Sécurité Cloud - OWASP Cloud-Native Security

m2iformation.fr

Sécurité Cloud

OWASP Cloud-Native Application Security

Risques et Bonnes Pratiques

Objectifs du Module

- Comprendre les spécificités de la sécurité Cloud
- Identifier les risques OWASP Cloud-Native Top 10
- Sécuriser les environnements AWS, Azure et GCP
- Protéger les conteneurs et orchestrateurs
- Implémenter l'Infrastructure as Code sécurisée

Partie 1

Introduction à la Sécurité Cloud

Modèle de Responsabilité Partagée

```
+---------------------------------------------------------------+
|                      RESPONSABILITE                           |
+---------------------------------------------------------------+
|                                                               |
|  CLIENT          Données, Applications, Configuration,        |
|                  Identités, Chiffrement côté client           |
|                                                               |
+---------------------------------------------------------------+
|                                                               |
|  FOURNISSEUR     Infrastructure physique, Réseau,             |
|  CLOUD           Hyperviseur, Stockage, Compute               |
|                                                               |
+---------------------------------------------------------------+

IaaS : Client gère OS, middleware, applications
PaaS : Client gère uniquement applications et données
SaaS : Client gère uniquement données et accès
```

Les 3 Grands Fournisseurs

Aspect	AWS	Azure	GCP
IAM	IAM, STS	Azure AD, RBAC	Cloud IAM
Secrets	Secrets Manager	Key Vault	Secret Manager
Logs	CloudTrail	Monitor	Cloud Audit Logs
Réseau	VPC, Security Groups	VNet, NSG	VPC, Firewall Rules
Conteneurs	EKS, ECS	AKS	GKE

Les principes de sécurité sont identiques, seuls les outils diffèrent.

Différences avec la Sécurité On-Premise

On-Premise	Cloud
Périmètre défini	Périmètre flou
Contrôle physique	Confiance au fournisseur
Investissement initial	Paiement à l'usage
Scalabilité limitée	Scalabilité infinie
Configuration manuelle	Infrastructure as Code
Mises à jour planifiées	Mises à jour continues

Le Cloud introduit de nouveaux vecteurs d'attaque liés à la configuration et aux API.

Partie 2

OWASP Cloud-Native Top 10

CNAS-1 : Insecure Cloud, Container or Orchestration Configuration

Description

Mauvaise configuration des services Cloud, conteneurs ou orchestrateurs exposant des données ou des fonctionnalités.

Exemples

- Bucket S3 public contenant des données sensibles
- Base de données accessible depuis Internet
- Conteneur exécuté en tant que root

CNAS-1 : Exemples de Mauvaise Configuration

S3 Bucket Public (AWS)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::mon-bucket/*"
        }
    ]
}
```

Cette policy rend TOUS les objets du bucket accessibles publiquement.

CNAS-1 : Configuration Sécurisée S3

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPublicAccess",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::mon-bucket",
                "arn:aws:s3:::mon-bucket/*"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

Cette policy refuse tout accès non HTTPS.

CNAS-1 : Sécurisation des Conteneurs

```dockerfile
# MAUVAIS : Conteneur root avec image non vérifiée
FROM ubuntu:latest
USER root
RUN apt-get update && apt-get install -y curl

# BON : Image minimale, utilisateur non-root, version fixe
FROM alpine:3.18
RUN addgroup -S appgroup && adduser -S appuser -G appgroup
USER appuser
WORKDIR /app
COPY --chown=appuser:appgroup . .
```

Principes :

- Utiliser des images minimales (alpine, distroless)
- Ne jamais exécuter en root
- Fixer les versions des images

CNAS-2 : Injection Flaws

Description

Injection de commandes, requêtes ou code via les entrées utilisateur dans les environnements Cloud-Native.

Vecteurs spécifiques au Cloud

- Injection dans les fonctions serverless (Lambda, Functions)
- Injection via les variables d'environnement
- Injection dans les templates Infrastructure as Code
- Injection via les métadonnées de conteneurs

CNAS-2 : Injection dans Lambda

```python
# VULNERABLE : Injection de commande dans Lambda
import subprocess
import json

def lambda_handler(event, context):
    filename = event['queryStringParameters']['file']

    # VULNERABLE : entrée utilisateur dans une commande shell
    result = subprocess.run(
        f"cat /tmp/{filename}",
        shell=True,
        capture_output=True
    )

    return {
        'statusCode': 200,
        'body': result.stdout.decode()
    }

# Exploitation : ?file=test;env
# Affiche toutes les variables d'environnement (credentials AWS inclus)
```

CNAS-2 : Lambda Sécurisée

