



Detection and Attribution of Anti-Forensic Techniques in Cloud Environments

Controlled Experiments on Microsoft Azure

Authors: Uzma Haneef · Sarah Zafar · Kamlesh Kumar Soothar · Muhammad Asghar Khan · Kamran Ali Memon

Presented by: Uzma Haneef · Sarah Zafar

Cyber Security Researchers

NED University of Engineering and Technology, Karachi, Pakistan

October 31, 2026

WHY IT MATTERS

Cloud forensics is hard, attribution is missing.

Why cloud forensics is hard:

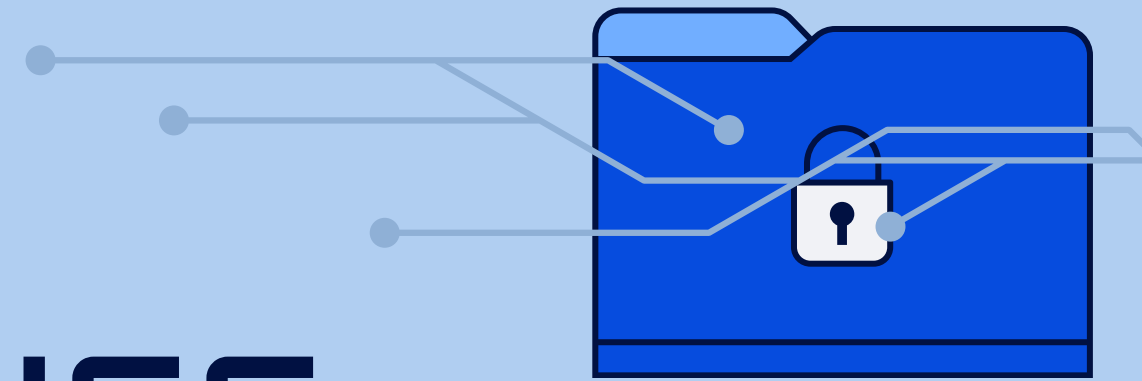
- Provider holds the evidence
- Shared (multi-tenant) hardware
- Data spread across regions
- Volatile instances and sessions
- Different laws per location

The gap we address:

Methods detect tampering but rarely say who did it: they use static logs, not correlation.

- Study of cloud anti-forensics
- Detection + attribution framework
- Controlled Azure experiments





THREE TECHNIQUES, ONE CLOUD

■ Log suppression

- Delete the diagnostic setting so later activity is never logged
- 2 trials, ~4,200 baseline events

■ Metadata spoofing

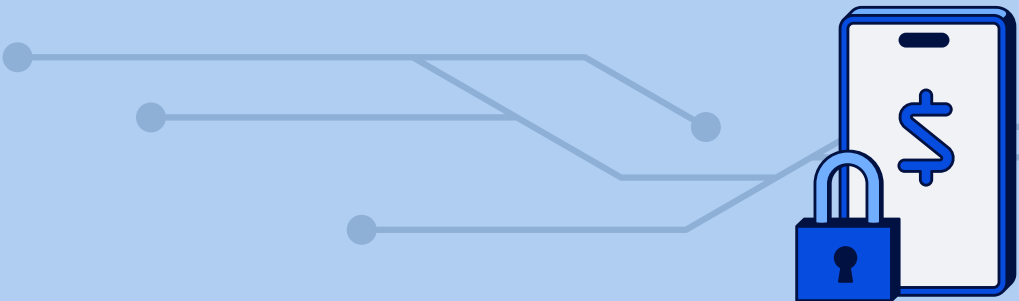
- Forge file timestamps to mislead the timeline
- 500 files, two difficulty levels

■ Obfuscation

- Hide content with encryption or encoding
- 450 blobs, range reads

Scope: one provider (Azure) · scripted workloads · synthetic data

TECHNIQUE 1 · LOG SUPPRESSION



ATTACK: DELETE THE DIAGNOSTIC SETTING

4,218	7 seconds	30 seconds
clean baseline events	longest normal gap	alert threshold ($\approx 4\times$ max)



