1				0		[]
awsRegion	1	1	1				0		[{"value": "us-west-1", "count": 1}]
date_hour	1	1	1	15	15	15	1	0	[{"value": "15", "count": 1}]
date_mday	1	1	1	20	20	20	1	0	[{"value": "20", "count": 1}]
date_minute	1	1	1	20	20	20	1	0	[{"value": "20", "count": 1}]
date_month	1	1	1				0		[{"value": "august", "count": 1}]
date_second	1	1	1	40	40	40	1	0	[{"value": "40", "count": 1}]
date_wday	1	1	1				0		[{"value": "monday", "count": 1}]
date_year	1	1	1	2018	2018	2018	1	0	[{"value": "2018", "count": 1}]
date_zone	1	1	1	0	0	0	1	0	[{"value": "0", "count": 1}]

The bottom screenshot shows the detailed view of the first event from the search results. The event details pane displays the following JSON structure:

```

{
  "eventName": "DescribeInstanceStatus",
  "eventSource": "ec2.amazonaws.com",
  "eventTime": "2018-08-20T15:15:20Z",
  "eventType": "AwsApiCall",
  "eventVersion": "1.05",
  "recipientAccountId": "622676721278",
  "requestID": "f4bd4e9b-e27c-4a52-93fa-fab7a76d3639",
  "requestParameters": {
    "accountID": "622676721278",
    "arn": "arn:aws:sts::622676721278:assumed-role/AWSServiceRoleForAutoScaling/AutoScaling",
    "invokedBy": "autoscaling.amazonaws.com",
    "principalId": "AROAIOHK7E4SHKYSVYLM:AutoScaling",
    "sessionContext": {
      "attributes": [
        {
          "creationDate": "2018-08-20T15:09:21Z",
          "mfaAuthenticated": false
        }
      ],
      "sessionIssuer": "[+]"
    },
    "type": "AssumedRole"
  }
}

```

3. What is the processor number used on the web servers?

Answer: E5-2676

The screenshot shows the Splunk 10.0.2 interface with a search bar containing the query: `index="botsv3" earliest=0 sourcetype="hardware"`. The results show three events, all from 8/20/18 at 3:26:25.000 PM. The first event is for a host with CPU details: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz, 30720 KB, 2 CPU_CACHE, 2 CPU_COUNT, and 1 HARD_DRIVES. The second event is for a host with CPU details: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz, 30720 KB, 2 CPU_CACHE, 2 CPU_COUNT, and 1 HARD_DRIVES. The third event is for a host with CPU details: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz, 30720 KB, 2 CPU_CACHE, 2 CPU_COUNT, and 1 HARD_DRIVES.

4. Bud accidentally makes an S3 bucket publicly accessible. What is the event ID of the API call that enabled public access?