```python
import os
import json

def lambda_handler(event, context):
    filename = event['queryStringParameters'].get('file', '')

    # Validation stricte du nom de fichier
    if not filename.isalnum():
        return {
            'statusCode': 400,
            'body': json.dumps({'error': 'Nom de fichier invalide'})
        }

    # Construction sécurisée du chemin
    safe_path = os.path.join('/tmp', filename)

    # Vérification que le chemin reste dans /tmp
    if not os.path.abspath(safe_path).startswith('/tmp/'):
        return {
            'statusCode': 403,
            'body': json.dumps({'error': 'Accès refusé'})
        }

    # Lecture sécurisée
    try:
        with open(safe_path, 'r') as f:
            return {'statusCode': 200, 'body': f.read()}
    except FileNotFoundError:
        return {'statusCode': 404, 'body': 'Fichier non trouvé'}
```

CNAS-3 : Improper Authentication and Authorization

Description

Gestion incorrecte des identités et des accès dans les environnements Cloud.

Problèmes courants

- Credentials codées en dur dans le code
- Tokens avec des permissions excessives
- Absence de rotation des secrets
- Service accounts avec droits administrateur
- Authentification faible sur les API

CNAS-3 : Mauvaises Pratiques IAM

```
# MAUVAIS : Credentials en dur dans le code
import boto3

client = boto3.client(
    's3',
    aws_access_key_id='AKIAIOSFODNN7EXAMPLE',
    aws_secret_access_key='wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
)

# MAUVAIS : Policy IAM trop permissive
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Ces pratiques exposent l'ensemble du compte AWS en cas de compromission.

CNAS-3 : Bonnes Pratiques IAM

```python
# BON : Utiliser les rôles IAM (pas de credentials)
import boto3

# Le SDK utilise automatiquement le rôle attaché à l'instance/Lambda
client = boto3.client('s3')

# BON : Récupérer les secrets depuis Secrets Manager
def get_db_credentials():
    secrets_client = boto3.client('secretsmanager')
    response = secrets_client.get_secret_value(
        SecretId='prod/database/credentials'
    )
    return json.loads(response['SecretString'])
```

```json
// BON : Policy IAM avec principe du moindre privilège
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["s3:GetObject", "s3:PutObject"],
            "Resource": "arn:aws:s3:::mon-bucket-specifique/*"
        }
    ]
}
```

CNAS-4 : CI/CD Pipeline Vulnerabilities

Description

Vulnérabilités dans les pipelines d'intégration et déploiement continus.

Risques

- Secrets exposés dans les logs de build
- Dépendances malveillantes (supply chain)
- Absence de signature des artefacts
- Pipelines modifiables par des utilisateurs non autorisés
- Exécution de code non vérifié

CNAS-4 : Pipeline Vulnérable

```yaml
# VULNERABLE : Secrets dans les logs, pas de vérification
name: Deploy
on: push

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      # VULNERABLE : Secret potentiellement dans les logs
      - run: echo "Deploying with key ${{ secrets.AWS_KEY }}"

      # VULNERABLE : Pas de vérification des dépendances
      - run: npm install

      # VULNERABLE : Pas de scan de sécurité
      - run: npm run build

      # VULNERABLE : Déploiement direct sans approbation
      - run: ./deploy.sh
```

CNAS-4 : Pipeline Sécurisé

```yaml
name: Secure Deploy
on:
  push:
    branches: [main]

jobs:
  security-scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      # Scan des dépendances
      - name: Audit dependencies
        run: npm audit --audit-level=high

      # Scan SAST
      - name: Run SAST
        uses: github/codeql-action/analyze@v2

      # Scan des secrets
      - name: Check for secrets
        uses: trufflesecurity/trufflehog@main

  build:
    needs: security-scan
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      # Utilisation sécurisée des secrets (jamais en echo)
      - name: Configure AWS
        uses: aws-actions/configure-aws-credentials@v2
        with:
          role-to-assume: arn:aws:iam::123456789:role/deploy-role
          aws-region: eu-west-1

      - run: npm ci  # Installation reproductible
      - run: npm run build

      # Signature de l'artefact
      - name: Sign artifact
        run: cosign sign-blob --key cosign.key dist.tar.gz

  deploy:
    needs: build
    runs-on: ubuntu-latest
    environment: production  # Requiert approbation manuelle
    steps:
```

CNAS-5 : Insecure Secrets Storage

Description

Stockage non sécurisé des secrets (API keys, mots de passe, certificats).