D1 · Event rule

Delete event in
Activity Log



D2 · Gap

Silence over
30 s



D3 · Correlation

Gap containing
a deletion



D4 · State check

Blobs changed vs.
events logged

TESTBED ARCHITECTURE

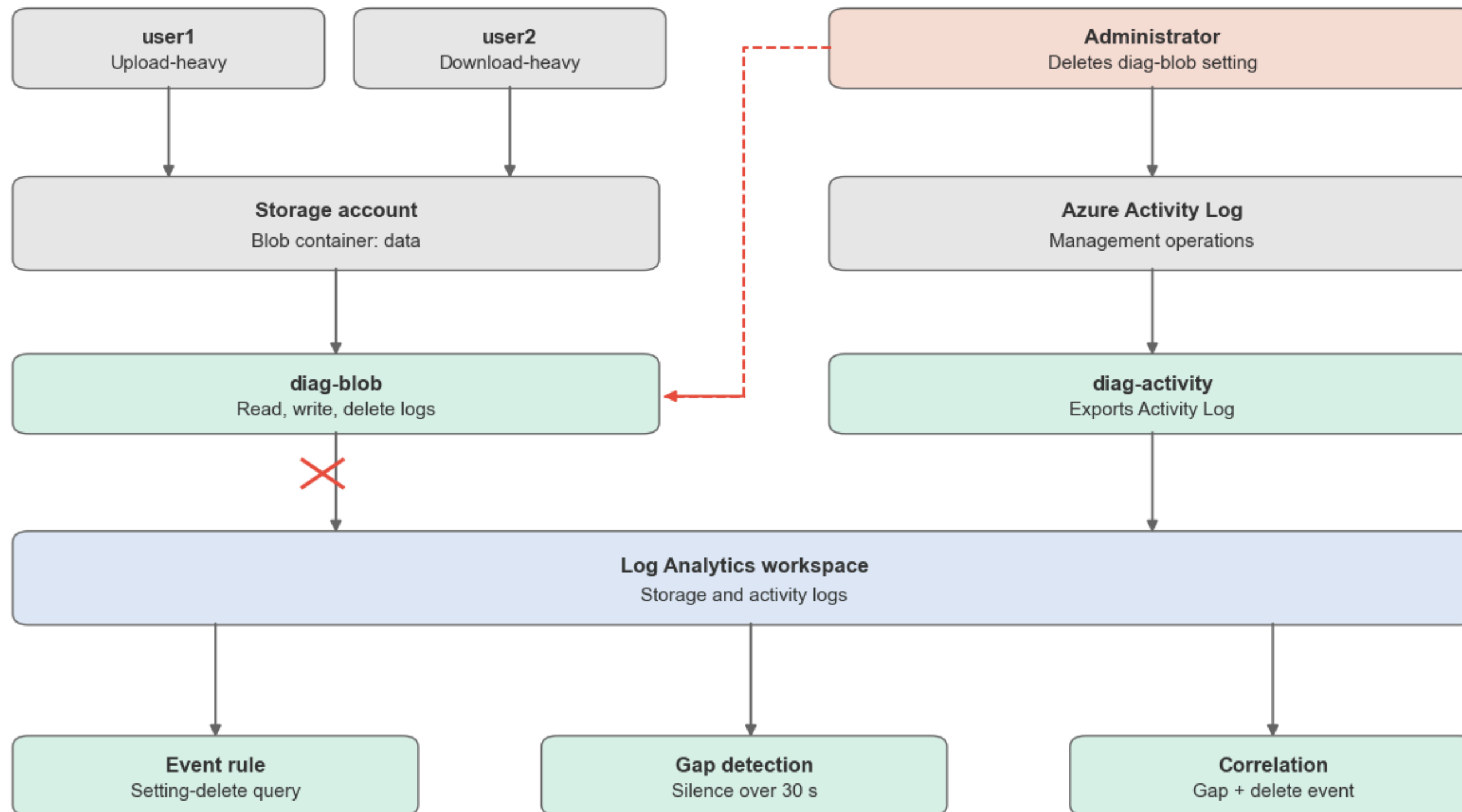


Figure 1: Testbed architecture and the logging attack.

MANUAL ATTACK LOG

```
2026-10-13T11:16:42.8683211Z,COOLDOWN END
PS C:\Users\Uzma\Downloads\Forensics_Research> type attack_log.csv
2026-10-13T07:01:45.3958957Z,CLEAN BASELINE START
2026-10-13T09:18:32.4576314Z,CLEAN BASELINE END
2026-10-13T09:18:40.7983263Z,CLEAN BASELINE END
2026-10-13T10:03:41.5642654Z,ATTACK1 START delete diag-blob
2026-10-13T10:18:36.1715856Z,ATTACK1 END logging restored
2026-10-13T10:30:02.8329528Z,COOLDOWN START
2026-10-13T10:44:05.6096979Z,COOLDOWN START
2026-10-13T10:44:19.5250281Z,COOLDOWN START
2026-10-13T11:16:37.7862356Z,COOLDOWN END
2026-10-13T11:16:42.8683211Z,COOLDOWN END
PS C:\Users\Uzma\Downloads\Forensics_Research>
```

What it shows:

- Ground-truth start and end times of the attack and cool-down, used to check Activity Log boundaries.

Figure 2: Manually recorded attack log (UTC).



BLACKOUT FOUND, ACCOUNT NAMED



14.4 min
Trial 1 blackout

0
Storage events
logged

≥ 84
blobs written
(versioning)

Alert	Gap window (UTC)	Length	Contains deletion?
1	08:46:17 → 10:29:32	103.3 min	Yes: true tampering
2	10:29:32 → 10:40:48	11.3 min	No: false alarm
3	10:40:48 → 10:44:34	3.8 min	No: false alarm

Alert 1 is ~7× the blackout (idle time before the attack). Alerts 2–3 come from two writes by another identity. The Activity Log names the account and source address (a session, not a person).

GAP PLUS CORRELATION (TRIAL 1)

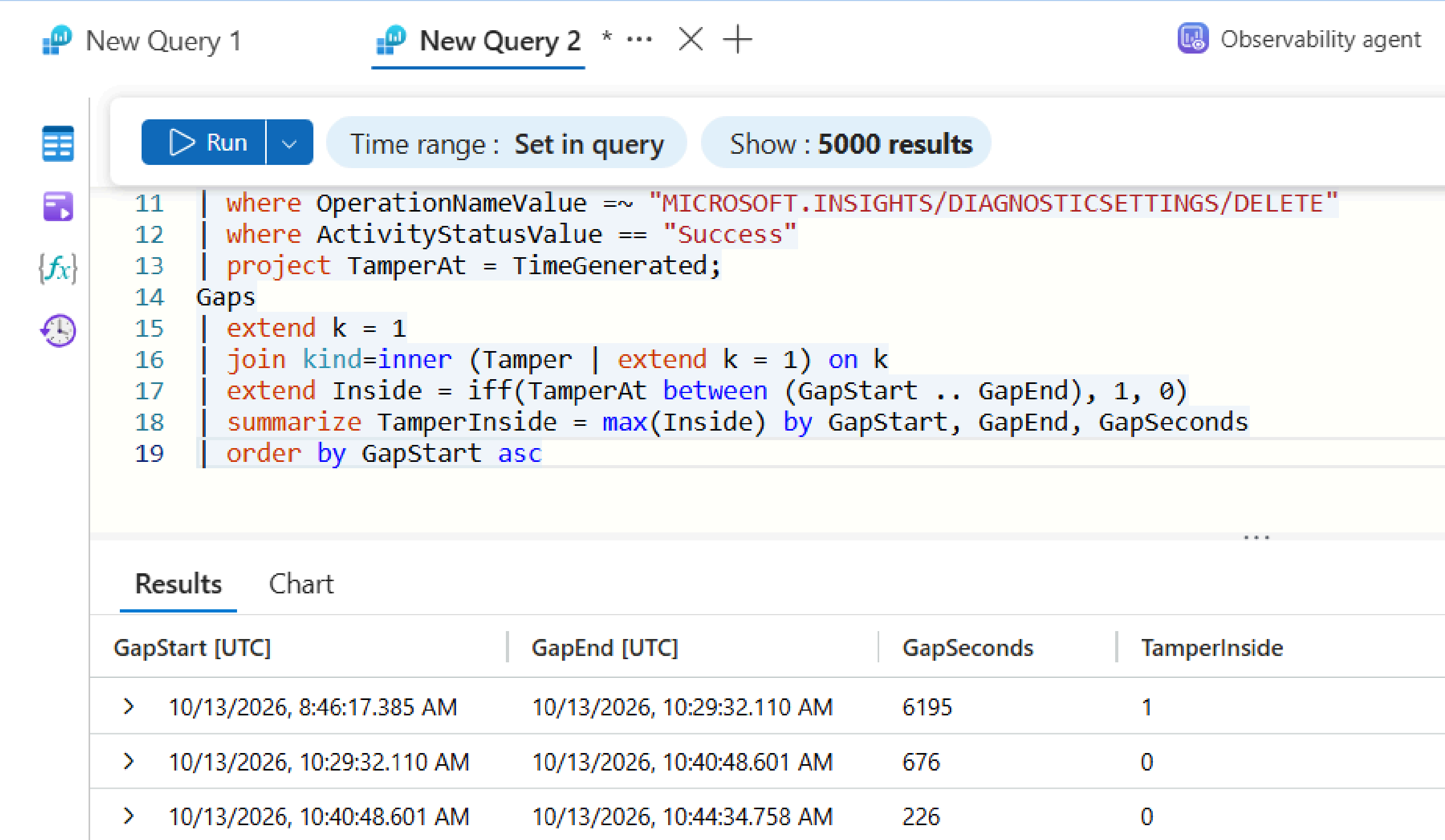
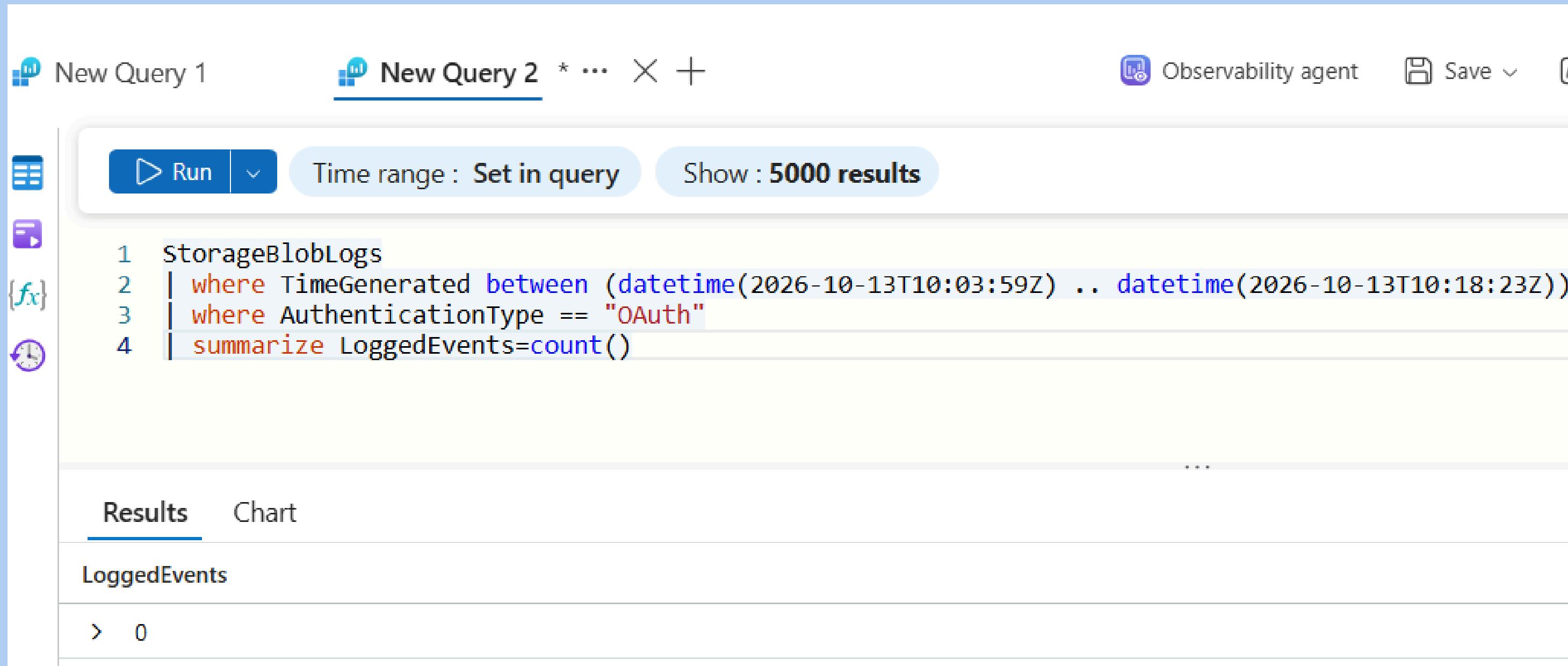


