

Lab 3.4

Infrastructure-as-Code Security with Checkov

Toepasbare Cloud Security voor IT-Professionals

Contents

1. Infrastructure-as-Code Security with Checkov	3
1.1. Context	3
1.2. Learning goals	3
1.3. Estimated time	3
1.4. Before you start	3
2. Run Checkov Locally	5
2.1. Review the Output Summary	5
3. Investigate Findings	6
3.1. Finding 1	6
3.2. Finding 2	7
4. Fix a Terraform Finding	8
5. Suppress a Finding with Justification	10
6. Review the Checkov Configuration	11
7. Add Checkov to the Azure DevOps Pipeline	12
7.1. Check the Pipeline Job	12
8. Commit and Run	13
9. Reflection Questions	14
9.1. Question 1	14
9.2. Question 2	14
9.3. Question 3	14
9.4. Question 4	15
10. Summary	16

1. Infrastructure-as-Code Security with Checkov

1.1. Context

In Lab 3.3 you scanned application code and dependencies. In this lab you scan Terraform infrastructure code with Checkov.

The goal is not to learn every Checkov rule. The goal is to practise a realistic workflow:

- Run an IaC security scan.
- Find the risky Terraform resource.
- Decide whether to fix, suppress, or document the finding.
- Add the scan to the Azure DevOps pipeline.

Your local workstation is Windows with PowerShell. The Azure DevOps pipeline still runs on an Ubuntu agent, so the pipeline snippets may use Bash/Linux commands.

1.2. Learning goals

By the end of this lab, you should be able to:

- Run Checkov locally against Terraform code.
- Interpret a failed CKV_AZURE_* check.
- Fix at least one Terraform security misconfiguration.
- Suppress a finding only when you can justify it.
- Explain why IaC scanning belongs before infrastructure promotion.

1.3. Estimated time

60 minutes

Suggested timing:

Time	Activity
10 minutes	Run Checkov locally
15 minutes	Investigate findings
15 minutes	Fix or suppress selected findings
15 minutes	Add Checkov to the pipeline
5 minutes	Reflection

1.4. Before you start

Make sure you have:

- ☐ Repository cloned locally
- ☐ Terraform files available in infrastructure/day-3
- ☐ .checkov.yml available in the repository root

- ☐ Python installed, or Docker Desktop installed

2. Run Checkov Locally

Open PowerShell in the repository root.

```
cd <REPOSITORY_ROOT>
```

If you prefer Docker (not verified):

```
docker run --rm -v "../src" bridgecrew/checkov --config-file /src/.checkov.yml
```

Note

The Docker command mounts your current repository folder into the container. This is useful if you do not want to install Checkov locally.

2.1. Review the Output Summary

Record the summary from the terminal.

Result	Count
Passed checks	
Failed checks	
Skipped checks	

Which Terraform directory did Checkov scan?

Your answer:

3. Investigate Findings

Pick two findings that look relevant for this lab. Prefer findings related to:

- Storage accounts
- App Service HTTPS/TLS settings
- Network access rules
- Key Vault configuration
- Diagnostic logging

3.1. Finding 1

Check ID:

Your answer, for example CKV_AZURE_3:

Resource and file path:

Your answer:

What is Checkov warning about?

Your answer:

Is this relevant for the dev environment? Why?

Your answer:

Decision: Fix, suppress, or accept temporarily?

Your answer:

3.2. Finding 2

Check ID:

Your answer:

Resource and file path:

Your answer:

What is Checkov warning about?

Your answer:

Is this relevant for the dev environment? Why?

Your answer:

Decision: Fix, suppress, or accept temporarily?

Your answer:

4. Fix a Terraform Finding

Choose one finding that should be fixed in code.

Open the Terraform file in VS Code and update the resource.

Examples of common fixes:

```
# Storage account: require HTTPS
https_traffic_only_enabled = true
min_tls_version            = "TLS1_2"
allow_nested_items_to_be_public = false

# App Service: require HTTPS
https_only = true

# Network rule: avoid unrestricted access where possible
source_address_prefix = "<specific-ip-range-or-service-tag>"
```

Check ID you fixed:

Your answer:

Original configuration:

Paste the relevant original Terraform lines:

Updated configuration:

Paste the fixed Terraform lines:

Run Checkov again:

```
docker run --rm -v "./src" bridgecrew/checkov --config-file /src/.checkov.yml
```


Did the finding disappear or change?

Your answer:

5. Suppress a Finding with Justification

Some checks may be too strict for this learning dev environment. Suppress only when you can explain why.

Use an inline Checkov skip comment near the resource:

```
# checkov:skip=CKV_AZURE_43:Customer-managed keys are not required for this temporary
dev lab environment
resource "azurerm_storage_account" "example" {
  ...
}
```

Check ID suppressed:

Your answer:

Justification:

Your answer. Include why this is acceptable and when it should be reviewed again:

Run Checkov again:

```
docker run --rm -v " ./src" bridgecrew/checkov --config-file /src/.checkov.yml
```

Is the finding now listed as skipped?

Your answer:

Inline or global suppression?

Prefer inline suppressions in Terraform files for lab work. They keep the reason close to the resource. Global suppressions in `.checkov.yml` are harder to review and can hide issues across the whole repository.

6. Review the Checkov Configuration

Open `.checkov.yml`.

Which directories are scanned?

Your answer:

Which output formats are configured?

Your answer:

Are any checks skipped globally?

Your answer:

7. Add Checkov to the Azure DevOps Pipeline

Open:

pipeline-snippets/lab-3-4-iac-scanning.md

Copy the SecurityIaCScanning job into the Validate stage of azure-pipelines.yml.

After Labs 3.2 and 3.3, the Validate stage should already include:

- QualityGates
- SecurityCodeAnalysis
- SecurityDependencyScanning

Add:

- SecurityIaCScanning

7.1. Check the Pipeline Job

In the snippet, find the Checkov command.

What command does the pipeline run?

Your answer:

Does the job use continueOnError: "true" or another audit-mode setting?

Your answer:

Why is audit mode useful while introducing a new security scanner?

Your answer:

8. Commit and Run

Commit your pipeline and Terraform changes:

```
git status
git add azure-pipelines.yml infrastructure/
git commit -m "feat: add IaC security scanning"
git push
```

Open the Azure DevOps pipeline run.

In the Validate stage, confirm that SecurityIaCScanning runs next to the other validation jobs.

Did the Checkov job run?

Your answer:

Did it publish a report artifact?

Your answer:

How many findings are shown in the pipeline logs?

Your answer:

9. Reflection Questions

9.1. Question 1

What is the difference between `terraform validate` and Checkov?

Your answer:

9.2. Question 2

Why is it better to find a public storage account in Terraform before running `terraform apply`?

Your answer:

9.3. Question 3

When is it acceptable to suppress a Checkov finding?

Your answer:

9.4. Question 4

You run Checkov for the first time on an existing repository and get 150 findings. What is a sensible strategy?

Your answer:

10. Summary

In this lab, you:

- Ran Checkov locally from PowerShell.
- Investigated Azure Terraform findings.
- Fixed or suppressed selected findings with justification.
- Added Checkov to the Azure DevOps Validate stage.
- Verified that the pipeline can publish IaC scan results.

Note

In the next lab, you will test the deployed API with OWASP ZAP. That scan runs after deployment because it needs a running application.