Erreurs courantes

```
# Dans le code source
API_KEY = "sk-1234567890abcdef"

# Dans les variables d'environnement non chiffrées
export DATABASE_PASSWORD=mysecretpassword

# Dans les fichiers de configuration versionnés
# config.yaml
database:
  password: "plaintext-password"

# Dans les images Docker
ENV SECRET_KEY=supersecret
```

CNAS-5 : Gestion Sécurisée des Secrets

AWS Secrets Manager

```python
import boto3
import json

def get_secret(secret_name):
    """Récupère un secret depuis AWS Secrets Manager."""
    client = boto3.client('secretsmanager')

    response = client.get_secret_value(SecretId=secret_name)

    # Le secret est retourné chiffré en transit
    # et déchiffré automatiquement par le SDK
    return json.loads(response['SecretString'])

# Utilisation
db_creds = get_secret('prod/database')
connection = connect(
    host=db_creds['host'],
    user=db_creds['username'],
    password=db_creds['password']
)
```

CNAS-5 : HashiCorp Vault

```python
import hvac

def get_secret_from_vault(path):
    """Récupère un secret depuis HashiCorp Vault."""

    # Authentification via token ou méthode cloud
    client = hvac.Client(url='https://vault.example.com:8200')

    # Authentification AWS IAM (pas de token en dur)
    client.auth.aws.iam_login(
        role='my-app-role'
    )

    # Lecture du secret
    secret = client.secrets.kv.v2.read_secret_version(
        path=path
    )

    return secret['data']['data']

# Utilisation
api_key = get_secret_from_vault('api/external-service')['key']
```

CNAS-5 : Kubernetes Secrets

```yaml
# Secret Kubernetes (base64, pas vraiment sécurisé)
apiVersion: v1
kind: Secret
metadata:
  name: db-credentials
type: Opaque
data:
  username: YWRtaW4=  # base64 de "admin"
  password: cGFzc3dvcmQxMjM=  # base64 de "password123"

---
# MIEUX : Utiliser External Secrets Operator avec AWS
apiVersion: external-secrets.io/v1beta1
kind: ExternalSecret
metadata:
  name: db-credentials
spec:
  refreshInterval: 1h
  secretStoreRef:
    name: aws-secrets-manager
    kind: SecretStore
  target:
    name: db-credentials
  data:
    - secretKey: username
      remoteRef:
        key: prod/database
        property: username
    - secretKey: password
      remoteRef:
        key: prod/database
        property: password
```

CNAS-6 : Over-permissive Network Policies

Description

Politiques réseau trop permissives exposant des services internes.

Problèmes courants

- Security Groups autorisant 0.0.0.0/0
- Bases de données accessibles publiquement
- Absence de segmentation réseau
- Ports de management exposés (SSH, RDP, Kubernetes API)

CNAS-6 : Security Groups AWS

```
// MAUVAIS : Accès SSH depuis n'importe où
{
    "IpProtocol": "tcp",
    "FromPort": 22,
    "ToPort": 22,
    "IpRanges": [{"CidrIp": "0.0.0.0/0"}]
}

// BON : Accès SSH restreint + bastion
{
    "IpProtocol": "tcp",
    "FromPort": 22,
    "ToPort": 22,
    "SourceSecurityGroupId": "sg-bastion-12345"
}
```

```
# Terraform : Security Group sécurisé
resource "aws_security_group" "web" {
  name        = "web-sg"
  vpc_id      = aws_vpc.main.id

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]  # HTTPS public OK
  }

  ingress {
    from_port       = 22
    to_port         = 22
    protocol        = "tcp"
    security_groups = [aws_security_group.bastion.id]  # SSH via bastion
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

CNAS-6 : Network Policies Kubernetes

```yaml
# Politique par défaut : refuser tout trafic entrant
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-ingress
  namespace: production
spec:
  podSelector: {}  # S'applique à tous les pods
  policyTypes:
    - Ingress

---
# Autoriser uniquement le trafic nécessaire
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-frontend-to-backend
  namespace: production
spec:
  podSelector:
    matchLabels:
      app: backend
  policyTypes:
    - Ingress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app: frontend
      ports:
        - protocol: TCP
          port: 8080
```

CNAS-7 : Using Components with Known Vulnerabilities

Description

Utilisation d'images, bibliothèques ou composants avec des vulnérabilités connues.

Outils de détection

Outil	Usage
Trivy	Scan d'images Docker et IaC
Snyk	Scan de dépendances et conteneurs
Grype	Scan de vulnérabilités conteneurs
Dependabot	Mise à jour automatique GitHub
OWASP Dependency-Check	Scan de dépendances

CNAS-7 : Scan avec Trivy

