Security Misconfiguration Analysis of Terraform

1. Introduction

Infrastructure as Code (IaC) has transformed the way cloud environments are provisioned, managed, and scaled. Among IaC tools, Terraform has gained significant popularity due to its simplicity and support for multi-cloud environments. However, the convenience of using code to deploy infrastructure also introduces security risks, especially when configurations are improperly defined. Misconfigurations can expose critical resources to unauthorized access, resulting in data breaches, service disruptions, and regulatory non-compliance.

To mitigate these risks, security tools like Checkov have become essential in DevSecOps pipelines. Checkov, an open-source static code analysis tool developed by Bridgecrew (Prisma Cloud), automatically scans IaC templates for security and compliance misconfigurations. This report presents a comprehensive analysis of Terraform configurations across multiple files using Checkov, executed on an Ubuntu environment. It aims to uncover and address common security issues that often go unnoticed during the development phase.

2. Objective

The objective of this exercise is to identify and evaluate security misconfigurations within Terraform-based infrastructure deployments using Checkov. The specific goals include:

- Scanning Terraform files for compliance with best security practices.
- Detecting common misconfigurations in EC2 instances, security groups, and storage resources.
- Providing detailed insights into failed checks and offering mitigation strategies.
- Demonstrating how Checkov can be integrated into development workflows to automate security validation.

By proactively identifying misconfigurations before deployment, this process helps strengthen the security posture of cloud environments and enhances the overall resilience of infrastructure.

3. Methodology

The analysis followed a systematic methodology encompassing environment setup, Terraform project creation, Checkov installation, and security scanning.

Environment Setup:

- Operating System: Ubuntu

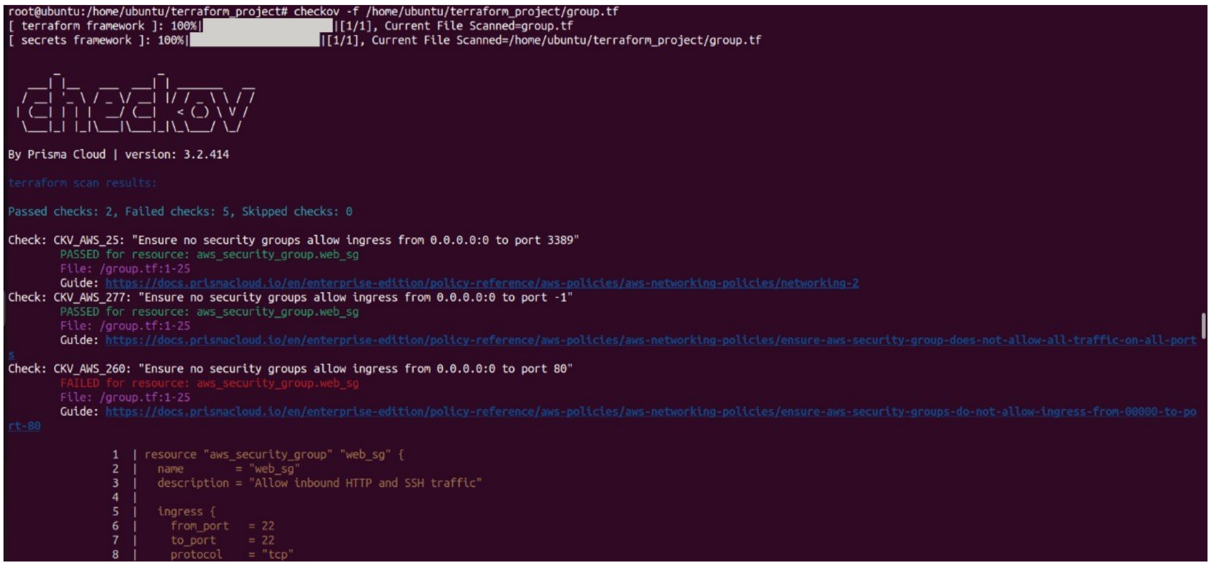
- Tools Installed: Terraform, Checkov (via pip install checkov)

Project Structure:

- Created a Terraform project directory containing the following files:
  - main.tf – EC2 instance configuration
  - group.tf – Security group rules
  - variables.tf – Variable declarations
  - outputs.tf – Output values

Checkov Scanning Process:

- Scanned each file individually using:
  - checkov -f /path/to/file.tf
- Performed a comprehensive directory scan using:
  - checkov -d /home/ubuntu/terraform_project
- Checkov evaluated the Terraform templates against a set of predefined policies related to AWS security best practices.



```
root@ubuntu:/home/ubuntu/terraform_project# checkov -f /home/ubuntu/terraform_project/group.tf
[ terraform framework ]: 100%|████████████████████|[1/1], Current File Scanned=group.tf
[ secrets framework ]: 100%|████████████████████|[1/1], Current File Scanned=/home/ubuntu/terraform_project/group.tf

       _               _
   ___| |__   ___  ___| | _______   __
  / __| '_ \ / _ \/ __| |/ / _ \ \ / /
 | (__| | | |  __/ (__|   < (_) \ V /
  \___|_| |_|\___|\___|_|\_\___/ \_/

By Prisma Cloud | version: 3.2.414

terraform scan results:

Passed checks: 2, Failed checks: 5, Skipped checks: 0

Check: CKV_AWS_25: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 3389"
        PASSED for resource: aws_security_group.web_sg
        File: /group.tf:1-25
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/networking-2
Check: CKV_AWS_277: "Ensure no security groups allow ingress from 0.0.0.0:0 to port -1"
        PASSED for resource: aws_security_group.web_sg
        File: /group.tf:1-25
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-security-group-does-not-allow-all-traffic-on-all-port
s
Check: CKV_AWS_260: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 80"
        FAILED for resource: aws_security_group.web_sg
        File: /group.tf:1-25
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-security-groups-do-not-allow-ingress-from-00000-to-po
rt-80

                1  | resource "aws_security_group" "web_sg" {
                2  |   name        = "web_sg"
                3  |   description = "Allow inbound HTTP and SSH traffic"
                4  |
                5  |   ingress {
                6  |     from_port   = 22
                7  |     to_port     = 22
                8  |     protocol    = "tcp"
```

Analysis and Review:

- Collected results for passed and failed checks.
- Identified high-risk issues and referenced Checkov policy guides for context.

4. Results & Findings

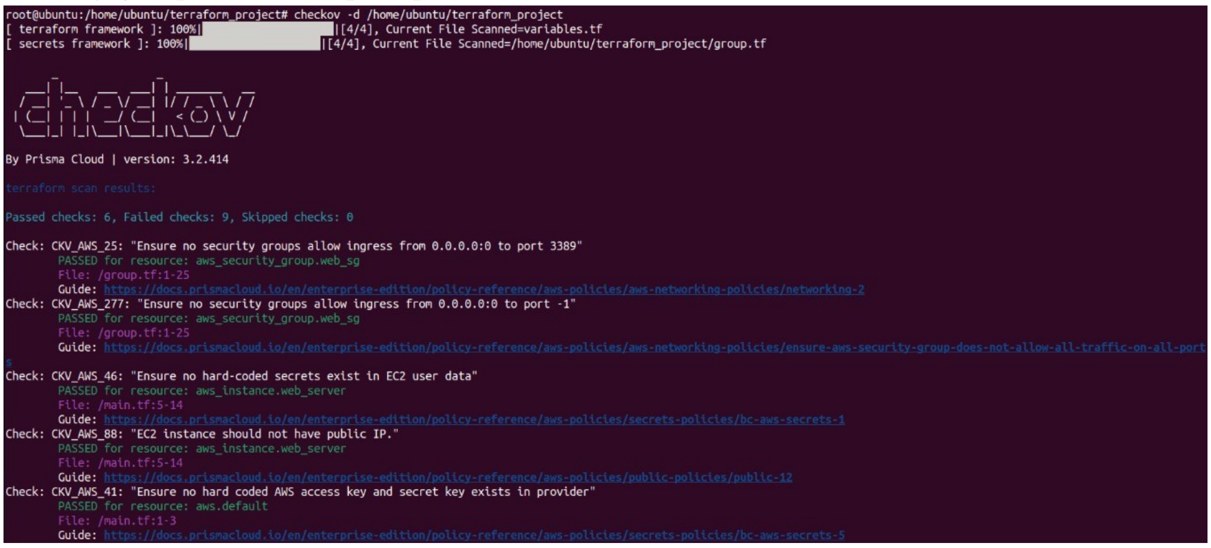
A detailed security analysis of the Terraform files revealed the following:

Overall Summary:

- Passed Checks: 6
- Failed Checks: 9
- Skipped Checks: 0

Key Passed Checks:

- No hard-coded AWS access keys (CKV_AWS_41)
- EC2 instance does not have a public IP (CKV_AWS_88)
- Secrets are not embedded in EC2 user data (CKV_AWS_46)
- Security group does not expose port 3389 (CKV_AWS_25)



```
root@ubuntu:/home/ubuntu/terraform_project# checkov -d /home/ubuntu/terraform_project
[ terraform framework ]: 100%|████████████████████|[4/4], Current File Scanned=variables.tf
[ secrets framework ]: 100%|████████████████████|[4/4], Current File Scanned=/home/ubuntu/terraform_project/group.tf

       _               _
   ___| |__   ___  ___| | _______   __
  / __| '_ \ / _ \/ __| |/ / _ \ \ / /
 | (__| | | |  __/ (__|   < (_) \ V /
  \___|_| |_|\___|\___|_|\_\___/ \_/

By Prisma Cloud | version: 3.2.414

terraform scan results:

Passed checks: 6, Failed checks: 9, Skipped checks: 0

Check: CKV_AWS_25: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 3389"
        PASSED for resource: aws_security_group.web_sg
        File: /group.tf:1-25
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/networking-2
Check: CKV_AWS_277: "Ensure no security groups allow ingress from 0.0.0.0:0 to port -1"
        PASSED for resource: aws_security_group.web_sg
        File: /group.tf:1-25
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-networking-policies/ensure-aws-security-group-does-not-allow-all-traffic-on-all-port
s
Check: CKV_AWS_46: "Ensure no hard-coded secrets exist in EC2 user data"
        PASSED for resource: aws_instance.web_server
        File: /main.tf:5-14
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/secrets-policies/bc-aws-secrets-1
Check: CKV_AWS_88: "EC2 instance should not have public IP."
        PASSED for resource: aws_instance.web_server
        File: /main.tf:5-14
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/public-policies/public-12
Check: CKV_AWS_41: "Ensure no hard coded AWS access key and secret key exists in provider"
        PASSED for resource: aws.default
        File: /main.tf:1-3
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/secrets-policies/bc-aws-secrets-5
```

Key Failed Checks and Implications:

1. **EC2 Instance Security:**
    - **CKV_AWS_79:** Instance Metadata Service Version 1 enabled; poses a risk of SSRF attacks.
    - **CKV_AWS_126:** Detailed monitoring disabled; hinders effective resource monitoring.
    - **CKV_AWS_8 & CKV_AWS_135:** Lack of encryption and EBS optimization; affects data security and performance.
    - **CKV2_AWS_41:** No IAM role attached; limits secure access to AWS services.
2. **Security Group Configuration:**
    - **CKV_AWS_24 & CKV_AWS_260:** Open access on ports 22 (SSH) and 80 (HTTP) from any IP; introduces high exposure risk.

- **CKV_AWS_382:** All traffic allowed in egress rules; violates principle of least privilege.
- **CKV_AWS_23:** Missing descriptions for security group rules; impacts maintainability and auditability.

5. Recommendations

Based on the above findings, the following recommendations are proposed to enhance infrastructure security:

1. **Enforce EC2 Hardening:**
   - Enable IMDSv2 and disable IMDSv1.
   - Enable detailed monitoring for all EC2 instances.
   - Encrypt all EBS volumes and enable EBS optimization.
   - Attach IAM roles with minimum required permissions to all instances.
2. **Tighten Network Security:**
   - Avoid using 0.0.0.0/0 for ingress and egress unless justified; restrict access to known IPs.
   - Apply network segmentation using VPCs and subnetting.
   - Define specific egress rules rather than allowing all traffic.
   - Include clear descriptions for all rules to support change management and auditing.
3. **Automation and Integration:**
   - Integrate Checkov into CI/CD pipelines to ensure continuous validation.
   - Use version control pre-commit hooks to block insecure code commits.
   - Combine Checkov with tools like tfsec and AWS Config for layered security checks.

6. Conclusion

The proactive use of Checkov to scan Terraform configurations uncovered several critical security misconfigurations, especially related to EC2 instance hardening and security group exposure. While some best practices were already followed, such as avoiding hard-coded credentials and limiting public IP usage, other areas like metadata protection and network access control need immediate attention.

Embedding Checkov into the infrastructure lifecycle ensures that security is treated as a first-class concern from the start. By remediating the identified issues and adhering to the recommendations provided, organizations can significantly reduce their attack surface, maintain compliance, and build a more secure and resilient cloud environment.

This exercise validates the importance of static code analysis in IaC and demonstrates how tools like Checkov can be pivotal in driving DevSecOps success.