Figure 3: Gap detection correlated with Activity Log deletions.

What it shows:

- Three gaps. Only the first has TamperInside = 1, so the deletion falls inside it.

ZERO EVENTS IN THE BLACKOUT

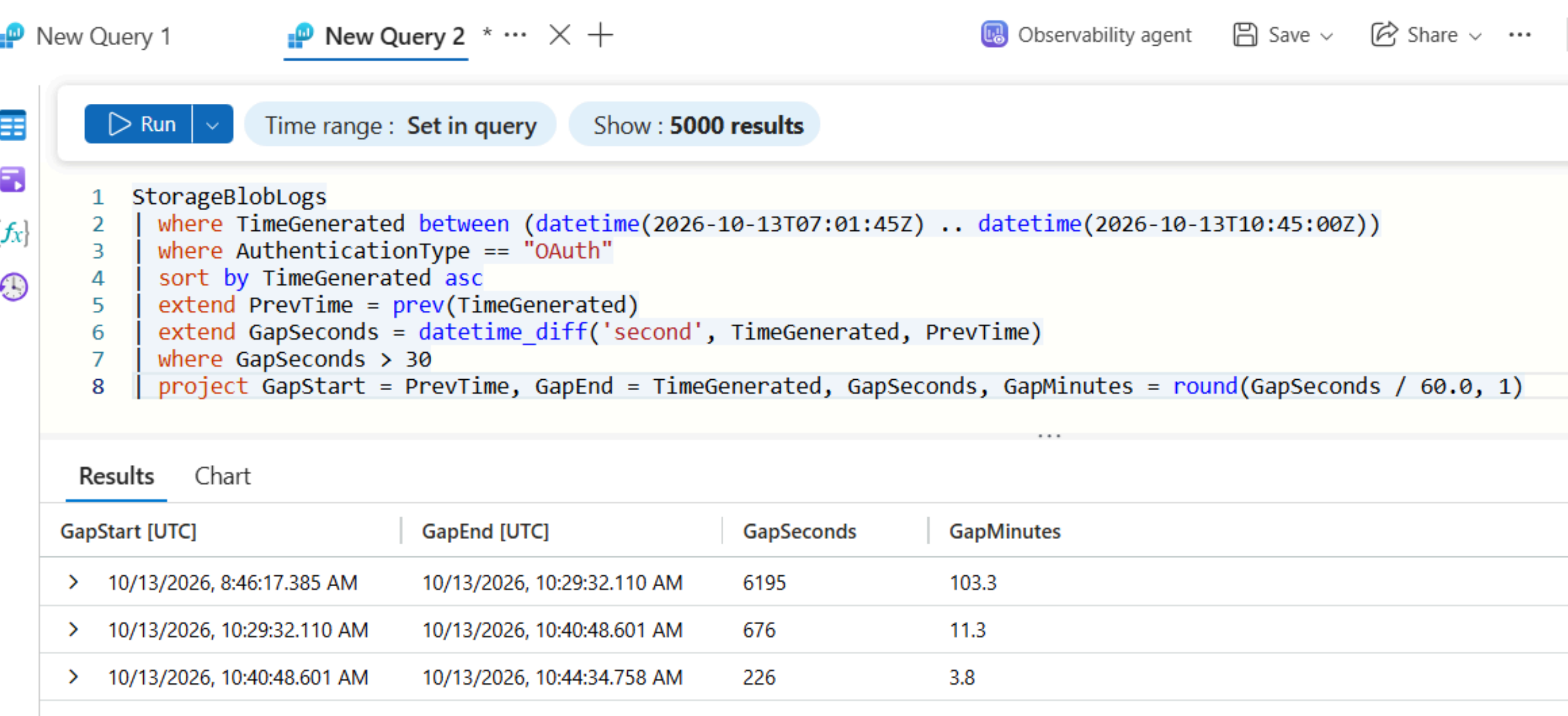


What it shows:

- LoggedEvents = 0 between the deletion (10:03:59) and the re-creation (10:18:23).