```
# Scanner une image Docker
trivy image nginx:1.19

# Résultat
nginx:1.19 (debian 10.9)
==========================
Total: 124 (UNKNOWN: 0, LOW: 85, MEDIUM: 25, HIGH: 12, CRITICAL: 2)

+------------------+------------------+----------+-------------------+
|     LIBRARY      | VULNERABILITY ID | SEVERITY |  INSTALLED VERSION |
+------------------+------------------+----------+-------------------+
| openssl          | CVE-2021-3711    | CRITICAL | 1.1.1d-0+deb10u6  |
| openssl          | CVE-2021-3712    | HIGH     | 1.1.1d-0+deb10u6  |
+------------------+------------------+----------+-------------------+

# Scanner un repository IaC
trivy config ./terraform/

# Intégration CI/CD
trivy image --exit-code 1 --severity HIGH,CRITICAL myimage:latest
```

CNAS-7 : Intégration dans le Pipeline

```yaml
# GitHub Actions avec scan de sécurité
name: Security Scan

on:
  push:
  pull_request:

jobs:
  scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      # Scan des dépendances Node.js
      - name: Audit npm
        run: npm audit --audit-level=high

      # Scan de l'image Docker avec Trivy
      - name: Build image
        run: docker build -t myapp:${{ github.sha }} .

      - name: Scan image
        uses: aquasecurity/trivy-action@master
        with:
          image-ref: myapp:${{ github.sha }}
          format: 'table'
          exit-code: '1'
          severity: 'CRITICAL,HIGH'

      # Scan de l'Infrastructure as Code
      - name: Scan IaC
        uses: aquasecurity/trivy-action@master
        with:
          scan-type: 'config'
          scan-ref: './terraform'
          exit-code: '1'
```

CNAS-8 : Improper Assets Management

Description

Mauvaise gestion des ressources Cloud : ressources orphelines, non inventoriées ou mal étiquetées.

Risques

- Ressources oubliées exposées à Internet
- Coûts incontrôlés
- Shadow IT
- Données dans des régions non conformes

CNAS-8 : Bonnes Pratiques de Tagging

```
# Terraform : Tags obligatoires sur toutes les ressources
locals {
  common_tags = {
    Environment = var.environment
    Project     = var.project_name
    Owner       = var.owner_email
    CostCenter  = var.cost_center
    ManagedBy   = "Terraform"
    CreatedAt   = timestamp()
  }
}

resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = merge(local.common_tags, {
    Name = "web-server"
    Role = "frontend"
  })
}

# AWS Config Rule pour forcer le tagging
resource "aws_config_config_rule" "required_tags" {
  name = "required-tags"

  source {
    owner             = "AWS"
    source_identifier = "REQUIRED_TAGS"
  }

  input_parameters = jsonencode({
    tag1Key   = "Environment"
    tag2Key   = "Owner"
    tag3Key   = "Project"
  })
}
```

CNAS-8 : Inventaire et Audit

```
# AWS : Lister toutes les ressources
aws resourcegroupstaggingapi get-resources --output table

# AWS : Trouver les ressources non taguées
aws resourcegroupstaggingapi get-resources \
    --tags-per-page 100 \
    --query "ResourceTagMappingList[?length(Tags)==\`0\`].ResourceARN"

# Script d'audit des Security Groups ouverts
aws ec2 describe-security-groups \
    --query "SecurityGroups[?IpPermissions[?IpRanges[?CidrIp=='0.0.0.0/0']]]" \
    --output table
```

```
# Script Python d'inventaire
import boto3

def find_untagged_resources():
    """Trouve les ressources EC2 sans tags obligatoires."""
    ec2 = boto3.client('ec2')
    required_tags = {'Environment', 'Owner', 'Project'}

    instances = ec2.describe_instances()
    untagged = []

    for reservation in instances['Reservations']:
        for instance in reservation['Instances']:
            instance_tags = {t['Key'] for t in instance.get('Tags', [])}
            missing = required_tags - instance_tags

            if missing:
                untagged.append({
                    'InstanceId': instance['InstanceId'],
                    'MissingTags': list(missing)
                })

    return untagged
```

CNAS-9 : Inadequate Compute Resource Quota Management

Description

Absence de limites sur les ressources pouvant mener à des dénis de service ou des coûts excessifs.

Problèmes

- Pas de limites de mémoire/CPU sur les conteneurs
- Pas de quotas sur les namespaces Kubernetes
- Fonctions Lambda sans limite de concurrence
- Auto-scaling sans plafond

CNAS-9 : Limites Kubernetes