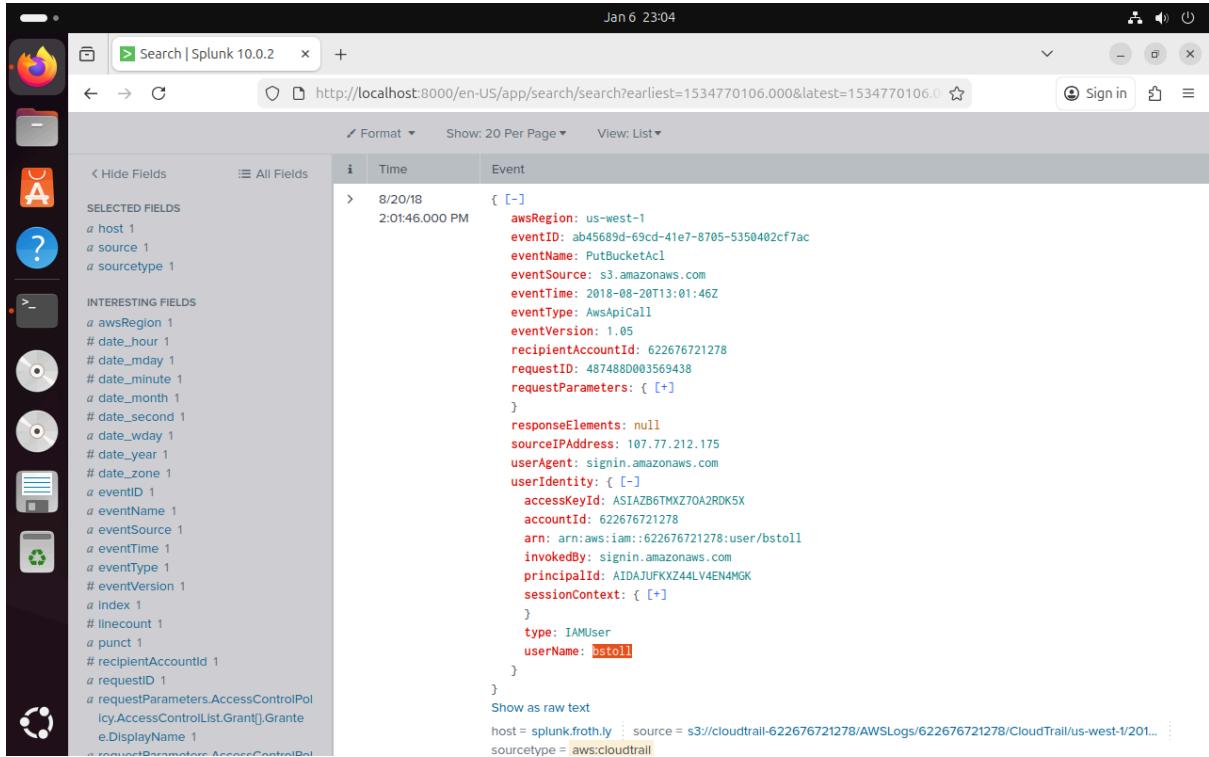
Answer: ab45689d-69cd-41e7-8705-5350402cf7ac

The screenshot shows the Splunk 10.0.2 interface with a search bar containing the query: `index="botsv3" earliest=0 sourcetype="aws:cloudtrail" eventName=PutBucketACL "AllUsers" | sort + _time | table _time eventID userIdentity.username requestParameters.bucketName`. The results show one event from 8/20/18 at 14:01:46. The event ID is highlighted in a red box: `ab45689d-69cd-41e7-8705-5350402cf7ac`. The user identity is `frothlywebcode`.

The earliest event shows the PutBucketAcl call.

