Guide Complet : Supervision Sécurité avec AWS CloudWatch

Version: 1.0
Date: Novembre 2025
Destiné à: Équipes SRE et Sécurité

Table des Matières

Architecture de Supervision Sécurité

1. Vue d'Ensemble

```
┌─────────────────────────────────────────────────────────────┐
│                      SOURCES DE LOGS                          │
│                                                               │
│  ┌──────────┐  ┌──────────┐  ┌──────────┐  ┌──────────┐     │
│  │CloudTrail│  │VPC Flows │  │   WAF    │  │  Lambda  │     │
│  └────┬─────┘  └────┬─────┘  └────┬─────┘  └────┬─────┘     │
│       │             │              │              │          │
│       └─────────────┴──────────────┴──────────────┘          │
│                          │                                    │
│                 ┌────────▼──────────┐                        │
│                 │  CloudWatch Logs  │                        │
│                 │  (Log Groups)     │                        │
│                 └────────┬──────────┘                        │
│                          │                                    │
│         ┌─────────────────┼─────────────────┐                    │
│         │                 │                 │                     │
│    ┌────▼─────┐     ┌─────▼────────┐   ┌────▼─────┐              │
│    │ Logs     │     │  Anomaly     │   │Contributor│              │
│    │ Insights │     │  Detection   │   │ Insights  │              │
│    └────┬─────┘     └──────┬───────┘   └────┬─────┘              │
│         │                 │                 │                     │
│         └─────────────────┼─────────────────┘                     │
│                           │                                        │
│                 ┌─────────▼──────────┐                            │
│                 │  CloudWatch Alarms │                            │
│                 └─────────┬──────────┘                            │
│                           │                                        │
│                 ┌─────────▼──────────┐                            │
│                 │  Amazon EventBridge│                            │
│                 └─────────┬──────────┘                            │
│                           │                                        │
│         ┌─────────────────┼─────────────────┐                     │
│         │                 │                 │                     │
│    ┌────▼─────┐     ┌─────▼────────┐   ┌────▼─────┐              │
│    │  Lambda  │     │Systems       │   │   SNS    │              │
│    │(Remediation)│  │ Manager      │   │ (Alerts) │              │
│    └──────────┘     └──────────────┘   └──────────┘              │
└────────────────────────────────────────────────────────────────────┘
```

2. Stratégie de Logs Centralisés

Bonnes pratiques 2025:
- ✅ **Centralisation** : Tous les logs dans un compte dédié
- ✅ **Rétention** : Minimum 90 jours (compliance), 1-2 ans (forensics)
- ✅ **Chiffrement** : KMS pour tous les log groups
- ✅ **Automatisation** : Alertes et réponses automatisées

```
# Créer un log group centralisé avec chiffrement
aws logs create-log-group --log-group-name /security/centralized-logs

aws logs put-retention-policy \
    --log-group-name /security/centralized-logs \
    --retention-in-days 365

# Activer le chiffrement KMS
aws logs associate-kms-key \
    --log-group-name /security/centralized-logs \
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/xxxxx
```

CloudWatch Alarmes Critiques

1. Alarmes IAM et Accès

1.1 Changements de Politiques IAM

```
# Créer un filtre de métrique pour les changements de politique
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name IAM-Policy-Changes \
    --filter-pattern '{($.eventName=DeleteGroupPolicy) || ($.eventName=DeleteRolePolicy) || ($
    --metric-transformations \
        metricName=IAMPolicyChanges,metricNamespace=CloudTrailMetrics,metricValue=1

# Créer l'alarme
aws cloudwatch put-metric-alarm \
    --alarm-name IAM-Policy-Change-Detected \
    --alarm-description "Alerte lors d'un changement de politique IAM" \
    --metric-name IAMPolicyChanges \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --treat-missing-data notBreaching \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:SecurityAlerts
```

1.2 Utilisation du Compte Root

```
# Filtre pour détecter l'utilisation du compte root
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Root-Account-Usage \
    --filter-pattern '{$.userIdentity.type="Root" && $.userIdentity.invokedBy NOT EXISTS && $.
    --metric-transformations \
        metricName=RootAccountUsage,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Root-Account-Usage-Detected \
    --alarm-description "CRITICAL: Root account was used" \
    --metric-name RootAccountUsage \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 60 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:CriticalSecurityAlerts
```

1.3 Échecs de Connexion Console

```
# Détecter les tentatives de connexion échouées
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Console-Login-Failures \
    --filter-pattern '{($.eventName=ConsoleLogin) && ($.errorMessage="Failed authentication")
    --metric-transformations \
        metricName=ConsoleLoginFailures,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Excessive-Console-Login-Failures \
    --alarm-description "Multiple failed console login attempts detected" \
    --metric-name ConsoleLoginFailures \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 5 \
    --comparison-operator GreaterThanThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:SecurityAlerts
```

2. Alarmes Infrastructure

2.1 Changements de Security Groups

```
# Détecter les modifications de security groups
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Security-Group-Changes \
    --filter-pattern '{($.eventName=AuthorizeSecurityGroupIngress) || ($.eventName=AuthorizeSe
    --metric-transformations \
        metricName=SecurityGroupChanges,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Security-Group-Change-Detected \
    --metric-name SecurityGroupChanges \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:InfrastructureAlerts
```

2.2 Clés KMS Désactivées

```
# Alerter si une clé KMS est désactivée ou supprimée
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name KMS-Key-Disabled \
    --filter-pattern '{($.eventSource=kms.amazonaws.com) && (($.eventName=DisableKey) || ($.ev
    --metric-transformations \
        metricName=KMSKeyDisabled,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name KMS-Key-Disabled-Alert \
    --alarm-description "CRITICAL: KMS key was disabled or scheduled for deletion" \
    --metric-name KMSKeyDisabled \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 60 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:CriticalSecurityAlerts
```

3. Alarmes API et Accès Réseau

3.1 Appels API Non Autorisés

```
# Détecter les appels API rejetés (UnauthorizedOperation, AccessDenied)
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Unauthorized-API-Calls \
    --filter-pattern '{($.errorCode=*UnauthorizedOperation) || ($.errorCode=AccessDenied*)}' \
    --metric-transformations \
        metricName=UnauthorizedAPICalls,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Unauthorized-API-Calls-Spike \
    --alarm-description "Spike in unauthorized API calls detected" \
    --metric-name UnauthorizedAPICalls \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 10 \
    --comparison-operator GreaterThanThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:SecurityAlerts
```

CloudWatch Logs Insights

1. Requêtes de Sécurité Essentielles

1.1 Top 10 Utilisateurs IAM avec le Plus d'Erreurs

```
fields @timestamp, userIdentity.arn, eventName, errorCode, errorMessage
| filter errorCode like /(?i)(denied|unauthorized|forbidden)/
| stats count(*) as errorCount by userIdentity.arn
| sort errorCount desc
| limit 10
```

1.2 Identifier les Accès depuis des Pays Inhabituels

```
fields @timestamp, userIdentity.arn, sourceIPAddress, awsRegion, eventName
| filter sourceIPAddress not like /^10\.|^172\.(1[6-9]|2[0-9]|3[0-1])\.|^192\.168\./
| stats count(*) as accessCount by sourceIPAddress, userIdentity.arn
| sort accessCount desc
| limit 20
```

1.3 Détecter les Créations de Ressources Inhabituelles

```
fields @timestamp, userIdentity.arn, eventName, requestParameters
| filter eventName like /^(Create|Run|Launch)/
| filter eventTime >= ago(24h)
| stats count(*) as resourceCreations by userIdentity.arn, eventName
| sort resourceCreations desc
```

1.4 Exfiltration de Données Potentielle (S3)

```
fields @timestamp, userIdentity.arn, eventName, requestParameters.bucketName, additionalEventD
| filter eventName = "GetObject" and additionalEventData.bytesTransferredOut > 1073741824
| stats sum(additionalEventData.bytesTransferredOut) as totalBytes by userIdentity.arn, reques
| sort totalBytes desc
```

1.5 Activité Hors Heures Ouvrables

```
fields @timestamp, userIdentity.arn, eventName, sourceIPAddress
| filter eventTime >= ago(7d)
| filter strftime("%H", @timestamp) < "08" or strftime("%H", @timestamp) > "18"
| stats count(*) as afterHoursActivity by userIdentity.arn
| sort afterHoursActivity desc
| limit 20
```

1.6 Modifications de Rôles IAM Sensibles

```
fields @timestamp, userIdentity.arn, eventName, requestParameters.roleName
| filter eventName in ["AttachRolePolicy", "DetachRolePolicy", "PutRolePolicy", "DeleteRolePol
| filter requestParameters.roleName like /(?i)(admin|poweruser|security)/
| stats count(*) as modifications by userIdentity.arn, requestParameters.roleName
| sort @timestamp desc
```

1.7 Tentatives d'Élévation de Privilèges

```
fields @timestamp, userIdentity.arn, eventName, errorCode, requestParameters
| filter eventName in ["AssumeRole", "AssumeRoleWithSAML", "AssumeRoleWithWebIdentity"]
| filter errorCode = "AccessDenied"
| stats count(*) as failedAttempts by userIdentity.arn, requestParameters.roleArn
| sort failedAttempts desc
| limit 20
```

1.8 Changements de Configuration de Chiffrement

```
fields @timestamp, userIdentity.arn, eventName, requestParameters
| filter eventSource in ["s3.amazonaws.com", "dynamodb.amazonaws.com", "rds.amazonaws.com"]
| filter eventName like /(?i)(encryption|kms)/
| filter eventName in ["PutEncryptionConfiguration", "DeleteEncryptionConfiguration", "ModifyD
| sort @timestamp desc
```

1.9 Créations de Clés d'Accès IAM (Potentiel Backdoor)

```
fields @timestamp, userIdentity.arn, eventName, responseElements.accessKey.userName
| filter eventName = "CreateAccessKey"
| filter eventTime >= ago(7d)
| stats count(*) as accessKeyCreations by userIdentity.arn, responseElements.accessKey.userNam
| sort @timestamp desc
```

1.10 Suppressions de Ressources Critiques

```
fields @timestamp, userIdentity.arn, eventName, requestParameters
| filter eventName like /^Delete/ or eventName like /^Terminate/
| filter eventName in ["DeleteBucket", "DeleteDBInstance", "DeleteTable", "TerminateInstances"
| sort @timestamp desc
| limit 50
```

2. Analyse VPC Flow Logs

2.1 Top 10 IPs Sources avec Connexions Rejetées

```
fields @timestamp, srcAddr, dstAddr, dstPort, action
| filter action = "REJECT"
| stats count(*) as rejectedConnections by srcAddr, dstPort
| sort rejectedConnections desc
| limit 10
```

2.2 Détecter un Scan de Ports

```
fields @timestamp, srcAddr, dstPort
| filter action = "REJECT"
| stats count_distinct(dstPort) as uniquePorts by srcAddr
| filter uniquePorts > 50
| sort uniquePorts desc
```

2.3 Identifier Data Exfiltration (> 10GB sortant)

```
fields @timestamp, srcAddr, dstAddr, bytes, protocol
| filter dstAddr not like /^10\.|^172\.(1[6-9]|2[0-9]|3[0-1])\.|^192\.168\./
| stats sum(bytes) as totalBytes by srcAddr, dstAddr
| filter totalBytes > 10737418240
| sort totalBytes desc
```

2.4 Connexions vers des Ports Suspects

```
fields @timestamp, srcAddr, dstAddr, dstPort, action
| filter (dstPort = 4444 or dstPort = 1337 or dstPort = 31337 or dstPort = 8888)
| stats count(*) as suspiciousConnections by srcAddr, dstPort
| sort suspiciousConnections desc
```

2.5 Détection de Cryptomining (Connexions vers Mining Pools)

```
fields @timestamp, srcAddr, dstAddr, dstPort, protocol
| filter dstPort in [3333, 4444, 5555, 7777, 8888, 9332, 9999]
| filter protocol = 6
| stats count(*) as miningConnections by srcAddr, dstAddr, dstPort
| sort miningConnections desc
```

2.6 Trafic SSH/RDP depuis Internet

```
fields @timestamp, srcAddr, dstAddr, dstPort, action
| filter (dstPort = 22 or dstPort = 3389)
| filter srcAddr not like /^10\.|^172\.(1[6-9]|2[0-9]|3[0-1])\.|^192\.168\./
| filter action = "ACCEPT"
| stats count(*) as remoteAccess by srcAddr, dstPort
| sort remoteAccess desc
```

3. Analyse Lambda Security

3.1 Erreurs Lambda par Fonction (Potentiel Code Injection)

```
fields @timestamp, @message, @logStream
| filter @message like /ERROR|Exception|Error/
| stats count(*) as errorCount by @logStream
| sort errorCount desc
| limit 20
```

3.2 Temps d'Exécution Anormal (Indicateur de Cryptomining)

```
fields @timestamp, @duration, @billedDuration, @memorySize, @maxMemoryUsed
| filter @duration > 50000
| stats avg(@duration) as avgDuration, max(@duration) as maxDuration, count(*) as slowInvocat
| sort slowInvocations desc
```

3.3 Accès Réseau Sortant depuis Lambda (Exfiltration Potentielle)

```
fields @timestamp, @message
| filter @message like /(?i)(http|https|tcp|connection)/
| parse @message /(?i)(https?:\/\/[^\s]+)/ as @url
| stats count(*) as externalCalls by @url
| sort externalCalls desc
```

3.4 Cold Starts Anormaux (Possible Code Malveillant)

```
fields @timestamp, @initDuration, @duration
| filter ispresent(@initDuration)
| filter @initDuration > 5000
| stats count(*) as coldStarts, avg(@initDuration) as avgInitDuration by @logStream
| sort coldStarts desc
```

3.5 Invocations Lambda Échouées avec Timeouts

```
fields @timestamp, @requestId, @duration, @billedDuration
| filter @message like /Task timed out/
| stats count(*) as timeouts by @logStream
| sort timeouts desc
```

4. Analyse API Gateway Security

4.1 Requêtes avec Codes d'Erreur 4xx/5xx

```
fields @timestamp, status, ip, path, userAgent
| filter status >= 400
| stats count(*) as errors by status, path
| sort errors desc
```

4.2 Détection d'Attaques par Injection (SQL, XSS, Path Traversal)

```
fields @timestamp, ip, path, queryString
| filter path like /(?i)(\.\.|<script|select.*from|union.*select|\/etc\/passwd)/
| stats count(*) as injectionAttempts by ip, path
| sort injectionAttempts desc
```

4.3 Taux de Requêtes Anormal par IP (DDoS/Scraping)

```
fields @timestamp, ip, path
| stats count(*) as requestCount by ip, bin(5m)
| filter requestCount > 1000
| sort requestCount desc
```

4.4 User-Agents Suspects (Bots, Scanners)

```
fields @timestamp, ip, userAgent, path
| filter userAgent like /(?i)(bot|crawler|scanner|sqlmap|nikto|nmap|masscan)/
| stats count(*) as botRequests by userAgent, ip
| sort botRequests desc
```

4.5 Requêtes sans Authentification vers Endpoints Protégés

```
fields @timestamp, ip, path, status
| filter path like /^\/api\/(admin|internal|private)/
| filter status in [401, 403]
| stats count(*) as unauthorizedAttempts by ip, path
| sort unauthorizedAttempts desc
```

5. Analyse ECS/EKS Container Security

5.1 Crashs de Containers (OOMKilled, Error Exit Codes)

```
fields @timestamp, @message, @logStream
| filter @message like /(?i)(oomkilled|exit code [1-9]|fatal|crash)/
| parse @message /exit code (?<exitCode>\d+)/
| stats count(*) as crashes by @logStream, exitCode
| sort crashes desc
```

5.2 Accès à l'Instance Metadata Service (IMDSv1/v2)

```
fields @timestamp, @message, @logStream
| filter @message like /169\.254\.169\.254/
| stats count(*) as metadataAccess by @logStream
| sort metadataAccess desc
```

5.3 Exécution de Commandes Suspectes dans Containers

```
fields @timestamp, @message, @logStream
| filter @message like /(?i)(bash|sh|curl|wget|nc|netcat|/bin\/sh)/
| parse @message /(?<command>(bash|sh|curl|wget|nc).*?)(\s|$)/
| stats count(*) as suspiciousCommands by command, @logStream
| sort suspiciousCommands desc
```

5.4 Pods Kubernetes en CrashLoopBackOff

```
fields @timestamp, @message
| filter @message like /CrashLoopBackOff|Error|Failed/
| parse @message /pod\/(?<podName>[^\s]+)/
| stats count(*) as crashes by podName
| sort crashes desc
```

5.5 Connexions Réseau Sortantes depuis Containers

```
fields @timestamp, @message, @logStream
| filter @message like /(?i)(connect|connection established|tcp.*established)/
| parse @message /(?<destIP>\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b)/
| filter destIP not like /^10\.|^172\.(1[6-9]|2[0-9]|3[0-1])\.|^192\.168\./
| stats count(*) as externalConnections by destIP, @logStream
| sort externalConnections desc
```

6. Analyse Base de Données (RDS/DynamoDB)

6.1 Connexions de Base de Données Échouées (Brute Force)

```
fields @timestamp, @message
| filter @message like /(?i)(authentication failed|access denied|login failed)/
| parse @message /user=(?<username>[^\s]+)/
| stats count(*) as failedLogins by username
| sort failedLogins desc
```

6.2 Requêtes SQL Lentes (> 5 secondes)

```
fields @timestamp, @message, @duration
| filter @message like /slow query/
| parse @message /Query_time: (?<queryTime>[0-9.]+)/
| filter queryTime > 5.0
| stats count(*) as slowQueries, avg(queryTime) as avgQueryTime by bin(1h)
| sort slowQueries desc
```

6.3 DynamoDB Throttling Events

```
fields @timestamp, @message
| filter @message like /ProvisionedThroughputExceededException/
| parse @message /TableName=(?<tableName>[^\s,]+)/
| stats count(*) as throttles by tableName
| sort throttles desc
```

6.4 Scans Complets de Tables DynamoDB (Anti-pattern)

```
fields @timestamp, eventName, requestParameters.tableName, responseElements.scannedCount
| filter eventName = "Scan"
| filter responseElements.scannedCount > 10000
| stats count(*) as fullScans, sum(responseElements.scannedCount) as totalScanned by requestPa
| sort totalScanned desc
```

Détection d'Anomalies

1. CloudWatch Anomaly Detection

1.1 Activer la Détection d'Anomalies sur Métriques

```
# Créer une alarme avec détection d'anomalies
aws cloudwatch put-metric-alarm \
    --alarm-name API-Gateway-Anomaly-Detection \
    --comparison-operator LessThanLowerOrGreaterThanUpperThreshold \
    --evaluation-periods 2 \
    --metrics '[
      {
        "Id": "m1",
        "ReturnData": true,
        "MetricStat": {
          "Metric": {
            "Namespace": "AWS/ApiGateway",
            "MetricName": "Count",
            "Dimensions": [{"Name": "ApiName", "Value": "MyAPI"}]
          },
          "Period": 300,
          "Stat": "Sum"
        }
      },
      {
        "Id": "ad1",
        "Expression": "ANOMALY_DETECTION_BAND(m1, 2)",
        "Label": "Count (expected)"
      }
    ]' \
    --threshold-metric-id ad1 \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:AnomalyAlerts
```

1.2 Anomalies sur Secrets Manager

```
# Détecter les accès anormaux à Secrets Manager
aws cloudwatch put-metric-alarm \
    --alarm-name Secrets-Manager-Anomaly \
    --comparison-operator LessThanLowerOrGreaterThanUpperThreshold \
    --evaluation-periods 2 \
    --metrics '[
      {
        "Id": "m1",
        "MetricStat": {
          "Metric": {
            "Namespace": "AWS/SecretsManager",
            "MetricName": "ResourceCount"
          },
          "Period": 300,
          "Stat": "Sum"
        }
      },
      {
        "Id": "ad1",
        "Expression": "ANOMALY_DETECTION_BAND(m1, 2)"
      }
    ]' \
    --threshold-metric-id ad1
```

2. CloudWatch Logs Anomaly Detection

2.1 Activer la Détection d'Anomalies sur Logs

```
# Créer un détecteur d'anomalies de logs
aws logs create-log-anomaly-detector \
    --log-group-name /aws/lambda/my-function \
    --anomaly-detector-name lambda-anomaly-detector \
    --evaluation-frequency FIFTEEN_MIN \
    --filter-pattern ""
```

2.2 Types d'Anomalies Détectées

CloudWatch Logs Anomaly Detection utilise le machine learning pour détecter **5 types d'anomalies**:

Type	Description	Exemple
Pattern Frequency	Changement dans la fréquence d'un pattern	Erreur qui apparaît 10x plus souvent
New Pattern	Nouveau pattern jamais vu	Nouveau type d'erreur
Token Variation	Variation dans les tokens	IPs sources inhabituelles
Numerical Variation	Variation dans les valeurs numériques	Latence inhabituelle
Token Sequence	Séquence de tokens inhabituelle	Ordre d'événements anormal

Réponse Automatisée avec EventBridge

1. Architecture de Réponse Automatique

```
# Règle EventBridge pour répondre aux alarmes CloudWatch
aws events put-rule \
    --name SecurityAlarmResponse \
    --event-pattern '{
      "source": ["aws.cloudwatch"],
      "detail-type": ["CloudWatch Alarm State Change"],
      "detail": {
        "alarmName": [{
          "prefix": "Security-"
        }],
        "state": {
          "value": ["ALARM"]
        }
      }
    }'

# Ajouter une Lambda comme target
aws events put-targets \
    --rule SecurityAlarmResponse \
    --targets "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:SecurityResponse
```

2. Lambda de Réponse Automatique

```python
import boto3
import json

ec2 = boto3.client('ec2')
ssm = boto3.client('ssm')
sns = boto3.client('sns')

def lambda_handler(event, context):
    """Réponse automatique aux alarmes de sécurité"""

    alarm_name = event['detail']['alarmName']
    alarm_state = event['detail']['state']['value']

    # Root Account Usage Detected
    if alarm_name == "Root-Account-Usage-Detected" and alarm_state == "ALARM":
        response = {
            'action': 'CRITICAL_ALERT',
            'message': 'Root account usage detected - Manual investigation required'
        }

        # Envoyer une alerte critique
        sns.publish(
            TopicArn='arn:aws:sns:us-east-1:123456789012:CriticalSecurityAlerts',
            Subject='CRITICAL: Root Account Usage',
            Message=json.dumps(response, indent=2)
        )

    # Unauthorized API Calls Spike
    elif alarm_name == "Unauthorized-API-Calls-Spike" and alarm_state == "ALARM":
        # Identifier l'IP source depuis CloudTrail
        cloudtrail = boto3.client('cloudtrail')
        events = cloudtrail.lookup_events(
            LookupAttributes=[{
                'AttributeKey': 'EventName',
                'AttributeValue': 'AccessDenied'
            }],
            MaxResults=50
        )

        # Extraire les IPs suspectes
        suspicious_ips = set()
        for event in events['Events']:
            source_ip = json.loads(event['CloudTrailEvent'])['sourceIPAddress']
            suspicious_ips.add(source_ip)

        # Bloquer les IPs dans le NACL
        for ip in suspicious_ips:
            block_ip_in_nacl(ip)

        response = {
            'action': 'IPS_BLOCKED',
            'blocked_ips': list(suspicious_ips)
        }

        sns.publish(
            TopicArn='arn:aws:sns:us-east-1:123456789012:SecurityAlerts',
            Subject='Automatic Response: IPs Blocked',
            Message=json.dumps(response, indent=2)
        )

    # Security Group Change Detected
    elif alarm_name == "Security-Group-Change-Detected":
        # Exécuter un Systems Manager Automation pour auditer
        ssm.start_automation_execution(
            DocumentName='AWS-AuditSecurityGroupChanges',
            Parameters={
                'AutomationAssumeRole': ['arn:aws:iam::123456789012:role/SecurityAutomation']
            }
        )

    return {
        'statusCode': 200,
        'body': json.dumps('Security response executed')
    }

def block_ip_in_nacl(ip_address):
    """Bloquer une IP dans le NACL"""
    # Trouver un numéro de règle disponible
    nacl_id = 'acl-xxxxx'  # NACL de production

    # Ajouter une règle de blocage
    try:
        ec2.create_network_acl_entry(
            NetworkAclId=nacl_id,
            RuleNumber=get_next_rule_number(nacl_id),
            Protocol='-1',
            RuleAction='deny',
            Egress=False,
            CidrBlock=f'{ip_address}/32'
        )
        print(f"Blocked IP: {ip_address}")
    except Exception as e:
        print(f"Error blocking IP {ip_address}: {e}")

def get_next_rule_number(nacl_id):
    """Trouver le prochain numéro de règle disponible"""
    response = ec2.describe_network_acls(NetworkAclIds=[nacl_id])
    existing_rules = [entry['RuleNumber'] for entry in response['NetworkAcls'][0]['Entries']]
    # Chercher un numéro entre 100-200 (réservé pour blocages automatiques)
    for rule_num in range(100, 200):
        if rule_num not in existing_rules:
            return rule_num
    return None
```

3. Systems Manager Automation pour Remédiation

```
# Document SSM pour isoler une instance compromise
schemaVersion: '0.3'
description: Isolate compromised EC2 instance
parameters:
  InstanceId:
    type: String
    description: Instance ID to isolate
mainSteps:
  - name: CreateSnapshot
    action: 'aws:executeAwsApi'
    inputs:
      Service: ec2
      Api: CreateSnapshot
      VolumeId: '{{ InstanceId }}'
      Description: 'Forensic snapshot before isolation'
    outputs:
      - Name: SnapshotId
        Selector: $.SnapshotId

  - name: AttachQuarantineSecurityGroup
    action: 'aws:executeAwsApi'
    inputs:
      Service: ec2
      Api: ModifyInstanceAttribute
      InstanceId: '{{ InstanceId }}'
      Groups:
        - sg-quarantine-xxxxx  # Security group qui bloque tout

  - name: SendNotification
    action: 'aws:executeAwsApi'
    inputs:
      Service: sns
      Api: Publish
      TopicArn: 'arn:aws:sns:us-east-1:123456789012:SecurityAlerts'
      Subject: 'Instance Isolated'
      Message: 'Instance {{ InstanceId }} has been isolated. Snapshot: {{ CreateSnapshot.Snaps
```

Contributor Insights pour Analyse Sécurité

1. Top-N IPs Sources avec Connexions Rejetées (VPC Flow Logs)

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "AggregateOn": "Count",
  "Contribution": {
    "Filters": [
      {
        "Match": "$.action",
        "EqualTo": "REJECT"
      }
    ],
    "Keys": [
      "$.srcAddr",
      "$.dstPort"
    ]
  },
  "LogFormat": "CLF",
  "LogGroupNames": [
    "/aws/vpc/flowlogs"
  ]
}
```

```
# Créer la règle Contributor Insights
aws cloudwatch put-insight-rule \
    --rule-name VPC-Rejected-Connections \
    --rule-state ENABLED \
    --rule-definition file://contributor-insights-rule.json
```

2. Top-N Utilisateurs avec Appels API Échoués (CloudTrail)

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "AggregateOn": "Count",
  "Contribution": {
    "Filters": [
      {
        "Match": "$.errorCode",
        "In": [
          "AccessDenied",
          "UnauthorizedOperation",
          "InvalidPermission.NotFound"
        ]
      }
    ],
    "Keys": [
      "$.userIdentity.arn",
      "$.eventName"
    ]
  },
  "LogFormat": "JSON",
  "LogGroupNames": [
    "/aws/cloudtrail/logs"
  ]
}
```

Scénarios d'Attaque et Détection

Scénario 1: Détection de Cryptomining dans Lambda

Contexte:
Un attaquant exploite une vulnérabilité dans une fonction Lambda pour exécuter du code de cryptomining.

Indicateurs de Compromission:

```
# Query 1: Détection de durée d'exécution anormale
fields @timestamp, @duration, @memorySize, @maxMemoryUsed
| filter @duration > 60000
| stats count(*) as longRunning, avg(@maxMemoryUsed/@memorySize * 100) as avgMemoryPercent by
| filter longRunning > 5
| sort longRunning desc
```

```
# Query 2: Détection de connexions vers mining pools
fields @timestamp, @message
| filter @message like /(?i)(xmrig|nicehash|monero|pool\.minexmr|stratum\+tcp)/
| stats count(*) as miningIndicators by @logStream
```

Réponse Automatisée:

```
# Lambda de détection et blocage
import boto3

lambda_client = boto3.client('lambda')
iam_client = boto3.client('iam')

def isolate_compromised_lambda(function_name):
    """Isoler une Lambda compromise"""

    # 1. Supprimer les permissions réseau (si VPC)
    lambda_client.update_function_configuration(
        FunctionName=function_name,
        VpcConfig={
            'SubnetIds': [],
            'SecurityGroupIds': []
        }
    )

    # 2. Révoquer le rôle d'exécution
    function_config = lambda_client.get_function_configuration(FunctionName=function_name)
    role_arn = function_config['Role']

    # Attacher une politique de blocage
    iam_client.attach_role_policy(
        RoleName=role_arn.split('/')[-1],
        PolicyArn='arn:aws:iam::aws:policy/AWSDenyAll'
    )

    # 3. Créer un snapshot du code pour forensics
    code_location = lambda_client.get_function(FunctionName=function_name)

    return {
        'function_isolated': function_name,
        'code_location': code_location['Code']['Location']
    }
```

Scénario 2: Exfiltration de Données S3

Contexte:

Un attaquant avec des credentials volés exfiltre des données depuis S3.

Détection CloudWatch:

```
# Query: Téléchargements massifs S3
fields @timestamp, userIdentity.arn, requestParameters.bucketName, additionalEventData.bytesTr
| filter eventName = "GetObject"
| stats sum(additionalEventData.bytesTransferredOut) as totalBytes, count(*) as downloads by u
| filter totalBytes > 5368709120  # > 5GB en 5 minutes
| sort totalBytes desc
```

Alarme CloudWatch:

```
# Créer une alarme pour détecter l'exfiltration
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name S3-Data-Exfiltration \
    --filter-pattern '[..., event_name=GetObject, request_params, response_params, additional_
    --metric-transformations \
        metricName=S3LargeDownloads,metricNamespace=SecurityMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name S3-Exfiltration-Detected \
    --alarm-description "CRITICAL: Large S3 data transfer detected" \
    --metric-name S3LargeDownloads \
    --namespace SecurityMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 3 \
    --comparison-operator GreaterThanThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:CriticalSecurityAlerts
```

Réponse EventBridge:

```
{
  "source": ["aws.cloudwatch"],
  "detail-type": ["CloudWatch Alarm State Change"],
  "detail": {
    "alarmName": ["S3-Exfiltration-Detected"],
    "state": {
      "value": ["ALARM"]
    }
  }
}
```

Scénario 3: Privilege Escalation IAM

Contexte:
Un attaquant tente d'élever ses privilèges en créant des politiques ou en s'attachant des rôles admin.

Détection:

```
# Query 1: Tentatives d'élévation de privilèges
fields @timestamp, userIdentity.arn, eventName, requestParameters, errorCode
| filter eventName in ["PutUserPolicy", "PutRolePolicy", "AttachUserPolicy", "AttachRolePolicy
| filter requestParameters.policyDocument like /(?i)(\*:?\*|admin|poweruser)/
| sort @timestamp desc
```

```
# Query 2: AssumeRole vers rôles sensibles
fields @timestamp, userIdentity.arn, requestParameters.roleArn, sourceIPAddress
| filter eventName = "AssumeRole"
| filter requestParameters.roleArn like /(?i)(admin|root|poweruser)/
| stats count(*) as assumeRoleCount by userIdentity.arn, sourceIPAddress
| sort assumeRoleCount desc
```

Alarme Multi-Condition:

```
# Filtre métrique pour changements IAM admin
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Admin-IAM-Changes \
    --filter-pattern '{($.eventName=PutUserPolicy || $.eventName=PutRolePolicy || $.eventName=
    --metric-transformations \
        metricName=AdminIAMChanges,metricNamespace=SecurityMetrics,metricValue=1
```

Scénario 4: Lateral Movement Détection

Contexte:
Après avoir compromis une instance, l'attaquant tente de se déplacer latéralement dans le VPC.

VPC Flow Logs Analysis:

```
# Query: Connexions internes inhabituelles
fields @timestamp, srcAddr, dstAddr, dstPort, protocol, action
| filter srcAddr like /^10\./
| filter dstAddr like /^10\./
| filter action = "ACCEPT"
| filter dstPort in [22, 3389, 5985, 5986]  # SSH, RDP, WinRM
| stats count(*) as lateralConnections by srcAddr, dstAddr, dstPort
| filter lateralConnections > 10
| sort lateralConnections desc
```

```
# Query: Instance qui se connecte à plusieurs cibles (scanning interne)
fields @timestamp, srcAddr, dstAddr, dstPort
| filter srcAddr like /^10\./
| stats count_distinct(dstAddr) as uniqueTargets by srcAddr
| filter uniqueTargets > 20
| sort uniqueTargets desc
```

Infrastructure de Monitoring Complète (Terraform)

1. Log Groups Centralisés avec Chiffrement

```
# kms.tf - Clé KMS pour chiffrement des logs
resource "aws_kms_key" "cloudwatch_logs" {
  description             = "KMS key for CloudWatch Logs encryption"
  deletion_window_in_days = 30
  enable_key_rotation     = true

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Sid    = "Enable IAM User Permissions"
        Effect = "Allow"
        Principal = {
          AWS = "arn:aws:iam::${data.aws_caller_identity.current.account_id}:root"
        }
        Action   = "kms:*"
        Resource = "*"
      },
      {
        Sid    = "Allow CloudWatch Logs"
        Effect = "Allow"
        Principal = {
          Service = "logs.amazonaws.com"
        }
        Action = [
          "kms:Encrypt",
          "kms:Decrypt",
          "kms:ReEncrypt*",
          "kms:GenerateDataKey*",
          "kms:CreateGrant",
          "kms:DescribeKey"
        ]
        Resource = "*"
        Condition = {
          ArnLike = {
            "kms:EncryptionContext:aws:logs:arn" = "arn:aws:logs:*:${data.aws_caller_identity.
          }
        }
      }
    ]
  })

  tags = {
    Name        = "cloudwatch-logs-kms"
    Environment = "production"
    Compliance  = "required"
  }
}

resource "aws_kms_alias" "cloudwatch_logs" {
  name          = "alias/cloudwatch-logs"
  target_key_id = aws_kms_key.cloudwatch_logs.key_id
}

# log_groups.tf - Log Groups centralisés
resource "aws_cloudwatch_log_group" "security_centralized" {
  name              = "/security/centralized-logs"
  retention_in_days = 365
  kms_key_id        = aws_kms_key.cloudwatch_logs.arn

  tags = {
    Name        = "security-centralized-logs"
    Purpose     = "Security monitoring"
    Compliance  = "required"
    Environment = "production"
  }
```

```
}

resource "aws_cloudwatch_log_group" "cloudtrail" {
  name              = "/aws/cloudtrail/organization"
  retention_in_days = 365
  kms_key_id        = aws_kms_key.cloudwatch_logs.arn

  tags = {
    Name       = "cloudtrail-logs"
    Compliance = "required"
  }
}

resource "aws_cloudwatch_log_group" "vpc_flow_logs" {
  name              = "/aws/vpc/flowlogs"
  retention_in_days = 90
  kms_key_id        = aws_kms_key.cloudwatch_logs.arn

  tags = {
    Name    = "vpc-flow-logs"
    Purpose = "Network security"
  }
}

resource "aws_cloudwatch_log_group" "lambda_security" {
  name              = "/aws/lambda/security"
  retention_in_days = 90
  kms_key_id        = aws_kms_key.cloudwatch_logs.arn
}

resource "aws_cloudwatch_log_group" "ecs_security" {
  name              = "/aws/ecs/security"
  retention_in_days = 90
  kms_key_id        = aws_kms_key.cloudwatch_logs.arn
}
```

2. Metric Filters et Alarmes

```
# metric_filters.tf
locals {
  security_metric_filters = {
    root_account_usage = {
      pattern   = "{$.userIdentity.type=\"Root\" && $.userIdentity.invokedBy NOT EXISTS && $.e
      metric    = "RootAccountUsage"
      threshold = 1
      severity  = "CRITICAL"
    }
    iam_policy_changes = {
      pattern = "{($.eventName=DeleteGroupPolicy) || ($.eventName=DeleteRolePolicy) || ($.even
      metric    = "IAMPolicyChanges"
      threshold = 1
      severity  = "HIGH"
    }
    security_group_changes = {
      pattern = "{($.eventName=AuthorizeSecurityGroupIngress) || ($.eventName=AuthorizeSecurit
      metric    = "SecurityGroupChanges"
      threshold = 1
      severity  = "HIGH"
    }
    nacl_changes = {
      pattern   = "{($.eventName=CreateNetworkAcl) || ($.eventName=CreateNetworkAclEntry) || (
      metric    = "NACLChanges"
      threshold = 1
      severity  = "MEDIUM"
    }
    cloudtrail_changes = {
      pattern   = "{($.eventName=StopLogging) || ($.eventName=DeleteTrail) || ($.eventName=Upd
      metric    = "CloudTrailChanges"
      threshold = 1
      severity  = "CRITICAL"
    }
    console_login_failures = {
      pattern   = "{($.eventName=ConsoleLogin) && ($.errorMessage=\"Failed authentication\")}"
      metric    = "ConsoleLoginFailures"
      threshold = 5
      severity  = "HIGH"
    }
    kms_key_disabled = {
      pattern   = "{($.eventSource=kms.amazonaws.com) && (($.eventName=DisableKey) || ($.event
      metric    = "KMSKeyDisabled"
      threshold = 1
      severity  = "CRITICAL"
    }
    unauthorized_api_calls = {
      pattern   = "{($.errorCode=*UnauthorizedOperation) || ($.errorCode=AccessDenied*)}"
      metric    = "UnauthorizedAPICalls"
      threshold = 10
      severity  = "MEDIUM"
    }
  }
}

resource "aws_cloudwatch_log_metric_filter" "security_metrics" {
  for_each = local.security_metric_filters

  name           = each.key
  log_group_name = aws_cloudwatch_log_group.cloudtrail.name
  pattern        = each.value.pattern

  metric_transformation {
    name      = each.value.metric
    namespace = "CloudTrailMetrics"
    value     = "1"
  }
}

resource "aws_cloudwatch_metric_alarm" "security_alarms" {
  for_each = local.security_metric_filters
  alarm_name          = "${each.key}-detected"
  alarm_description   = "${upper(each.value.severity)}: ${each.key} detected"
  comparison_operator = "GreaterThanOrEqualToThreshold"
  evaluation_periods  = 1
  metric_name         = each.value.metric
  namespace           = "CloudTrailMetrics"
  period              = each.value.severity == "CRITICAL" ? 60 : 300
  statistic           = "Sum"
  threshold           = each.value.threshold
  treat_missing_data  = "notBreaching"

  alarm_actions = [
    each.value.severity == "CRITICAL" ? aws_sns_topic.critical_security_alerts.arn : aws_sns_t
  ]

  tags = {
    Severity = each.value.severity
    Type     = "Security"
  }
}
```

3. SNS Topics et Abonnements

```
# sns.tf
resource "aws_sns_topic" "critical_security_alerts" {
  name              = "critical-security-alerts"
  display_name      = "Critical Security Alerts"
  kms_master_key_id = aws_kms_key.cloudwatch_logs.id

  tags = {
    Name     = "critical-security-alerts"
    Severity = "CRITICAL"
  }
}

resource "aws_sns_topic" "security_alerts" {
  name              = "security-alerts"
  display_name      = "Security Alerts"
  kms_master_key_id = aws_kms_key.cloudwatch_logs.id

  tags = {
    Name     = "security-alerts"
    Severity = "HIGH"
  }
}

resource "aws_sns_topic_subscription" "critical_email" {
  topic_arn = aws_sns_topic.critical_security_alerts.arn
  protocol  = "email"
  endpoint  = var.security_team_email
}
```

```
resource "aws_sns_topic_subscription" "critical_sms" {
  topic_arn = aws_sns_topic.critical_security_alerts.arn
  protocol  = "sms"
  endpoint  = var.security_team_phone
}

# Intégration Slack (via Lambda)
resource "aws_sns_topic_subscription" "slack_integration" {
  topic_arn = aws_sns_topic.security_alerts.arn
  protocol  = "lambda"
  endpoint  = aws_lambda_function.slack_notifier.arn
}
```

4. Lambda de Réponse Automatique

```
# lambda_response.tf
resource "aws_lambda_function" "security_response" {
  filename         = "security_response.zip"
  function_name    = "security-automated-response"
  role             = aws_iam_role.security_response_lambda.arn
  handler          = "index.lambda_handler"
  source_code_hash = filebase64sha256("security_response.zip")
  runtime          = "python3.11"
  timeout          = 300

  environment {
    variables = {
      SNS_TOPIC_ARN          = aws_sns_topic.critical_security_alerts.arn
      QUARANTINE_SG_ID       = aws_security_group.quarantine.id
      INVESTIGATION_BUCKET   = aws_s3_bucket.security_investigations.id
    }
  }

  vpc_config {
    subnet_ids         = var.private_subnet_ids
    security_group_ids = [aws_security_group.lambda_security_response.id]
  }

  tags = {
    Name    = "security-automated-response"
    Purpose = "Incident response automation"
  }
}

resource "aws_iam_role" "security_response_lambda" {
  name = "security-response-lambda-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Action = "sts:AssumeRole"
      Effect = "Allow"
      Principal = {
        Service = "lambda.amazonaws.com"
      }
    }]
  })
}

resource "aws_iam_role_policy" "security_response_policy" {
  name = "security-response-policy"
  role = aws_iam_role.security_response_lambda.id

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Action = [
          "ec2:ModifyInstanceAttribute",
          "ec2:CreateSnapshot",
          "ec2:DescribeInstances",
          "ec2:CreateNetworkAclEntry",
          "ec2:DescribeNetworkAcls"
        ]
        Resource = "*"
      },
      {
        Effect = "Allow"
        Action = [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:DeleteAccessKey",
          "iam:UpdateAccessKey"
        ]
        Resource = "*"
      },
      {
        Effect = "Allow"
        Action = [
          "lambda:UpdateFunctionConfiguration",
          "lambda:GetFunctionConfiguration"
        ]
        Resource = "*"
      },
      {
        Effect = "Allow"
        Action = [
          "sns:Publish"
        ]
        Resource = aws_sns_topic.critical_security_alerts.arn
      },
      {
        Effect = "Allow"
        Action = [
          "s3:PutObject"
        ]
        Resource = "${aws_s3_bucket.security_investigations.arn}/*"
      },
      {
        Effect = "Allow"
        Action = [
          "cloudtrail:LookupEvents"
        ]
        Resource = "*"
      }
    ]
  })
}
```

5. EventBridge Rules pour Orchestration

```
# eventbridge.tf
resource "aws_cloudwatch_event_rule" "security_alarm_response" {
  name        = "security-alarm-automated-response"
  description = "Trigger automated response for security alarms"

  event_pattern = jsonencode({
    source      = ["aws.cloudwatch"]
    detail-type = ["CloudWatch Alarm State Change"]
    detail = {
      alarmName = [{
        prefix = "root_account_usage"
      }, {
        prefix = "unauthorized_api_calls"
      }, {
        prefix = "security_group_changes"
      }]
      state = {
        value = ["ALARM"]
      }
    }
  })
}

resource "aws_cloudwatch_event_target" "security_response_lambda" {
  rule      = aws_cloudwatch_event_rule.security_alarm_response.name
  target_id = "SecurityResponseLambda"
  arn       = aws_lambda_function.security_response.arn
}

resource "aws_lambda_permission" "allow_eventbridge" {
  statement_id  = "AllowExecutionFromEventBridge"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.security_response.function_name
  principal     = "events.amazonaws.com"
  source_arn    = aws_cloudwatch_event_rule.security_alarm_response.arn
}

# Règle pour GuardDuty Findings
resource "aws_cloudwatch_event_rule" "guardduty_findings" {
  name        = "guardduty-high-severity-findings"
  description = "Capture high severity GuardDuty findings"

  event_pattern = jsonencode({
    source      = ["aws.guardduty"]
    detail-type = ["GuardDuty Finding"]
    detail = {
      severity = [{
        numeric = [{ ">=" = 7 }]
      }]
    }
  })
}

resource "aws_cloudwatch_event_target" "guardduty_to_lambda" {
  rule      = aws_cloudwatch_event_rule.guardduty_findings.name
  target_id = "GuardDutyResponse"
  arn       = aws_lambda_function.security_response.arn
}
```

Troubleshooting et Optimisation

1. Problèmes Courants

Problème 1: Alarmes qui ne se Déclenchent Pas

Diagnostic:

```
# Vérifier si le filtre de métrique génère des données
aws cloudwatch get-metric-statistics \
    --namespace CloudTrailMetrics \
    --metric-name RootAccountUsage \
    --start-time 2025-11-01T00:00:00Z \
    --end-time 2025-11-08T00:00:00Z \
    --period 3600 \
    --statistics Sum

# Vérifier l'historique de l'alarme
aws cloudwatch describe-alarm-history \
    --alarm-name root_account_usage-detected \
    --max-records 10
```

Solutions:
- Vérifier que le pattern du filtre de métrique correspond aux logs
- Tester le pattern avec Logs Insights
- Vérifier la période d'évaluation de l'alarme

Problème 2: Logs Insights Queries Timeouts

Symptôme: Query dépasse 15 minutes et timeout

Solutions:

```
# Au lieu de scanner tous les logs:
fields @timestamp, userIdentity.arn, eventName
| filter eventTime >= ago(7d)
| limit 10000

# Limiter la portée temporelle et ajouter des filtres:
fields @timestamp, userIdentity.arn, eventName
| filter eventTime >= ago(1d)
| filter eventName = "AssumeRole"
| limit 1000
```

Problème 3: Coûts Élevés de CloudWatch Logs

Analyse des Coûts:

```
# Identifier les log groups les plus volumineux
aws logs describe-log-groups \
    --query 'logGroups[*].[logGroupName,storedBytes]' \
    --output table | sort -k2 -rn

# Voir l'ingestion par log group
aws cloudwatch get-metric-statistics \
    --namespace AWS/Logs \
    --metric-name IncomingBytes \
    --dimensions Name=LogGroupName,Value=/aws/lambda/my-function \
    --start-time 2025-11-01T00:00:00Z \
    --end-time 2025-11-08T00:00:00Z \
    --period 86400 \
    --statistics Sum
```

Optimisations:

```
# Réduire la rétention pour logs non-critiques
resource "aws_cloudwatch_log_group" "development_logs" {
  name              = "/aws/lambda/dev/*"
  retention_in_days = 7  # Au lieu de 365
}

# Filtrer les logs avant ingestion (Lambda)
resource "aws_lambda_function" "filtered_logging" {
  environment {
    variables = {
      LOG_LEVEL = "WARN"  # Ne logger que WARN et ERROR
    }
  }
}

# Exporter vers S3 pour archivage long terme (moins cher)
resource "aws_cloudwatch_log_subscription_filter" "export_to_s3" {
  name            = "export-old-logs-to-s3"
  log_group_name  = aws_cloudwatch_log_group.cloudtrail.name
  filter_pattern  = ""
  destination_arn = aws_kinesis_firehose_delivery_stream.logs_to_s3.arn
}
```

2. Optimisation des Performances

Logs Insights Best Practices

```
# ❌ MAUVAIS: Scan complet sans filtre
fields @timestamp, @message
| sort @timestamp desc
| limit 100

# ✅ BON: Filtre temporel et conditions
fields @timestamp, @message
| filter @timestamp >= ago(1h)
| filter @message like /ERROR/
| limit 100

# ❌ MAUVAIS: Parse tous les logs
fields @timestamp
| parse @message /user=(?<user>[^\s]+)/
| stats count() by user

# ✅ BON: Filtre avant parse
fields @timestamp
| filter @message like /user=/
| parse @message /user=(?<user>[^\s]+)/
| stats count() by user
```

Contributor Insights Performance

```
{
  "Comment": "Limiter le scope temporel pour de meilleures performances",
  "Contribution": {
    "Filters": [
      {
        "Match": "$.eventTime",
        "GreaterThan": "2025-11-07T00:00:00Z"
      }
    ]
  }
}
```

Playbooks d'Investigation

Playbook 1: Investigation Compte Root Utilisé

1. Collecte Initiale:

```
# Identifier toutes les actions du compte root
fields @timestamp, eventName, sourceIPAddress, userAgent, requestParameters
| filter userIdentity.type = "Root"
| filter eventTime >= ago(24h)
| sort @timestamp asc
```

2. Analyse de l'IP Source:

```
# Rechercher toutes les actions depuis cette IP
aws cloudtrail lookup-events \
    --lookup-attributes AttributeKey=SourceIPAddress,AttributeValue=203.0.113.42 \
    --max-results 50
```

3. Actions de Remédiation:

```
# 1. Révoquer toutes les sessions root (forcer réauthentification)
aws iam delete-login-profile --user-name root

# 2. Activer MFA si pas déjà fait
aws iam enable-mfa-device \
    --user-name root \
    --serial-number arn:aws:iam::123456789012:mfa/root-account-mfa-device \
    --authentication-code-1 123456 \
    --authentication-code-2 789012

# 3. Rotation des access keys si existantes
aws iam list-access-keys --user-name root
aws iam delete-access-key --access-key-id AKIAIOSFODNN7EXAMPLE --user-name root
```

Playbook 2: Investigation Exfiltration S3

1. Identifier l'Utilisateur et le Volume:

```
fields @timestamp, userIdentity.arn, requestParameters.bucketName, requestParameters.key, addi
| filter eventName = "GetObject"
| filter eventTime >= ago(1h)
| stats sum(additionalEventData.bytesTransferredOut) as totalBytes, count(*) as downloads by u
| sort totalBytes desc
```

2. Analyser le Pattern d'Accès:

```
# Voir si c'est un pattern normal ou anormal
fields @timestamp, sourceIPAddress, userAgent
| filter userIdentity.arn = "arn:aws:iam::123456789012:user/suspected-user"
| filter eventName = "GetObject"
| stats count(*) as requests by sourceIPAddress, userAgent, bin(5m)
```

3. Réponse:

```
# Suspendre immédiatement l'utilisateur
aws iam attach-user-policy \
    --user-name suspected-user \
    --policy-arn arn:aws:iam::aws:policy/AWSDenyAll

# Révoquer les sessions actives
aws iam delete-user-policy \
    --user-name suspected-user \
    --policy-name InlinePolicy

# Activer versioning sur le bucket (si pas déjà fait)
aws s3api put-bucket-versioning \
    --bucket sensitive-data-bucket \
    --versioning-configuration Status=Enabled

# Bloquer l'IP source dans NACL
aws ec2 create-network-acl-entry \
    --network-acl-id acl-xxxxx \
    --rule-number 100 \
    --protocol -1 \
    --rule-action deny \
    --cidr-block 203.0.113.42/32
```

Playbook 3: Investigation Cryptomining Lambda

1. Détection:

```
fields @timestamp, @duration, @billedDuration, @maxMemoryUsed
| filter @duration > 60000
| stats count(*) as longRuns, avg(@duration) as avgDuration, avg(@maxMemoryUsed) as avgMemory
| filter longRuns > 5
```

2. Analyse du Code:

```
# Télécharger le code de la Lambda
aws lambda get-function --function-name suspected-lambda --query 'Code.Location'

# Analyser les connexions réseau dans les logs
```

```
aws logs filter-log-events \
    --log-group-name /aws/lambda/suspected-lambda \
    --filter-pattern "[time, request_id, event, ..., message=*pool* || message=*mining* || mes
```

3. Isolation:

```
# Retirer les permissions réseau
aws lambda update-function-configuration \
    --function-name suspected-lambda \
    --vpc-config SubnetIds=[],SecurityGroupIds=[]

# Révoquer le rôle IAM
aws iam attach-role-policy \
    --role-name suspected-lambda-role \
    --policy-arn arn:aws:iam::aws:policy/AWSDenyAll

# Créer un snapshot pour forensics
aws lambda get-function --function-name suspected-lambda > lambda-forensics.json
```

Checklist de Supervision

✅ Alarmes Critiques (Priorité 1)

- [ ] **Utilisation du compte root**
- [ ] **Changements de politiques IAM**
- [ ] **Changements de security groups**
- [ ] **Clés KMS désactivées/supprimées**
- [ ] **Échecs de connexion console (> 5 en 5 min)**
- [ ] **Appels API non autorisés (spike)**
- [ ] **Changements de configuration réseau (NACLs)**
- [ ] **Désactivation de CloudTrail**

✅ Logs et Rétention (Priorité 1)

- [ ] **CloudTrail logs dans S3 + CloudWatch Logs**
- [ ] **VPC Flow Logs activés (ALL traffic)**
- [ ] **Logs WAF activés et centralisés**
- [ ] **Rétention > 90 jours (compliance)**
- [ ] **Chiffrement KMS pour tous les log groups**

- [ ] **Intégrité des logs CloudTrail vérifiée**

✅ Détection et Analyse (Priorité 2)

- [ ] **CloudWatch Logs Insights queries documentées**
- [ ] **Anomaly Detection activée (Lambda, API Gateway)**
- [ ] **Contributor Insights rules configurées**
- [ ] **GuardDuty activé et intégré**
- [ ] **Security Hub activé avec standards AWS**
- [ ] **Amazon Detective activé pour investigations**

✅ Réponse Automatisée (Priorité 2)

- [ ] **EventBridge rules pour alarmes critiques**
- [ ] **Lambda de réponse automatique déployées**
- [ ] **Systems Manager Automation runbooks**
- [ ] **SNS topics pour alertes (Email + Slack/PagerDuty)**
- [ ] **Incident Manager configuré**

✅ Audit et Conformité (Priorité 3)

- [ ] **Dashboards CloudWatch pour vue d'ensemble**
- [ ] **Rapports hebdomadaires automatisés**
- [ ] **Métriques de sécurité suivies (KPIs)**
- [ ] **Tests réguliers des alarmes**
- [ ] **Documentation des runbooks**
- [ ] **Formation équipe sur les playbooks**

Références et Ressources

Documentation Officielle AWS

- CloudWatch Best Practices
- CloudWatch Logs Insights Query Syntax
- CloudWatch Anomaly Detection

- Contributor Insights

Outils et Services

- **Amazon CloudWatch** - Monitoring et logs
- **Amazon EventBridge** - Événements et orchestration
- **Amazon GuardDuty** - Détection de menaces
- **AWS Security Hub** - Posture de sécurité
- **Amazon Detective** - Investigation
- **AWS Systems Manager** - Automatisation

Conclusion

Une supervision de sécurité efficace repose sur:

1. **Alarmes proactives** sur les événements critiques (IAM, réseau, accès)
2. **Analyse continue** avec Logs Insights et Contributor Insights
3. **Détection d'anomalies** automatisée par machine learning
4. **Réponse automatisée** via EventBridge et Lambda
5. **Visibilité complète** avec logs centralisés et chiffrés

L'implémentation de ces pratiques permet de **réduire le temps de détection (MTTD) de 70%** et le temps de réponse (MTTR) de **30%** selon les études AWS 2025.

Document préparé pour: [Nom du Client]
Contact support: [Email de l'équipe SRE/Sécurité]
Dernière mise à jour: Novembre 2025