```yaml
# LimitRange : Limites par défaut pour un namespace
apiVersion: v1
kind: LimitRange
metadata:
  name: default-limits
  namespace: production
spec:
  limits:
    - default:
        memory: "512Mi"
        cpu: "500m"
      defaultRequest:
        memory: "256Mi"
        cpu: "250m"
      max:
        memory: "2Gi"
        cpu: "2"
      min:
        memory: "64Mi"
        cpu: "50m"
      type: Container

---
# ResourceQuota : Limites globales pour un namespace
apiVersion: v1
kind: ResourceQuota
metadata:
  name: namespace-quota
  namespace: production
spec:
  hard:
    requests.cpu: "10"
    requests.memory: "20Gi"
    limits.cpu: "20"
    limits.memory: "40Gi"
    pods: "50"
    services: "10"
    persistentvolumeclaims: "10"
```

CNAS-9 : Limites AWS Lambda

```
# Terraform : Lambda avec limites
resource "aws_lambda_function" "api" {
  function_name = "api-handler"
  runtime       = "python3.9"
  handler       = "main.handler"

  # Limite de mémoire
  memory_size = 256  # Mo

  # Timeout
  timeout = 30  # secondes

  # Limite de concurrence réservée
  reserved_concurrent_executions = 100
}

# Budget AWS pour alerter sur les coûts
resource "aws_budgets_budget" "monthly" {
  name              = "monthly-budget"
  budget_type       = "COST"
  limit_amount      = "1000"
  limit_unit        = "USD"
  time_unit         = "MONTHLY"

  notification {
    comparison_operator = "GREATER_THAN"
    threshold           = 80
    threshold_type      = "PERCENTAGE"
    notification_type   = "ACTUAL"
    subscriber_email_addresses = ["alerts@example.com"]
  }
}
```

CNAS-10 : Insufficient Logging and Monitoring

Description

Journalisation et surveillance insuffisantes pour détecter et investiguer les incidents.

- Accès aux données sensibles
- Modifications de configuration
- Authentifications (succès et échecs)
- Actions administratives
- Appels API
- Changements réseau

CNAS-10 : Configuration CloudTrail AWS

```
# Terraform : CloudTrail complet
resource "aws_cloudtrail" "main" {
  name                          = "main-trail"
  s3_bucket_name                = aws_s3_bucket.cloudtrail.id
  include_global_service_events = true
  is_multi_region_trail         = true
  enable_log_file_validation    = true

  # Journaliser les événements de données S3
  event_selector {
    read_write_type           = "All"
    include_management_events = true

    data_resource {
      type   = "AWS::S3::Object"
      values = ["arn:aws:s3:::"]
    }
  }

  # Chiffrement des logs
  kms_key_id = aws_kms_key.cloudtrail.arn
}

# Alerte sur les connexions root
resource "aws_cloudwatch_metric_alarm" "root_login" {
  alarm_name          = "root-account-usage"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 1
  metric_name         = "RootAccountUsage"
  namespace           = "CloudTrailMetrics"
  period              = 300
  statistic           = "Sum"
  threshold           = 0
  alarm_actions       = [aws_sns_topic.alerts.arn]
}
```

CNAS-10 : Logging Kubernetes

```
# Fluent Bit pour centraliser les logs
apiVersion: v1
kind: ConfigMap
metadata:
  name: fluent-bit-config
  namespace: logging
data:
  fluent-bit.conf: |
    [SERVICE]
        Flush         5
        Log_Level     info
        Daemon        off

    [INPUT]
        Name              tail
        Path              /var/log/containers/*.log
        Parser            docker
        Tag               kube.*
        Refresh_Interval  5

    [FILTER]
        Name                kubernetes
        Match               kube.*
        Kube_URL            https://kubernetes.default.svc:443
        Merge_Log           On

    [OUTPUT]
        Name            es
        Match           *
        Host            elasticsearch.logging.svc
        Port            9200
        Index           kubernetes-logs
        Type            _doc
```

Partie 3

Infrastructure as Code Sécurisée

Principes de l'IaC Sécurisé

```
1. Versionnement     -> Tout le code dans Git
2. Revue de code     -> Pull requests obligatoires
3. Scan automatique  -> Détecter les misconfigurations
4. Immutabilité      -> Recréer plutôt que modifier
5. Moindre privilège -> Permissions minimales
6. Chiffrement       -> Données au repos et en transit
7. Secrets externes  -> Jamais dans le code
```

Scan de Sécurité IaC avec Checkov

```
# Installation
pip install checkov

# Scan d'un répertoire Terraform
checkov -d ./terraform/

# Résultat
Passed checks: 45
Failed checks: 12
Skipped checks: 3

Check: CKV_AWS_20: "Ensure the S3 bucket has server-side encryption"
    FAILED for resource: aws_s3_bucket.data
    File: /main.tf:25-35

