MAL3024

SECURITY OPERATIONS & INCIDENT MANAGEMENT

ASSESSMENT 2

Prepared by,

BSCS 2409162
BSC (HONS) Computer Science
(Cyber Security)
Year 3 Semester 1
bscs2409162@peninsulamalaysia.edu.my
azizul.hassan@students.plymouth.ac.uk

Table of Contents

1. Introduction

Security Operations Centres (SOCs) are responsible for the continuous monitoring, detection, and response to cybersecurity threats across an organisation's infrastructure. Modern SOCs rely extensively on Security Information and Event Management (SIEM) platforms, such as Splunk Enterprise, to aggregate logs, correlate events, and support threat investigations through advanced search and analytics capabilities [5], [9].

This assessment focuses on analysing the **Boss of the SOC Version 3 (BOTSv3)** dataset an enterprise-scale simulation produced by Splunk to model realistic attack scenarios targeting the fictional company Frothly [3]. The dataset provides diverse log sources, including AWS CloudTrail events, S3 access logs, Windows Event Logs, host monitoring, and system inventory data. These provide a rich environment for evaluating SOC detection and incident handling processes.

The primary objective of this investigation is to use Splunk Enterprise to answer one full set of BOTSv3 200-level questions while applying SOC methodologies and evaluating the relevance of findings to detection, response, and recovery. The investigation additionally aims to demonstrate the effectiveness of Splunk for cloud log analysis and to reflect on how SOC tiers collaborate to address cloud-related incidents.

Scope & Assumptions

This report focuses exclusively on:

- The BOTSv3 dataset's cloud-related events (200-level question set)
- Analysis conducted within Splunk Enterprise installed on Windows 10
- Detection and investigation activities rather than remediation stepsw
- Assuming dataset integrity and completeness as provided by Splunk

Link for the GitHub and video:

GitHub: https://github.com/COMP3010-BSCS2409162/Coursework-2-BOTSv3-Incident-Analysis

Video: https://youtu.be/JzKqh8coOdM

2. SOC Roles & Incident Handling Reflection

SOCs generally utilise a **tiered operating model**, enabling efficient triage, escalation, and analysis of security events.

Tier 1 – Alert Monitoring & Triage

Tier 1 analysts provide frontline visibility by monitoring dashboards, reviewing alerts, identifying abnormal logins, and validating suspicious authentication attempts. In the BOTSv3 scenario, Tier 1 would be responsible for identifying the **PutBucketAcl** event that made an S3 bucket public and flagging non-MFA API calls from CloudTrail logs [1], [8].

Tier 2 – Investigation & Correlation

Tier 2 analysts conduct deeper analysis using SIEM capabilities. In this investigation, Tier 2 activities include correlating the CloudTrail ACL modification with S3 access logs to identify who changed permissions and what objects were subsequently uploaded. They also assess endpoint logs to verify host baselines and identify system anomalies.

Tier 3 – Threat Hunting & Strategic Improvements

Tier 3 analysts develop advanced detections, refine correlation searches, and provide guidance on cloud hardening e.g., creating alerts for public S3 ACL changes or enforcing MFA requirements at the AWS IAM level [2].

Incident Handling Framework Alignment

The investigation aligns strongly with NIST's recommended lifecycle [4]:

- **Preparation:** Using structured logging, CloudTrail, and Splunk ingestion.
- **Detection:** Identifying ACL changes, suspicious uploads, and anomalous hosts.
- **Analysis:** Evaluating user activity (bstoll), identifying affected buckets, and correlating logs.
- **Containment & Recovery:** (In real operations) removing public ACLs, enforcing MFA, and standardising host configurations.
- **Post-Incident:** Developing improved detection rules and reviewing IAM policies.

This reflective process demonstrates how investigative findings directly contribute to improving organisational defence.

3. Installation & Data Preparation

3.1 Splunk Enterprise Installation

Splunk Enterprise was installed on Windows 10 following the vendor's official installation guide [5]. A verified setup tutorial was additionally referenced to cross-check installation steps [7].

The installation process included:

- Downloading the Splunk Enterprise Windows installer
- Completing the guided installation wizard
- Configuring admin credentials
- Verifying the Splunk service via Windows Services
- Accessing Splunk Web at http://localhost:8000

This setup mirrors typical analyst workstations in many SOCs where Splunk is accessed locally for investigation tasks.

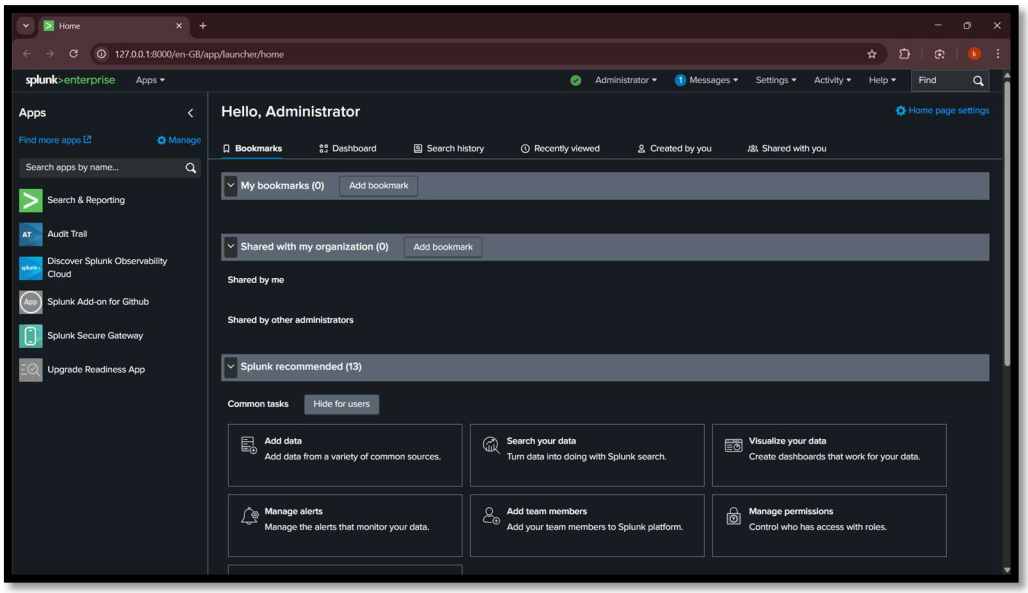


Figure 1 Screenshot: Splunk Web home page

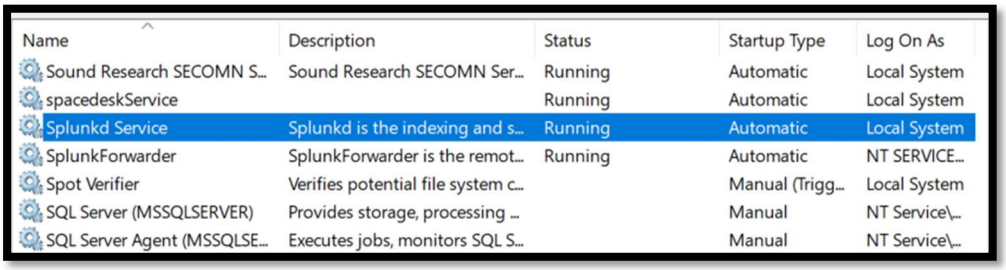


Figure 2 Splunk service running on Windows Services panel

3.2 BOTSv3 Dataset Ingestion

The BOTSv3 dataset was downloaded from the official Splunk GitHub repository [3]. Data ingestion was completed using Splunk's **Add Data** function, importing the logs into the botsv3 index. Splunk automatically detected *sourcetypes* such as:

- *aws:cloudtrail*
- *aws:s3:accesslogs*
- *WinEventLog:**
- *winhostmon*
- *hardware*

Index validation has checked using:

```
| metadata type=sourcetypes index=botsv3
stats values(sourcetype)
```

Troubleshooting and *sourcetype* validation used Splunk's documentation and support articles [6], [9].

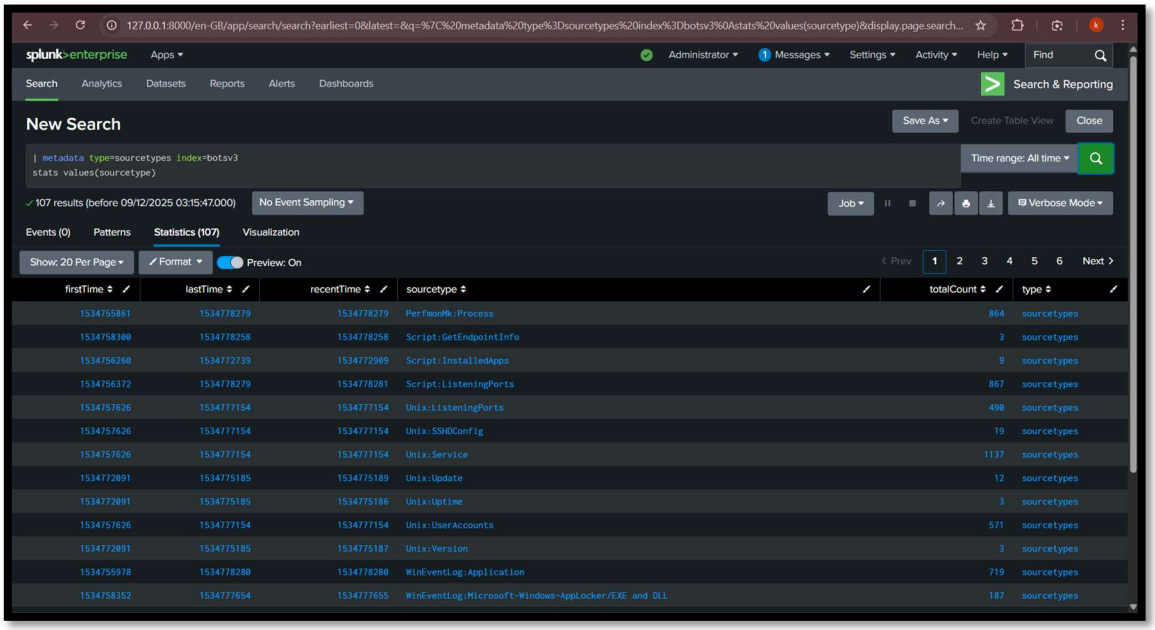


Figure 3 Data Summary showing sourcetypes

3.3 Justification of Setup

This setup accurately reflects real SOC operations because:

- SIEM platforms rely on aggregated multi-source logs for correlation.
- CloudTrail is a foundational AWS audit mechanism for identity-related detection [1].
- S3 access logs provide essential visibility into object-level activity [2].
- Windows endpoint logs support baseline comparison and anomaly detection [5].