Figure 4: Zero OAuth storage events logged during the blackout.

GAP DETECTION ALONE (TRIAL 1)



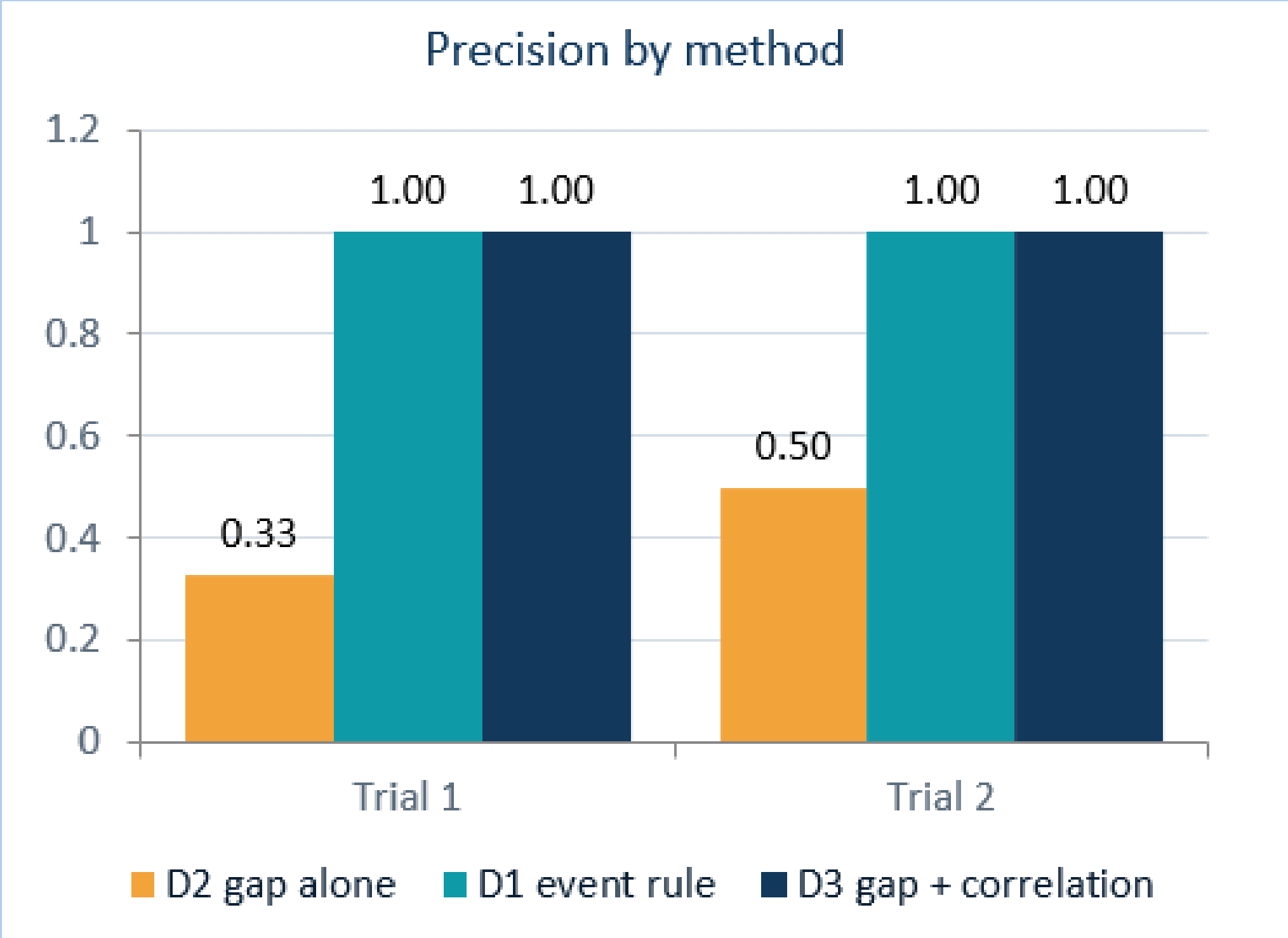
What it shows:

- Three alerts: 103.3, 11.3 and 3.8 min. Only the first contains the blackout. Precision 1/3.

Figure 5: Gap detection alone.