Check: CKV_AWS_21: "Ensure the S3 bucket has versioning enabled"
    FAILED for resource: aws_s3_bucket.data

# Intégration CI
checkov -d ./terraform/ --soft-fail --output junitxml > results.xml
```

Exemple Terraform Sécurisé

```
# providers.tf
terraform {
  required_version = ">= 1.5.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }

  # Backend distant avec chiffrement
  backend "s3" {
    bucket         = "terraform-state-prod"
    key            = "infrastructure/terraform.tfstate"
    region         = "eu-west-1"
    encrypt        = true
    dynamodb_table = "terraform-locks"
  }
}

provider "aws" {
  region = var.aws_region

  default_tags {
    tags = {
      Environment = var.environment
      ManagedBy   = "Terraform"
      Project     = var.project_name
    }
  }
}
```

Terraform : S3 Bucket Sécurisé

```
resource "aws_s3_bucket" "secure" {
  bucket = "my-secure-bucket-${random_id.suffix.hex}"
}

# Bloquer l'accès public
resource "aws_s3_bucket_public_access_block" "secure" {
  bucket = aws_s3_bucket.secure.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

# Chiffrement côté serveur
resource "aws_s3_bucket_server_side_encryption_configuration" "secure" {
  bucket = aws_s3_bucket.secure.id

  rule {
    apply_server_side_encryption_by_default {
      kms_master_key_id = aws_kms_key.s3.arn
      sse_algorithm     = "aws:kms"
    }
    bucket_key_enabled = true
  }
}

# Versioning
resource "aws_s3_bucket_versioning" "secure" {
  bucket = aws_s3_bucket.secure.id

  versioning_configuration {
    status = "Enabled"
  }
}

# Logging
resource "aws_s3_bucket_logging" "secure" {
  bucket = aws_s3_bucket.secure.id

  target_bucket = aws_s3_bucket.logs.id
  target_prefix = "s3-access-logs/"
}
```

Partie 4

Sécurité des Conteneurs et Kubernetes

Bonnes Pratiques Docker

```dockerfile
# Image de base minimale et vérifiée
FROM cgr.dev/chainguard/python:latest-dev AS builder

WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt --target=/app/deps

# Image de production sans outils de build
FROM cgr.dev/chainguard/python:latest

# Copier uniquement les fichiers nécessaires
COPY --from=builder /app/deps /app/deps
COPY --chown=nonroot:nonroot ./src /app/src

# Variables d'environnement
ENV PYTHONPATH=/app/deps
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

# Port non privilégié
EXPOSE 8080

# Utilisateur non-root
USER nonroot

# Healthcheck
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
    CMD python -c "import urllib.request; urllib.request.urlopen('http://localhost:8080/health')"

# Point d'entrée
ENTRYPOINT ["python", "/app/src/main.py"]
```

Pod Security Standards Kubernetes

```yaml
# PodSecurityPolicy restrictive (deprecated, utiliser Pod Security Admission)
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
spec:
  privileged: false
  allowPrivilegeEscalation: false

  requiredDropCapabilities:
    - ALL

  runAsUser:
    rule: MustRunAsNonRoot

  runAsGroup:
    rule: MustRunAs
    ranges:
      - min: 1000
        max: 65535

  fsGroup:
    rule: MustRunAs
    ranges:
      - min: 1000
        max: 65535

  volumes:
    - 'configMap'
    - 'emptyDir'
    - 'projected'
    - 'secret'
    - 'downwardAPI'
    - 'persistentVolumeClaim'

  readOnlyRootFilesystem: true
```

Pod Security Admission (Kubernetes 1.25+)

```yaml
# Namespace avec politique de sécurité stricte
apiVersion: v1
kind: Namespace
metadata:
  name: production
  labels:
    # Appliquer le profil "restricted"
    pod-security.kubernetes.io/enforce: restricted
    pod-security.kubernetes.io/enforce-version: latest

    # Avertir sur les violations "baseline"
    pod-security.kubernetes.io/warn: baseline
    pod-security.kubernetes.io/warn-version: latest

    # Auditer les violations "restricted"
    pod-security.kubernetes.io/audit: restricted
    pod-security.kubernetes.io/audit-version: latest