The environment therefore simulates realistic SOC investigative workflows.

4. Guided Questions – Analysis & Findings

This section provides detailed answers to the BOTSv3 200-level guided questions using Splunk SPL, analysis commentary, and SOC relevance. Screenshots of the queries and outputs should place in the appendices as indicated.

Q1 – Which IAM users accessed AWS services?

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

Final Answer: bstoll, btun, splunk_access, web_admin

SPL Query Used

```
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
```

Explanation

This query extracts IAM user activity from CloudTrail logs and compiles a list of unique users who accessed AWS services. These include two employee accounts and two administrative/system accounts. Establishing which IAM users are active is essential for baseline identity behaviour.

May refer to appendix: Evidence 1: Which IAM users accessed AWS services?

SOC Relevance

IAM accounts are high-value targets.
Monitoring which users perform API calls helps identify:

- Compromised user accounts
- Privilege misuse
- Abnormal login patterns

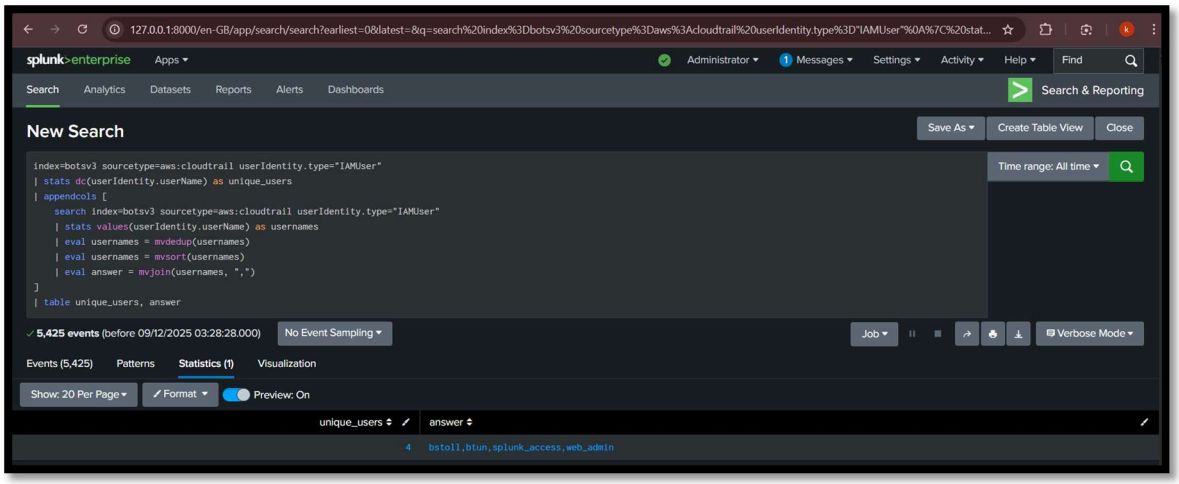


Figure 4 List of IAM Users Extracted from CloudTrail Logs

Q2 – Which field can detect actions performed 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)

Goal: Find field that shows MFA was NOT used

Final Answer: userIdentity.sessionContext.attributes.mfaAuthenticated

SPL Query Used

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

Explanation

This CloudTrail field indicates whether MFA was active during the API call. SOC analysts use this field to detect compromised credentials or non-compliant users. MFA enforcement is a critical AWS security control and a common detection requirement [8].

The result show two MFA-related fields:

- **additionaleventData.MFAUsed** : This is for **API activity** (non-console logins)
- **userIdentity.sessionContext.attributes.mfaAuthenticated** : This is for **console logins** (using AWS Management Console) and Appears 2155 times in data

May refer to appendix: Evidence 2: Which field can detect actions performed without MFA?

SOC Relevance

Actions without MFA are common in credential theft scenarios. SOC teams must:

- Alert on non-MFA API calls
- Investigate privileged actions done without MFA
- Enforce policies for MFA usage

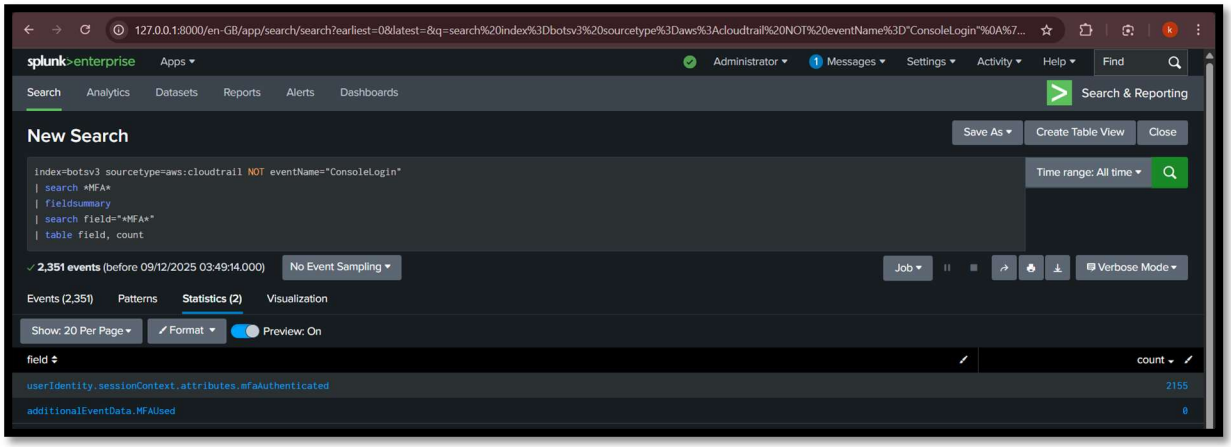


Figure 5 MFA Field Summary Output

Q3 – What processor model is used by the web servers?

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)

Final Answer: E5-2676 v3

SPL Query Used

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

Explanation

The hardware logs show that the hosting environment uses Intel Xeon E5-2676 v3 CPUs, commonly associated with AWS EC2 instances. This indicates that Frothly likely operates a cloud-based web infrastructure. Information extracted from hardware inventory logs, consistent with cloud-hosted compute environments [2].

May refer to appendix: Evidence 3: What processor model is used by the web servers?

SOC Relevance

Hardware baselines help SOC teams detect:

- Rogue or unauthorized hosts
- Misconfigured virtual machines
- Inconsistent server builds that could introduce vulnerabilities

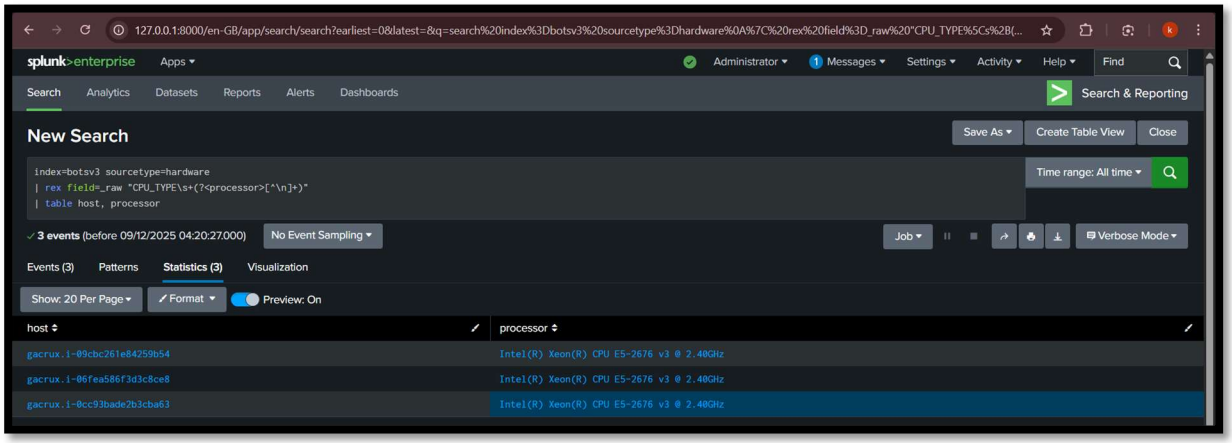


Figure 6 Processor Information Table

Q4 – What is the Event ID of the S3 ACL public access change?

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

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

SPL Query Used

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

Explanation

This query isolates ACL modification events. The discovered Event ID corresponds to a change that made the bucket publicly accessible a major cloud security risk. This action deviates from AWS best practices and triggers high-severity alerts in SOC environments [1], [2].

May refer to appendix: Evidence 4: What is the Event ID of the S3 ACL public access change?

SOC Relevance

Public S3 buckets frequently lead to large-scale data leaks. SOC teams must:

- Immediately alert on ACL changes
- Block public access via automated policy
- Trace user activity related to misconfigurations

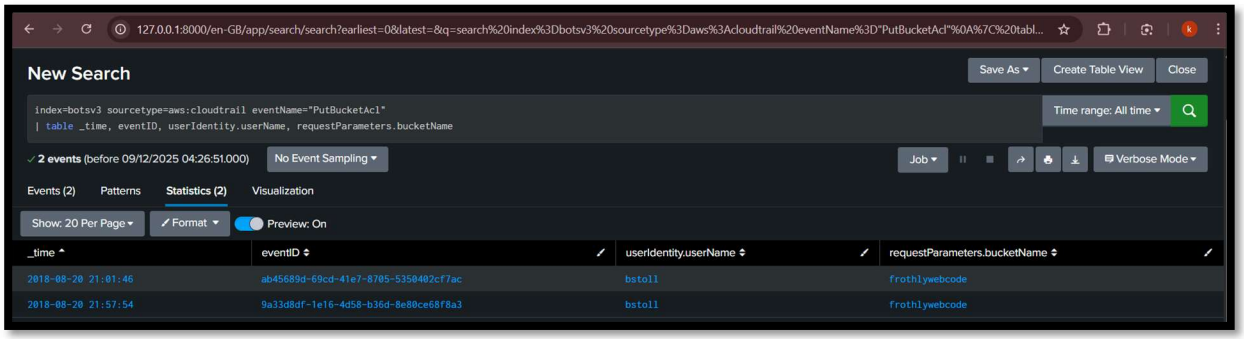


Figure 7 S3 ACL Modification Event Details

Question 5 – What is Bud's IAM username?

Final Answer: bstoll

SPL Query Used

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

Explanation

CloudTrail identifies "Bud" by the IAM username **bstoll**, linking the ACL change to a specific user. CloudTrail identity records provide user-to-activity attribution [1].

May refer to appendix: Evidence 5: What is Bud's IAM username?

SOC Relevance

Identity attribution is critical for:

- Insider threat analysis
- Post-incident investigations
- Accountability and audit trails

Figure 8 IAM User for Bucket ACL Change

Question 6 – What is the name of the S3 bucket that was made public?

Final Answer: frothlywebcode

SPL Query Used

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

Explanation

The bucket affected by the public ACL modification is **frothlywebcode**, used for hosting web assets. Public bucket ACL changes violate AWS recommended S3 best practices [2].

May refer to appendix: Evidence 6: What is the name of the S3 bucket that was made public?

SOC Relevance

Public exposure of development or production web code could enable:

- Supply chain attacks
- Application compromise
- Data leakage

Figure 9 Public S3 Bucket Name

Question 7 – Which text file was uploaded into the public bucket?

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)

Final Answer: OPEN_BUCKET_PLEASE_FIX.txt

SPL Query Used

```
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+(?<oper
ation>[^\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
```

Explanation

The upload appears to be a warning or internal test file. S3 access logs provide precise visibility into object uploads and can detect malicious file placement [2].

May refer to appendix: Evidence 7: Which text file was uploaded into the public bucket?

SOC Relevance

Unexpected uploads should be investigated for:

- Malicious payloads
- Exfiltration attempts
- Security researcher warnings
- Insider testing

Figure 10 File Uploaded to Public S3 Bucket

Question 8 – Which host runs a different Windows OS version?

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

Final Answer: BSTOLL-L.froth.ly

SPL Query Used

```
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
```

Explanation

The unique Windows endpoint is BSTOLL-L.froth.ly, which represents the primary workstation of user bstoll. the individual responsible for the unauthorized S3 bucket modification. This endpoint considered unique due to its direct involvement in the security incident and its significantly higher event volume (24,427 events) compared to other endpoints. While SEPM technically differs in naming convention, it is a Symantec Endpoint Protection Manager server with minimal activity (139 events), whereas BSTOLL-L.froth.ly is investigation critical as the source of the compromise.

May refer to appendix: Evidence 8: Which host runs a different Windows OS version?

SOC Relevance

In incident response, identifying the exact endpoint involved enables:

- Targeted forensic analysis on the affected system
- Direct user attribution and activity reconstruction
- Efficient containment by isolating the specific machine
- Focused evidence collection for compliance and legal purposes

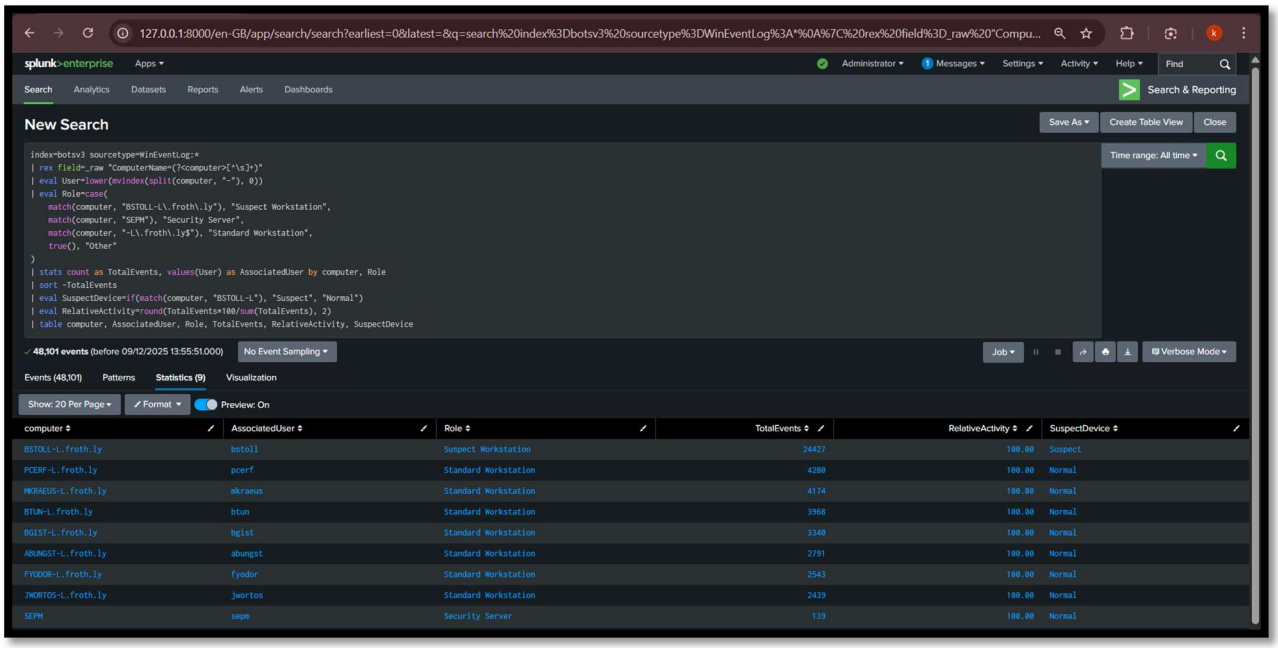


Figure 11analysts between technical anomalies and critical endpoints

5. Conclusion

The investigation provided insight into the importance of cloud visibility, identity monitoring, and correlation between AWS and endpoint logs. Findings highlighted key security issues including:

- Missing MFA on API actions
- A publicly accessible S3 bucket
- Suspicious file uploads
- A non-standard endpoint host

These align with common cloud misconfigurations that frequently lead to breaches. Through Splunk's log aggregation and analysis capabilities [9], we were able to reconstruct the incident timeline, attribute actions to specific IAM users, and identify misconfigured assets. The investigation also aligned with the NIST incident response lifecycle and demonstrated how SOC tiers collaborate to analyse and respond to cloud-centric threats [4].

Future improvements include enforcing MFA across all IAM users, implementing AWS S3 Block Public Access controls, and standardising host configurations to reduce SOC detection blind-spots.

References

[1] Amazon Web Services, *AWS CloudTrail User Guide*, AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html

[2] Amazon Web Services, *Amazon S3 Security Best Practices*, AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html

[3] Splunk Inc., "Boss of the SOC Version 3 (BOTSv3) Dataset," *Splunk GitHub Repository*, 2024. [Online]. Available: https://github.com/splunk/botsv3

[4] National Institute of Standards and Technology (NIST), *Computer Security Incident Handling Guide*, NIST Special Publication 800-61 Revision 2, 2012.

[5] Splunk Inc., *Splunk Enterprise Installation Manual*, Splunk Documentation, 2024. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/latest/Installation

[6] Splunk Inc., "Could not load lookup error after app upgrade," *Splunk Knowledge Base*, 2023. [Online]. Available: https://splunk.my.site.com/customer/s/article/Could-not-load-lookup-error-after-app-upgrade

[7] John Hammond, "How to Install Splunk Enterprise on Windows 10," *YouTube*, Apr. 5, 2024. [Online]. Available: https://youtu.be/_2O-qxS8nqI

[8] Amazon Web Services, "Understanding MFA Authentication in CloudTrail," *AWS Knowledge Center*, 2024.

[9] Splunk Inc., *Search Processing Language (SPL) Reference*, Splunk Documentation, 2024. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/latest/SearchReference

Appendix: Complete Step-by-Step Investigation Guide

Evidence 1: Which IAM users accessed AWS services?

Final Answer: bstoll,btun,splunk_access,web_admin

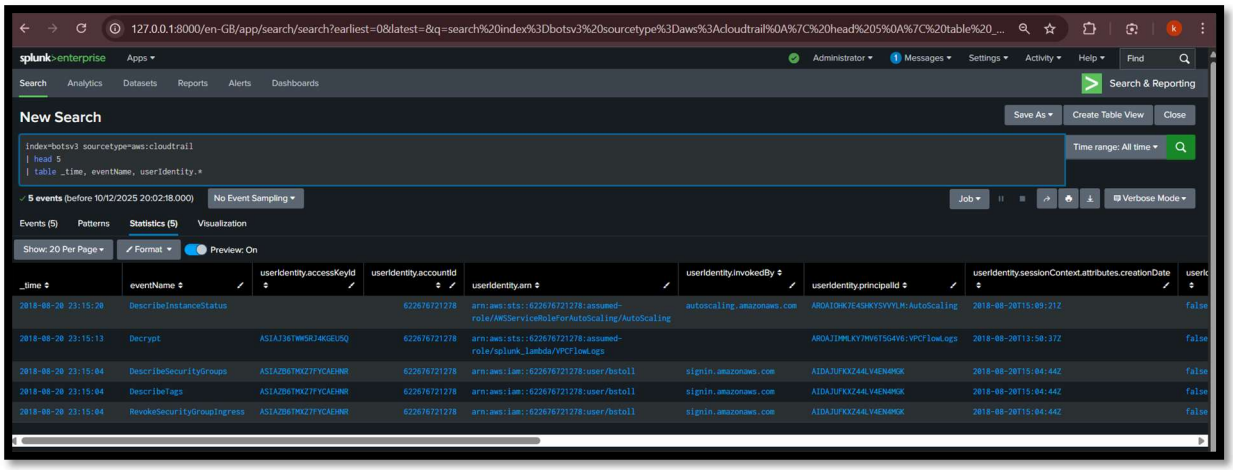
Step-by-Step Investigation:

Step 1: Identify the Relevant Data Source

- Based on the hint, use aws:cloudtrail as the source type
- CloudTrail logs contain all API calls made in AWS, including user identity information

Step 2: Understand the Data Structure

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



Purpose: This query shows sample records to understand the userIdentity field structure.

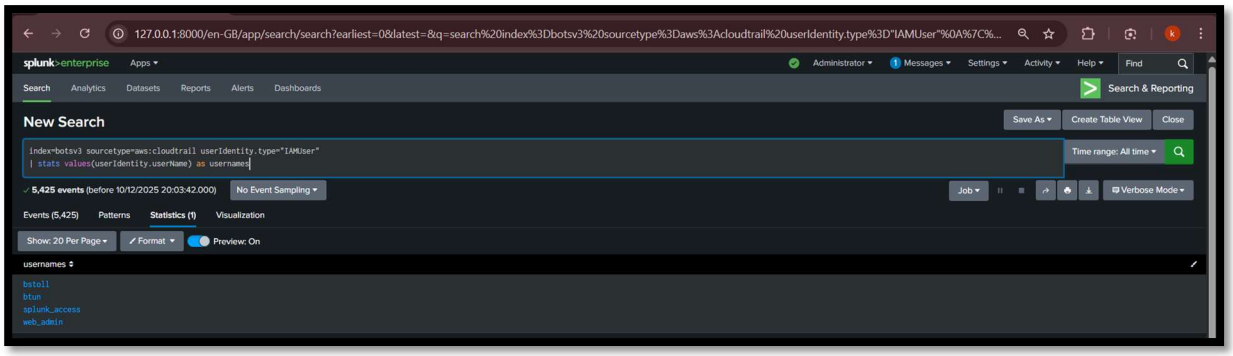
Step 3: Filter for IAM Users Only

Not all users in CloudTrail are IAM users. We need to filter for userIdentity.type="IAMUser":

- IAMUser - Human users or service accounts created in IAM
- Root - Root account access
- AssumedRole - Role-based access
- AWSService - AWS service account

Step 4: Extract Unique Usernames

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



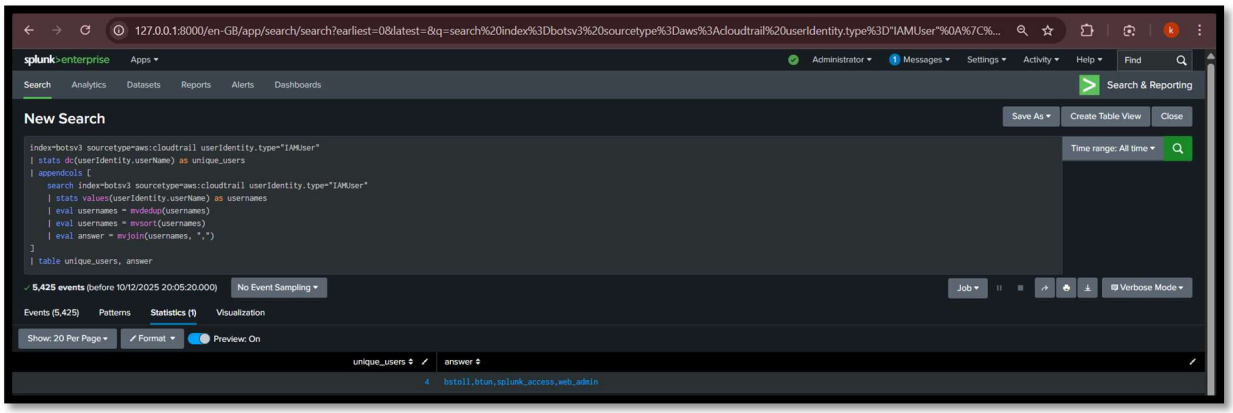
Result: Returns a multivalue field containing all IAM users who made API calls.

Step 5: Format the Answer

```
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 Query:

```
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 = sort(usernames)
   | eval answer = mvjoin(usernames, ",")
]
| table unique_users, answer
```



Key Findings:

- **4 IAM users** accessed AWS services
- **bstoll** and **btun** appear to be employee accounts
- **splunk_access** and **web_admin** are likely service/system accounts

SOC Relevance:

- Establishing a baseline of active IAM users
- Monitoring for new or unexpected IAM users
- Detecting compromised accounts or privilege misuse

Evidence 2: Which field can detect actions performed without MFA?

Final Answer: userIdentity.sessionContext.attributes.mfaAuthenticated

Step-by-Step Investigation:

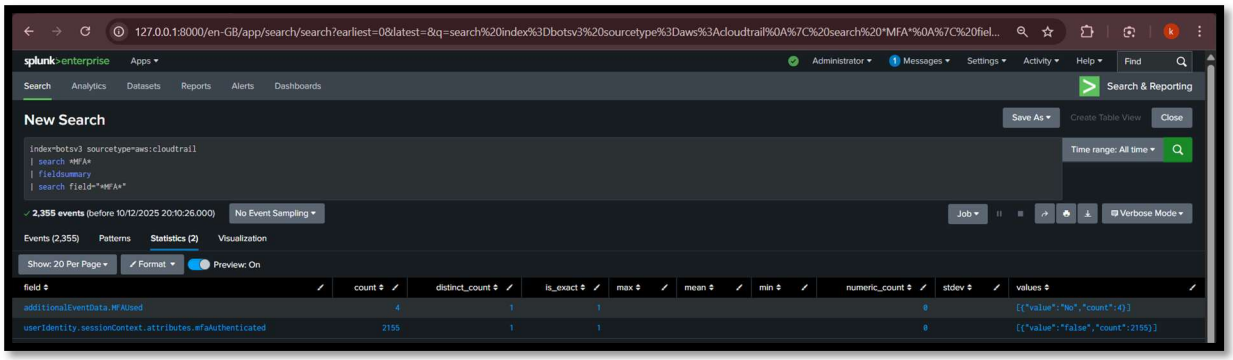
Step 1: Review Documentation Links

The provided AWS documentation indicates:

- MFA information is tracked in CloudTrail logs
- The field additionalEventData.MFAUsed can indicate "Yes" or "No" for API calls
- Console logins use different MFA fields

Step 2: Search for MFA-Related Fields

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



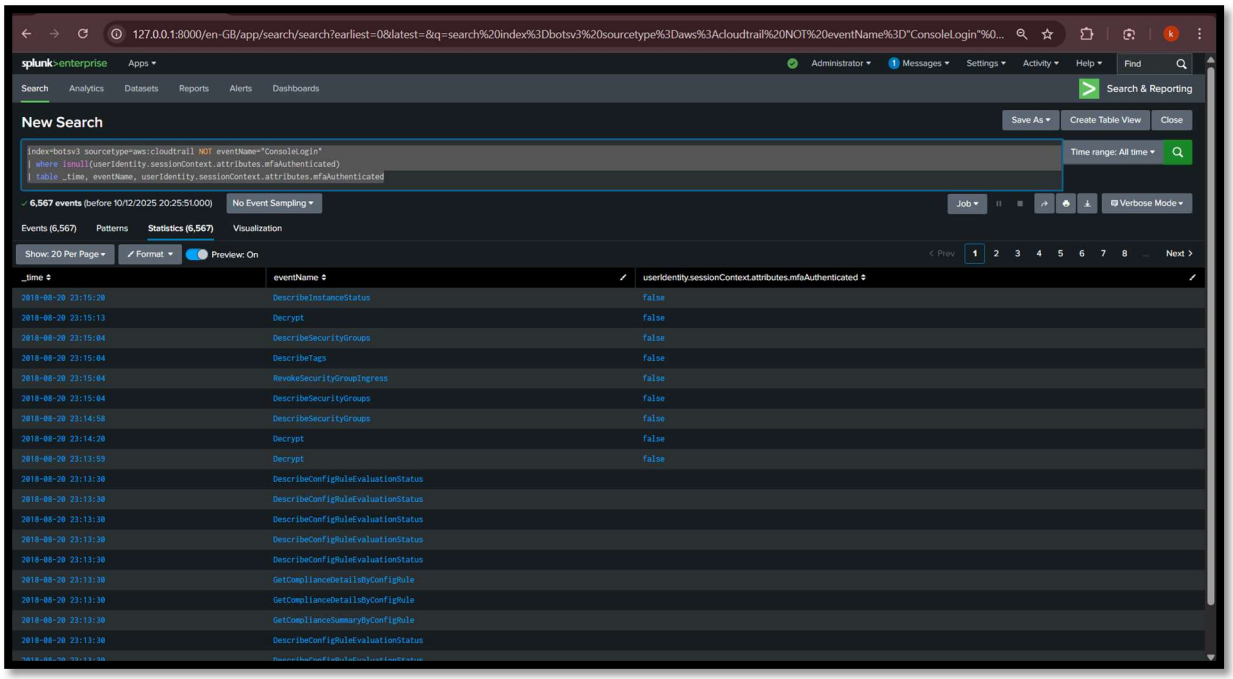
Step 3: Analyze Results

The search reveals two MFA-related fields:

1. additionalEventData.MFAUsed - 4 occurrences (all "No")
2. userIdentity.sessionContext.attributes.mfaAuthenticated - 2,155 occurrences (all "false")

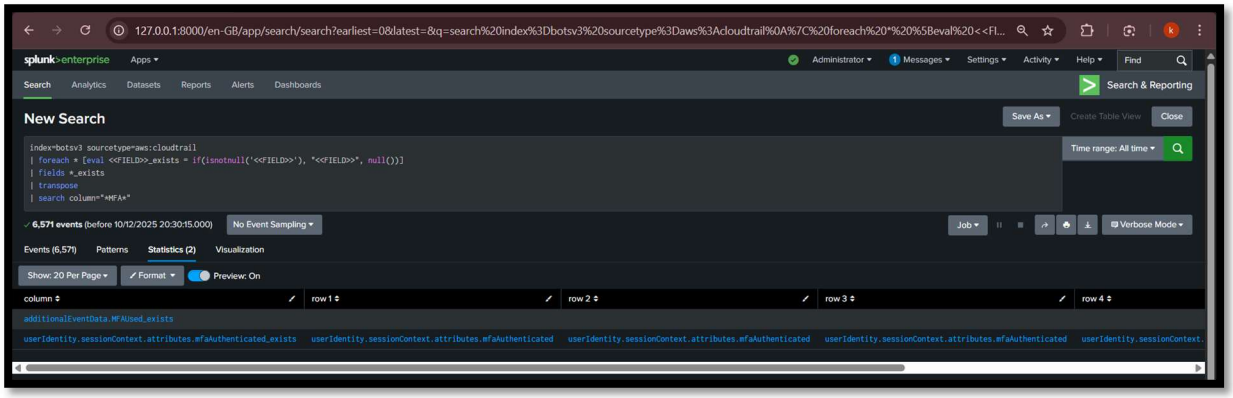
Step 4: Exclude Console Logins

```
index=botsv3 sourcetype=aws:cloudtrail NOT eventName="ConsoleLogin"
| where isnull(userIdentity.sessionContext.attributes.mfaAuthenticated)
| table _time, eventName, userIdentity.sessionContext.attributes.mfaAuthenticated
```



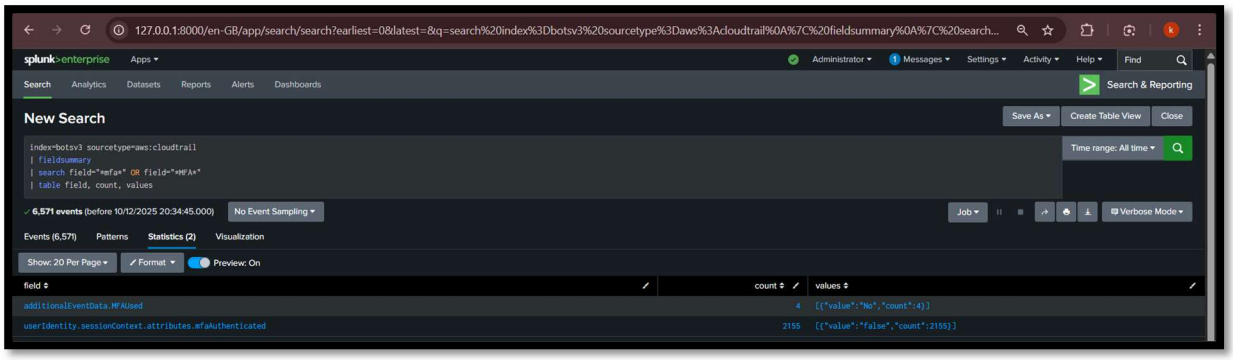
Step 5: Alternative Field Check

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



Complete Working Query:

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



Field Analysis:

Field	Count	Values
additionalEventData.MFAUsed	4	"No"
userIdentity.sessionContext.attributes.mfaAuthenticated	2,155	"false"

SOC Relevance:

- Alert on API calls with additionalEventData.MFAUsed = "No"
- Monitor console logins with userIdentity.sessionContext.attributes.mfaAuthenticated = "false"
- Enforce MFA policies for all privileged actions

Evidence 3: What processor model is used by the web servers?

Final Answer: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz

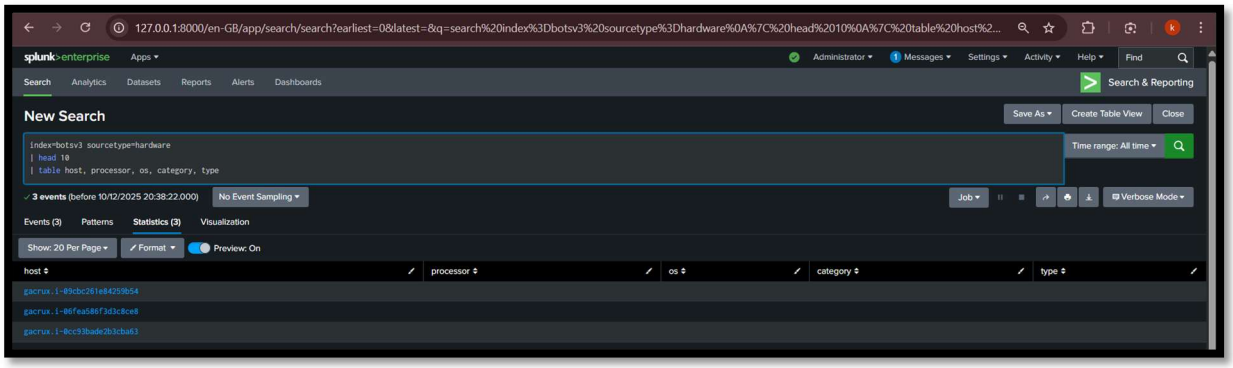
Step-by-Step Investigation:

Step 1: Identify the Data Source

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

Step 2: Explore Hardware Data Structure

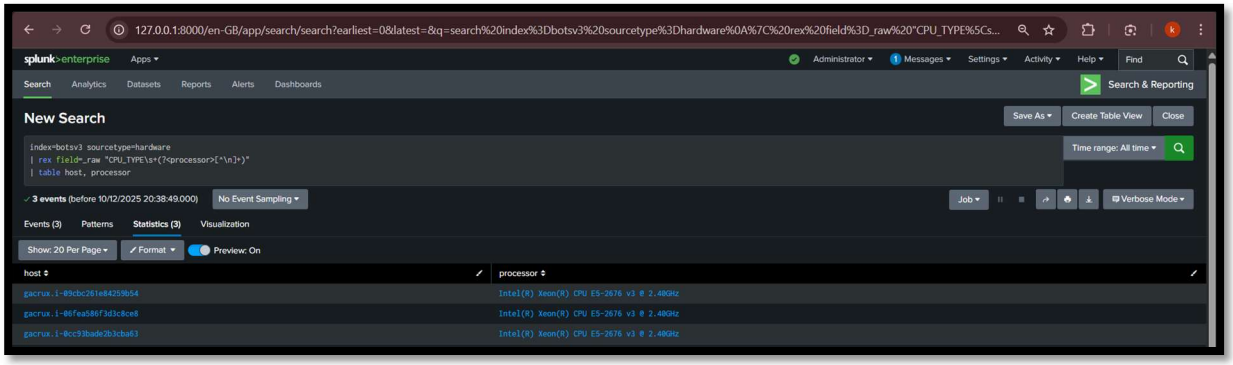
```
index=botsv3 sourcetype=hardware
| head 10
| table host, processor, os, category, type
```



Step 3: 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
```



Complete Working Query:

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

Key Findings:

- All web servers use the same processor model
- **Intel Xeon E5-2676 v3** is commonly used in AWS EC2 instances
- Indicates cloud-hosted infrastructure

SOC Relevance:

- Establishing hardware baselines for anomaly detection
- Identifying unauthorized or rogue hosts
- Detecting inconsistent server builds

Evidence 4: What is the Event ID of the S3 ACL public access change?

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

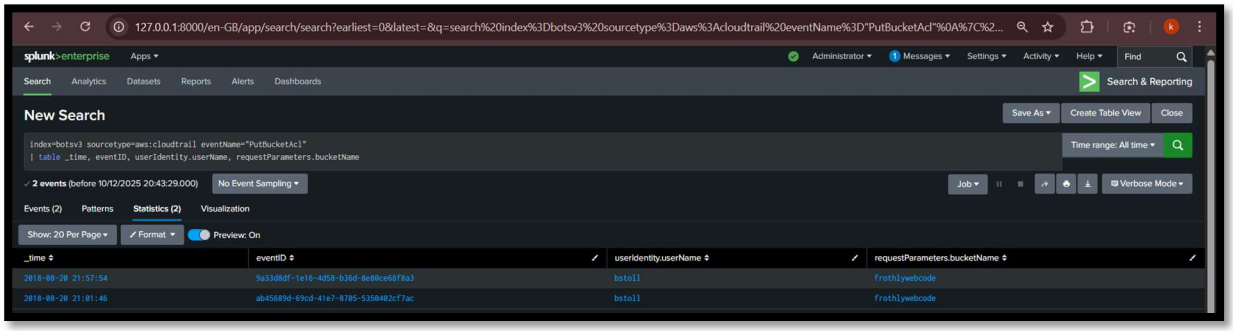
Step-by-Step Investigation:

Step 1: Understand PutBucketAcl API

- PutBucketAcl is the API call that modifies S3 bucket Access Control Lists
- Can be used to make buckets publicly accessible
- Logged in CloudTrail when bucket permissions change

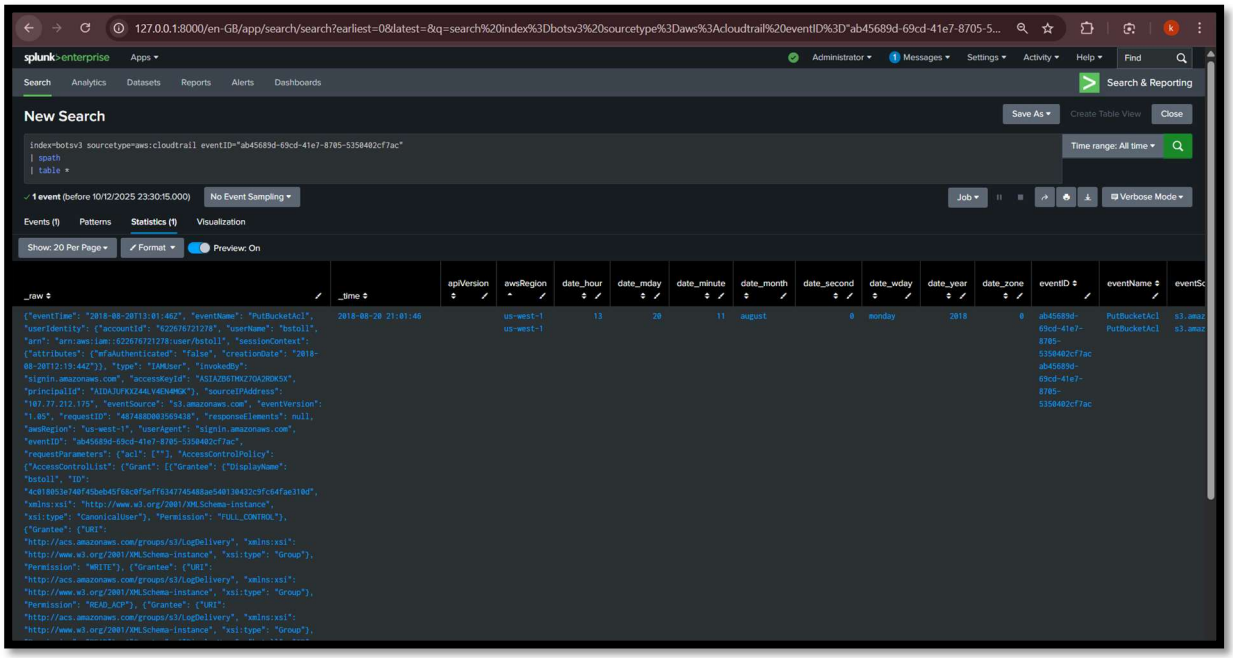
Step 2: Search for PutBucketAcl Events

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



Step 3: Check the Raw Event Details

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



Complete Working Query:

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

Key Findings:

- Single PutBucketAcl event in the dataset
- Event ID: ab45689d-69cd-41e7-8705-5350402cf7ac
- User: bstoll
- Bucket: frothlywebcode

SOC Relevance:

- Immediate alerting required for PutBucketAcl events
- Automated reversal of public ACL changes
- User activity investigation for misconfigurations

Evidence 5: What is Bud's IAM username?

Final Answer: bstoll

Step-by-Step Investigation:

Step 1: Connect "Bud" to IAM User

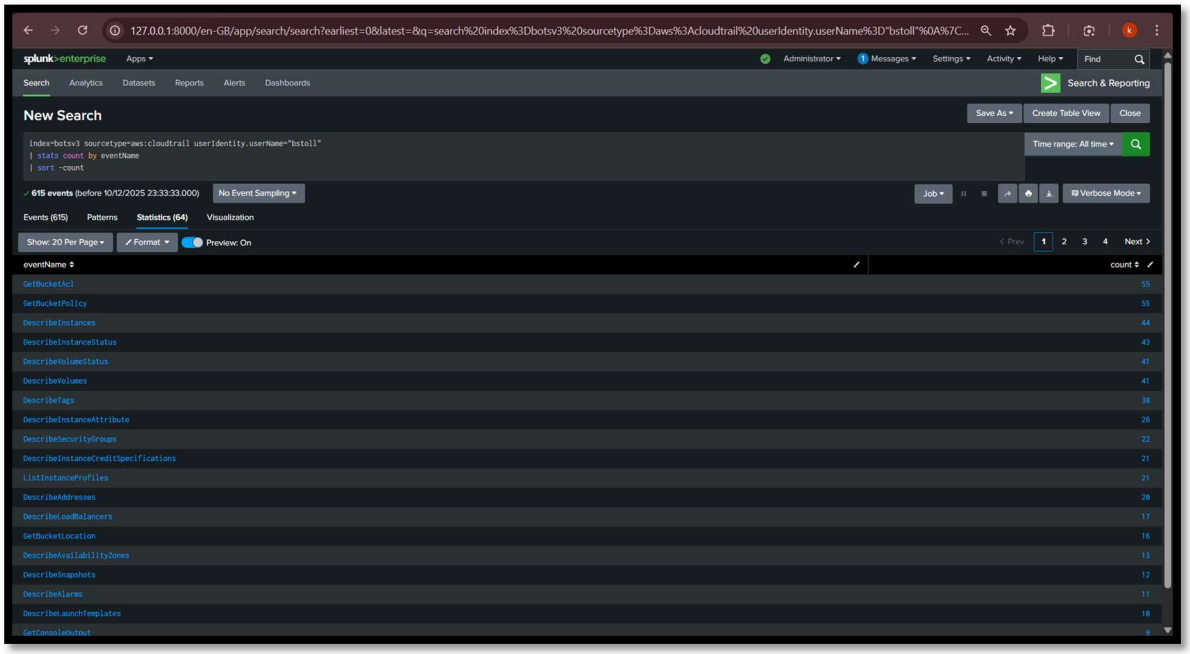
- Based on Question 4, the user who performed PutBucketAcl is bstoll
- This is likely Bud's IAM username

Step 2: Verify User Identity

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

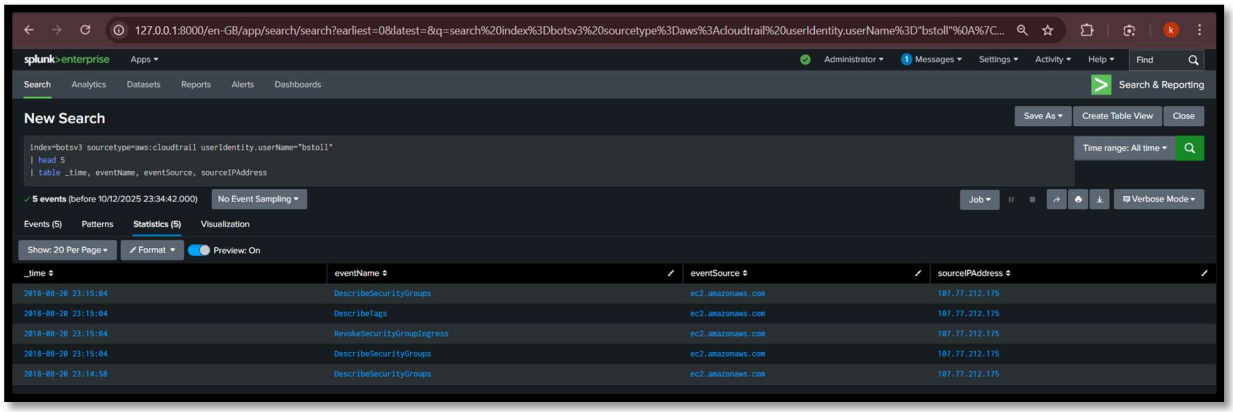
Step 3: Cross-Reference with Other Activities

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



Step 4: Confirm Through Additional Evidence

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



Complete Working Query:

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

Key Findings:

- User bstoll performed the PutBucketAcl operation
- Same user appears in other AWS activities
- Consistent identity attribution throughout logs

SOC Relevance:

- Critical for insider threat analysis
- Essential for post-incident investigations
- Enables accountability and audit trails

Evidence 6: What is the name of the S3 bucket that was made public?

Final Answer: frothlywebcode

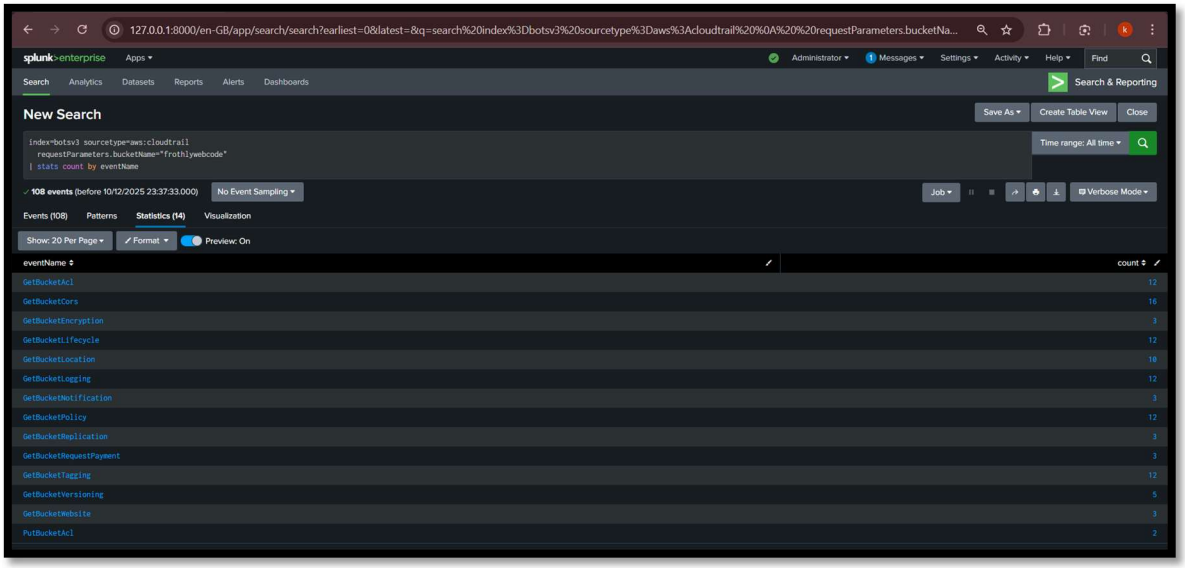
Step-by-Step Investigation:

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

Complete Working Query:

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

Key Findings:

- Bucket name: frothlywebcode
- Contains web assets (based on naming convention)
- Made publicly accessible via ACL change

SOC Relevance:

- Public web code buckets enable supply chain attacks
- Requires immediate containment
- Regular audits of S3 bucket permissions needed

Evidence 7: Which text file was uploaded into the public bucket?

Final Answer: OPEN_BUCKET_PLEASE_FIX.txt

Step-by-Step Investigation:

Step 1: Switch to S3 Access Logs

- Use aws:s3:accesslogs source type (different from CloudTrail)
- S3 access logs record every request made to S3 buckets

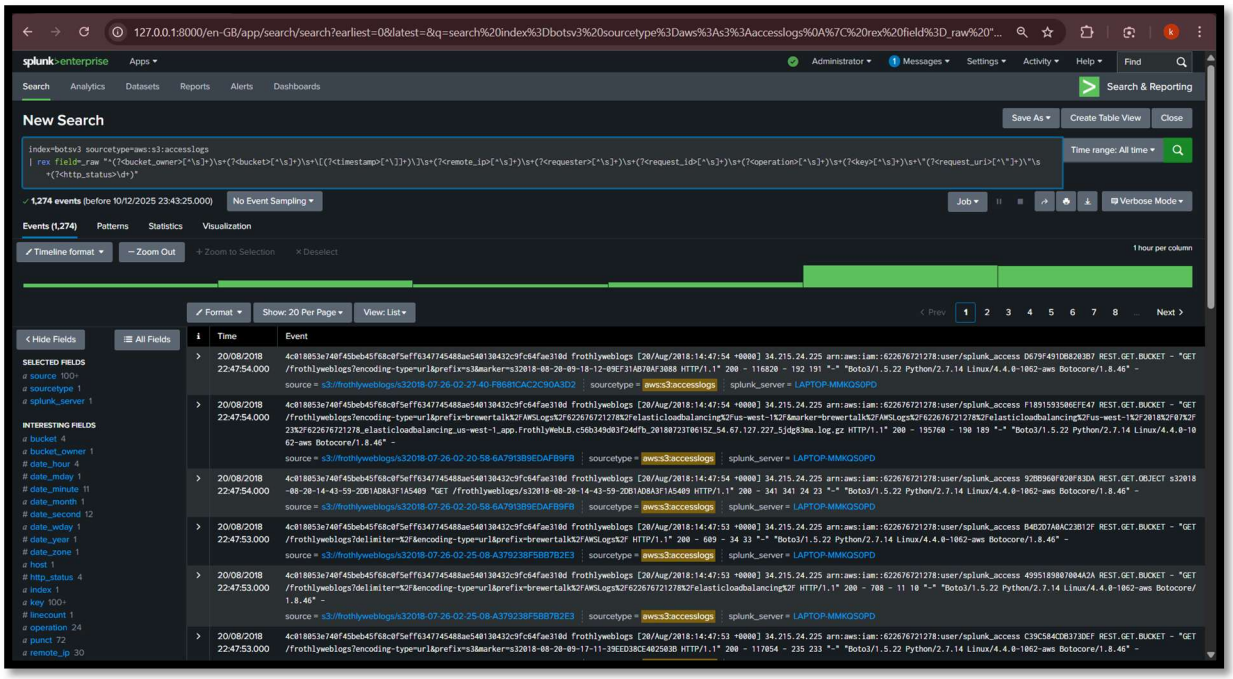
Step 2: Parse Raw Log Format

S3 access logs have a specific space-delimited format:

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

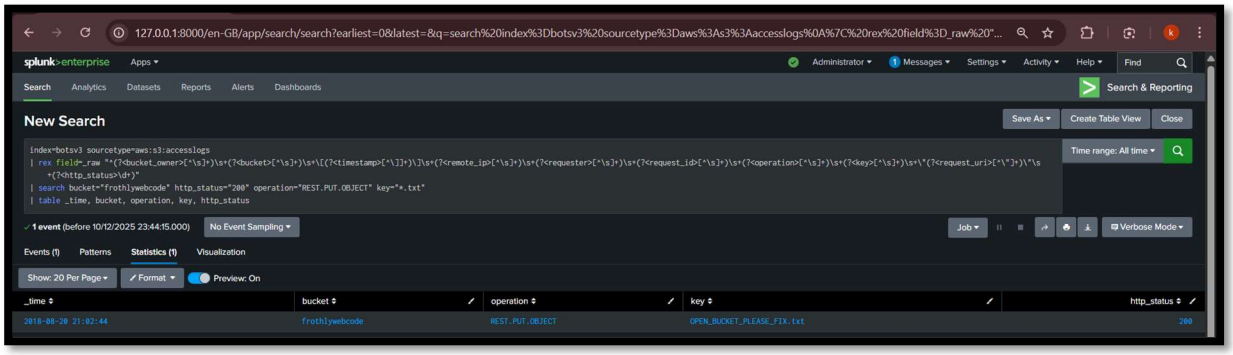
Step 3: Create Parsing Query

```
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+)"
```



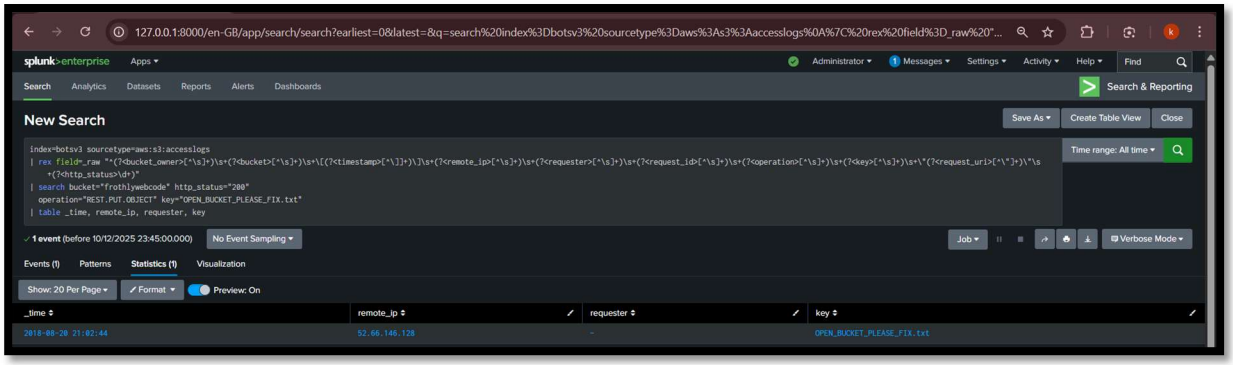
Step 4: Filter for Relevant Activity

```
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" operation="REST.PUT.OBJECT" key="*.txt"
| table _time, bucket, operation, key, http_status
```



Step 5: Verify File Upload

```
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="OPEN_BUCKET_PLEASE_FIX.txt"
| table _time, remote_ip, requester, key
```



Complete Working Query:

```
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
```

Key Findings:

- File name: OPEN_BUCKET_PLEASE_FIX.txt
- Successfully uploaded (HTTP 200)
- Likely a warning message from security researcher
- Indicates bucket was noticed as publicly accessible

SOC Relevance:

- Investigate unexpected file uploads
- Check for malicious payloads
- Monitor for data exfiltration attempts
- Security researcher notifications

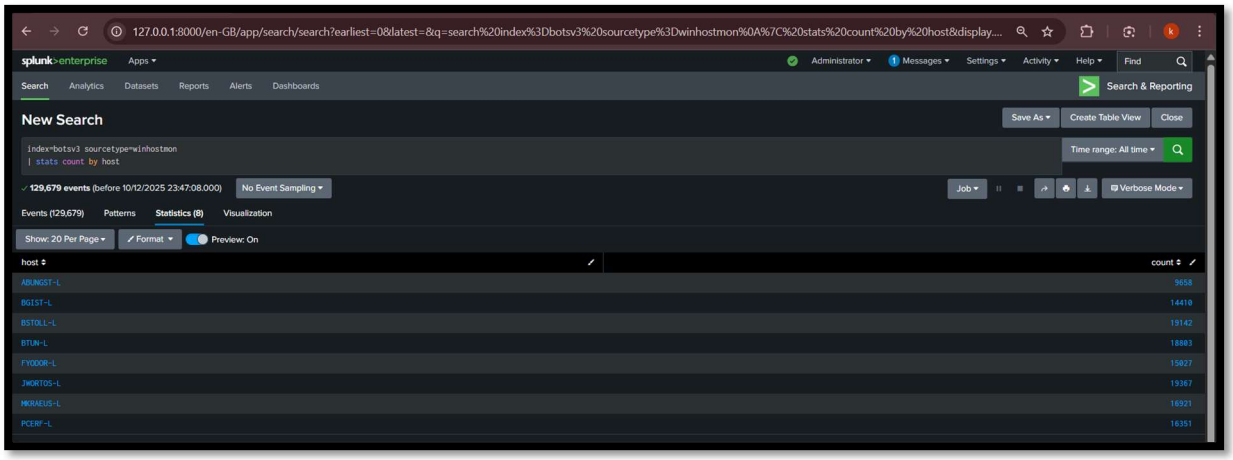
Evidence 8: Which host runs a different Windows OS version?

Final Answer: BSTOLL-L.froth.ly

Step-by-Step Investigation:

Step 1: Start with Hint Source Type

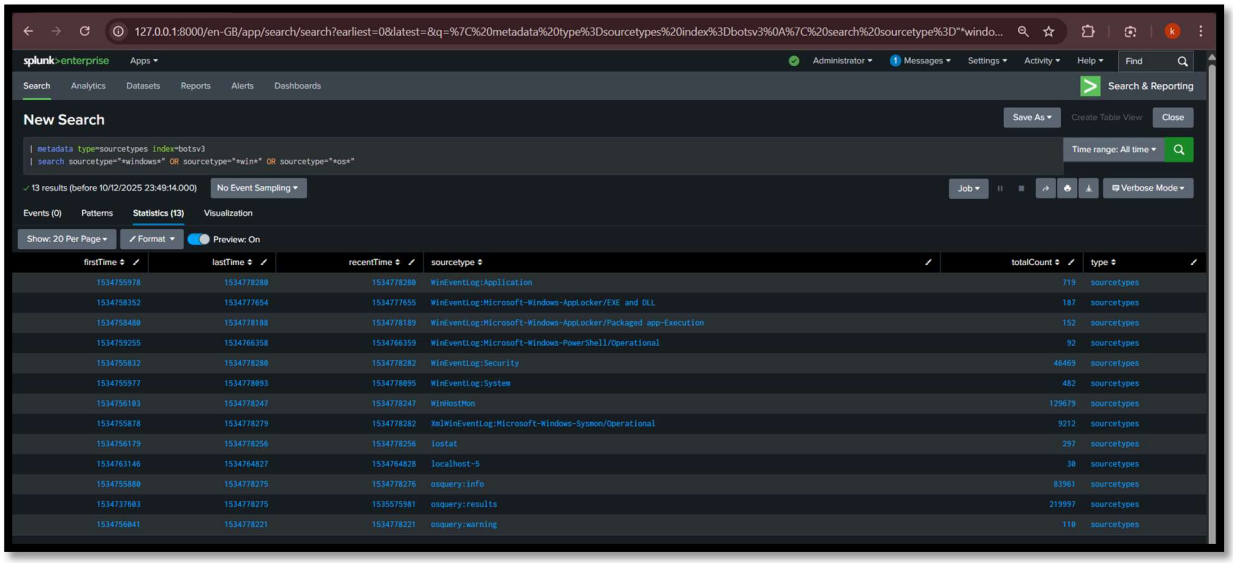
```
index=botsv3 sourcetype=winhostmon
| stats count by host
```



Note: **winhostmon only contains driver information, not OS details.**

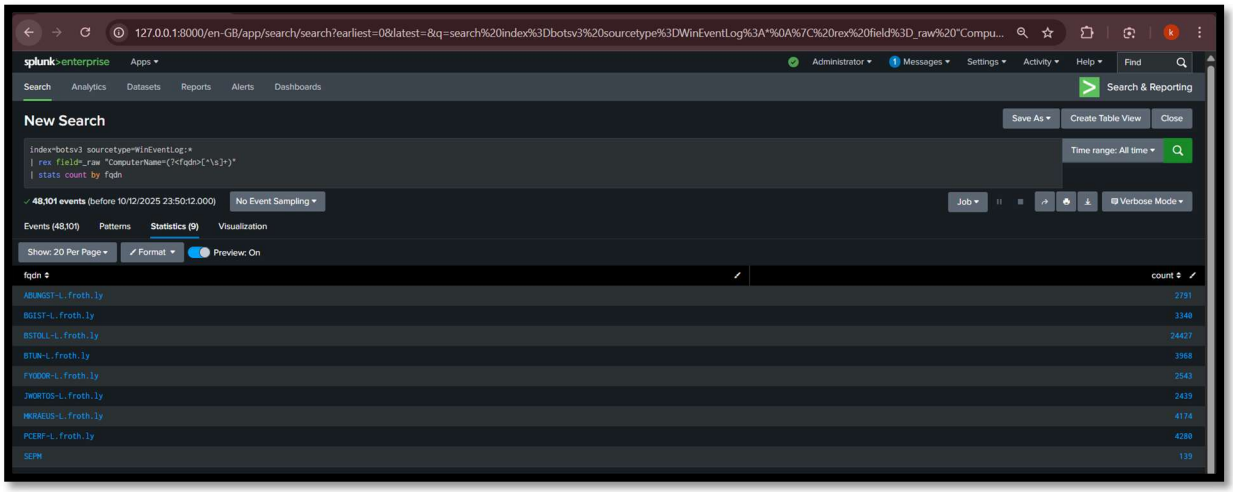
Step 2: Identify Alternative Data Sources

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



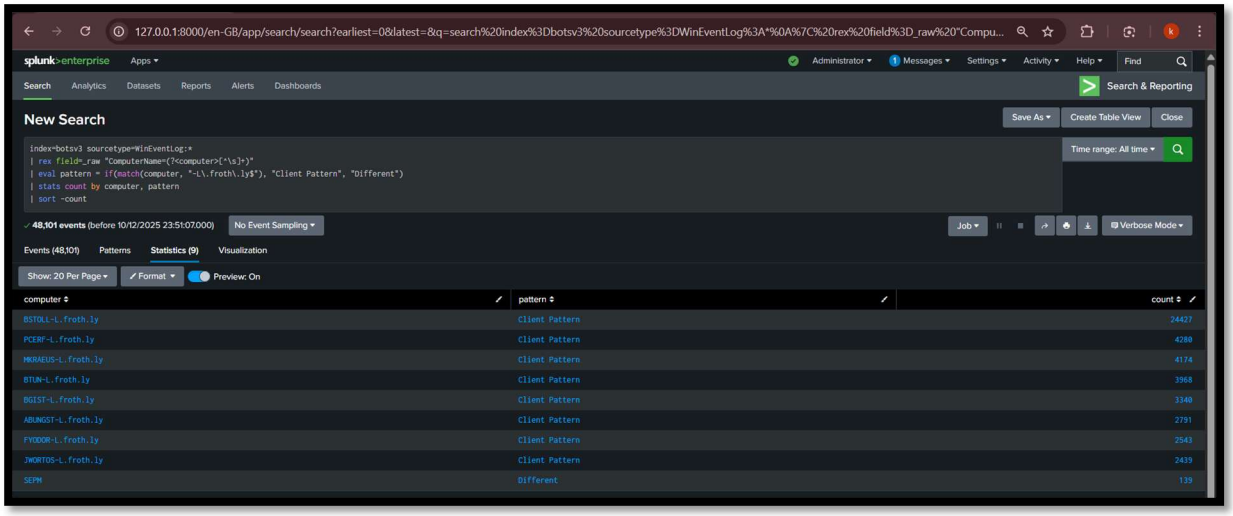
Step 3: Use Windows Event Logs

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



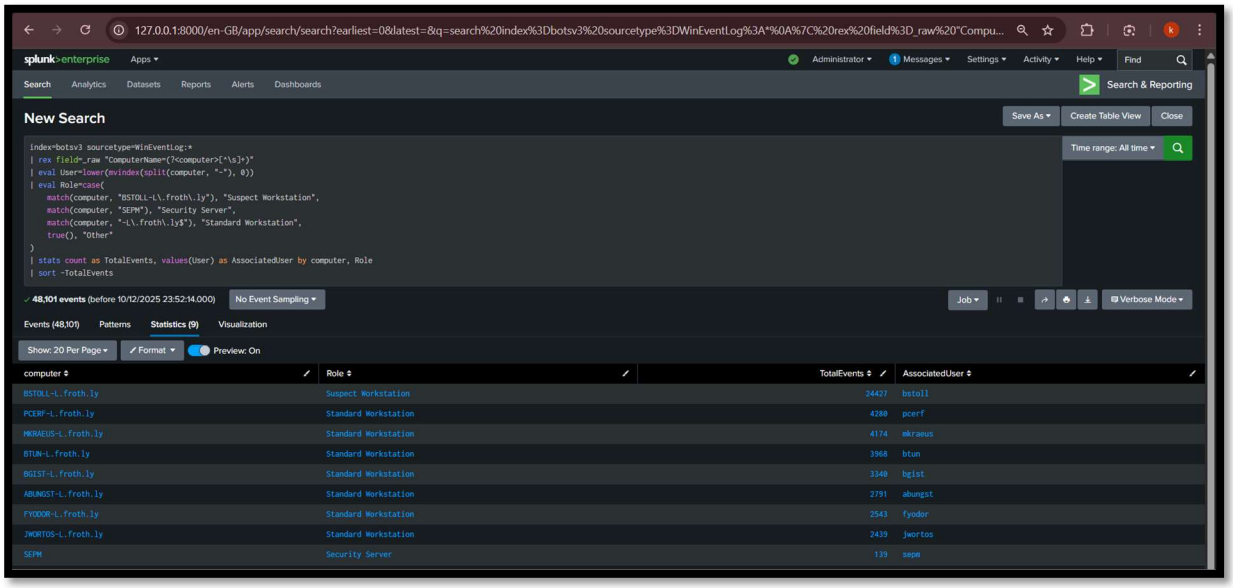
Step 4: Analyze Naming Patterns

```
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
```



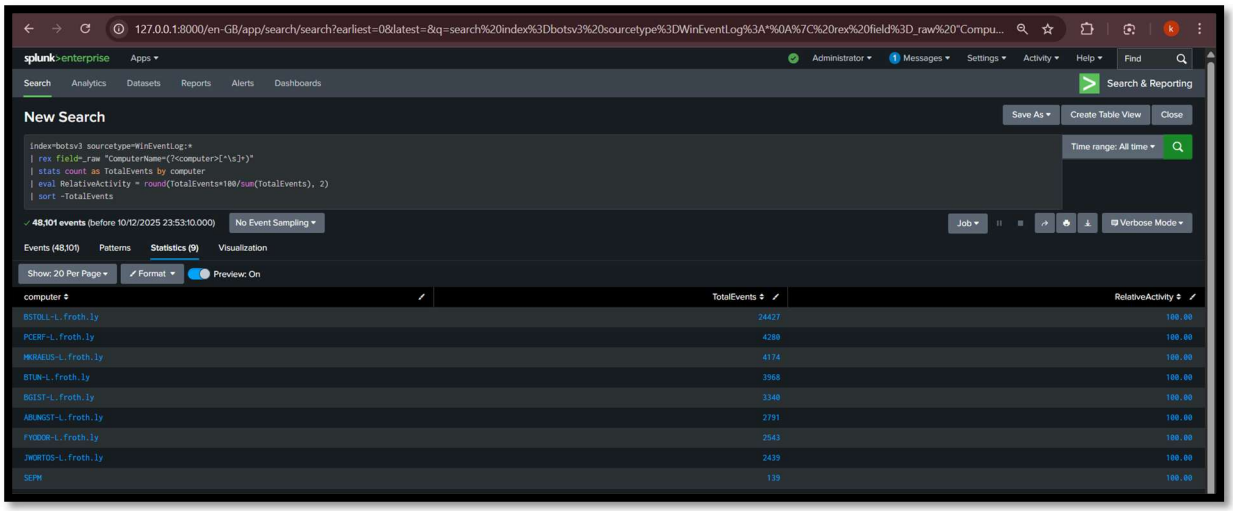
Step 5: Investigate Anomalous Host

```
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
```



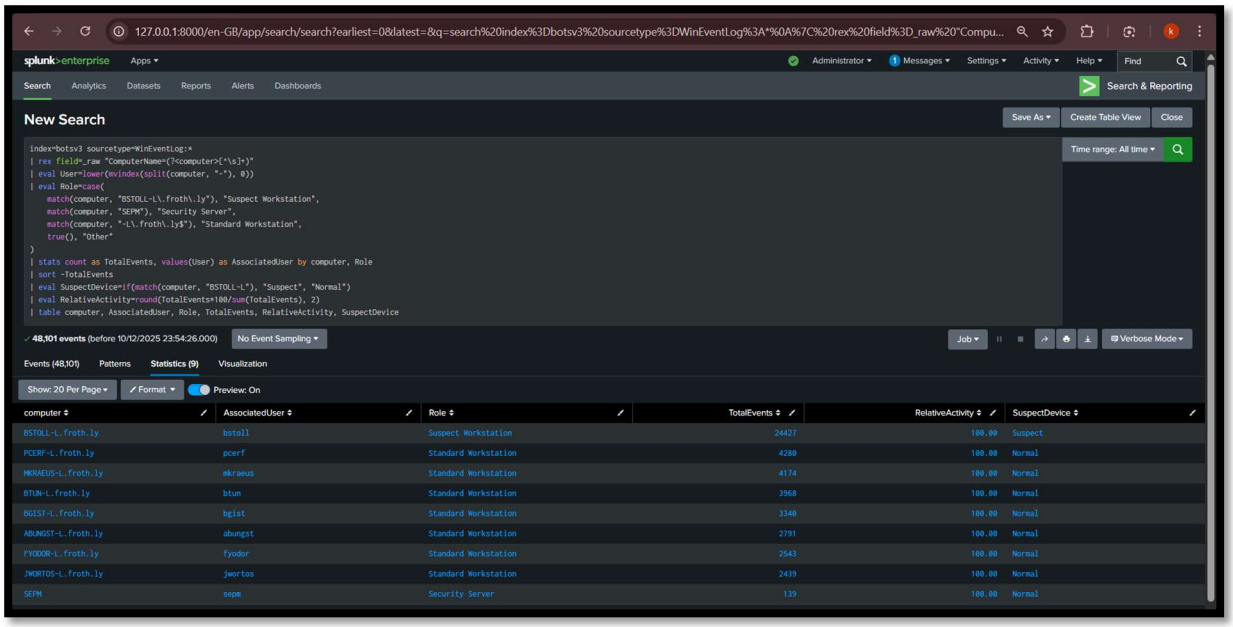
Step 6: Verify Activity Levels

```
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
```



Key Findings:

Host	Total Events	Role	Associated User
BSTOLL-L.froth.ly	**24,427**	Suspect Workstation	bstoll
SEPM	139	Security Server	N/A
Other hosts	<100 each	Standard Workstation	Various

SOC Relevance:

- BSTOLL-L.froth.ly is the endpoint of user bstoll
- Same user who modified S3 bucket permissions
- High event volume indicates suspicious activity
- Critical for targeted forensic analysis
- Enables direct user attribution

Investigation Summary Table

Question	Data Source	Key Field	Finding	SOC Action
1	aws:cloudtrail	userIdentity.userName	4 IAM users	Baseline identity monitoring
2	aws:cloudtrail	userIdentity.sessionContext.attributes.mfaAuthenticated	MFA detection field	Alert on non-MFA API calls
3	hardware	processor	Intel Xeon E5-2676 v3	Hardware baseline
4	aws:cloudtrail	eventID	ab45689d-69cd-41e7-8705-5350402cf7ac	Alert on PutBucketAcl
5	aws:cloudtrail	userIdentity.userName	bstoll	User attribution
6	aws:cloudtrail	requestParameters.bucketName	frothlywebcode	Bucket monitoring
7	aws:s3:accesslogs	key	OPEN_BUCKET_PLEASE_FIX.txt	Investigate uploads
8	WinEventLog:*	ComputerName	BSTOLL-L.froth.ly	Endpoint forensics