DevSecOps Workshop

Segurança em Infrastructure as Code

João Oliveira

2022184283

Laboratório de Cibersegurança

MEI | FCT | UC

Dezembro 2025

Agenda

- DevSecOps
- Repositório
- Vulnerabilidades
- Ferramentas de Análise de Segurança
- Fases do Workshop
  - Fase 1 - Setup
  - Fase 2 - Análise
    - Vulnerabilidades detetadas pelo Trivy
  - Fase 3 - Correções
  - Fase 4 - Security Gates & Scan adicional
    - Vulnerabilidades detectadas pelo Checkov
- Q&A

DevSecOps

Os conceitos deste Workshop visam a expandir os laboratórios #3 e #4, aplicando as técnicas de DevSecOps (e outras adicionais) ao seguinte cenário:

- **Código declarativo para infraestrutura em Cloud Providers (CP) usando ferramentas de Infrastructure as Code (IaC)**

O Workshop tem os seguintes objetivos

Infrastructure As Code (IaC)
Demonstrar código com ferramentas IaC (e.g.: Terraform) potencialmente inseguro

Shift-Left Principle usando ferramentas DevSecOps
Mover a segurança para as fases iniciais do desenvolvimento.

Políticas “Security as Code”
Políticas de segurança definidas e automatizadas usando o Checkov

“Breaking the Build”
Security Scans que permitem falhar em caso de código inseguro

Repositório

```
devsecops-workshop-repo/
├── .github/
│   └── workflows/
│       └── cicd.yml            # CI/CD Pipeline
│           ├── Terraform Format Check
│           ├── Terraform Init & Validate
│           ├── Gitleaks (secret scanning)
│           ├── Trivy (IaC scanning + SARIF upload)
│           └── Checkov (policy-as-code + SARIF upload)
│
├── terraform/
│   ├── main.tf                 # Infrastructure Resources
│   │   ├── google_storage_bucket (insecure_bucket)
│   │   ├── google_compute_firewall (allow_all_ssh)
│   │   ├── google_compute_instance (insecure_instance)
│   │   ├── google_service_account (insecure_sa)
│   │   ├── google_project_iam_member (insecure_sa_editor)
│   │   ├── google_sql_database_instance (insecure_db)
│   │   └── google_sql_user (insecure_db_user)
│   │
│   ├── variables.tf            # Input Variables
│   │   ├── project_id, region, zone
│   │   ├── db_password (hardcoded)
│   │   └── api_key (hardcoded)
│   │
│   ├── outputs.tf              # Outputs
│   │   ├── bucket_name, bucket_url
│   │   ├── instance_name
│   │   ├── database_name, database_public_ip
│   │   └── service_account_email
│   │
│   └── terraform.tfvars.example  # Example Variables
│
├── patches/
│   └── fix.patch               # Security Fixes
│
├── README.md                   # Workshop Instructions
├── Workshop_Lab_Guide.pdf      # Detailed Lab Guide
└── .gitignore
```

Ferramentas de Análise de Segurança

Checkov

Policy-as-Code

Ampla cobertura. Policies adicionais ao Trivy.

Trivy*

Multi-Scanner

Misconfigurations + CVEs.

“Swiss-army” security tool

Gitleaks

Secret Detection

Explorado no laboratório #3.

Deteta credenciais no código

*** O Trivy integra agora o Tfsec, a maior ferramenta para análise de segurança em configurações IaC**

Fases do Workshop

FASE 1

Preparação do Ambiente

1. Pre-requisitos
2. Fork do repositório de partida
3. Ativar uma GH Codespace ou clone local
4. Ativar GitHub Actions

FASE 2

Análise Estática de Segurança

1. Ativar a execução dos Workflows

Na GUI do Github Actions

2. Esperar

Pela execução do Workflow

3. Verificar os resultados

Em Github CodeQL (tab “Security”)

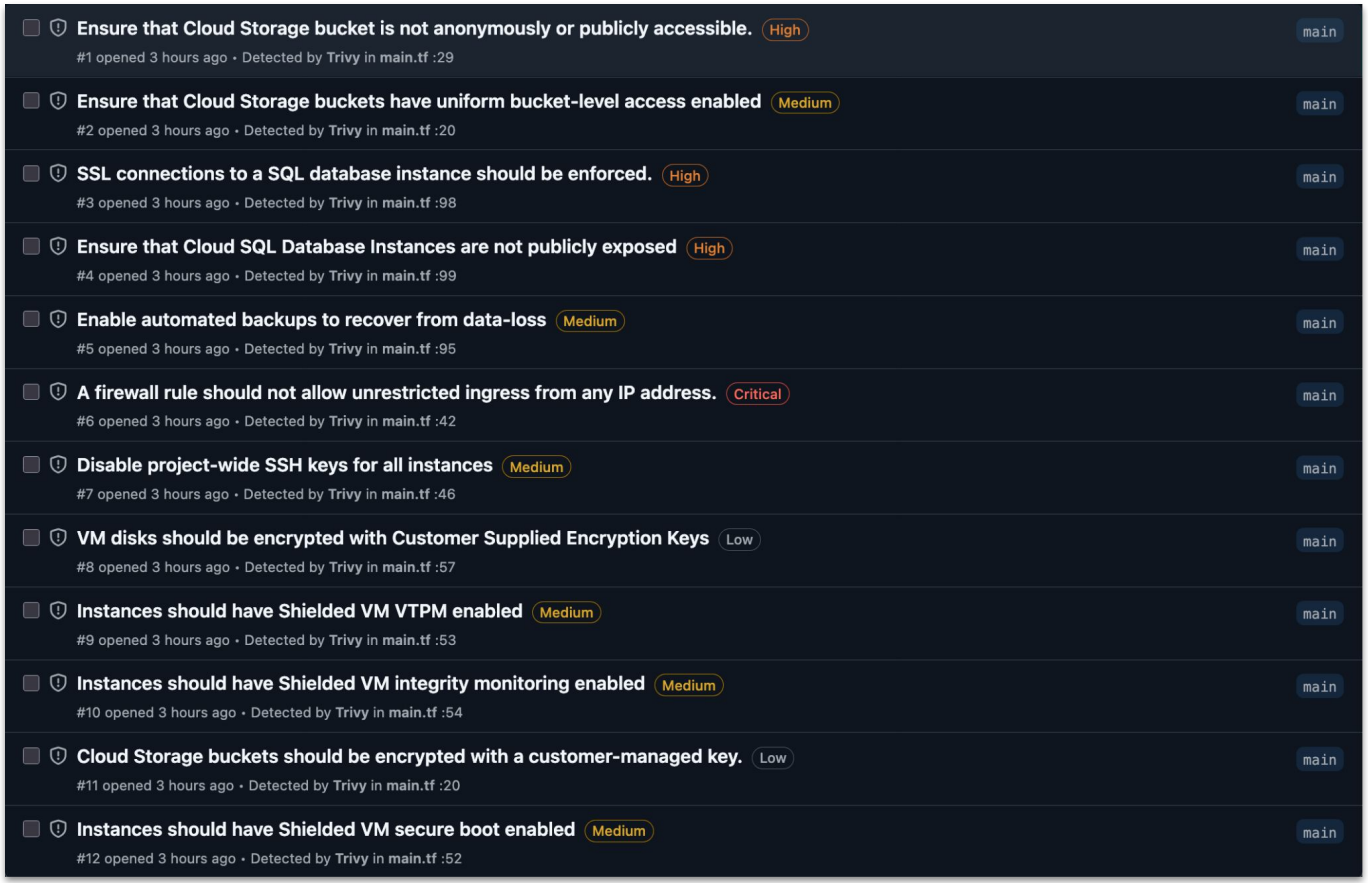
4. Análise de Resultados

Documentar vulnerabilidades encontradas

```
# Vulnerability scanning using Trivy but for config files
- name: Run Trivy
  uses: aquasecurity/trivy-action@master
  with:
    scan-type: 'config'
    scan-ref: 'terraform'
    severity: 'CRITICAL,HIGH,MEDIUM,LOW'
    exit-code: '0'
    format: 'sarif'
    output: 'trivy-results.sarif'

- name: Upload Trivy results to GitHub Security
  uses: github/codeql-action/upload-sarif@v4
  if: always()
  with:
    sarif_file: 'trivy-results.sarif'
```

Vulnerabilidades - Trivy



FASE 3

Implementação de Correções

Storage Bucket (insecure_bucket):

- Enabled uniform bucket-level access
- Enabled versioning
- Added access logging
- Enforced public access prevention
- Added KMS encryption
- Removed public IAM access (allUsers)

Firewall (allow_restricted_ssh):

- Restricted SSH to IAP range only (35.235.240.0/20)

Compute Instance (insecure_instance):

- Enabled Shielded VM (secure boot, vTPM, integrity monitoring)
- Added disk KMS encryption
- Enabled OS Login
- Blocked project-wide SSH keys
- Reduced service account scopes

IAM (insecure_sa):

- Replaced Editor role with specific roles (compute.instanceAdmin, storage.objectViewer)

Database (insecure_db):

- Enabled backups
- Disabled public IP
- Enforced SSL/TLS (TRUSTED_CLIENT_CERTIFICATE_REQUIRED)

Variables:

- Removed hardcoded secrets
- Marked sensitive variables as sensitive = true

Outputs:

- Replaced database_public_ip with connection_name
- Marked service_account_email as sensitive

FASE 4

Security Gates e Scan Adicional (Checkov)

```yaml
# Vulnerability scanning using Trivy but for config files
- name: Run Trivy
  uses: aquasecurity/trivy-action@master
  with:
    scan-type: 'config'
    scan-ref: 'terraform'
    severity: 'CRITICAL,HIGH,MEDIUM,LOW'
    exit-code: '1'
    format: 'sarif'
    output: 'trivy-results.sarif'

- name: Upload Trivy results to GitHub Security
  uses: github/codeql-action/upload-sarif@v4
  if: always()
  with:
    sarif_file: 'trivy-results.sarif'

# IaC Security Scanner
- name: Run Checkov
  uses: bridgecrewio/checkov-action@v12
  with:
    directory: terraform
    framework: terraform
    soft_fail: false
    output_format: sarif
    output_file_path: checkov-results.sarif

- name: Upload Checkov results to GitHub Security
  uses: github/codeql-action/upload-sarif@v4
  if: always()
  with:
    sarif_file: checkov-results.sarif
```

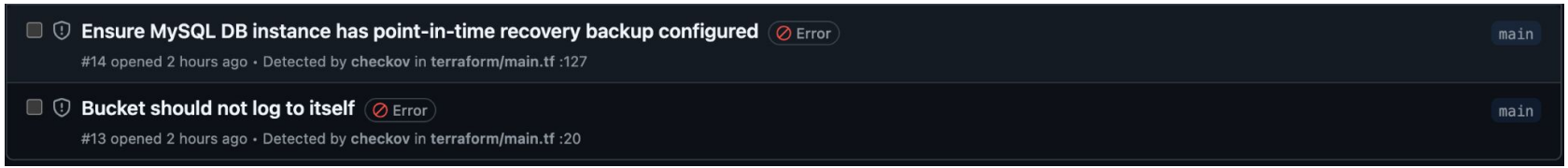
Gate FALHA

Problemas detectados = Build falha

Gate PASSA

Código seguro = Build conclui com sucesso

Vulnerabilidades - Checkov



Obrigado!

Q&A

João Oliveira

2022184283

Laboratório de Cibersegurança

MEI | FCT | UC

Dezembro 2025