5. What is Bud's username

Answer: bstoll



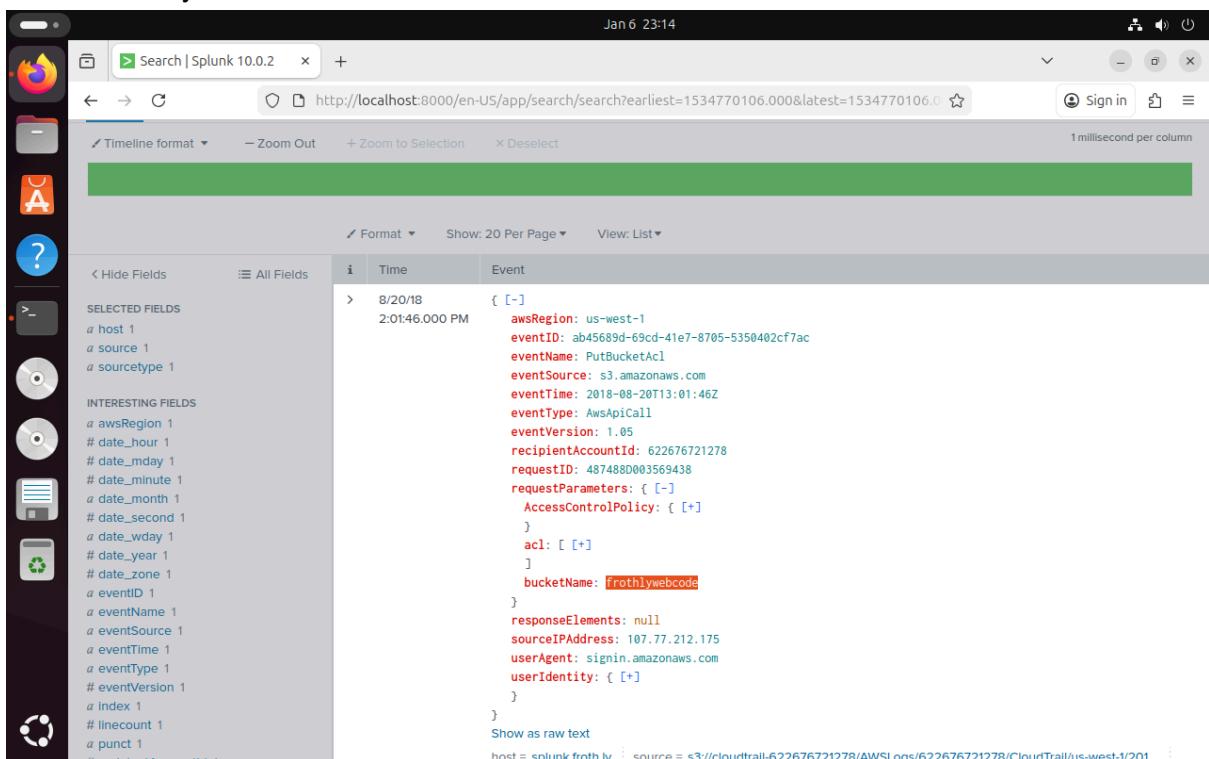
The screenshot shows a Splunk search interface with the URL <http://localhost:8000/en-US/app/search/search?earliest=1534770106.000&latest=1534770106.0>. The search results table has columns for i (index), Time, and Event. One event is listed:

```
i | Time | Event
> 8/20/18 2:01:46.000 PM {
  awsRegion: us-west-1
  eventId: ab45689d-69cd-41e7-8705-5350402cf7ac
  eventName: PutBucketAcl
  eventSource: s3.amazonaws.com
  eventTime: 2018-08-20T13:01:46Z
  eventType: AwsApiCall
  eventVersion: 1.05
  recipientAccountId: 622676721278
  requestID: 487488D003569438
  requestParameters: {
    responseElements: null
    sourceIPAddress: 107.77.212.175
    userAgent: signin.amazonaws.com
    userIdentity: {
      accessKeyId: ASIAZB6TMXZ70A2RDK5X
      accountId: 622676721278
      arn: arn:aws:iam::622676721278:user/bstoll
      invokedBy: signin.amazonaws.com
      principalId: AIDAJUFKXZ44LV4EN4MGK
      sessionContext: {
        type: IAMUser
        userName: bstoll
      }
    }
  }
  Show as raw text
}
```

host = splunk.frothly.ly : source = s3://cloudtrail-622676721278/AWSLogs/622676721278/CloudTrail/us-west-1/201...
sourceType = aws:cloudtrail

6. What is the name of the S3 bucket that was made publicly accessible?

Answer: frothlywebcode



The screenshot shows a Splunk search interface with the URL <http://localhost:8000/en-US/app/search/search?earliest=1534770106.000&latest=1534770106.0>. The search results table has columns for i (index), Time, and Event. One event is listed:

```
i | Time | Event
> 8/20/18 2:01:46.000 PM {
  awsRegion: us-west-1
  eventId: ab45689d-69cd-41e7-8705-5350402cf7ac
  eventName: PutBucketAcl
  eventSource: s3.amazonaws.com
  eventTime: 2018-08-20T13:01:46Z
  eventType: AwsApiCall
  eventVersion: 1.05
  recipientAccountId: 622676721278
  requestID: 487488D003569438
  requestParameters: {
    AccessControlPolicy: [
      acl: [
        {
          bucketName: frothlywebcode
        }
      ]
    ]
    responseElements: null
    sourceIPAddress: 107.77.212.175
    userAgent: signin.amazonaws.com
    userIdentity: {
      accessKeyId: ASIAZB6TMXZ70A2RDK5X
      accountId: 622676721278
      arn: arn:aws:iam::622676721278:user/frothlywebcode
      invokedBy: signin.amazonaws.com
      principalId: AIDAJUFKXZ44LV4EN4MGK
      sessionContext: {
        type: IAMUser
        userName: frothlywebcode
      }
    }
  }
  Show as raw text
}
```

host = splunk.frothly.ly : source = s3://cloudtrail-622676721278/AWSLogs/622676721278/CloudTrail/us-west-1/201...
sourceType = aws:cloudtrail

7. What is the name of the text file that was successfully uploaded into the S3 bucket while it was publicly accessible?

Answer: OPEN_BUCKET_PLEASE_FIX.txt

The screenshot shows the Splunk 10.0.2 interface with a search bar containing the query `index="botsv3" earliest=0 sourcetype="aws:s3:accesslogs" frothlywebcode *.txt*`. The results section displays three events from January 6, 2018, at 23:37. The first event is a REST call to `/OPEN_BUCKET PLEASE FIX.txt` with a host of `a host 1`. The second event is another REST call to the same endpoint with a host of `a host 2`. Both events show the source as `s3://frothlyweblogs/s32018-07-26-01-25-30-F2258C3FF62970B6` and sourcetype as `aws:s3:accesslogs`.

This confirms external exfiltration-style activity while the bucket was public.

8. What is the FQDN of the endpoint that is running a different Windows operating system edition than the others?

Answer: BSTOLL-L.froth.ly

The screenshot shows the Splunk 10.0.2 interface with a search bar containing the query `index="botsv3" earliest=0 sourcetype="winhostmon" |stats values(OS) as OS by host | eventstats count(OS) as os_count by OS |sort os_count |table host OS os_count`. The results section displays 129,679 events from January 6, 2018, at 23:49. A statistics table is shown with columns for host, OS, and os_count. The table lists several hosts with their corresponding OS editions and counts. The host `BSTOLL-L` is listed with `Microsoft Windows 10 Enterprise` and a count of `1`, while other hosts like `ABUNGST-L`, `BGIST-L`, `BTUN-L`, `FYODOR-L`, `JWORTOS-L`, `MKRAEUS-L`, and `PCERF-L` are listed with `Microsoft Windows 10 Pro` and a count of `7`.

host	OS	os_count
BSTOLL-L	Microsoft Windows 10 Enterprise	1
ABUNGST-L	Microsoft Windows 10 Pro	7
BGIST-L	Microsoft Windows 10 Pro	7
BTUN-L	Microsoft Windows 10 Pro	7
FYODOR-L	Microsoft Windows 10 Pro	7
JWORTOS-L	Microsoft Windows 10 Pro	7
MKRAEUS-L	Microsoft Windows 10 Pro	7
PCERF-L	Microsoft Windows 10 Pro	7

The image consists of two vertically stacked screenshots of the Splunk Enterprise web interface. Both screenshots show a search results page with the following details:

- Search Bar:** The search query is `index="botsv3" earliest=0 host="BSTOLL-L" sourcetype="wineventlog:*`. The results are filtered to show 24,427 events (before 1/7/26 12:13:21.000 AM) with No Event Sampling.
- Time Range:** All time
- Event List:** The results are displayed in a table with columns: Time and Event. One event is expanded to show its full details:

8/20/18 4:17:58.000 PM	08/20/2018 03:17:58 AM LogName=Security SourceName=Microsoft Windows security auditing. EventCode=4689 EventType=0 Show all 21 lines
8/20/18 4:17:29.000 PM	08/20/2018 03:17:29 AM LogName=Security
- Selected Fields:** host 1, source 5, sourcetype 5
- Interesting Fields:** Account_Domain 6, Account_Name 11, Application_Name 40
- Event Actions:** A table showing selected fields and their values, along with checkboxes for other event fields. The selected fields are:

Type	Field	Value	Actions
Selected	host	BSTOLL-L	▼
	source	WinEventLog:Security	▼
	sourcetype	WinEventLog:Security	▼
Event	Account_Domain	AzureAD	▼
	Account_Name	BudStoll	▼
	ComputerName	BSTOLL-L.frothly	▼
	EventCode	4689	▼
	EventType	0	▼
	Exit_Status	0x0	▼
	Keywords	Audit Success	▼
	LogName	Security	▼
	Logon_ID	0x4B339	▼
	Message	A process has exited. Subject: Security ID: AzureAD\BudSt	▼

Incident Narrative

Questions 4–7 form a timeline of an insider mistake leading to data exposure. User bstoll (Bud) executed a PutBucketAcl API call granting "AllUsers" read/write access to the frothlywebcode bucket. Shortly after, an external entity uploaded OPEN_BUCKET PLEASE_FIX.txt, demonstrating successful exploitation. The fault lies in a lack of least-privilege bucket policies and non-continuous monitoring of CloudTrail and S3 access logs.

SOC and Incident Handling Reflection

In a Security Operations Centre (SOC), incident detection and response typically follow a tiered analyst structure to ensure efficient handling of alerts at scale.

Tier 1 analysts serve as the first line of defence, monitoring dashboards and SIEM tools like Splunk for initial alerts. In the Frothly incident, they would detect anomalous CloudTrail events such as the PutBucketAcl API call granting public access, or unusual S3 access log entries indicating external uploads. Their role involves: validating the alert, picking out contextual data (e.g., user details from IAM), and escalating confirmed incidents to higher tiers while documenting initial findings.

Tier 2 analysts would conduct deeper investigation, performing the type of correlation demonstrated in this analysis, linking the configuration change by user bstoll to the subsequent unauthorised upload of OPEN_BUCKET_PLEASE_FIX.txt. Then advanced Splunk searches would be carried out across sourcetypes (CloudTrail, S3 access logs, endpoint logs) and reconstruct timelines.

Tier 3 analysts would focus on root cause analysis, identifying whether the misconfiguration resulted from human error, lack of training or a potential insider threat and recommend long-term remediation. This tiered approach aims to meet common practices of a Computer Security Incident Response Team (CSIRT). Structured escalation ensures rapid containment while preserving evidence for forensic review. The use of Splunk as a unified platform for log ingestion, search, and correlation are key components of a SOC monitoring infrastructure.

Conclusion

This analysis of the BOTSV3 dataset has successfully uncovered a security incident involving the unintentional exposure of an S3 bucket (frothlywebcode) to public access, triggered by a PutBucketAcl API call from the IAM user bstoll. This misconfiguration enabled an external actor to upload the file OPEN_BUCKET_PLEASE_FIX.txt shortly afterward, demonstrating clear exploitation and a potential data exfiltration risk.

Key lessons from this investigation include the dangers of insufficient access controls in AWS services and the importance of proactive monitoring of API calls. When not properly managed, even seemingly minor actions by authorised users can lead to significant exposure. Strict policies and automated safeguards can help alleviate this risk. Centralised logging would have given the ability to correlate events across CloudTrail and S3 access logs, improving situational awareness.

For Frothly's Security Operations Centre (SOC) strategy, this case should draw attention to the importance of layered detection mechanisms and rapid response capabilities. Implementing real-time alerting on high-risk API actions would enable Tier 1 analysts to flag anomalies early, while tools like Splunk support Tier 2/3 investigations. For response, adopting automated playbooks would reduce mean time to remediate (MTTR), such as immediate notifications to administrators upon detection of public bucket changes.

To prevent recurrence and strengthen overall resilience, Frothly should adopt the following recommendations:

- Enforce multi-factor authentication (MFA) for all IAM users to mitigate risks from compromised credentials.
- Enable S3 Block Public Access at the account level to prevent buckets from being made publicly accessible, regardless of individual settings.
- Configure CloudTrail-based alerts for sensitive events such as PutBucketAcl and PutBucketPolicy to ensure immediate visibility.
- Conduct regular automated audits of bucket permissions using tools like AWS Config or third-party scanners.
- Deploy automated solutions, such as AWS Lambda functions triggered by CloudWatch Events, to revert unauthorised public access configurations instantly.
- Conduct employee training in order to lower the risks of breaches that come as a result of human error.

By addressing these areas, Frothly can significantly enhance its incident detection and response abilities, strengthening their SOC operation.