Security Monitoring in the Cloud
Session #3

Application & Data Security Monitoring

Jan 2026

Key Outcomes

- Implement observability for cloud applications using logs and traces.
- Detect unauthorized access to cloud storage and data services.
- Create actionable alerts for anomalous data activity.
- Integrate monitoring outputs with SIEM workflows.

Course Material: https://github.com/0x74696D/security_monitoring_tp

Agenda

01 **Observability for Cloud based Applications**
Logging patterns and integrations

02 **Monitoring Data Services**
Storage, DBs, Queues, etc.

03 **Alerting**
Best practices and threat scenarios

Observability for Cloud Applications

Observability for Cloud Applications

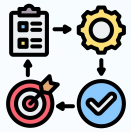
1.0.0

Cloud Logging for Applications

Security logs:

- Auth events (login success/fail)

- Permission changes, role changes

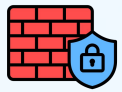


- Sensitive operations (exports, deletes, config changes)

- Client IP, user ID, request IDs

GCP Cloud Logging or **AWS CloudWatch** Logs for:

- Serverless workloads (Cloud Run Functions/Jobs, Lambda, Batch, Step Functions, etc.)
- Containerized workloads (Cloud Run Services, ECS)
- Kubernetes workloads (GKE, EKS)
- Load balancers, API Gateway



+ CDN logs (Cloud HTTP(S) Load Balancer logs for GCP or Access Logs for Cloudfront)
+ Firewall logs (Google Cloud Armor, WAF)

Observability for Cloud Applications

1.1.0

Logging Patterns

🎯 Who did what, from where, to which resource, and was it allowed.

Question	Example
Who	user_id, service_account
What	action (login, export, delete, grant_role)
Where	source_ip, region, user_agent
When	timestamp (always!)
On what	resource_id (file, record, dataset)
Result	success / failure / denied
Context	request_id / trace_id

Observability for Cloud Applications

1.2.0

Best Practices

Structured Logging

→ machine-readable, not free-text blobs
→ standardized fields across systems
→ security-relevant events, not raw noise: high-value events

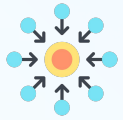
```
{
  "timestamp": "2026-01-24T10:31:42Z",
  "event_type": "auth_failure",
  "user_id": "12345",
  "source_ip": "203.0.113.42",
  "service": "api-gateway",
  "destination": "auth_service_prod",
  "reason": "invalid_password",
  "environment": "prod”,
  "request_id": "019bf076-916f-7d10-88c4-f776c9bd0587"

}
```

User failed to login from 203.0.113.42 at 10:31:42

Observability for Cloud Applications

1.2.1

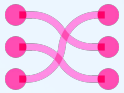
Best Practices



Centralized
→ not live only on the host
→ RBAC, encryption , monitoring
→ retention policies: hot/cold/archive
→ **Correlation**

Alert on behavior, not single events

- 20 failed logins in 5 minutes
- Login from new country + privilege escalation
- API key used from two regions simultaneously



Correlate logs with metrics & traces

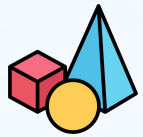
- Auth failures + traffic spikes
- WAF blocks + application errors
- IAM changes + unusual API usage

Monitoring Data Services

Monitoring Data Services

2.0.0

Monitoring blob access



GCS access logs or AWS S3 access logs & audit logs or CloudTrail:

storage.objects.get, storage.objects.list, storage.objects.delete

What to monitor:

- Bulk downloads
- Access from unusual identities / service accounts
- Access from unexpected regions or IPs

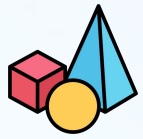
Example scenarios:

- Exfiltration from a “backups” bucket
- Public bucket misconfig → mass access

Monitoring Data Services

Monitoring Databases & Data Services

2.1.0



Common data backends:

- SQL databases (PaaS) Cloud SQL / RDS, Aurora
- Data Warehouse BigQuery / Redshift
- No SQL DB: Firestore, DynamoDB

What's important to log:

- Login attempts (DB auth failures)
- Privilege changes (GRANT/REVOKE)
- Mass SELECTs, EXPORTs, COPY TO statements

Monitoring Data Services

2.2.0

Monitoring Databases & Data Services

Data Service	Cloud Provider	Logging Service	What Gets Logged (Security-Relevant)
Cloud SQL	GCP	**Cloud Audit Logs** + DB engine logs	Logins, failed auth, schema changes, connections
BigQuery	GCP	**Cloud Audit Logs (Data Access)**	Queries, extracts, exports, dataset access
Cloud Storage (GCS)	GCP	**Cloud Audit Logs (Data Access)**	Object reads, writes, deletes, bulk downloads
Firestore / Datastore	GCP	**Cloud Audit Logs**	Document reads/writes, admin operations
Aurora (MySQL/Postgres)	AWS	**CloudWatch Logs**	SQL queries, connections, auth failures
RDS (MySQL/Postgres)	AWS	**CloudWatch Logs**	Login attempts, slow queries, errors
S3	AWS	**CloudTrail + S3 Access Logs**	Object access, list/get/delete operations
DynamoDB	AWS	**CloudTrail + CloudWatch Metrics**	Table access, scan/query operations

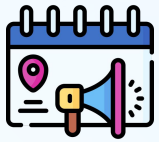
Alerting

Alerting

3.0.0

From Observability to Detection (What Becomes an Alert?)

Not every log is an alert → building *signal* from noise



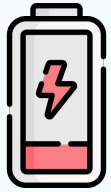
- **What events should be alerting:**
  - Login failures threshold (brute force)
  - Privilege escalation actions
  - Bulk exports
  - Anomalous regions or IPs

- **Alert types**:
  - Threshold-based (N events in T minutes)
  - Pattern-based (specific queries or endpoints)
  - Behavior-based (deviation from baseline)

Alerting

Best Practices for Alert Design

3.1.0

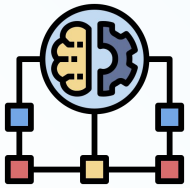


Avoiding alert fatigue

- Prioritize “actionable” over “perfect coverage”
- Use severity levels (info / low / med / high / critical)

Designing for investigation

- Include context (user, IP, trace ID, resource)
- Include runbook links / recommended actions



Map to Frameworks

- MITRE stages (recon, access, exfil)
- Incident response workflows

Alerting

Alert Design Steps

Detect malicious behavior, not isolated events.

Attacker Behavior | Required Telemetry | Atomic Detections | Correlation in Time & Context | Actionability

Step 1 - Start from Attacker Behavior

What is the attacker trying to do?

- Access credentials?
- Escalate privileges?
- Move laterally?
- Exfiltrate data?

→ Translate behavior into **observable signals**

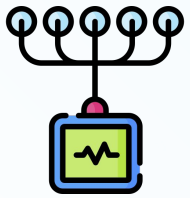
Alerting

Alert Design Steps

3.1.2

Detect malicious behavior, not isolated events.

Attacker Behavior → **Required Telemetry** → Atomic Detections → Correlation in Time & Context → Actionability



Step 2 - Identify Required Telemetry

Map each behavior to logs

Behavior	Telemetry
Identity abuse	IAM / API audit logs
Cluster abuse	Kubernetes audit logs
Runtime execution	Container / node logs
Lateral movement	Network flow logs
Data access	Storage / DB audit logs

Alerting

Alert Design Steps

Detect malicious behavior, not isolated events.

Attacker Behavior → Required Telemetry → **Atomic Detections** → Correlation in Time & Context → Actionability

Step 3 - Build Atomic Detections

Create small, reliable signals:

- *“Service account used from new IP”*
- *“Pod exec in prod namespace”*
- *“Privileged pod created”*
- *“Bulk data download”*

These should be **low-noise**, **high-confidence**.

Alerting

Alert Design Steps

Detect malicious behavior, not isolated events.

Attacker Behavior → Required Telemetry → Atomic Detections → **Correlation in Time & Context** → Actionability

Step 4 - Correlate in Time & Context

- *Link signals across layers*
  - *Same identity*
  - *Same workload*
  - *Same time window*
  - *Same resource*

Alerting

Alert Design Steps

Detect malicious behavior, not isolated events.

Attacker Behavior → Required Telemetry → Atomic Detections → Correlation in Time & Context → **Tuning**

Step 5 - Tune for Actionability

- *Suppress known automation*
- *Add thresholds & time windows*
- *Attach investigation context*
- *Map to MITRE & response playbooks*

Question ?

Additional Resources

1. OWASP Logging Best Practices
2. Guide to Computer Security Log Management

Hands-on time after a small break !

Detect reconnaissance activity within GKE

https://github.com/0x74696D/security_monitoring_tp