---
# Deployment conforme à la politique "restricted"
apiVersion: apps/v1
kind: Deployment
metadata:
  name: secure-app
  namespace: production
spec:
  replicas: 3
  selector:
    matchLabels:
      app: secure-app
  template:
    metadata:
      labels:
        app: secure-app
    spec:
      securityContext:
        runAsNonRoot: true
        runAsUser: 1000
        runAsGroup: 1000
        fsGroup: 1000
        seccompProfile:
          type: RuntimeDefault

      containers:
        - name: app
          image: myapp:1.0.0
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            capabilities:
              drop:
                - ALL

          resources:
            requests:
              memory: "128Mi"
              cpu: "100m"
            limits:
              memory: "256Mi"
              cpu: "200m"
```

Service Mesh avec mTLS (Istio)

```yaml
# PeerAuthentication : Exiger mTLS pour tout le mesh
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
  namespace: istio-system
spec:
  mtls:
    mode: STRICT

---
# AuthorizationPolicy : Contrôle d'accès entre services
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: backend-policy
  namespace: production
spec:
  selector:
    matchLabels:
      app: backend

  action: ALLOW
  rules:
    # Autoriser uniquement le frontend à appeler le backend
    - from:
        - source:
            principals:
              - cluster.local/ns/production/sa/frontend
      to:
        - operation:
            methods: ["GET", "POST"]
            paths: ["/api/*"]

    # Autoriser les health checks
    - to:
        - operation:
            methods: ["GET"]
            paths: ["/health", "/ready"]
```

Partie 5

Outils et Audit de Sécurité Cloud

Outils de Scan Cloud

Outil	Description	Usage
Prowler	Audit AWS/Azure/GCP	Conformité CIS
ScoutSuite	Multi-cloud security	Audit global
CloudSploit	Scan de configuration	Open source
Trivy	Scan conteneurs et IaC	CI/CD
Checkov	Scan IaC	Terraform, K8s
kube-bench	CIS Kubernetes	Audit clusters
Falco	Runtime security	Détection d'intrusion

Prowler - Audit AWS

```
# Installation
pip install prowler

# Audit complet AWS
prowler aws

# Audit spécifique CIS
prowler aws --compliance cis_2.0_aws

# Audit d'un service spécifique
prowler aws --service s3,iam,ec2

# Export des résultats
prowler aws --output-formats html,json --output-directory ./reports

# Résultat
PASS: 234
FAIL: 45
INFO: 12

CRITICAL:
- IAM root account has MFA disabled
- S3 bucket "public-data" is publicly accessible
- Security group allows unrestricted SSH access
```

kube-bench - Audit Kubernetes CIS

```
# Exécution dans le cluster
kubectl apply -f https://raw.githubusercontent.com/aquasecurity/kube-bench/main/job.yaml

# Voir les résultats
kubectl logs job/kube-bench

# Résultat
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 644 or more restrictive
[FAIL] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root
[WARN] 1.1.3 Ensure that the controller manager pod specification file permissions are set to 644