TECHNIQUE 1 . TRIAL 2 AND SUMMARY

REPEATED WITH SWAPPED WORKLOADS

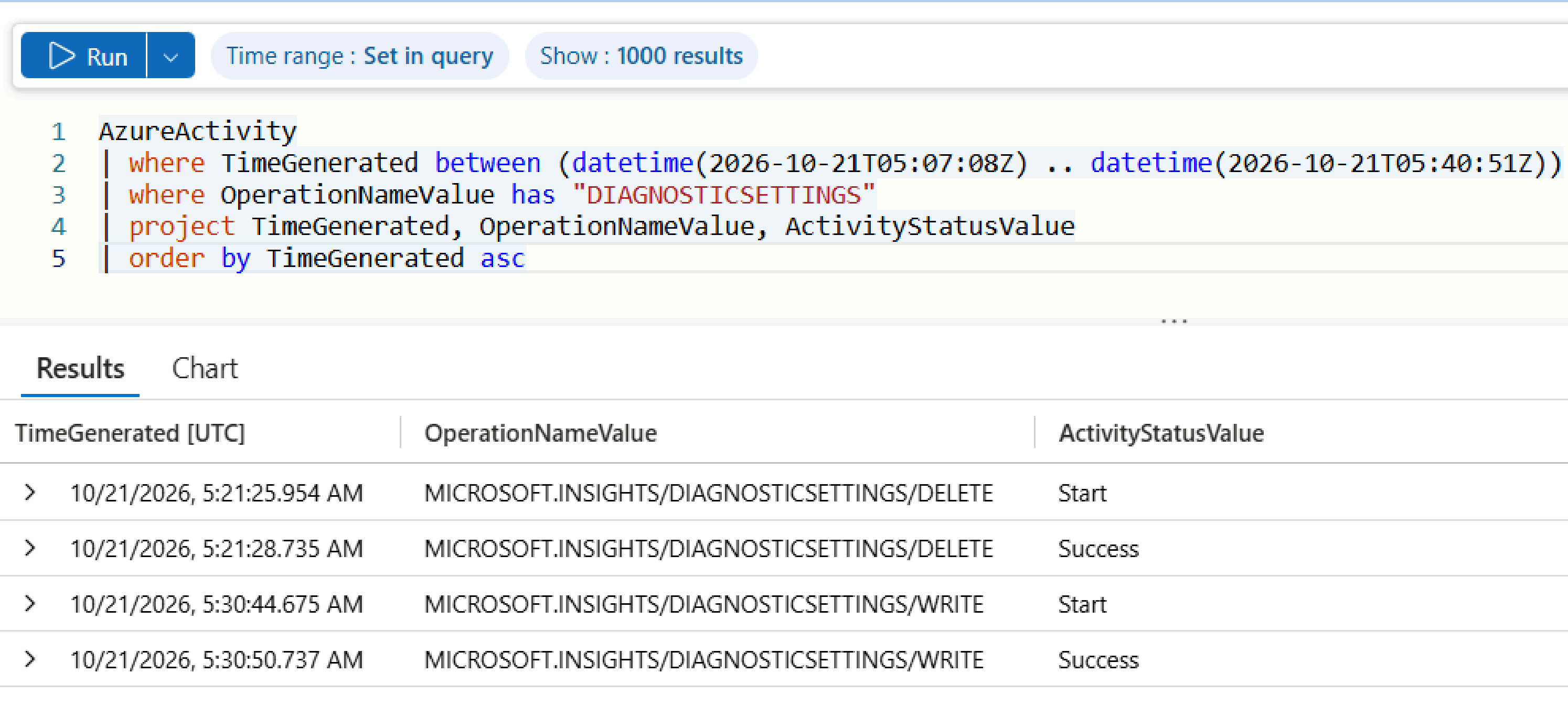


Trial	Blackout	Gap alerts
1	14.4 min	3
2	9.37 min	2

Trial 2 result

- Deleted 05:21:28, recreated 05:30:50 UTC. Blackout length depends on attacker timing. Gap alone 1/2; with correlation 1/1. D1 and D3 name the account.

AND RE-CREATION (TRIAL 2)

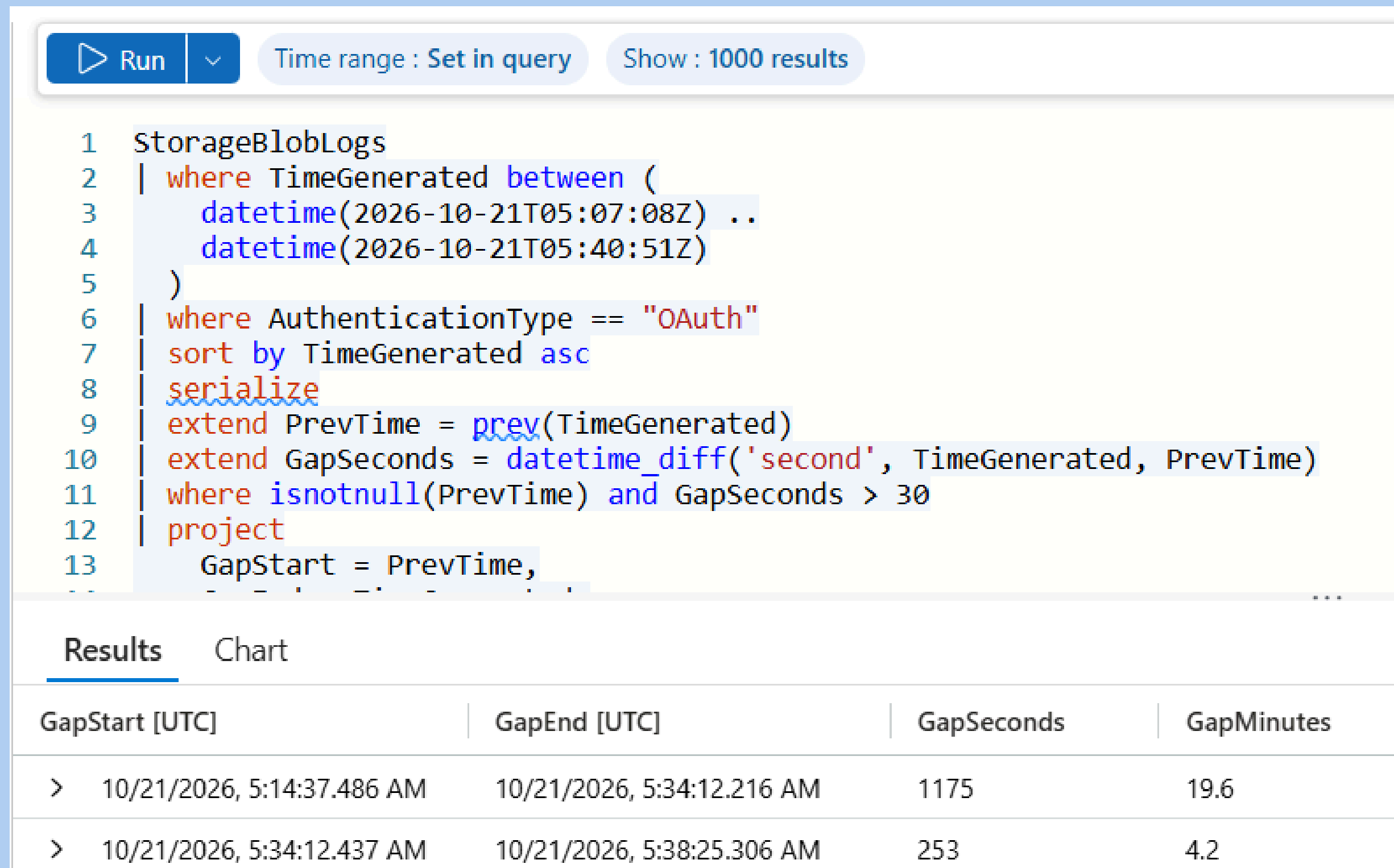


What it shows:

- Activity Log shows DELETE then WRITE. Blackout of 9.37 min, shorter than Trial 1.

Figure 6: diag-blob deleted at 05:21:28 and re-created at 05:30:50 UTC (Trial 2).

GAP ALONE (TRIAL 2)

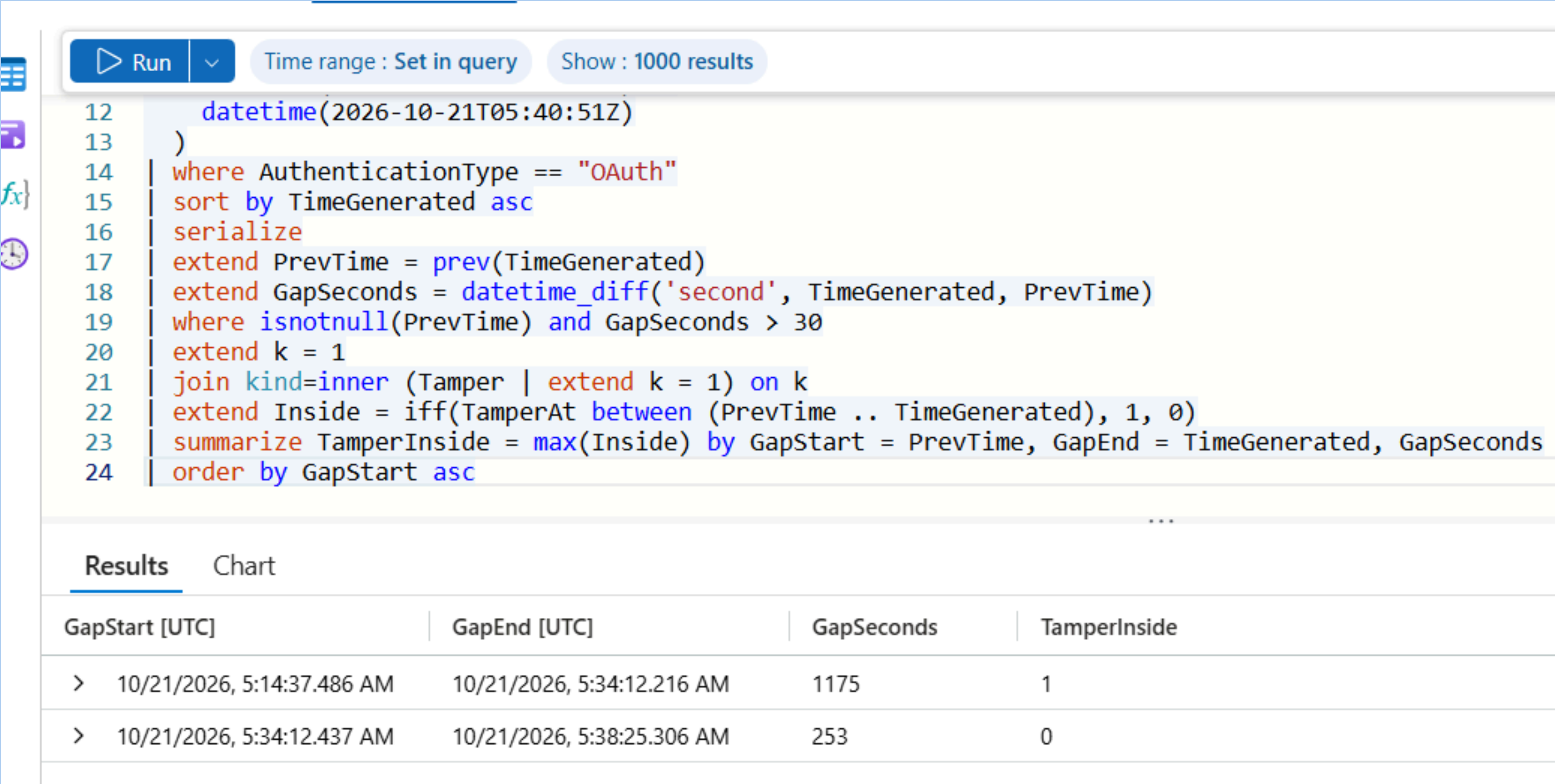


What it shows:

- Two alerts of 19.6 and 4.2 min. One is a false alarm. Precision 1/2.

Figure 7: Gap detection alone, Trial 2 (two alerts, one true positive).

GAP PLUS CORRELATION (TRIAL 2)

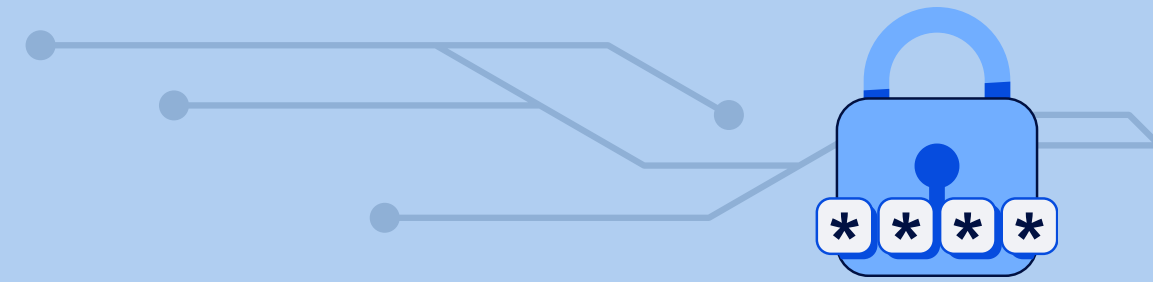


What it shows:

- Correlation removes the false alert. One alert remains and matches the tampering. Precision 1/1.

Figure 8: Gap detection correlated with Activity Log deletions, Trial 2 (one alert, matching true tampering).

TECHNIQUE 2 . METADATA SPOOFING



ATTACK: DELETE THE DIAGNOSTIC SETTING

Attack:

- 500 ext4 files. 250 random files (seed 42) forged with dates from 2015 to 2021, 4.8–11.7 years before creation; 250 untouched.

Local detector (any one fires):

- Modification time before birth time
- Change time over 1 day after modification
- Zero nanoseconds in modification time

Why it works

touch rewrites modification time, but change time and birth time stay true to real events.

LOCAL DETECTOR RESULTS

```
(azcli-env)-(uzma@Uzma)-[~/ts_experiment]
$ python3 metrics.py
cat metrics_summary.csv
df -T ~
TP=250  FP=0  TN=250  FN=0
Precision: 1.000
Recall:    1.000
F1 score:  1.000
False Positive Rate: 0.000
Exported to metrics_summary.csv
TP,FP,TN,FN,Precision,Recall,F1,FPR
250,0,250,0,1.0,1.0,1.0,0.0
Filesystem      Type 1K-blocks      Used Available Use% Mounted on
/dev/sda1       ext4  98497780 28528164  64920068  31% /
```

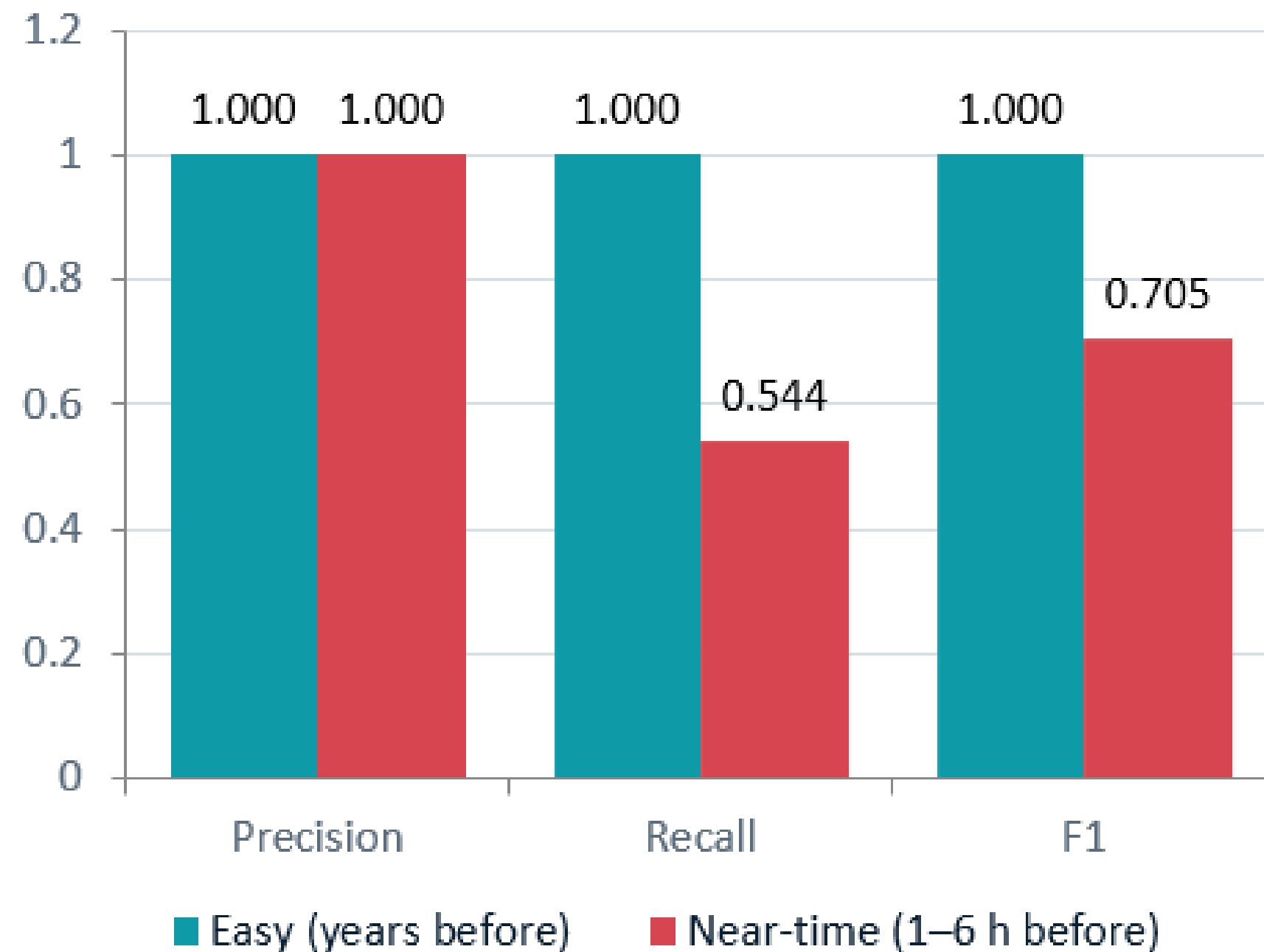
What it shows:

- TP 250, FP 0, TN 250, FN 0. Precision, recall and F1 all 1.000. Confirms ext4.

Figure 9: Local detector results on 500 files (TP = 250, FP = 0, TN = 250, FN = 0) and file system type (ext4).

TECHNIQUE 2 . RESULTS

EASY FORGERIES CAUGHT, NEAR-TIME ONES SLIP THROUGH



114 / 250

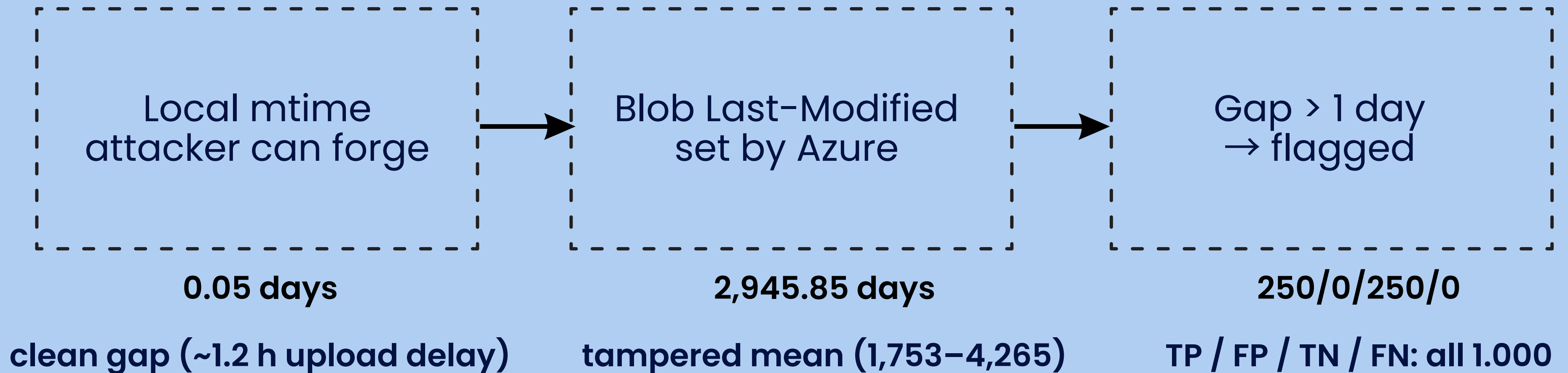
near-time
forgeries missed

Two difficulty levels:

- Easy: forged 4.8–11.7 years before creation (seed 42), all scores 1.000.
- Near-time: 1–6 hours before (seed 99), precision 1.000, recall 0.544, F1 0.705.

TECHNIQUE 2 . CROSS-SOURCE CHECK

COMPARE AGAINST AZURE'S LAST-MODIFIED



Caveats

Last-Modified is upload time, so backups or migrated files can false-alarm (needs a baseline).
Scored on easy forgeries only. PC and Azure clocks not compared. Single run.

CROSS-SOURCE OUTPUT

```
(azcli-env)-(uzma@Uzma)-[~/ts_experiment]
$ head -5 cross_source_output.csv
filename,tampered_ground_truth,local_mtime,blob_last_modified,gap_days,flagged_by_cross_source
files/file_0000.txt,1,1571194200.0,1791916779.0,2554.66,1
files/file_0001.txt,1,1530914760.0,1791916856.0,3020.86,1
files/file_0002.txt,0,1791912236.541417,1791916767.0,0.05,0
files/file_0003.txt,1,1594542360.0,1791916804.0,2284.43,1
```

Figure 10: First rows of the cross-source output: local modification time, blob Last-Modified, and gap in days.

What it shows:

- Per file: local mtime, blob Last-Modified, gap in days and flag.
- Tampered files show gaps of years.

CROSS-SOURCE RESULTS



```
(azcli-env)-(uzma@Uzma)-[~/ts_experiment]
$ python3 cross_source_metrics.py
cat cross_source_metrics.csv
TP=250 FP=0 TN=250 FN=0
Precision=1.000 Recall=1.000 F1=1.000
TP,FP,TN,FN,Precision,Recall,F1
250,0,250,0,1.0,1.0,1.0
```

Figure 11: Cross-source detector results on 500 files (TP = 250, FP = 0, TN = 250, FN = 0).

What it shows:

- TP 250, FP 0, TN 250, FN 0. Precision, recall and F1 all 1.000 on the easy set.

TECHNIQUE 3 . OBFUSCATION



HIDE CONTENT, SCAN ONLY A PREFIX

Dataset: 450 blobs

200 benign (text, JSON, gzip, zip)
and 250 obfuscated (Base64,
hex, AES-256, gzip+Base64,
XOR). All uploaded as text/plain.

Threat model and scan

Attacker: compromised
identity with write rights.
Defender: read-only, scans
prefixes (128 B to full), 450×6
= 2,700 range reads.

Alert if any signal fires:

Entropy > 7.5
bits/byte

Base64 charset
+ entropy ≥ 4.0

Hex charset + entropy $\geq$
3.0

text/ type, few printable
bytes

BLOB COUNT VERIFICATION



```
(azcli-env)~(uzma@Uzma)-[~/ob_experiment]
$ python3 scan_blobs.py
450 blobs listed in summary.csv
SCANSTART(UTC): 2026-10-15T09:04:19.642975+00:00
SCANEND(UTC): 2026-10-15T09:24:26.510948+00:00
wrote 2700 rows to scan_results.csv
sed Available Use
/dev/sda1 ext4 98497780 28528164 64920068 31
```

Figure 12: Blob count verification confirming all 450 files uploaded to Azure Blob Storage.

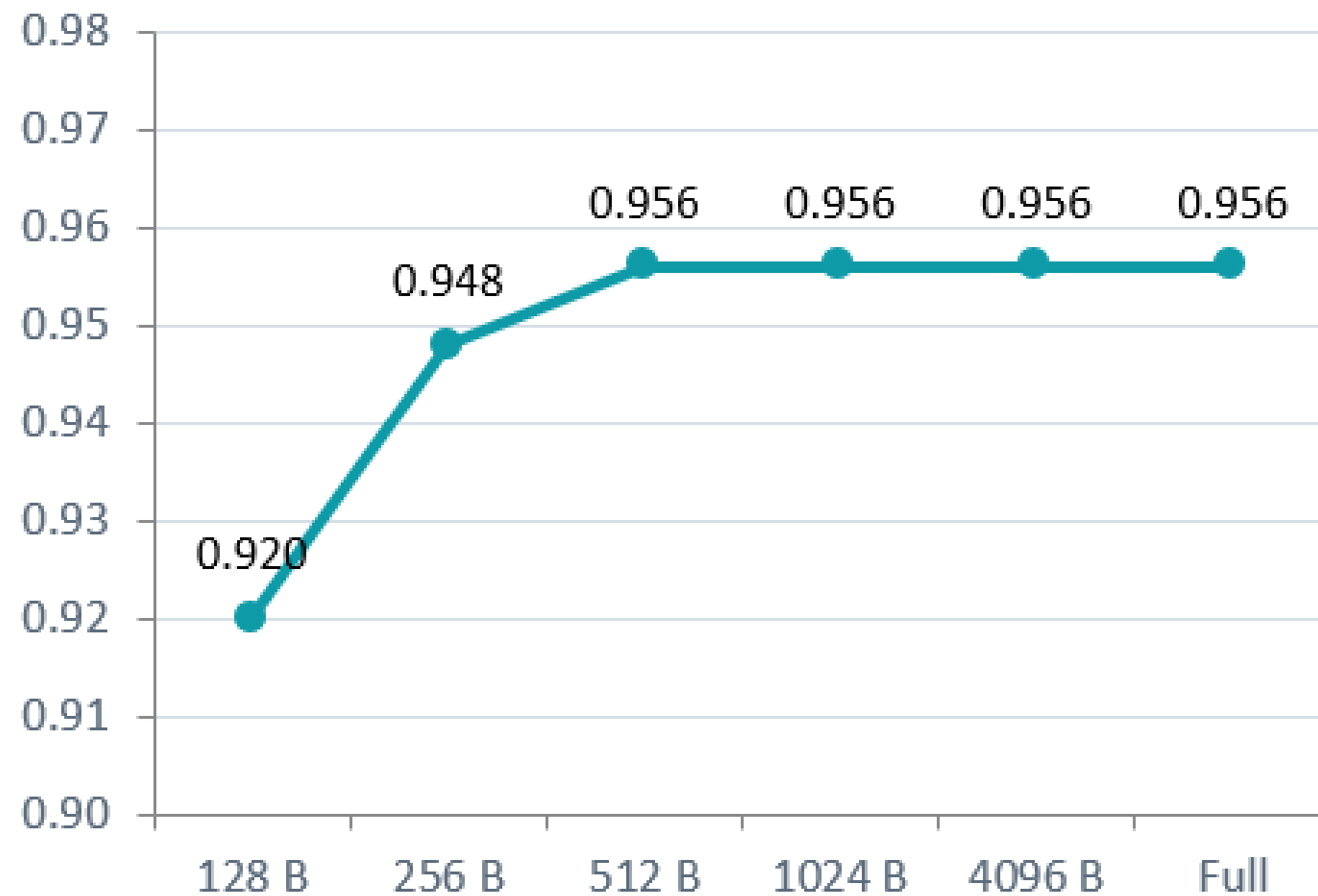
What it shows:

- 450 blobs listed.
- 2,700 scan rows written (450 blobs × 6 read sizes).

TECHNIQUE 3 . RESULTS

A SMALL PREFIX CATCHES MOST OF IT

Recall by prefix length



0.920

at 128 B (6.3%)

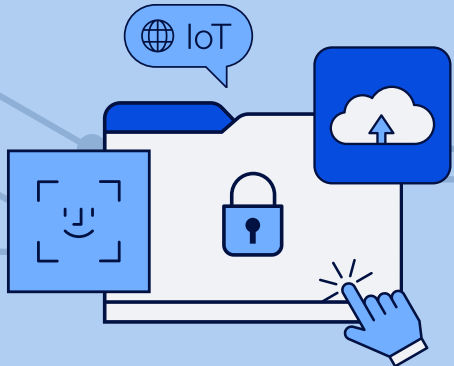
0.956

from 512 B (24.9%)

Zero false positives

- All 200 benign blobs stayed clean at every size. Reading more than 512 B added no detections for these blobs.

XOR IS THE BLIND SPOT



Class	128 B	256 B	512 B	1024 B	4096 B	Full
AES-256	50/50	50/50	50/50	50/50	50/50	50/50
Base64	50/50	50/50	50/50	50/50	50/50	50/50
Gzip + Base64	50/50	50/50	50/50	50/50	50/50	50/50
Hexadecimal	41/50	48/50	50/50	50/50	50/50	50/50
XOR (single-byte)	39/50	39/50	39/50	39/50	39/50	39/50

XOR stays at 78% even on the full blob: a heuristic limit, not missing data. XOR output has high entropy but no structural signature.



WHAT WORKED, AND WHERE IT DIDN'T

Technique	Best method	Best result	Weak spot
Log suppression	Gap + correlation, event rule	Precision 1/1 in 2 trials; names account	Gap alone: 1/3, 1/2
Metadata spoofing	Local 3 signals; cross-source	All metrics 1.000 on easy forgeries	Near-time recall 0.544
Obfuscation	Range-read prefix scan	Recall 0.956 at 512 B; 0 FP	XOR 39/50

TAKEAWAYS AND LIMITS

Key takeaways

- Correlating Activity and storage logs removes false alarms and names the account
- Provider-assigned evidence beats forgeable metadata
- Prefix reads screen for obfuscation cheaply

Limits and next steps

- Scripted workloads, synthetic data, one provider, 2 log trials
- Weak spots: near-time forgeries, XOR
- Next: other providers, dedicated attacker identity, real-time alerting

Thank you · Any Questions?