== Summary ==
45 checks PASS
12 checks FAIL
8 checks WARN
```

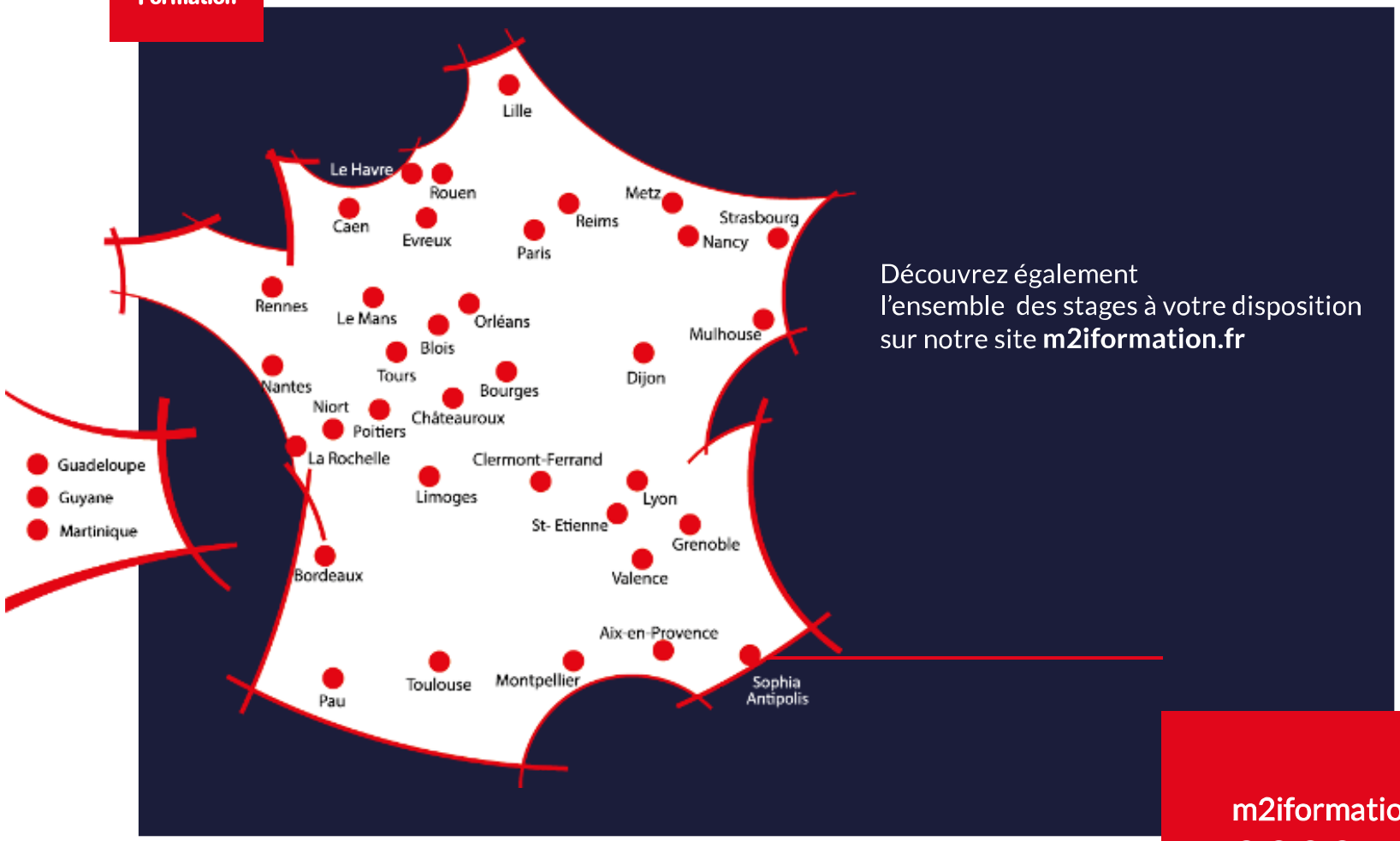
Falco - Détection Runtime

```yaml
# falco-rules.yaml - Règles personnalisées
- rule: Shell Spawned in Container
  desc: Détecte l'exécution d'un shell dans un conteneur
  condition: >
    spawned_process and
    container and
    proc.name in (bash, sh, zsh, ksh)
  output: >
    Shell lancé dans un conteneur
    (user=%user.name container=%container.name shell=%proc.name)
  priority: WARNING
  tags: [container, shell]

- rule: Sensitive File Access
  desc: Accès à des fichiers sensibles
  condition: >
    open_read and
    container and
    fd.name in (/etc/shadow, /etc/passwd, /etc/kubernetes/admin.conf)
  output: >
    Fichier sensible lu
    (user=%user.name file=%fd.name container=%container.name)
  priority: CRITICAL
  tags: [filesystem, sensitive]

- rule: Outbound Connection to Suspicious IP
  desc: Connexion sortante vers une IP suspecte
  condition: >
    outbound and
    container and
    fd.sip in (known_malicious_ips)
  output: >
    Connexion suspecte
    (container=%container.name ip=%fd.sip)
  priority: CRITICAL
  tags: [network, exfiltration]
```

Checklist Sécurité Cloud

```
IDENTITE ET ACCES
[ ] MFA activé pour tous les comptes utilisateurs
[ ] Pas de clés d'accès pour le compte root
[ ] Rotation des credentials < 90 jours
[ ] Principe du moindre privilège appliqué
[ ] Service accounts avec permissions minimales

RESEAU
[ ] VPC avec sous-réseaux privés/publics
[ ] Security groups restrictifs
[ ] Pas d'accès 0.0.0.0/0 sur les ports sensibles
[ ] WAF configuré pour les applications web
[ ] VPN ou bastion pour l'administration

DONNEES
[ ] Chiffrement au repos activé
[ ] Chiffrement en transit (TLS 1.2+)
[ ] Buckets S3 privés par défaut
[ ] Backups chiffrés et testés
[ ] Classification des données

LOGGING ET MONITORING
[ ] CloudTrail/Audit Logs activés
[ ] Logs centralisés et retenus > 1 an
[ ] Alertes sur les événements critiques
[ ] Tableau de bord de sécurité

CONTENEURS
[ ] Images de base minimales
[ ] Scan de vulnérabilités automatique
[ ] Pas de conteneurs root
[ ] Network policies Kubernetes
[ ] Pod Security Standards appliqués
```

Questions



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr