Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Hallvard Torsvik Bamrud
Martin Clementz
Henrik Werner Lervåg
Oscar Stentun Stadskleiv

An Empirical Analysis of the Differences Between Terraform Security Scanners

Bachelor's thesis in Digital Infrastructure and Cyber Security
Supervisor: Tor Ivar Melling
May 2024

NTNU
Norwegian University of
Science and Technology

Hallvard Torsvik Bamrud
Martin Clementz
Henrik Werner Lervåg
Oscar Stentun Stadskleiv

An Empirical Analysis of the Differences Between Terraform Security Scanners

Bachelor’s thesis in Digital Infrastructure and Cyber Security
Supervisor: Tor Ivar Melling
May 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

An Empirical Analysis of the Differences Between Terraform Security Scanners

Hallvard Torsvik Bamrud
Martin Clementz
Henrik Werner Lervåg
Oscar Stentun Stadskleiv

Spring 2024

Abstract

Infrastructure as Code (IaC) tools such as Terraform, are powerful tools that could help infrastructure provisioners create and manage digital infrastructure. The use of these tools enables the provisioner to configure their infrastructure for their specific purpose. However, these tools do not guarantee a secure infrastructure, as misconfigurations could be overlooked. The use of Static Application Security Testing (SAST) tools can help identify misconfigurations and vulnerabilities before the infrastructure is deployed. In this thesis, our forcus is on evaluating three of the most popular SAST tools for Terraform configurations which use the Azure Resource Manager (AzureRM); Checkov, Terrascan and tfsec.

For this purpose, we created a data collection and testing framework to collect "maintained" projects. We identified 800 repositories that fit our criteria, and used these to evaluate and compare the three SAST tools.

To evaluate the tools we consider factors such as their rule coverage, ease of use, and their abilities to find security concerns in a variety of different repositories. We used a data-centric approach for this evaluation. Based on our findings, Checkov was the strongest of the tools, finding a total of 28 787 security concerns in 639 of the repositories. Terrascan identified 4 484 in 581 repositories, and tfsec identified 4 660 in 334 repositories. Checkov consistently identified more security concerns for all categories, except the secrets category where it performed equal to tfsec. Terrascan and tfsec were quite close in regards to the amount of security concerns found. However, tfsec had on average a higher amount found in the repositories where it identified at least one security concern. In addition to having the highest amount of rules, highest coverage and most security concern detections, Checkov was also the most stable of the tools tested.

After reviewing our results we found that Checkov was the most useful of the tools. We suggest that infrastructure provisioners use a combination of Checkov and Terrascan, as this gives a 99.7% coverage rate, which justifies the added performance cost.

Sammendrag

Infrastruktur som kode-verktøy (IaC-verktøy) som Terraform er kraftige verktøy som kan hjelpe driftere med å opprette og vedlikeholde sin digitale infrastruktur. Bruken av disse verktøyene tillater drifterne å konfigurere infrastrukturen i henhold til deres spesifikke behov. Likevel, vil ikke disse verktøyene garantere en sikker infrastruktur, og feilkonfigureringer kan bli oversett. Ved å ta i bruk statisk kodeanalyse-verktøy (SAST-verktøy) kan drifterne identifisere potensielle feilkonfigureringer og svakheter før infrastrukturen rulles ut. I dette studiet, er hovedfokuset å sammenligne tre av de mest populære SAST verktøyene for Terraform konfigurasjoner som bruker Azure Resource Manager (AzureRM); Checkov, Terrascan og tfsec.

Til dette formålet har vi laget et rammeverk for datainnsamling og testing, som samlet inn "vedlikeholdte" prosjekter. Vi identifiserte 800 kodebaser som passet til våre kriterier. Disse kodebasene brukte vi for å evaluere og sammenligne de tre SAST verktøyene.

For å evaluere verktøyene har vi evaluert antall kodebaser hvor verktøyene klarte å identifisere minst én sikkerhetsrisiko. Her så vi på faktorer som regeldekning, brukervennlighet og hvor mange sikkerhetsrisikoer de var i stand til å finne. Vi brukte en datasentrisk tilnærming i evalueringsprosessen. Basert på våre funn, er Checkov det beste verktøyet, med totalt 28 787 funn av sikkerhetsrisikoer i 639 av kodebasene. Terrascan identifiserte 4 484 i 581 kodebaser, og tfsec identifiserte 4 660 i 334 av kodebasene. Checkov identifiserte konsekvent flere sikkerhetsrisikoer i alle kategorier, bortsett fra i kategorien "secrets" hvor den presterte på nivå med tfsec. Terrascan og tfsec presterte nokså likt i forhold til antall sikkerhetsrisikoer de klarte å identifisere. Tfsec hadde imidlertid et høyere gjennomsnittlig antall funn i kodebaser hvor den fant minst en sikkerhetsrisiko. I tillegg til å ha flest antall regler, høyest dekningsgrad og flest identifiserte sikkerhetsrisikoer, var Checkov også det mest stabile verktøyet vi testet.

Basert på resultatene våre, fant vi ut at Checkov var det nyttigste verktøyet. Vi anbefaler infrastrukturdriftere å bruke en kombinasjon av Checkov og Terrascan, da dette gir en dekningsgrad på 99.7%, som rettferdiggjør den ekstra ytelseskostnaden.

Preface

This thesis is authored by Hallvard Torsvik Bamrud, Martin Clementz, Henrik Werner Lervåg and Oscar Stentun Stadskleiv as a bachelor's thesis in the bachelor's degree programme Digital Infrastructure and Cyber Security at The Norwegian University of Science and Technology in Trondheim. The thesis was written in the spring of 2024. As the authors of the thesis we wish to thank our supervisor Tor Ivar Melling for his guidance, recommendations and advice during the writing of this thesis. This thesis is a modified version of Tor Ivar Melling's bachelor thesis suggestion for IaC.

Contents

Acronyms and Glossary

RQ - Research Question

IaC - Infrastructure as Code

SAST - Static Application Security Testing

HCL - HashiCorp Configuration Language

CI/CD - Continuous Integration and Continuous Delivery

OPA - Open Policy Agent Framework

CLI - Command-Line Interface

API - Application Programming Interface

Git - Distributed version control platform for files

Rate limiting - Method to limit network traffic where the amount of requests are limited to a max amount per unit of time.

Alert fatigue - Fatigue originating from reading through too many alerts

Fork - “A fork is a new repository that shares code and visibility settings with the original ‘upstream’ repository.” [1]

1 Introduction

In this section we provide the background and topics for our thesis as well as our research questions. We establish the objective and scope, and explain some verbiage choices relevant for the reader. Finally, we outline the layout of this thesis.

1.1 Background

Infrastructure as Code (IaC) tools allows infrastructure provisioners to create and manage complex cloud infrastructure over time with version control systems. The ability to configure complete and complex systems demands technical knowledge from the provisioner on how to configure their IaC-code securely. There are multiple Static Application Security Testing (SAST) tools that promise to solve this problem and catch misconfigurations in IaC-code. These tools are often incorporated in continuous integration tests and allow provisioners to catch security issues in their configuration before they are applied to their infrastructure.

Choosing what security tool to use when testing IaC code is challenging. To our knowledge there are no empirical study with a main focus on comparing the strengths and weaknesses of each tool. The provisioner has to choose based on experience and open sources to find out which tool or set of tools could identify most of the misconfiguration in their IaC code. Instead of choosing one tool, the provisioner could also combine multiple tools to detect more misconfigurations. This could help detect more security concerns, but at the cost of duplicate findings, alert fatigue and performance.

1.2 Thesis Topic

This thesis covers the topics IaC, cloud infrastructure security, SAST tools, Terraform and data collection from public repositories. Our main focus will be to try to determine which single SAST tool or combination of tools would be the optimal solution to help infrastructure provisioners secure their IaC configurations.

1.3 Problem Statement

This thesis delves into some of the complexities of securing IaC repositories using SAST tools. To address this, we have formulated the following problem statement:

What are the differences between the Terraform security scanners Checkov, Terrascan and tfsec, and which single tool or combination of these tools do we recommend to implement into IaC workflows?

1.3.1 Research Questions

We have broken the problem statement down into the following research questions (RQs):

RQ1: What are the challenges and difficulties of collecting and subsequently analysing data from publicly accessible code repositories? With this question we wish to explore challenges which present themselves when working on a dataset made of publicly accessible code repositories, with our three chosen SAST tools.

RQ2: What is the Performance Impact of Using Terraform Security Scanners on Build and Deployment Pipelines? This question explores the impact on Continuous Integration/Continuous Deployment (CI/CD) processes, including performance and speed.

RQ3: What are the main differences between the different SAST tools in terms of their rulesets and performance? This RQ aims to showcase the differences between the tools, with a focus on rulesets, categories and performance. This will be done by discussing the tools individually before comparing them to each other.

RQ4: What single scanner or combination of scanners provide the best security guidelines for IaC repositories? This question will act as the conclusion of our work, answering which tool or combination of tools we found to be most effective. We will also give our advise on what combination of tools we suggest provisioners should use.

1.4 Objective and Scope

Objective

This thesis aims to conduct a comprehensive comparison of three of the most popular security scanners for IaC configuration files: Checkov, Terrascan and tfsec. The primary goal is to document the strengths and weaknesses of each tool, to help infrastructure provisoners choose which tool or tools they should incorporate in their workflow. This research aims to contribute to a broader understanding of how security is managed in IaC environments, and provide useful insights into the current state of IaC security. These observations could help guide the development of future tools and practices for enhancing security in automated infrastructure management.

Scope

This thesis looks at publicly available repositories from two popular Git providers, GitHub and GitLab. Private repositories are not in the scope, as we do not have access to a large amount of private repositories containing IaC code. Publicly available repositories are easier to run SAST tools on, as their licence grants us access to download their source code.

We further narrow our scope by only looking at one popular IaC language in combination with a single cloud provider. Through discussion with our supervisor we decided to limit the scope to analysing configurations that use Terraform with the AzureRM provider.

Further narrowing of the scope was applied to only look at three SAST tools for Terraform: Checkov (v3.2.74), Terrascan (v1.19.1) and tfsec (v1.28.5). These tools are widely used and are among the most popular code scanning tools for Terraform. Looking at the GitHub star count from their own repositories confirms that the tools are widely used. Checkov has 6.6 thousand stars [2], Terrascan has 4.5 thousand [3] and tfsec has 6.6 thousand [4]. More specific considerations for deciding the scope are listed below.

- **Project deadline**. This thesis is written as a bachelor thesis. We therefore have limited time to conduct our research into all security aspects of the IaC tools landscape. This was the main reason we decided to use only three security scanners, and to just look at one IaC tool in combination with one cloud provider.
- **IaC tool popularity**. According to Statista, Terraform is the second most used configuration language, with a usage of 30% [5], making any security findings relevant to many users. It is also ranked by 6sense as the number one in market share in the configuration management market, with a 32% market share [6]. The configuration language's popularity makes it realistic to find enough open source data to analyse, as well as making our findings relevant for many users.
- **Cloud provider popularity**. Azure is the second largest provider of cloud services in terms of market share globally [7], making security findings relevant to many users. The reason for choosing Azure over AWS is that we are more familiar with Azure from prior experiences.
- **Lack of research**. We were not able to find any research that completed a comprehensive empirical comparison of Terraform security scanners. Some comparisons exist, which is discussed further in section 2.3, but not to the scale of this thesis.

1.5 Verbiage

Security Concern

In this thesis we have chosen to use the term "security concern" to describe the security flaws, holes, faults, weaknesses, vulnerabilities and concerns we find in our dataset. It is important to note that the "concerns" are perceived by the software we use, as flaws, but in the context they are used they may not pose any risk. Rahman et.al. performed a study, looking at the security smells they could identify in open source repositories. After the study they asked the owners of the repositories whether they agree with their findings in fact being flaws. 30.2% of participants disagreed with the researchers [8]. Our study uses a different dataset and different checks from the aforementioned paper. However, we do recognise that what we and our software sees as flaws may not pose a risk, and if the developers are aware and cautious of them, they may not be flaws or concerns at all. Therefore, when we use the term "security concern" going forward, this is in the context of it being something the scanning tools flag as flaws. However, whether they are actual concerns for the security, might be up for a debate. What we find might not be considered as a concern from the developer at all, but rather their intended design.

Checks

Checkov refers to the tests they run as *checks*, as do tfsec, but Terrascan uses the word *policy* for the tests they run. To avoid any confusion we have chosen to standardise this and use the word *check* from here on out in the thesis when we refer to either of these.

1.6 Thesis Outline

Introduction
In the introduction, we introduce our thesis through background, thesis topic, problem statement, research questions, objective and scope and thesis outline.

Theory:
In the theory section, we explain some fundamental and more complex theory used in the thesis. This will give a good base of understanding, to make the thesis more comprehensible. This section also includes an overview of some of the related works that inspired our thesis.

Method:
In the method section, we explain how data collection and data analysis is conducted. The entire data collection process is explained. This includes criteria for which repositories are selected. In the data analysis part of this section the process of each security scanner is explained, as well as the creation of the combined categories, created to unify the results from the different tools.

Results:
In the results section, we present the results from the analysis. This includes graphs to convey the results clearly, as well as comparisons between the different scanners.

Discussion:
In the discussion section, we discuss the results and further elaborate on the reasoning behind some of the choices that were made. RQ 1, 2 and 3 will be answered and

further discussed in this chapter, with a subsection each.

Conclusion:
In the conclusion section, we conclude the thesis by stating ideas for future work and giving our answer to RQ4 as well as concluding the project as a whole.

2 Theory

This chapter contains the theory that the thesis is based on. It consists of two parts. The first part lays the foundation needed to follow the thesis. This is done by elaborating on concepts and definitions used. It contains terms the reader should know and also contributes to the understanding of why the methods used in the next chapter was chosen. The second part takes a look at related work, where similar projects and reports are mentioned. These reports created the base for this thesis by enlightening what results had been found prior and what areas could be expanded upon.

2.1 Concepts and Definitions

IaC

Kief Morris explains Infrastructure as Code as "an approach to infrastructure automation based on practices from software development. It emphasizes consistent, repeatable routines for provisioning and changing systems and their configuration. You make changes to code, then use automation to test and apply those changes to your systems."[9]. This enables developers to apply changes in the infrastructure by the press of a button. However it also means a single bad line of code could lead to an entry point for an attacker.

Terraform

Terraform is a declarative configuration language [10] that uses HashiCorp Configuration Language (HCL). Declarative in this sense means the developer only needs to state what they want in the completed infrastructure: amounts of servers and virtual machines, what network devices should connect to and more. The developer can build the entire infrastructure or remove it by a simple command at any given moment. Terraform uses "providers", a plugin for interaction with remote systems [11]. Many major cloud providers have already created blocks that can be used with their cloud systems, and anyone can develop and distribute their own provider.

Azure

Microsoft Azure is the second largest cloud provider, with a 24% market share in the Cloud provider market [7]. The Azure Resource Manager (AzureRM) is the fourth most installed Terraform provider [12]. Azure is a cloud provider which allows its users to commission resources in the cloud that can be used for infrastructure, development and compute, among other things [13].

Continuous Integration and Continuous Delivery

Continuous integration and continuous delivery (CI/CD) “aims to streamline and accelerate the software development lifecycle” [14]. CI refers to integrating code changes into the main branch of a shared code repository frequently, and having automated testing when changes are committed and/or merged. CD refers to automating infrastructure provisioning and the application release process. [15]

GitHub Actions

GitHub Actions allows developers to automate, customise and execute software development workflows in their repositories. GitHub Actions are written with YAML syntax, and developers can create their own actions, or use premade actions created by other developers [16].

GitHub Actions Matrices

Matrices in GitHub Actions allow developers to create multiple jobs automatically by defining input parameters. The jobs created will include all possible combinations of those inputs. The jobs are independent of each other and can be run simultaneously. For example a matrix with a one-dimensional list with two items as input will create two jobs.[17]

Static Code Analysis

Static code analysis is most commonly performed by running static code analysis tools on non-running code, with the objective of uncovering possible weaknesses or vulnerabilities [18].

Closed Card Sorting

Card sorting is an exercise in which the participants get a set of cards/data and label them into groups. In closed card sorting these groups or categories are predefined, whereas in open card sorting there are not predefined groups or categories [19].

2.2 Terraform Security Scanner Tools

Checkov

Checkov is a tool used to statically analyse IaC code [20]. It searches for security concerns in the code and reports these concerns with a severity level and what security concern rule the configuration violated. Checkov allows developers to create custom checks using Python and YAML.

Terrascan

Terrascan is a static code analysis tool for detecting security concerns in Terraform configurations. It supports different Terraform providers and can be extended with custom checks written in the Rego language with JSON rule files. Terrascan has a CLI tool and an official GitHub Action which provisioners can use in their CI/CD pipelines [21].

Tfsec

Tfsec is a static code analysis tool for detecting security concerns in Terraform configurations. It is built on the defsec engine and is extendable with new rules and Terraform providers [22, 23]. Tfsec is an open source project which accepts contributions, making it possible for the community to add new security checks written with JSON or YAML [24].

2.3 Related Work

There have been several studies on IaC security using datasets consisting of public code repositories. We took inspiration from some of these studies for our own repository collection [25, 26, 27, 28]. Iosif et al. (2022) looked at Terraform code with the AWS provider, testing with Checkov, Terrascan and tfsec [28]. Verdet et al. (2023) tested Terraform code with the AzureRM provider (in addition to AWS and Google Cloud) using Checkov [27]. These studies deal with a lot of the same type of data as we do, in addition to utilising the same tools, thus providing valuable insights we have utilised in our own thesis. In their study Iosif et al. (2022) also conducted a short comparison of the three SAST tools they used. This provided valuable perspectives for our thesis, as well as motivation to perform a more in-depth comparison between the SAST tools. They concluded that the best solution would be to use a combination of the three tools. In addition to these studies there are several less formal works, such as posts in tech-blogs that have conducted comparisons of IaC scanning tools [29, 30, 31]. While these do not have the depth and formality of a research paper, their results are still valuable to us both to compare our own findings with, and to get a sense of what characteristics other users value in a scanning tool.

3 Method

In this section we delve into the methods we used to conduct the research for this thesis. We discuss which criteria we have set for the repository collection process, how the technical data collection and testing processes have been conducted and lastly which methods have been used for the data analysis.

3.1 Data Collection

The data collected for this thesis is gathered from GitHub and GitLab, and is comprised of Terraform files that uses the AzureRM provider. In order to only use relevant data we have put restrictions on which repositories to accept, using the criteria we establish in this section. Another part of the data collection process is how we gather the repositories that meet our requirements and subsequently how these are tested. Lastly we will look into the security aspects of the data collection.

3.1.1 The Criteria

In this thesis our focus is on maintained projects, similar to the engineered projects defined by Munaiah, N. et al. as "a software project that leverages sound software engineering practices in one or more of its dimensions such as documentation, testing, and project management" [32]. The repositories we consider to be maintained projects are held to a somewhat lower standard than what Munaiah, N. et al. defines as their criteria for engineered projects. The reason for this is that we wish to have a large enough dataset to properly test the three scanning tools. The point of the study is to test how these tools respond to different sorts of security concerns, not how professionally the repositories are managed. Therefore, having less professional or engineered projects included in the dataset was something we did not consider to invalidate the research. We do however wish to test the tools on active and legitimate projects, so we have chosen criteria that will filter out peoples homework, experimental configurations etc.

Taking inspiration from prior research [32, 33, 34] we established the criteria below.

- **Criteria 1 - Not a fork:**
  The repository can not be a fork.
- **Criteria 2 - Number of contributors:**
  For a repository to be accepted it will need to have more contributors than the threshold of 3.
- **Criteria 3 - Total number of commits:**
  The repository will need to have more than 20 commits.
- **Criteria 4 - Duration:**
  The repository has to be older than 26 weeks

- **Criteria 5 - Recent activity:**
  For the repository to be considered active it needs to have had activity within the last year.
- **Criteria 6 - Issues:**
  The project needs to have 10 or more issues.

3.1.2 Technical Data Collection Process

We created a data collection and testing framework to collect and process the repositories that fell within the scope of this thesis. The framework finds desired repositories from GitHub and GitLab and run specified security checks on discovered repositories matching the collection criteria. The results are stored in a database for further analysis. This framework can be found in appendix A.

The framework is built to be extendable with new repository discovery methods for the same or new Git providers. The framework is easily extendable with new static code analysis tests on the discovered repositories. The data collection and testing parts of the framework are run independently.

The testing framework works in stages. When a stage is finished it moves on to the next stage. This was achieved by building the framework on top of GitHub Actions. GitHub Actions provides a simple abstraction for parallelising workloads across virtual machines [35] and the ability to run the workloads using GitHub runners [36] or locally run them using Act [37]. Figure 1 shows a simplified structure of how the framework's data collection process works.

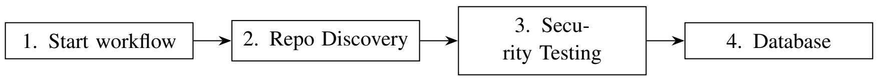


Figure 1: Data collection process

The first step of the workflow in figure 1 uses the GitHub Actions "workflow_dispatch" feature [38] to initialise the data collection process with a specific repository discovery method, using the method's name as input. This makes it possible to initialise different data collection processes independently. In our thesis we are focusing on GitHub and GitLab, and added the names of the repository discovery methods we created as inputs.

Initialising a discovery method starts the corresponding discovery process in step 2 in figure 1. The purpose of a discovery method is to find all the public repositories from the selected Git hosting provider that matches our criteria which was established in section 3.1.1. However, due to the Git providers API limitations, which are discussed further in section 5.1 we found it necessary to add more discovery methods for a single Git provider.

The first discovery method for GitHub consist of querying their code search API. This API allows authenticated users to get repositories that contain code the user wants to find. We used the search query `azurerm extension:tf` finding all `.tf` files on GitHub containing the string "azurerm". This would according to the API [39] get the correct results, but when testing their API, we got only 669 repositories from this query. We have a hypothesis that there should be well over 669 repositories on GitHub containing HCL files with the string "azurerm", so we created a second discovery method for GitHub.

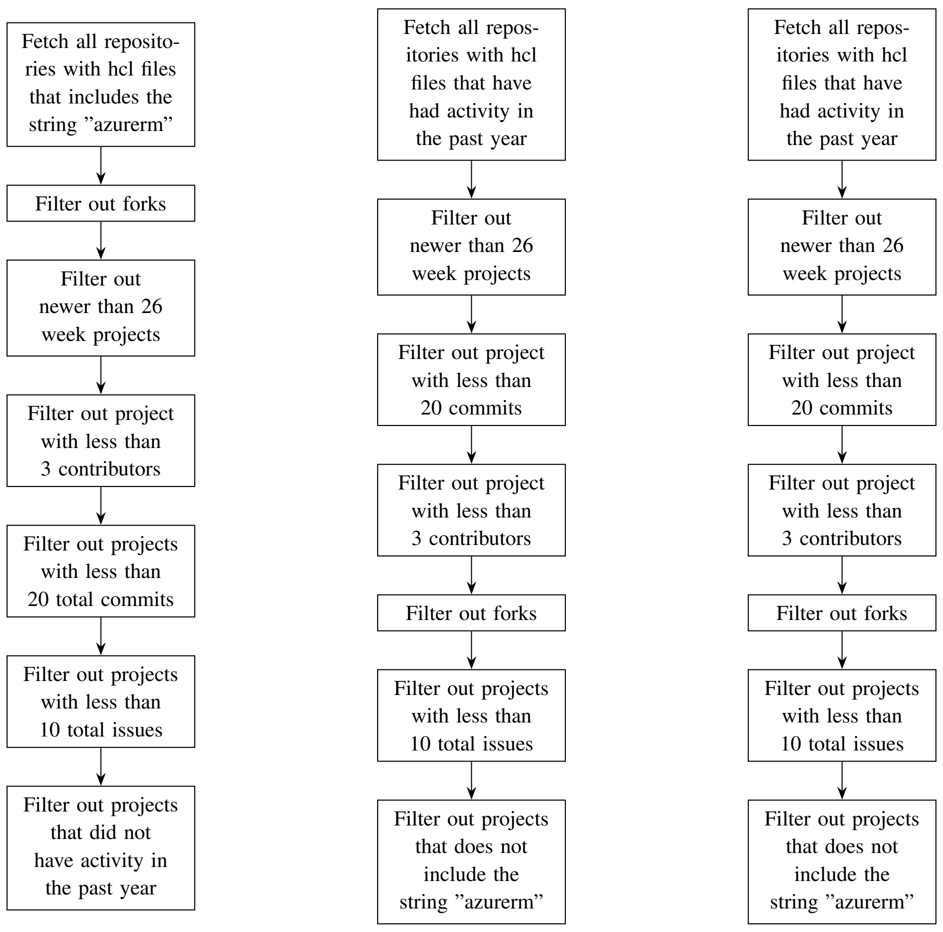
The second discovery method for GitHub consist of querying their repository search API. This API allows authenticated users to get repositories matching criteria the user wants. We used the search query `language:hcl pushed:\>2023-04-01` matching all repositories that have HCL as their most used language and some Git activity since last year. The API found 127 089 repositories matching this criteria.

Due to limitations on the search API it would not be possible to get more than 1 000 results per query [40]. To get each search query from 127 089 results down to 1 000 we used the query functionality to query date ranges `pushed:FROM-DATE..TO-DATE`. This would limit the scope of the search without compromising on the findings. We implemented a divide and conquer algorithm [41] to do this search efficiently and get all the repositories that had been updated in the past year. The algorithm works by calling the search function recursively and dividing the time scope in half until the total results are under 1 000. The found repositories are returned and merges together into one list.

For GitLab we created one discovery method that consists of querying their repository search API. This API allows unauthenticated and authenticated users to get repositories marching the user's criteria. We searched for all repositories having HCL as their primary language and their last activity being after the 1st of April 2023.

The repository discovery methods need to implement their own filtering ruleset because of the differences in the Git providers' API. In figure 2 we have outlined how the different methods filter out repositories after the initial discovery. The only difference is the order of when the filter rules are applied. The same ruleset is applied to all methods which will result in the same final results regardless of when a repository was filtered out.

The repository discovery methods are implemented in TypeScript and runs using the Bun JavaScript runtime [42]. The use of a typed high level programming language makes the code easy to read and work with. The use of Bun is to improve the performance and memory while doing network requests [43]. Network requests are the main computation this process does since it mostly fetches data from the APIs.


Fetch all repositories with hcl files that includes the string ”azurerm”
Filter out forks
Filter out newer than 26 week projects
Filter out project with less than 3 contributors
Filter out projects with less than 20 total commits
Filter out projects with less than 10 total issues
Filter out projects that did not have activity in the past year
Fetch all repositories with hcl files that have had activity in the past year
Filter out newer than 26 week projects
Filter out project with less than 20 commits
Filter out project with less than 3 contributors
Filter out forks
Filter out projects with less than 10 total issues
Filter out projects that does not include the string ”azurerm”
Fetch all repositories with hcl files that have had activity in the past year
Filter out newer than 26 week projects
Filter out project with less than 20 commits
Filter out project with less than 3 contributors
Filter out forks
Filter out projects with less than 10 total issues
Filter out projects that does not include the string ”azurerm”

(a) GitHub Code search (b) GitHub repository search (c) GitLab

Figure 2: Repository discovery methods

The repository fetching job outputs a list containing URLs for repositories that match our criteria.

Through the filtering process the results for each step is uploaded to the database. If a repository passes a filter it will log that in the database.

Discovery Process Results

The discovery process for each of the discovery methods worked as expected. Table 1 shows how the results from the repository collection are affected by the filtering rules applied to the initial dataset from the APIs. The rules in the first column refers to the rules from the stages in figure 2.

The table shows that most of the filtering rules was applied successfully. We can see that some repositories was discarded at each stage. This is expected as each rule combines with the previous and sets a overall stricter criteria on the repository.

Filtering rule 3 in the GitHub discovery methods seems to not work. This rule is "not a fork" and we expected that at least some repositories would be discarded by this rule. This is further discussed in 5.5. The implementation of this rule was based on the GitHub API response schema [40] and checked the boolean value of `fork` on the repository object. This value was false on all repositories we discovered from GitHub.

Rule	GitHub code search	GitHub repository search	GitLab
0	669	127 089	9 844
1	669	74 012	5 674
2	557	21 398	2 727
3	269	6 000	1 377
4	266	6 000	1 045
5	143	4 094	77
6	115	706	7
Discovered repositories	115	706	7

Table 1: Repository discovery results for our different discovery strategies

Because we implemented two strategies for GitHub, we expect there to be overlap in the discovered repositories. Table 2 shows the amount of unique repositories discovered from each provider.

	GitHub	GitLab
Total	793	7

Table 2: The total amount of discovered repositories

Technical Testing Process

The technical testing process occurs in step 3 in figure 1, and a more in-depth overview of what occurs in this step can be seen in figure 3. The testing steps for each of the tools follows the same pattern, but there are slight differences in how each of the tools are used within the workflow.

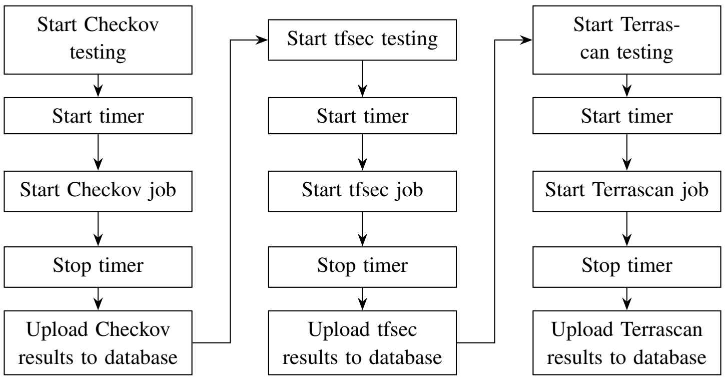


Figure 3: Security testing process

Before the workflow can run the security tests the framework transforms the list containing URLs into GitHub Action matrices [35]. Using matrices on Action jobs allows them to be run in parallel, which enables the framework to do concurrent static code analysis on the discovered repositories.

The process of transforming the list of repositories to a valid matrix is done by creating nested matrices. A matrix containing other matrices is used to bypass GitHub Actions' limit on matrix sizes of 256 items. While a 1-layer matrix can create 256 jobs, a 2-layer nested matrix can create 65 536 (256 x 256) jobs.

The security testing job is now set up correctly to run parallel security testing on the discovered repositories. We chose to test a maximum of 22 repositories at the same time, making the whole security testing process take 1 hour 19 minutes and 47 seconds.

The security testing workflow that runs the three scanner tools on a repository are set up to run the tools sequentially. It works by first downloading the testing framework and installing the necessary dependencies. After the framework is set up correctly it will download the source code of the repository it is going to test.

The fist security scanning tool to be tested is Checkov. We first have to specify which checks we want to run, which is all AzureRM checks from Checkov. After this a timer is started, and then the Checkov GitHub Action from thebridgecrew [44] is run, which is where the checks are run. When the checks are finished the timer is stopped, and then the results are uploaded to the database.

The tfsec checks are run after the Checkov checks. The tfsec steps are similar to the Checkov steps, with the exception that we do not need to specify which checks we want to be run. The first step is to start the timer, then we run the tfsec GitHub Action from Aqua Security [45]. When the checks are finished we stop the timer, and then we upload the results to the database.

The last tool we run is Terrascan. These tests have to be run in a different manner than the previous two. There exists a Terrascan GitHub Action [46], but it does not allow us to change the format of the output (besides creating a SARIF file). We need the output of the tests in JSON format for us to be able to parse through them and upload the results to our database. For this reason we have not used the Terrascan GitHub Action, but rather installed the tool in the workflow and run it through a command. Although the steps for running these tests are somewhat different to the steps for Checkov and tfsec, it will work in practically the same way as if we had used the Terrascan GitHub action. The first step we do is to install the Terrascan tool. We then start the timer and then the tests are run with a command that specifies only to run Azure tests on Terraform files. We then run a test to see that the checks have finished and the results have been written to a JSON file. After this we stop the timer and lastly the results in the JSON file are parsed and uploaded to the database.

3.1.3 Security Aspects of the Data Collection Process

The repositories we are checking for security concerns are real public code repositories with code that are potentially being used in active infrastructures. If we release information about any of these repositories having security concerns it could harm the repositories. It is therefore important for us to treat this data with care and in a way that does not leak any information to the outside world. We have therefore taken some security precautions to ensure the security of the repositories in our dataset.

Private repository: The code repository we have written our own code in is a private GitHub repository which are only accessible by the authors of this thesis.

Password protected database: The database we have used for storing data about the different repositories and test results is a password protected database. The password is a 20 character long randomly generated password including special characters. Only the authors of this thesis have access to the database and the raw results.

No publication of repository-specific data: We will not be publishing any repository specific data, such as name of repositories, names of contributors etc. Therefore we will not publish any information which can connect a specific repository to a specific security concern. We will also delete the raw data after the delivery of our thesis.

3.2 Data Analysis Methodology

This section describes the scope of our security testing and how we collect the results. We use three different security tools that has their own set of misconfiguration checks. This section lists each tool's checks grouped by their internal category. The checks are further categorised to fit a common categorisation set by us. In appendix C, D, E and F all checks that we run are listed, with their original and new category specified.

3.2.1 Categorise Security Checks

Each tool has their own internal way of representing a misconfiguration, which makes it difficult to compare the raw output from the tools. We created common categories to solve the problem of comparing security findings. The common categories are based on a combination of the tools' internal categorisation from Checkov and Terrascan. The individual tests have been distributed in the common categories based on their former categories for Checkov and most Terrascan checks. For Terrascan checks with the former category "Infrastructure Security" and all tfsec checks, we used closed card sorting to place them in the new categories. Tfsec does not have internal categories and there is a large span of different tests in the "Infrastructure security" category from Terrascan, which is why we sorted them this way. Table 3 shows the common categories we established.

Common categories	Checkov	Terrascan	tfsec
General Security	General security Kubernetes Application Security Misc*	Infrastructure Security Security Best Practices Compliance Validation	-
Networking	Networking	Infrastructure Security	-
Identity and Access Management	Identity and Access Management	Identity and Access Management	-
Encryption and Data Protection	Encryption	Data Protection	-
Logging and Monitoring	Logging	Logging and Monitoring	-
Backup	Backup and Recovery Convention	Resilience	-
Secrets	Secrets	-	-

Table 3: Common and original categories
*: We made this category to cover Checkov policies with custom categories.
-: No category

In table 4, the amount of misconfiguration rules per category per tool is shown. We see that Checkov has an equal amount of, or more rules than the others in all categories, except networking, where Terrascan has the most. It is also worth noting that tfsec has no checks in the backup category, and Terrascan has no checks in the secrets category. All of the rules were given categories after they were uploaded to the database, using an SQL update query.

Categories	Checkov	Terrascan	tfsec	Total
General Security	98	8	7	113
Networking	92	138	12	242
Identity and Access Management	23	11	18	52
Encryption and Data Protection	31	5	2	38
Logging and Monitoring	17	13	11	41
Backup	15	3	0	18
Secrets	2	0	2	4
Total	278	178	52	508

Table 4: Amount of rules per category for the different tools

3.2.2 Technical Analysis Methodologies

All relevant data from GitHub and GitLab is stored in a database. From this database, we can retrieve our results, and filter the results to properly record statistics. As we are comparing the tools and checks in different categories, we made sure to have columns specifying these in the database, enabling concise database queries. To get a good overview of the differences and similarities, we used Microsoft Excel to format graphs and tables in a different view than just the raw data results. We chose to look at the result through several different statistical views for the project. This includes column and bar graphs, both for the individual tools, and a comparisons through clustered graphs.

4 Results

In this chapter, we will take a look at the results of the individual tools one by one, followed by a comparison of their results.

4.1 Checkov

Checkov was able to identify at least one security concern in 639 of the 800 repositories giving a coverage of 79.86%. In total, it was able to identify 28 787 security concerns in these repositories. This gives an average of 45.05 security concerns for each repository that has at least one concern, or an overall average of 35.98 based on all repositories. The distribution of these checks between categories can be seen in figure 4. This shows that Checkov was able to find a lot within the general security and networking categories, while only one concern was found in the secrets category.

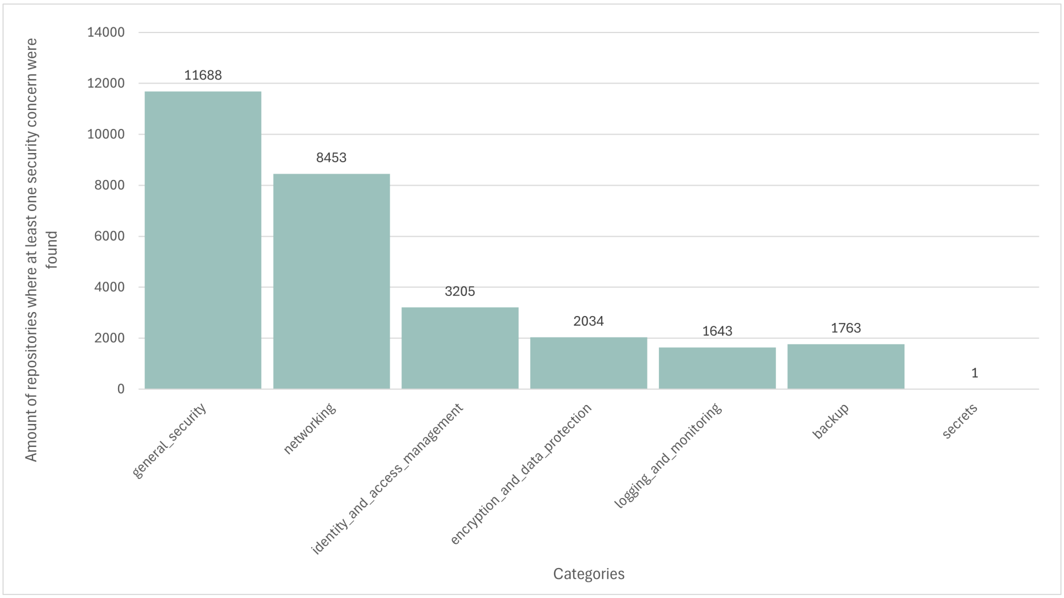


Figure 4: Graph showing the distribution of security concerns Checkov was able to identify by category.

There is a major difference between the different repositories when it comes to size, complexity and different implementations. Although Checkov identified the average amount of security concerns per repository with at least one concern to be 45.05, the median for these repositories were only 17. This shows that there is a vast difference between the repositories where it identified the most, opposed to the least amount. As this is the case, we made a graph that shows how many repositories contain a certain amount of concerns. Figure 5 shows the distribution of these concerns. From the graph, we can see that there is a clear correlation between the amount of repositories and the amount of concerns found. Checkov identified less than 45 security concerns in 80% of the repositories where it identified at least one security concern. There is a spike with 17 repositories that has between 171 and 175 security concerns. This is a spike that came from a vulnerable-by-design repository which has been cloned and used as a base for multiple demo repositories.

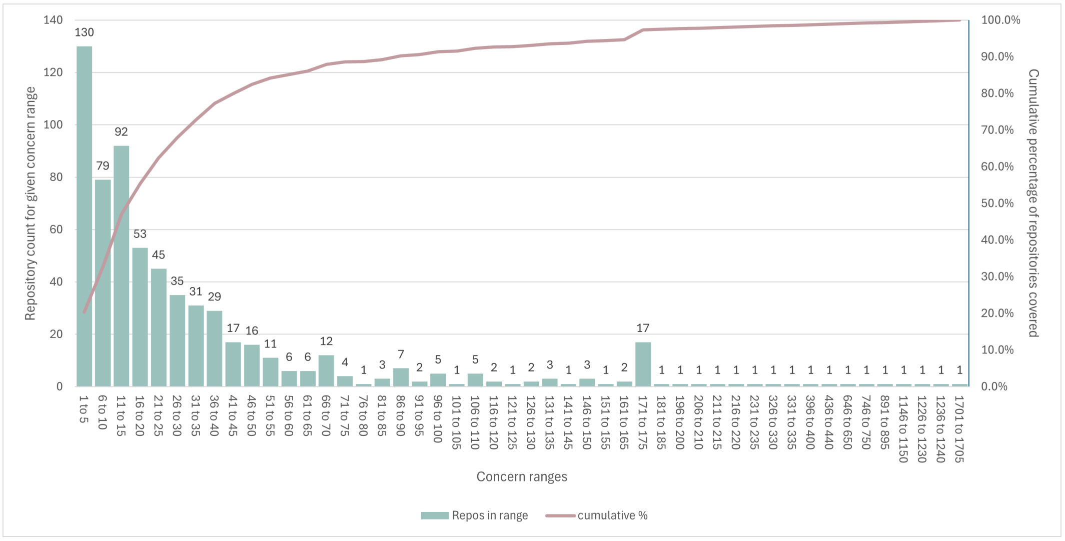


Figure 5: Checkov: Amount of repositories within ranges of security concern count, with a cumulative percentage of total amount of repositories represented.

4.2 Terrascan

Terrascan was able to identify at least one security concern in 581 of the 800 repositories, giving it a coverage of 72.63%. It was able to identify a total of 4 481 security concerns in these repositories, resulting in an average of 7.72 security concerns per repository where it identified at least one, or an overall average of 5.61 based on all repositories. The distribution of concerns by category can be seen in figure 6. From this we can see that Terrascan finds a considerable amount in the networking and identity and access management categories, with fewer results in the other categories, and none in the secrets category as it did not have any checks in this category.

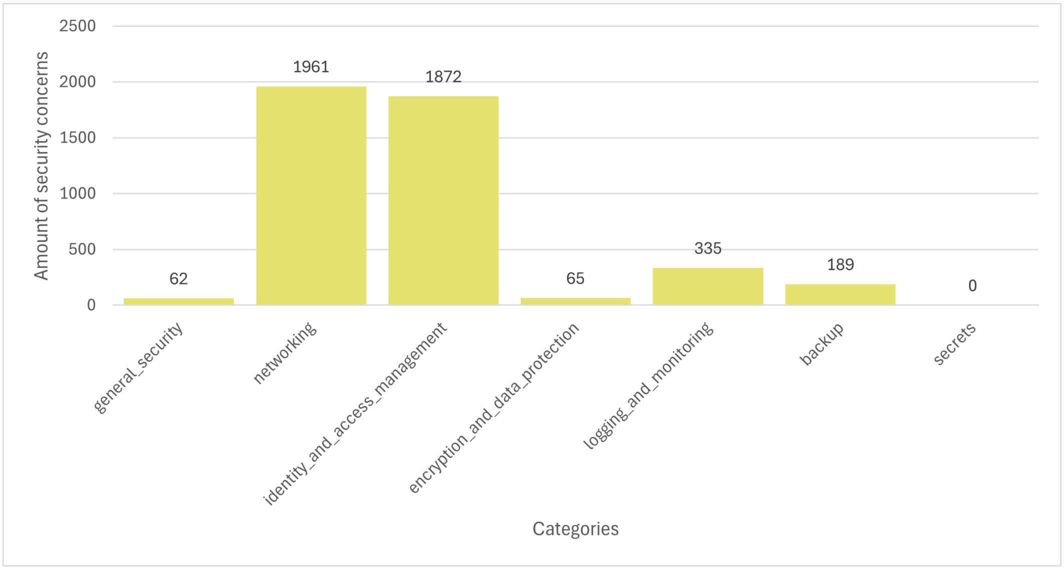


Figure 6: Graph showing the distribution of security concerns Terrascan was able to identify by category.

Due to the major difference between the different repositories when it comes to size, complexity and different implementations, a count or average might not be the best metric to go by. Although Terrascan was able to identify an average of 7.72 for repositories it detected at least one security concern in, the median for these repositories were only 4. The difference in average as opposed to median shows that there is a great deal of variance, and that there are a few repositories which increase the average substantially. The distribution of repositories that contain a certain amount of concerns are shown in figure 7. Here we can see that there is a clear correlation between the amount of repositories and the amount of concerns found. The vast majority of repositories only contain a few security concerns each, where 80% of repositories contains less than 10 security concerns.

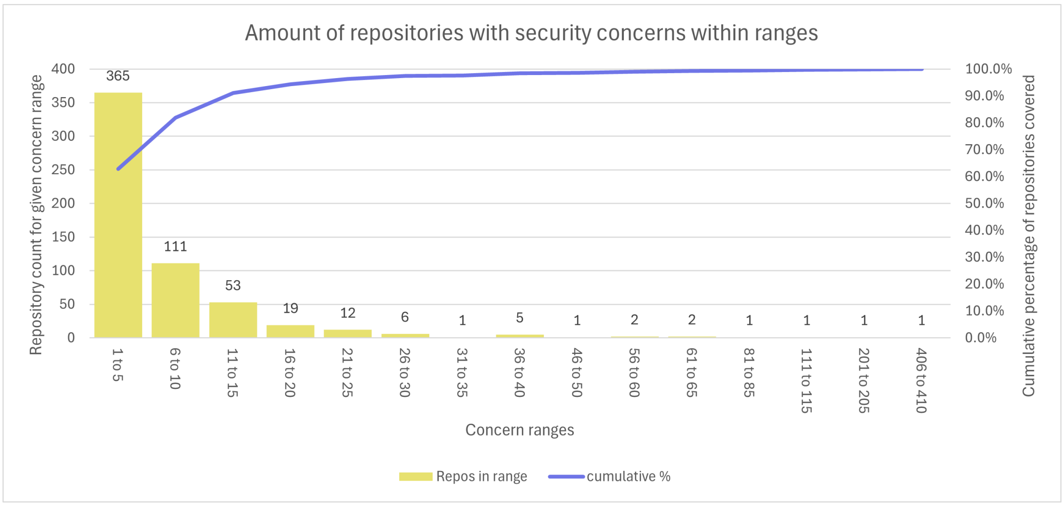


Figure 7: Terrascan: Amount of repositories within ranges of security concern count, with a cumulative percentage of total amount of repositories represented.

4.3 Tfsec

Of the 800 repositories analysed, tfsec was able to identify at least one security concern in 334 of these. Giving a coverage percentage of 41.75%. It was able to identify a total of 4 660 security concerns in these repositories. This gives an average of 13.95 security concerns per repository where it was able to identify at least one, or an overall average for all repositories of 5.83. The distribution between categories of these checks, is shown in figure 8. Tfsec was able to find a lot within identity and access management, and networking, with only one found in secrets. It did not have any checks that fell within the backup category.

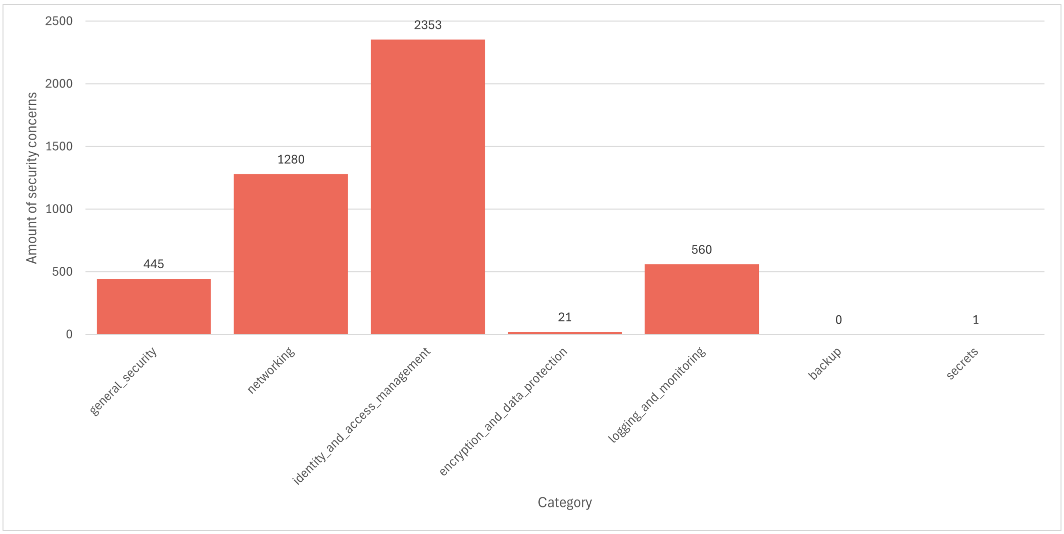


Figure 8: Graph showing the distribution of security concerns tfsec was able to identify by category.

Tfsec identified the average amount of security concerns per repository to be 13.95 (for repositories where it was able to identify at least one), with a median value of 5. This shows that there is a substantial difference between the repositories where it identified the most, as opposed to the least amount of security concerns. Figure 9 shows the distribution of the security concerns. The graph shows a clear correlation between the amount of repositories and the amount of concerns found. In tfsec, as with the other tools, it is evident that the vast majority of repositories contain only a few security concerns, with 80% of the repositories having fewer than 15 security concerns. There is a spike with 15 repositories that has between 86 and 90 security concerns. This is a spike that came from a vulnerable-by-design repository which has been cloned and used as a base for multiple demo repositories.

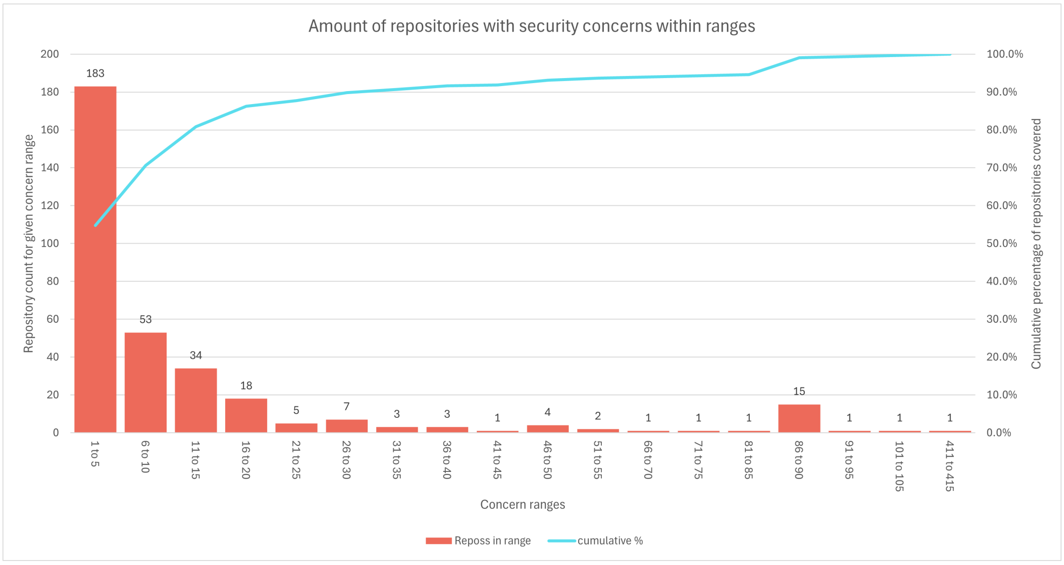


Figure 9: Tfsec: Amount of repositories within ranges of security concern count, with a cumulative percentage of total amount of repositories represented.

4.4 Comparison

4.4.1 Security Concern Detection Coverage and Overlap

In assessing security concern detection across repositories, it's vital to consider the coverage and overlap between the three tools. The coverage is based on how many of the 800 repositories they were able to find at least one security concern within. Checkov was able to identify this in 639 repositories, Terrascan 581, and tfsec was only able to identify security concerns in 334 of the repositories.

The coverage percentages are measured as repositories where the tool identified one or more security concerns divided by the total count of repositories. Checkov boasts the highest coverage rate at 79.9%, followed relatively closely by Terrascan at 72.6% and tfsec trailing behind at 41.8%. More on why tfsec scores such a low coverage rate, is discussed in section 5.3.

The average time is based on the time to scan through a complete repository, and only includes times from repositories where the tool was able to identify security concerns. The average times are visualised in figure 10. Tfsec was the fastest tool with an average time to scan a repository at a mere 4.16 seconds, with Terrascan at 12.60 and Checkov following behind at 13.94 seconds. From these observations, tfsec uses roughly a third of the time compared to the other tools, which are relatively similar in scanning time.

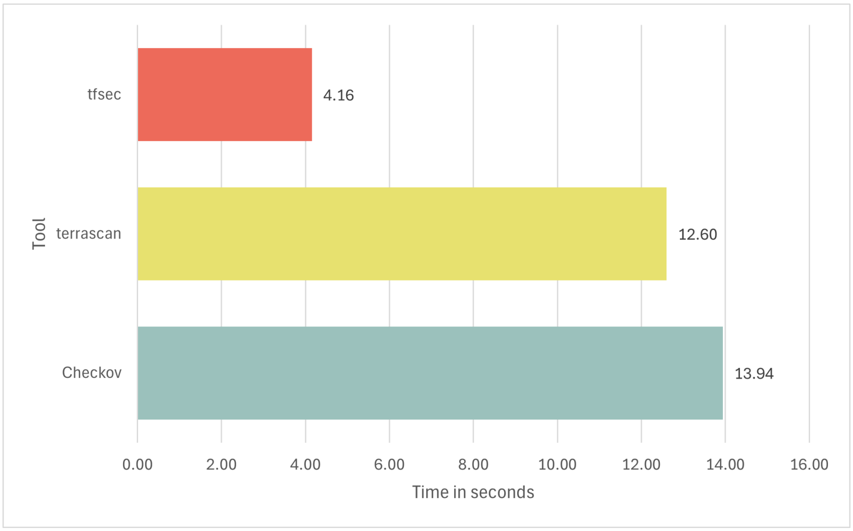


Figure 10: Average time to run the tool against a repository

Table 5 provides a more comprehensive breakdown of security concern detection and overlap between the three scanning tools. Each column depicts a specific scenario. These scenarios include repositories devoid of security concerns detected by any of the tools, repositories where either just one or two of the tools were able to identify any security concerns, and where all tools detected security concerns. The ”Combined” row aggregates the repository count.

For instance, none of the tools were able to identify any security concerns in 112 repositories, and they all agreed that there were security concerns in 281 repositories. The overlap is most apparent in the columns where either two or all three were able to identify security concerns.

In the column where 2 of the tools were able to find security concerns, there is significant overlap. There was a total of 304 repositories in this column of the table. The sum of all the repositories the individual tools found is 608, which indicates that there is a notable degree of redundancy in the findings, further indicating a substantial overlap in their detection capabilities. Again, Checkov proves to be the most thorough as it found security concerns in 298 of the total 304 repositories. Meaning that Terrascan and tfsec were only able to agree on 6 repositories where Checkov was unable to find any security concerns. Overall, this indicates that crossreferencing among the tools can increase their reliability, reducing the risk of false negatives.

Tools	**0**		**1**		**2**		**3**		**Total**	
Combined	112	14.0%	103	12.9%	304	38.0%	281	35.1%	800	100.0%
Checkov			60	7.5%	298	37.3%				
Terrascan			41	5.1%	259	32.4%				
tfsec			2	0.3%	51	6.4%				

Table 5: Number of tools finding at least one error

In table 6 we takes a look at what percentage of the repositories the tools found zero security concerns in. Another interesting metric to observe, is the difference between the total rules and the amount of rules that were triggered once or more. In Checkov and tfsec, most of the rules from the ruleset found security concerns. In Terrascan however, only 45 of the total 178 got any hits. Why this was the case, will be further discussed in 5.3

Metric	Checkov	Terrascan	tfsec
Zero security concern	20.1%	27.4%	58.3%
Avg. security concern #	45.05	7.72	14.95
# total rules	278	178	52
# rules that were triggered once or more	259	45	51

Table 6: SAST tool comparison: clean repositories, average security concerns, and rule trigger statistics

4.4.2 Security Concern Distribution based on our Categories

When comparing the performance of Checkov, Terrascan, and tfsec across different categories of security concerns, significant variations emerge, highlighting their respective capabilities and focus areas.

As table 7 and figure 11 shows, the performance differences are quite noticeable. Checkov demonstrates a robust detection capability across a wide range of categories, particularly excelling in general security (11 688) and networking (8 453). This indicates its comprehensive coverage in these critical areas. Terrascan, on the other hand, displays notable strengths in identity and access management and networking, uncovering 1 872 and 1 961 security concerns respectively. Tfsec also contributes to the identification of security concerns, particularly in identity and access management (2 353) and networking (1 280).

In some categories, the difference in performance is highly noticeable. For instance, Checkov identifies 2 034 security concerns in encryption and data protection, whereas Terrascan and tfsec only find 65 and 21 respectively. This suggests that Checkov may offer a more extensive coverage or sensitivity to security concerns related to encryption and data protection practices. This disparity between how many security concerns they were able to find is present in other categories as well, with Checkov often finding way more security concerns than the other tools.

Despite Checkov and tfsec having checks for the secrets category, only two hits were observed in total, where the two tools found one each in different repositories. This indicates that these tests might be too lenient, or that only a few of the tested repositories contained any security concerns that would be detected for this category.

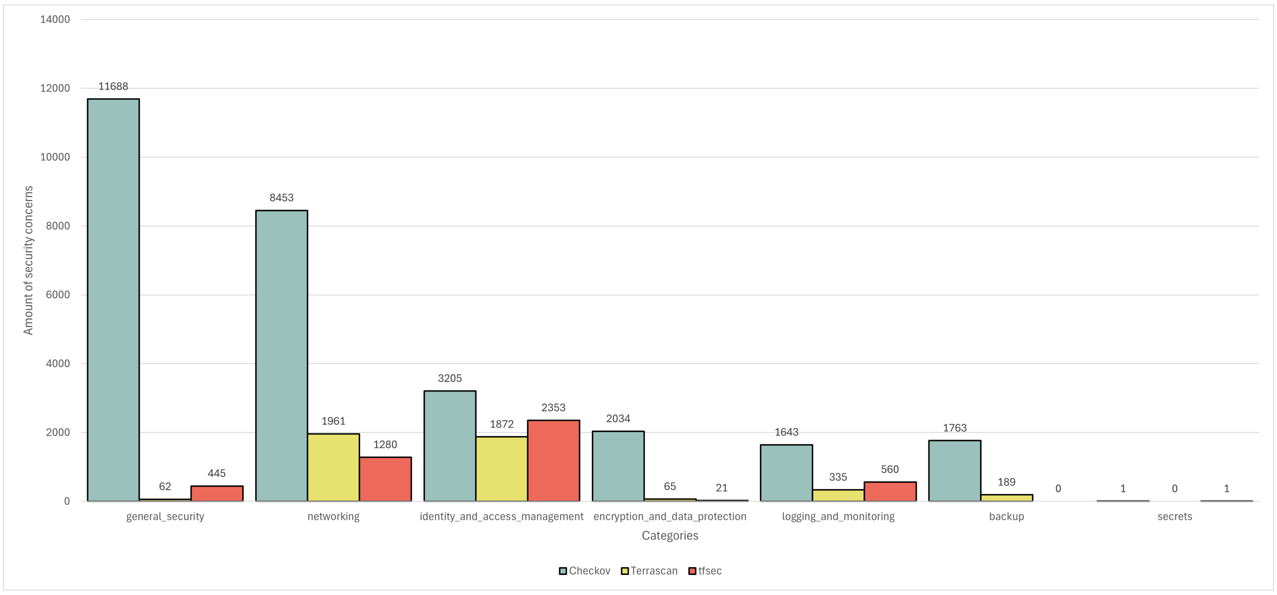


Figure 11: Total amount of security concerns found by each tool in each category

Tools	Checkov	Terrascan	tfsec	Total
General Security	11 688	62	445	12 195
Networking	8 453	1 961	1 280	11 694
Identity and Access Management	3 201	1 872	2 353	7 430
Encryption and Data Protection	2 034	65	21	2 120
Logging and Monitoring	1 643	335	560	2 538
Backup	1 763	189	0	1 952
Secrets	1	0	1	2
Total	28 787	4 484	4 660	37 931

Table 7: Total amount of security concerns the tools were able to identify by category

Moreover, analysing the total number of security concerns detected and the average per repository provides additional insights into the tools' efficacy. Checkov identifies a total of 28 787 security concerns, with an average of 45.05 security concerns per repository with at least one security concern. Terrascan uncovers 4 484 security concerns in total, averaging 7.72 security concerns per repository with at least one security concern, while tfsec detects 4 660 security concerns in total, averaging 13.95 security concerns per repository with at least one security concern. Table 8 shows the amount of unique repositories that gets at least one hit from tests within the given categories. More details can be found in table 6. These figures underline the variations in the observed amounts of security concerns found by each tool.

Tools	Checkov	Terrascan	tfsec
Identity and Access Management	371	516	281
Networking	525	369	240
General Security	610	28	43
Logging and Monitoring	215	162	117
Backup	380	108	0
Encryption and Data Protection	359	22	20
Secrets	1	0	1

Table 8: Amount of unique repositories with a security concern in a given category per tool

4.4.3 A Closer Look at Specific Checks

In this section, the focus is on the specific checks. We look at how many times a check has encountered a security concern in the entire dataset. In figure 12 the 10 checks with the highest hit count is shown.

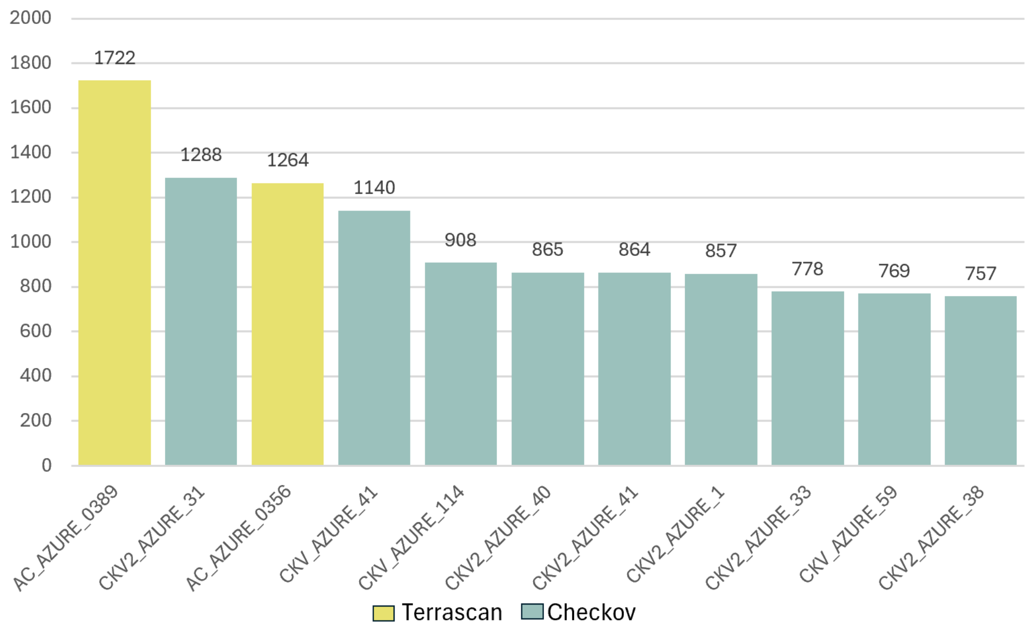


Figure 12: Graph showing the 10 checks with the highest hit count

Both Checkov and Terrascan checks are represented in this graph, with Terrascan having the check with the highest amount of hits. As seen earlier, Checkov has the most amount of checks with a high number of hits, having 8 of the top 10. While it seems like Terrascan has multiple checks with a high number of hits, that is not entirely the case. Even though the two yellow bars in the graph has 1 722 and 1 264 hits, the next Terrascan check only has 264. Tfsec has no checks with as many hits compared to the others, with the most prominent one having 531 hits, which places it as number 16.

5 Discussion

In this part of the thesis we will discuss the results we have gotten from the study, as well as our experience working with this material. We will answer RQs 1-3, and discuss how we have come to these answers. Then we will discuss assumptions and interpretations, threats to validity and ethical considerations related to our study.

5.1 RQ1 - Challenges of Collecting Data

RQ1: What are the challenges and difficulties of collecting and subsequently analysing data from publicly accessible code repositories?

RQ1 answer summary: The challenges and difficulties we experienced in regard to repository collection were in identifying the repositories we wanted to analyse, and adhering to the limitations of the GitHub and GitLab APIs. These were primarily limitations in filter capabilities and rate limiting of requests. The challenges we faced while testing the data revolved around using a self-made testing framework, whether the information extracted had substance, differences in the SAST tools' rulesets and indications that some tools were failing. We experienced some challenges with configuring Terrascan in the testing framework, but we were able to work around these challenges and make the tool run in our framework.

Repository Collection

The main challenge of collecting data from public repositories lies in identifying the repositories we want to analyse. The challenges stem from limitations placed upon the public-facing APIs from the Git providers. This includes not having the capabilities to filter and query data for an advanced filtering process. Limitations we identified included the following: not having the filtering capabilities needed, limitations on data extracted per query, rate limiting requests, giving incorrect data, server errors on normal requests, and poor documentation of these behaviours and limits.

The public-facing API from both Git providers we tested did not have the filtering capabilities to filter for the criteria we set in section 3.1.1. Even though both providers have built-in filtering options, these were not sufficient for this thesis' needs. This limitation led to the repository discovery process having to query the public facing APIs several times per repository, to get enough information to satisfy each discovery method's filters.

The consequence of the need to query the API endpoints many times was reaching the rate limit. Since our initial testing showed that there were 127 089 repositories on GitHub, we parallelised the filtering process code to run on different threads to make it faster. With our 8 threads we made about 14 400 request to the GitHub API per hour, but the limit was 5 000 req/hour [47]. This limitation was bypassed with the use of multiple GitHub account access tokens and making three threads share a common token. This limitation was solvable, but important to keep in mind when collecting data from open sources.

Both Git providers' APIs had limits on the amount of data that could be retrieved per query. They shared a common pagination interface where the maximum amount of items per query was 100. This meant that the initial discovery process for GitLab iterated through 1 270 pages of repositories, discovering 100 repositories per page. The GitHub search API had further limitations, such as the maximum amount of results for a given search query being 1 000, equating to 10 pages. This meant that it was not possible to retrieve data from page 11, even though there was more data. This led to us developing a divide and conquer method for retrieving initial repositories. Even though we found ways around the limitations, we still found it challenging to work around these APIs.

Other challenges when it came to API limitation were unexpected internal server errors. We experienced random internal server errors when using the GitLab search API with normal queries. These queries seemed to work after some retries even though nothing was changed on our part. We suspect that this unexpected behaviour was caused by a bug in the GitLab platform, something we could not do anything about. Our solution was to try again up to 10 times if a HTTP status code 500 (Internal Server Error) occurs and wait 10 seconds between each retry. Although the server said that it failed, this solution managed to retrieve all the repositories. For those who are going to collect data from these sources in the future it is worth noting that you can not always trust the http status codes.

Looking at the filtering results in table 1, it is apparent that some of the filters were not applied correctly or had no effect on GitHub discovery methods. The ”not a fork” criteria had no effect on the repositories. We consider this odd behaviour since this criteria manged to filter out 91% of repositories from the previous step (70 repositories) in the GitLab filtering script. This makes us suspect that we 1. implemented the check incorrectly, 2. GitHub's API specification is wrong or 3. GitHub's search API does not list forks. We believe the first option is unlikely since the code checks the `fork` attribute, which when true, signifies that the repository is a fork. The second option is not probable because we tested that the API and the fork attribute was true when the repository was a fork, and false otherwise. We consider the third option to be the most probable. The API verified that no forked repositories had made it past our filters. This is again an indication that there simply were no forked repositories to filter out rather than there being an error with the filter.

Collecting Information from Discovered Repositories

The main challenge of extracting data from repositories lied in both creating our testing framework and configuring this correctly, as well as whether or not the information extracted had substance. Even though the security tools used to analyse the repositories were set up according to their documentation, there could have been information we missed or nuances in their setup that could have affected the results. These facts are important to keep in mind when looking at the results.

As discussed in section 1.5 we have used the term "security concern" to reference the violations that are flagged by the SAST tools. We also discussed how these security violations are configurations that the SAST tools view to be issues or vulnerabilities, but are not necessarily issues that pose any real threat. Tools finding such security concerns could therefore get a higher count of discovered security concerns, even though what they found may not be real issues. When looking at the results the reader should have this in mind, as the security concern count may be skewed to make SAST tools with looser rulesets appear better than those with stricter rules. This is due to them flagging more security concerns that are merely bad or questionable practices, but not real issues. On the other hand, we have not manually gone in-depth into the security concern hits and can not say for certain that this is the case. We do not however, view this as a threat to the validity of our research, as the scope of this study is not to solely review critical violations.

One of the concerns we have about our framework is that the security concern findings for tfsec was not uploaded to the database correctly. Our findings show that tfsec did not find any non provider-specific issues. Tfsec has a total of 51 security concern detection rules with a "azure" label and only 1 detection rule with a "general" label. Our findings show that the detection rule with a "general" label was not found in any of the 334 repositories tfsec found security concerns in. Seeing as it is only 1 security rule in the secrets category that is not provider specific, it is possible that this rule simply did not get any hits, rather than there being an error in the run.

We ran the security tools according to their documentation and configured them to be resilient to faulty repositories, however, some of the tools were still failing on certain repositories. This was a big challenge we could not solve, as it was the tool itself that broke during the testing phase. When the tools broke, they did not give a result that we could upload to the database, resulting in no findings for that tool on that repository. Our study only compares the effect of each tool and whether it identified security concerns. If the tool breaks on a repository it does not find any concerns and will therefore not upload any findings to the database. We recommend that others who conduct similar studies keep track of when the security scanning tool breaks.

Challenges with the SAST Tools

For Checkov and tfsec testing we have used the official GitHub Actions. There is also an official GitHub Action for Terrascan, but this does not allow change of the output format. There is an option to generate a `.sarif` file with the test results, but this file does not contain the same information as when Terrascan is run from the command line with the JSON format selected. In this study we needed the test results delivered in JSON format, so we had to run the tool with a command instead of the GitHub Action.

Terrascan has 5 possible exit codes: 0, 1, 3, 4, 5. Besides 0 and 1, the exit codes are used to inform whether Terrascan identified violations but not errors, errors but not violations, or a combination of both [48]. This use of non-zero exit codes causes the GitHub Action to perceive the job as having errors, which again causes it to mark the job as failed and skip related jobs. This is a problem because the job has not actually failed, it has worked as it is supposed to, but GitHub Actions still views it as a failed run. We worked around this by using the `continue-on-error: true` flag, which makes the job continue even when it gets an error. This is not a good fix, as the job will also continue if there is a legitimate error. It is not necessarily a bad idea to have the error codes configured like this. In most cases users will not use Terrascan to scan hundreds of repositories like we do, but rather check only one repository. Having exit codes which throws errors when security concerns are discovered can therefore be a good thing, as it can prevent the faulty code to be pushed to the production environment. We would however, have liked to see there being a `soft-fail` option, as there is in Checkov [49] and tfsec [50], which would allow the tool to continue the job with errors.

5.2 RQ2 - Performance Impact

RQ2: What is the Performance Impact of Using Terraform Security Scanners on Build and Deployment Pipelines?

RQ2 answer summary: All the tools demand little time to complete their scans and can run concurrently with other processes in a workflow, minimising potential delay. The performance impact of integrating SAST tools into a CI/CD pipeline is therefore low.

Our analysis indicates that the slowest tool requires an average of 13.94 seconds to check a repository, while the fastest tool only needs 4.16 seconds per repository. Moreover, these tests can be run independent of existing pipelines and can therefore run concurrently alongside existing processes, minimising potential delay. This, combined with the low amount of time even the slowest tool needs, makes the performance impact of integrating SAST tools into an existing CI/CD pipeline low. It is worth noting that the speed results we obtained from these scans may have been affected by the number of repositories being tested simultaneously.

Furthermore, the utilisation of SAST tools can significantly benefit developers striving to establish secure IaC environments. By integrating these tools early in the development cycle, developers can proactively identify and rectify security vulnerabilities. This could help ensure a robust foundation for their infrastructure deployments. This proactive approach not only enhances security practices but also streamlines the development process by addressing potential security concerns before they escalate.

5.3 RQ3 - Main Differences Observed Between the Different SAST Tools

RQ3: What are the main differences between the different SAST tools in terms of their rulesets and performance?

RQ3 answer summary: There is a difference in the amount of rules the tools have in their rulesets, security concerns the tools were able to find and repositories the tools were able to find security concerns in. Checkov scores the highest in all of these measures, Terrascan places second in size of ruleset and coverage, while tfsec places second in total security concern count. There are also differences in coverage when categories are taken into consideration, but here as well, Checkov places highest in all categories except for the secret category where it has the same amount as tfsec.

As shown in sections 4.4.1 and 4.4.2, there is a distinct difference in the amounts of security concerns the tools were able to find in the repositories. Part of this comes down to how many of the repositories they were able to complete their scans for. As mentioned in section 5.1, the investigation into the main differences between the three SAST tools begins with an overview of the results, highlighting differences in detection capabilities across repositories. This stark contrast in coverage sets the stage for a deeper exploration into the underlying factors driving these discrepancies.

Checkov

Checkov has a ruleset consisting of 278 total rules for Azure. This is by far the largest ruleset out of the three tools. If we look at table 4, we can see that it has quite good coverage in all categories except for secrets, where it only has two rules.

Comparing the ruleset with the results, a similar pattern emerges. As shown in section 4.4.1, Checkov was able to identify a wide range of concerns, and consistently found more security concerns than the other tools across the categories. There is also a clear correlation between the number of checks in a given category and the amount of security concerns it is able to find in the given category. The main focus areas of Checkov seems to be in the general security and networking categories, where it found the most amount of security concerns along with having the highest amount of rules.

In our testing, Checkov also seemed to be the most stable of the three tools. It had the lowest number of problems encountered, where it timed out or threw errors. Checkov also has some convenient features such as AI integration to automatically give suggestions for code replacement whenever it finds an error [51].

Terrascan

Terrascan's ruleset was the second largest with a rule count of 178 for Azure. From table 4, it is clear that in contrast to Checkov, these were not evenly distributed, with 138 networking rules, leaving only 40 spread out for the other categories. However, the number of tests does not fully explain this large disparity, as the vast majority of these rules are more strict than what the other tools have for their rules in the same category. As for the other categories, the remaining rules are relatively evenly split. However, Terrascan completely lack specific rules related to finding security concerns in the secrets category.

We found that only 45 rules out of the total 178 rules were triggered once or more using Terrascan's ruleset. This could be due to the fact that Terrascan has very specific checks. For example a lot of the network checks are configured to only check one port each. With network being the category with the most checks, it is possible that none of the repositories had most of the port misconfigurations. This can explain why some of the 133 of the checks correctly got zero hits. However, as we have not done an in-depth analysis of the checks that got no hits, we are unable to confirm this.

When comparing the ruleset with the results seen in section 4.4.1, there is some level of correlation between the number of rules and the number of security concerns found. However, it is not as clear as with Checkov. Networking ended up having the highest number of findings, while identity and access management ended up following closely behind despite having way fewer rules. It also seemed like the tool underperformed in several of the other categories, coming in behind the other tools.

Tfsec

Tfsec had the smallest ruleset of the three tools by a significant margin, with only 52 tests for Azure, including their non provider-specific secrets rule. This leaves only 51 tests specifically designated for Azure. The rules are relatively evenly spread, with identity and access management and networking having the highest rule counts. However there are some clear lacks, especially for backup where there was no checks, and for encryption and data protection as well as secrets where it only had two tests for each category.

Comparing the ruleset with the results shown in section 4.4.1, we can see that there is a clear correlation, with no significant outliers. We can also see that despite its lower rule count and that it was unable to find security concerns in as many repositories, the total number of concerns it was able to find was quite comparable to Terrascan. Tfsec and Terrascan seemed to be relatively evenly matched in most categories. Counting only repositories with at least one security concern hit, tfsec ended up finding more security concerns per repository on average than Terrascan. Tfsec found on average 14.0 whereas Terrascan only found 7.7. This could indicate that each rule is more thorough than the rules from the other tools.

Tfsec found security concerns in the fewest repositories, with only 334, compared to Checkov's 639 and Terrascan's 589. At first glance, this seemed to be caused by the tool not functioning properly. However, tfsec has a feature that when enabled, will ignore invalid Terraform files in a repository and continue testing the valid files. We did not disable this feature since we wanted to test functional IaC projects. The outcome was that tfsec was not able to find any configuration mistakes in repositories with invalid Terraform configuration, while the other tools which are more resilient by default, reported errors even in invalid Terraform files. We consider it likely that our results might have been swayed in favour of the other tools that complete their checks, even when faced with repositories containing invalid Terraform files.

It should be mentioned that tfsec is in some ways outdated as Aqua security, the developers of tfsec, has begun to merge the tool into their more comprehensive Trivy suite. Aqua's attention is no longer on tfsec, and only Trivy continues to get new official updates. The merging of these tools seemed unfinished, therefore we chose to use the tried and tested tfsec in our study. Trivy also seems to not be in full release state, as its current version number is 0.51.1 [52] (as of the 19th of May 2024). We recommend that others who intend to conduct similar studies consider using Trivy as a SAST tool for Terraform.

5.4 Assumptions and Interpretations

Seeing as our study is an empirical data-centric study, we have strived to make as few assumptions as possible regarding our results. There have however been some assumptions in regards to what the results suggest in terms of SAST tool quality. We have equated violation count and coverage with quality, thereby assuming that each test result is valid and not a duplicate.

We have also made another significant choice in terms of interpretations, which is that we measure all security concerns equally and do not consider severity level.

This means that we do not have insight into how the coverage is distributed between the tools if we were to use severity level as a metric. This could uncover other valuable insights in regards to which tools is better at finding security concerns in each severity level.

5.5 Threats to Validity

Similar Repositories

With our filtration we did not collect forks in our repository collection. However, upon closer manual inspection of the security concerns, we found repositories with similar results. When further investigating these repositories, we found that one was a clone of the other, and that there were likely more clones in our dataset. In our experience clones are very difficult to automatically filter out in the dataset. This means that a few of the repositories in practice might get their security concern statistics added twice or more, as their clones might contain the same security concerns.

However, as the similar repository still fits all of the criteria, it could be important to allow it into the analysis. A cloned repository that is still active, should strive to limit its security concerns even though they came from the cloned repository. The problem occurs when a repository is cloned within the last year, and then abandoned. The commits and names of the contributors could be cloned from another repository, without anything new having been added. In that case it would just end up being duplicate data in our analysis, which could skew the results in favour of one of the tools.

Risks of False Positives and Negatives

In our study we have checked 800 public code repositories and got 37 931 security concern hits. This is a lot of repositories and security concerns. We have not reviewed the results we got besides the statistical analysis we did. This means that we have not manually or otherwise gone into the repositories and the individual test results to confirm that each hit is valid, and that no security concern have been overlooked. Therefore there is a possibility that there are false positives or negatives in our test data that we have not accounted for. This could skew the results we have gotten, and because we do not know if this is the case, we also do not know in which direction the data could potentially have been skewed. On the other hand we can also not confirm that this has happened at all, and it is therefore possible that there are no false positives or negatives.

5.6 Ethical Considerations

In this project, the main ethical consideration is the privacy of the repositories that were collected data from. We opted to mention neither the names of repositories nor repository owners in this study. The reason for this anonymisation, is to avoid leaking information about which repositories contain security concerns. Even though the repositories are public, we strive to avoid disclosing any more information than what was necessary to conduct our study.

6 Conclusion

6.1 Limitations and Future Work

As our research focused solely on Checkov, Terrascan and tfsec, there is potential for future work to analyse and compare other tools in addition to these. Tfsec has migrated to Trivy, meaning in a future analysis, the use of Trivy could also replace tfsec completely. Future tools and updates could become better than currently available ones, and could be worth studying.

While the tools we tested identifies security concerns, future work could focus on identifying more strictly defined security issues. Our conclusion is based on the tools' own reported security concerns, but future work could do the comparison of tools on a dataset with known security issues that has known consequences. This research would contribute to identifying what SAST tools are best at finding more critical issues, eliminating the impact variation in rulesets has on the results. A way that this can be done is by only looking at security concerns above a certain severity level.

We identified that failing security tools was a challenge when it came to collecting data from open source repositories. We recommend that future work consider adding logging to the data collection process to keep track of how the tools behave on the given repositories, and whether it breaks or not. This will help the researchers identify and fix issues with their data collection process and allow them to be more confident in their results.

6.2 RQ4 - Recommendations

RQ4: What single scanner or combination of scanners provide the best security guidelines for IaC repositories?

RQ4 answer summary: We conclude that a combination of Checkov and Terrascan provides the most comprehensive guideline for a safe IaC repository.

This section will elaborate on what we found to be the strongest tool, and what combination of tools will best secure a repository consisting of Terraform code using the AzureRM provider.

Section 4.4.1 shows security concern detection coverage and overlap. The results from this section clearly shows which tool has the overall highest coverage, which is Checkov. Another factor to consider is that Checkov appeared to be the most stable of the tools. We encountered errors or challenges with both Terrascan and tfsec, and these tools also appeared to occasionally break, causing them to not complete the scans. We did not encounter such issues with Checkov, and it was generally easy and stable to use. This is why we deem it as the most useful tool.

However, we experienced that as Terrascan was able to find security concerns in some of the repositories where Checkov found no concerns, we argue that a combination of the two would be the best solution when both coverage and performance is taken into consideration. This is because this combination provides almost perfect coverage (99.7%) which justifies the added performance cost. This is further supported by the fact that Terrascan complements Checkov by having precise checks with a narrow focus in the network category, whereas Checkov has broader checks in this category.

Tfsec completes its tests the fastest. However, as tfsec intentionally avoids checking repositories with invalid Terraform configurations, it is difficult to get a complete overview of how well it really works. In our study, tfsec only found security concerns in 2 of the 800 repositories where neither Checkov nor Terrascan found any concerns. This only adds up to a potential lost coverage percentage of 0.3% by not including tfsec.

If the only goal of the provisioner is to test the repository as thoroughly as possible, using all three tools would be the optimal solution. However, reading through the outputs from all the tools can lead to alert fatigue. If they value high performance and less alerts, our results suggest that tfsec finds security concerns in too few repositories where Checkov and Terrascan are unable to identify any, to justify using it. From our results, we conclude that the best single SAST tool to include is Checkov. However, if there is a need for better coverage, a combination of Checkov and Terrascan is recommended.

7 Bibliography

References

[1] GitHub. *Fork a repository*. (N/A). URL: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo. (Visited: 20.05.2024).

[2] Bridgecrew. *Checkov*. 2024. URL: https://github.com/bridgecrewio/checkov. (Visited: 06.05.2024).

[3] Tenable. *Terrascan*. 2024. URL: https://github.com/tenable/terrascan. (Visited: 06.05.2024).

[4] Aqua Security. *tfsec*. 2024. URL: https://github.com/aquasecurity/tfsec. (Visited: 06.05.2024).

[5] Lionel Sujay Vailshery. "Usage of cloud configuration tools worldwide in 2023, current and planned". In: (2023). URL: https://www.statista.com/statistics/511293/worldwide-survey-cloud-devops-tools/. (Visited: 28.04.2024).

[6] 6sense. *Terraform*. 2024. URL: https://6sense.com/tech/configuration-management/terraform-market-share.

[7] Felix Richter. "Cloud Infrastructure Market - Amazon Maintains Cloud Lead as Microsoft Edges Closer". In: (2024). URL: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/. (Visited: 28.04.2024).

[8] Akond Rahman, Chris Parnin, and Laurie Williams. "The Seven Sins: Security Smells in Infrastructure as Code Scripts". In: (2019), p. 172. DOI: 10.1109/ICSE.2019.00033.

[9] Kief Morris. *Infrastructure as Code Dynamic Systems for the Cloud Age*. O'Reilly Media, Inc., 2020. ISBN: 9781098114671.

[10] hashicorp. *Terraform Language Documentation*. (N/A). URL: https://developer.hashicorp.com/terraform/language. (Visited: 06.05.2024).

[11] Hashicorp. *Provider Requirements*. 2024. URL: https://developer.hashicorp.com/terraform/language/providers/requirements. (Visited: 06.05.2024).

[12] Ryan Fee. *The Top 20 Terraform Providers*. 2021. URL: https://www.scalr.com/blog/top-20-terraform-providers. (Visited: 20.05.2024).

[13] Azure. *What is Azure?* (N/A). URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/. (Visited: 06.05.2024).

[14] Redhat. *What is CI/CD?* Dec. 2023. URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd. (Visited: 06.05.2024).

[15] GitLab. *What is CI/CD?* (N/A). URL: https://about.gitlab.com/topics/ci-cd/. (Visited: 06.05.2024).

[16] Github. *GitHub Actions documentation*. (N/A). URL: https://docs.github.com/en/actions. (Visited: 06.05.2024).

[17] Github. *Using a matrix for your jobs*. (N/A). URL: https://docs.github.com/en/actions/using-jobs/using-a-matrix-for-your-jobs. (Visited: 06.05.2024).

[18] Ryan Dewhurst. *Static Code Analysis*. (N/A). URL: https://owasp.org/www-community/controls/Static_Code_Analysis. (Visited: 06.05.2024).

[19] Samhita Tankala and Katie Sherwin. *Card Sorting: Uncover Users' Mental Models for Better Information Architecture*. Feb. 2023. URL: https://www.nngroup.com/articles/card-sorting-definition/. (Visited: 06.05.2024).

[20] Checkov. *What is Checkov?* (N/A). URL: https://www.checkov.io/1.Welcome/What%20is%20Checkov.html. (Visited: 06.05.2024).

[21] Terrascan. *Contributing*. May 2022. URL: https://github.com/tenable/terrascan/blob/master/CONTRIBUTING.md. (Visited: 06.05.2024).

[22] Aqua Security. *Architecture*. Nov. 2023. URL: https://github.com/aquasecurity/defsec/blob/master/ARCHITECTURE.md. (Visited: 06.05.2024).

[23] Aqua Security. *Contributing*. Nov. 2023. URL: https://github.com/aquasecurity/defsec/blob/master/CONTRIBUTING.md. (Visited: 06.05.2024).

[24] Aqua Security. *Custom Checks*. 2024. URL: https://aquasecurity.github.io/tfsec/v1.28.5/guides/configuration/custom-checks/. (Visited: 06.05.2024).

[25] Nuthan Munaiah et al. "Curating GitHub for engineered software projects." In: *Empir Software Eng 22, 3219–3253* (2017). URL: https://doi.org/10.1007/s10664-017-9512-6.

[26] Akond Rahman, Chris Parnin, and Laurie Williams. "The Seven Sins: Security Smells in Infrastructure as Code Scripts". In: (2019), pp. 164–175. DOI: 10.1109/ICSE.2019.00033.

[27] Alexandre Verdet et al. *Exploring Security Practices in Infrastructure as Code: An Empirical Study*. 2023. arXiv: 2308.03952 [cs.CR].

[28] Andrei-Cristian Iosif et al. "A Large-Scale Study on the Security Vulnerabilities of Cloud Deployments". In: *Ubiquitous Security*. Ed. by Guojun Wang et al. Singapore: Springer Singapore, 2022, pp. 171–188. ISBN: 978-981-19-0468-4.

[29] Sam Gabrail. *Which IaC Scanning Tool is the Best?: Comparing Checkov vs tfsec vs Terrascan*. (N/A). URL: https://www.env0.com/blog/best-iac-scan-tool. (Visited: 06.05.2024).

[30] Abhishek Dubey. *IaC Security Analysis: Checkov vs. tfsec vs. Terrascan – A Comparative Evaluation*. Apr. 2024. URL: https://opstree.com/blog/2024/04/30/iac-security-analysis-checkov-vs-tfsec-vs-terrascan-a-comparative-evaluation/. (Visited: 06.05.2024).

[31] Mariusz Michalowski. *Infrastructure as Code (IaC) Scanning Tools*. Jan. 2024. URL: https://hackernoon.com/infrastructure-as-code-iac-scanning-tools. (Visited: 06.05.2024).

[32] Nuthan Munaiah et al. "Curating GitHub for engineered software projects." In: *Empir Software Eng 22, 3219–3253* (2017), p. 3222. URL: https://doi.org/10.1007/s10664-017-9512-6.

[33] Akond Rahman, Chris Parnin, and Laurie Williams. "The Seven Sins: Security Smells in Infrastructure as Code Scripts". In: (2019), p. 170. DOI: 10.1109/ICSE.2019.00033.

[34] Amritanshu Agrawal et al. "We don't need another hero?: the impact of "heroes" on software development". In: *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. ICSE '18. ACM, May 2018, p. 247. DOI: 10.1145/3183519.3183549. URL: http://dx.doi.org/10.1145/3183519.3183549.

[35] GitHub. *Using a matrix for your jobs*. (N/A). URL: https://docs.github.com/en/actions/using-jobs/using-a-matrix-for-your-jobs. (Visited: 06.05.2024).

[36] GitHub. *Using GitHub-hosted runners*. (N/A). URL: https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners. (Visited: 06.05.2024).

[37] Nektos. *Act*. 2024. URL: https://github.com/nektos/act. (Visited: 06.05.2024).

[38] GitHub. *Events that trigger workflows*. (N/A). URL: https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows#workflow_dispatch. (Visited: 06.05.2024).

[39] GitHub. *Searching code (legacy)*. (N/A). URL: https://docs.github.com/en/search-github/searching-on-github/searching-code#search-by-language. (Visited: 20.05.2024).

[40] GitHub. *REST API endpoints for search*. Nov. 2022. URL: https://docs.github.com/en/rest/search/search?apiVersion=2022-11-28. (Visited: 06.05.2024).

[41] Douglas R Smith. "The design of divide and conquer algorithms". In: *Science of Computer Programming* 5 (1985), pp. 37–58.

[42] Bun. *Bun is a fast JavaScript all-in-one toolkit*. (2024). URL: https://bun.sh/. (Visited: 06.05.2024).

[43] Md Feroj Ahmod. “JAVASCRIPT RUNTIME PERFORMANCE ANALYSIS: NODE AND BUN”. In: (2023). URL: https://urn.fi/URN:NBN:fi:tuni-202306136706.

[44] Bridgecrew. *checkov-action*. 2022. URL: https://github.com/bridgecrewio/checkov-action. (Visited: 27.04.2024).

[45] Aqua Security. *tfsec-action*. 2023. URL: https://github.com/aquasecurity/tfsec-action. (Visited: 06.05.2024).

[46] Tenable. *terrascan-action*. 2021. URL: https://github.com/tenable/terrascan-action. (Visited: 06.05.2024).

[47] GitHub. *Rate limits for the REST API*. Nov. 2022. URL: https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api?apiVersion=2022-11-28. (Visited: 08.05.2024).

[48] tenable. *Documentation*. 2022. URL: https://runterrascan.io/docs/_print/. (Visited: 06.05.2024).

[49] PRISMA CLOUD. *Hard and soft fail*. (N/A). URL: https://www.checkov.io/2.Basics/Hard%20and%20soft%20fail.html. (Visited: 20.05.2024).

[50] Aqua security. *Parameters*. 2021. URL: https://aquasecurity.github.io/tfsec/v0.61.1/getting-started/usage/. (Visited: 20.05.2024).

[51] Checkov. *OpenAI*. (N/A). URL: https://www.checkov.io/4.Integrations/OpenAI.html. (Visited: 06.05.2024).

[52] Aqua Security. *aquasecurity/trivy/releases*. 2024. URL: https://github.com/aquasecurity/trivy/releases. (Visited: 08.05.2024).

List of Figures

List of Tables

Appendices

A Repository Discovery and Data Collection Framework

Our code repository is attached in the submitted Zip-file.

B Project Handbook

Our project handbook is attached in the submitted Zip-file.

C List of Checks in Checkov

Old category	New category	Rule name	Rule ID	Rule description
General security	General security	N/A	CKV_AZURE_1	Ensure Azure Instance does not use basic authentication(Use SSH Key Instead)
Encryption	Encryption and data protection	N/A	CKV_AZURE_2	Ensure Azure managed disk have encryption enabled
Encryption	Encryption and data protection	N/A	CKV_AZURE_3	Ensure that 'supportsHttpsTrafficOnly' is set to 'true'
Kubernetes	General security	N/A	CKV_AZURE_4	Ensure AKS logging to Azure Monitoring is Configured
Kubernetes	General security	N/A	CKV_AZURE_5	Ensure RBAC is enabled on AKS clusters
Kubernetes	General security	N/A	CKV_AZURE_6	Ensure AKS has an API Server Authorized IP Ranges enabled
Kubernetes	General security	N/A	CKV_AZURE_7	Ensure AKS cluster has Network Policy configured
Kubernetes	General security	N/A	CKV_AZURE_8	Ensure Kubernetes Dashboard is disabled
Networking	Networking	N/A	CKV_AZURE_9	Ensure that RDP access is restricted from the internet
Networking	Networking	N/A	CKV_AZURE_10	Ensure that SSH access is restricted from the internet
Logging	Logging and monitoring	N/A	CKV_AZURE_11	Ensure no SQL Databases allow ingress from 0.0.0.0/0 (ANY IP)
Logging	Logging and monitoring	N/A	CKV_AZURE_12	Ensure that Network Security Group Flow Log retention period is 'greater than 90 days'
Identity and access management	Identity and access management	N/A	CKV_AZURE_13	Ensure App Service Authentication is set on Azure App Service
Networking	Networking	N/A	CKV_AZURE_14	Ensure web app redirects all HTTP traffic to HTTPS in Azure App Service
Networking	Networking	N/A	CKV_AZURE_15	Ensure web app is using the latest version of TLS encryption
Identity and access management	Identity and access management	N/A	CKV_AZURE_16	Ensure that Register with Azure Active Directory is enabled on App Service
Networking	Networking	N/A	CKV_AZURE_17	Ensure the web app has 'Client Certificates (Incoming client certificates)' set
Networking	Networking	N/A	CKV_AZURE_18	Ensure that 'HTTP Version' is the latest if used to run the web app
Networking	Networking	N/A	CKV_AZURE_19	Ensure that standard pricing tier is selected
Networking	Networking	N/A	CKV_AZURE_20	Ensure that security contact 'Phone number' is set
Networking	Networking	N/A	CKV_AZURE_21	Ensure that 'Send email notification for high severity alerts' is set to 'On'
Networking	Networking	N/A	CKV_AZURE_22	Ensure that 'Send email notification for high severity alerts' is set to 'On'
Logging	Logging and monitoring	N/A	CKV_AZURE_23	Ensure that 'Auditing' is set to 'Enabled' for SQL servers
Logging	Logging and monitoring	N/A	CKV_AZURE_24	Ensure that 'Auditing' Retention is 'greater than 90 days' for SQL servers
General security	General security	N/A	CKV_AZURE_25	Ensure that 'Threat Detection types' is set to 'All'
General security	General security	N/A	CKV_AZURE_26	Ensure that 'Send Alerts To' is enabled for MSSQL servers
General security	General security	N/A	CKV_AZURE_27	Ensure that 'Email service and co-administrators' is 'Enabled' for MSSQL servers
Networking	Networking	N/A	CKV_AZURE_28	Ensure 'Enforce SSL connection' is set to 'ENABLED' for MySQL Database Server
Networking	Networking	N/A	CKV_AZURE_29	Ensure 'Enforce SSL connection' is set to 'ENABLED' for PostgreSQL Database Server
Networking	Networking	N/A	CKV_AZURE_30	Ensure server parameter 'log_checkpoints' is set to 'ON' for PostgreSQL Database Server
Networking	Networking	N/A	CKV_AZURE_31	Ensure configuration 'log_connections' is set to 'ON' for PostgreSQL Database Server
Networking	Networking	N/A	CKV_AZURE_32	Ensure server parameter 'connection_throttling' is set to 'ON' for PostgreSQL Database Server
Networking	Networking	N/A	CKV_AZURE_33	Ensure Storage logging is enabled for Queue service for read, write and delete requests
Networking	Networking	N/A	CKV_AZURE_34	Ensure that 'Public access level' is set to Private for blob containers
Networking	Networking	N/A	CKV_AZURE_35	Ensure default network access rule for Storage Accounts is set to deny
Networking	Networking	N/A	CKV_AZURE_36	Ensure 'Trusted Microsoft Services' is enabled for Storage Account access
Logging	Logging and monitoring	N/A	CKV_AZURE_37	Ensure that Activity Log Retention is set 365 days or greater
Logging	Logging and monitoring	N/A	CKV_AZURE_38	Ensure audit profile captures all the activities
Identity and access management	Identity and access management	N/A	CKV_AZURE_39	Ensure that no custom subscription owner roles are created
General security	General security	N/A	CKV_AZURE_40	Ensure that the expiration date is set on all keys
General security	General security	N/A	CKV_AZURE_41	Ensure that the expiration date is set on all secrets
Backup and recovery convention	Backup	N/A	CKV_AZURE_42	Ensure the key vault is recoverable
Misc	General security	N/A	CKV_AZURE_43	Ensure Storage Accounts adhere to the naming rules
Networking	Networking	N/A	CKV_AZURE_44	Ensure Storage Account is using the latest version of TLS encryption
Secrets	Secrets	N/A	CKV_AZURE_45	Ensure that no sensitive credentials are exposed in VM custom_data
Networking	Networking	N/A	CKV_AZURE_47	Ensure 'Enforce SSL connection' is set to 'ENABLED' for MariaDB servers
Networking	Networking	N/A	CKV_AZURE_48	Ensure 'public network access enabled' is set to 'False' for MariaDB server
General security	General security	N/A	CKV_AZURE_49	Ensure Azure linux scale set does not use basic authentication(Use SSH Key Instead)
General security	General security	N/A	CKV_AZURE_50	Ensure Virtual Machine Extensions are not Installed
Networking	Networking	N/A	CKV_AZURE_52	Ensure MSSQL is using the latest version of TLS encryption
Networking	Networking	N/A	CKV_AZURE_53	Ensure 'public network access enabled' is set to 'False' for mySQL servers
Networking	Networking	N/A	CKV_AZURE_54	Ensure MySQL is using the latest version of TLS encryption
General security	General security	N/A	CKV_AZURE_55	Ensure that Azure Defender is set to On for Servers
General security	General security	N/A	CKV_AZURE_56	Ensure that function apps enables Authentication
General security	General security	N/A	CKV_AZURE_57	Ensure that CORS disallows every resource to access app services
Networking	Networking	N/A	CKV_AZURE_58	Ensure that Azure Synapse workspaces enables managed virtual networks
Networking	Networking	N/A	CKV_AZURE_59	Ensure that Storage accounts disallow public access
General security	General security	N/A	CKV_AZURE_60	Ensure that Azure Defender is set to On for App Service
General security	General security	N/A	CKV_AZURE_61	Ensure function apps are not accessible from all regions
Logging	Logging and monitoring	N/A	CKV_AZURE_62	Ensure that App service enables HTTP logging
Networking	Networking	N/A	CKV_AZURE_63	Ensure that Azure File Sync disables public network access
Logging	Logging and monitoring	N/A	CKV_AZURE_64	Ensure that App service enables detailed error messages
Logging	Logging and monitoring	N/A	CKV_AZURE_65	Ensure that App service enables failed request tracing
General security	General security	N/A	CKV_AZURE_66	Ensure that 'HTTP Version' is the latest, if used to run the Function app
Networking	Networking	N/A	CKV_AZURE_67	Ensure that PostgreSQL server disables public network access
General security	General security	N/A	CKV_AZURE_68	Ensure that Azure Defender is set to On for Azure SQL database servers
Networking	Networking	N/A	CKV_AZURE_69	Ensure that Function apps is only accessible over HTTPS
General security	General security	N/A	CKV_AZURE_70	Ensure that Managed identity provider is enabled for app services
General security	General security	N/A	CKV_AZURE_71	Ensure that remote debugging is not enabled for app services
Encryption	Encryption and data protection	N/A	CKV_AZURE_72	Ensure that Automation account variables are encrypted
Encryption	Encryption and data protection	N/A	CKV_AZURE_73	Ensure that Azure Data Explorer (Kusto) uses disk encryption

Encryption	Encryption and data protection	N/A	CKV_AZURE_74	Ensure that Azure Data Explorer uses double encryption
Encryption	Encryption and data protection	N/A	CKV_AZURE_75	Ensure that Azure Batch account uses key vault to encrypt data
Networking	Networking	N/A	CKV_AZURE_76	Ensure that UDP Services are restricted from the Internet
Application security	General security	N/A	CKV_AZURE_77	Ensure FTP deployments are disabled
General security	General security	N/A	CKV_AZURE_78	Ensure that Azure Defender is set to On for SQL servers on machines
General security	General security	N/A	CKV_AZURE_79	Ensure that ‘Net Framework’ version is the latest, if used as a part of the web app
General security	General security	N/A	CKV_AZURE_80	Ensure that ‘PHP version’ is the latest, if used to run the web app
General security	General security	N/A	CKV_AZURE_81	Ensure that ‘Python version’ is the latest, if used to run the web app
General security	General security	N/A	CKV_AZURE_82	Ensure that ‘Java version’ is the latest, if used to run the web app
General security	General security	N/A	CKV_AZURE_83	Ensure that Azure Defender is set to On for Storage
General security	General security	N/A	CKV_AZURE_84	Ensure that Azure Defender is set to On for Kubernetes
General security	General security	N/A	CKV_AZURE_85	Ensure that Azure Defender is set to On for Container Registries
General security	General security	N/A	CKV_AZURE_86	Ensure that Azure Defender is set to On for Key Vault
General security	General security	N/A	CKV_AZURE_87	Ensure that app services use Azure Files
Networking	Networking	N/A	CKV_AZURE_88	Ensure that Azure Cache for Redis disables public network access
Networking	Networking	N/A	CKV_AZURE_89	Ensure that only SSL are enabled for Cache for Redis
General security	General security	N/A	CKV_AZURE_90	Ensure that Virtual Machines use managed disks
Encryption	Encryption and data protection	N/A	CKV_AZURE_91	Ensure that managed disks use a specific set of disk encryption sets for the customer-managed key encryption
Backup and recovery convention	Backup	N/A	CKV_AZURE_92	Ensure that My SQL server enables geo-redundant backups
General security	General security	N/A	CKV_AZURE_93	Ensure that automatic OS image patching is enabled for Virtual Machine Scale Sets
Encryption	Encryption and data protection	N/A	CKV_AZURE_94	Ensure that MySQL server enables infrastructure encryption
Encryption	Encryption and data protection	N/A	CKV_AZURE_95	Ensure that Virtual machine scale sets have encryption at host enabled
Networking	Networking	N/A	CKV_AZURE_96	Ensure that Azure Container group is deployed into virtual network
Networking	Networking	N/A	CKV_AZURE_97	Ensure Cosmos DB accounts have restricted access
Networking	Networking	N/A	CKV_AZURE_98	Ensure that Cosmos DB accounts have customer-managed keys to encrypt data at rest
Networking	Networking	N/A	CKV_AZURE_99	Ensure that Azure Cosmos DB disables public network access
Backup and recovery convention	Backup	N/A	CKV_AZURE_100	Ensure that PostgreSQL server enables geo-redundant backups
General security	General security	N/A	CKV_AZURE_101	Ensure that Azure Data Factory uses Git repository for source control
Networking	Networking	N/A	CKV_AZURE_102	Ensure that Azure Data factory public network access is disabled
Encryption	Encryption and data protection	N/A	CKV_AZURE_103	Ensure that Data Lake Store accounts enables encryption
Networking	Networking	N/A	CKV_AZURE_104	Ensure that Azure Event Grid Domain public network access is disabled
Networking	Networking	N/A	CKV_AZURE_105	Ensure that API management services use virtual networks
Networking	Networking	N/A	CKV_AZURE_106	Ensure that Azure IoT Hub disables public network access
Networking	Networking	N/A	CKV_AZURE_107	Ensure that key vault allows firewall rules settings
Networking	Networking	N/A	CKV_AZURE_108	Ensure that key vault enables purge protection
Logging	Logging and monitoring	N/A	CKV_AZURE_109	Ensure that key vault enables soft delete
Backup and recovery convention	Backup	N/A	CKV_AZURE_110	Ensure that key vault key is backed by HSM
Networking	Networking	N/A	CKV_AZURE_111	Ensure that SQL server disables public network access
General security	General security	N/A	CKV_AZURE_112	Ensure that key vault secrets have “content_type” set
Networking	Networking	N/A	CKV_AZURE_113	Ensure that AKS enables private clusters
Networking	Networking	N/A	CKV_AZURE_114	Ensure that AKS uses Azure Policies Add-on
Networking	Networking	N/A	CKV_AZURE_115	Ensure that AKS uses disk encryption set
Networking	Networking	N/A	CKV_AZURE_116	Ensure that Network Interfaces disable IP forwarding
Networking	Networking	N/A	CKV_AZURE_117	Ensure that Network Interfaces don’t use public IPs
Application security	General security	N/A	CKV_AZURE_118	Ensure that Application Gateway enables WAF
Networking	Networking	N/A	CKV_AZURE_119	Ensure that Azure Front Door enables WAF
Networking	Networking	N/A	CKV_AZURE_120	Ensure that Application Gateway uses WAF in “Detection” or “Prevention” modes
Networking	Networking	N/A	CKV_AZURE_121	Ensure that Azure Front Door uses WAF in “Detection” or “Prevention” modes
Networking	Networking	N/A	CKV_AZURE_122	Ensure that Azure Cognitive Search disables public network access
Encryption	Encryption and data protection	N/A	CKV_AZURE_123	Ensures that Service Fabric use three levels of protection available
General security	General security	N/A	CKV_AZURE_124	Ensures that Active Directory is used for authentication for Service Fabric
General security	General security	N/A	CKV_AZURE_125	Ensure that My SQL server enables Threat detection policy
General security	General security	N/A	CKV_AZURE_126	Ensure that PostgreSQL server enables Threat detection policy
Backup and recovery convention	Backup	N/A	CKV_AZURE_127	Ensure that MariaDB server enables geo-redundant backup
Encryption	Encryption and data protection	N/A	CKV_AZURE_128	Ensure that PostgreSQL server enables infrastructure encryption
General security	General security	N/A	CKV_AZURE_129	Ensure that ‘Security contact emails’ is set
Secrets	Secrets	N/A	CKV_AZURE_130	Ensure that ‘Security contact emails’ is set
General security	General security	N/A	CKV_AZURE_131	Ensure cosmosdb does not allow privileged escalation by restricting management plane changes
Application security	General security	N/A	CKV_AZURE_132	Ensure Front Door WAF prevents message lookup in Log4j2. See CVE-2021-44228 aka log4jshell
Networking	Networking	N/A	CKV_AZURE_133	Ensure that Cognitive Services accounts disable public network access
Application security	General security	N/A	CKV_AZURE_134	Ensure Application Gateway WAF prevents message lookup in Log4j2. See CVE-2021-44228 aka log4jshell
Backup and recovery convention	Backup	N/A	CKV_AZURE_135	Ensure that PostgreSQL Flexible server enables geo-redundant backups
Identity and access management	Identity and access management	N/A	CKV_AZURE_136	Ensure ACR admin account is disabled
Identity and access management	Identity and access management	N/A	CKV_AZURE_137	Ensures that ACR disables anonymous pulling of images
Networking	Networking	N/A	CKV_AZURE_138	Ensure ACR set to disable public networking
Identity and access management	Identity and access management	N/A	CKV_AZURE_139	Ensure that Local Authentication is disabled on CosmosDB
Identity and access management	Identity and access management	N/A	CKV_AZURE_140	Ensure AKS local admin account is disabled
Identity and access management	Identity and access management	N/A	CKV_AZURE_141	Ensure Machine Learning Compute Cluster Local Authentication is disabled
Networking	Networking	N/A	CKV_AZURE_142	Ensure AKS cluster nodes do not have public IP addresses
Networking	Networking	N/A	CKV_AZURE_143	Ensure that Public Access is disabled for Machine Learning Workspace
Networking	Networking	N/A	CKV_AZURE_144	Ensure Function app is using the latest version of TLS encryption
Logging	Logging and monitoring	N/A	CKV_AZURE_145	Ensure server parameter ‘log_retention’ is set to ‘ON’ for PostgreSQL Database Server
Networking	Networking	N/A	CKV_AZURE_146	Ensure PostgreSQL is using the latest version of TLS encryption
Networking	Networking	N/A	CKV_AZURE_147	Ensure Redis Cache is using the latest version of TLS encryption
Encryption	Encryption and data protection	N/A	CKV_AZURE_148	Ensure that Virtual machine does not enable password authentication
General security	General security	N/A	CKV_AZURE_149	Ensure Machine Learning Compute Cluster Minimum Nodes Set To 0
Encryption	Encryption and data protection	N/A	CKV_AZURE_150	Ensure Windows VM enables encryption
Encryption	Encryption and data protection	N/A	CKV_AZURE_151	Ensure Client Certificates are enforced for API management
Encryption	Encryption and data protection	N/A	CKV_AZURE_152	Ensure Client Certificates are enforced for API management
Networking	Networking	N/A	CKV_AZURE_153	Ensure web app redirects all HTTP traffic to HTTPS in Azure App Service Slot
Networking	Networking	N/A	CKV_AZURE_154	Ensure the App service slot is using the latest version of TLS encryption
Networking	Networking	N/A	CKV_AZURE_155	Ensure debugging is disabled for the App service slo
Logging	Logging and monitoring	N/A	CKV_AZURE_156	Ensure default Auditing policy for a SQL Server is configured to capture and retain the activity logs

General security	General security	N/A	CKV_AZURE_157	Ensure that Synapse workspace has data_exfiltration_protection_enabled
Networking	Networking	N/A	CKV_AZURE_158	Ensure that databricks workspace is not publi
Logging	Logging and monitoring	N/A	CKV_AZURE_159	Ensure function app builtin logging is enabled
Networking	Networking	N/A	CKV_AZURE_160	Ensure that HTTP (port 80) access is restricted from the interne
Networking	Networking	N/A	CKV_AZURE_161	Ensures Spring Cloud API Portal is enabled on for HTTP
Networking	Networking	N/A	CKV_AZURE_162	Ensures Spring Cloud API Portal Public Access Is Disable
General security	General security	N/A	CKV_AZURE_163	Enable vulnerability scanning for container images.
General security	General security	N/A	CKV_AZURE_164	Ensures that ACR uses signed/trusted images
Networking	Networking	N/A	CKV_AZURE_165	Ensure geo-replicated container registries to match multi-region container deployments
Misc	General security	N/A	CKV_AZURE_166	Ensure container image quarantine, scan, and mark images verified
General security	General security	N/A	CKV_AZURE_167	Ensure a retention policy is set to cleanup untagged manifests.
Kubernetes	General security	N/A	CKV_AZURE_168	Ensure Azure Kubernetes Cluster (AKS) nodes should use a minimum number of 50 pods.
Kubernetes	General security	N/A	CKV_AZURE_169	Ensure Azure Kubernetes Cluster (AKS) nodes use scale sets
General security	General security	N/A	CKV_AZURE_170	Ensure that AKS use the Paid Sku for its SLA
Networking	Networking	N/A	CKV_AZURE_171	Ensure AKS cluster upgrade channel is chose
General security	General security	N/A	CKV_AZURE_172	Ensure autorotation of Secrets Store CSI Driver secrets for AKS clusters
Encryption	Encryption and data protection	N/A	CKV_AZURE_173	Ensure API management uses at least TLS 1.2
Networking	Networking	N/A	CKV_AZURE_174	Ensure API management public access is disable
General security	General security	N/A	CKV_AZURE_175	Ensure Web PubSub uses a SKU with an SLA
Identity and access management	Identity and access management	N/A	CKV_AZURE_176	Ensure Web PubSub uses managed identities to access Azure resources
General security	General security	N/A	CKV_AZURE_177	Ensure Windows VM enables automatic updates
General security	General security	N/A	CKV_AZURE_178	Ensure linux VM enables SSH with keys for secure communication
General security	General security	N/A	CKV_AZURE_179	Ensure VM agent is installed
General security	General security	N/A	CKV_AZURE_180	Ensure that data explorer uses Sku with an SLA
Identity and access management	Identity and access management	N/A	CKV_AZURE_181	Ensure that data explorer/Kusto uses managed identities to access Azure resources securely.
Networking	Networking	N/A	CKV_AZURE_182	Ensure that VNET has at least 2 connected DNS Endpoint
Networking	Networking	N/A	CKV_AZURE_183	Ensure that VNET uses local DNS addresse
Identity and access management	Identity and access management	N/A	CKV_AZURE_184	Ensure 'local_auth_enabled' is set to 'False'
Networking	Networking	N/A	CKV_AZURE_185	Ensure 'Public Access' is not Enabled for App configuratio
Encryption	Encryption and data protection	N/A	CKV_AZURE_186	Ensure App configuration encryption block is set.
Backup and recovery convention	Backup	N/A	CKV_AZURE_187	Ensure App configuration purge protection is enabled
General security	General security	N/A	CKV_AZURE_188	Ensure App configuration Sku is standard
Networking	Networking	N/A	CKV_AZURE_189	Ensure that Azure Key Vault disables public network acces
Networking	Networking	N/A	CKV_AZURE_190	Ensure that Storage blobs restrict public acces
Identity and access management	Identity and access management	N/A	CKV_AZURE_191	Ensure that Managed identity provider is enabled for Azure Event Grid Topic
Identity and access management	Identity and access management	N/A	CKV_AZURE_192	Ensure that Azure Event Grid Topic local Authentication is disabled
Networking	Networking	N/A	CKV_AZURE_193	Ensure public network access is disabled for Azure Event Grid Topi
Identity and access management	Identity and access management	N/A	CKV_AZURE_194	Ensure that Managed identity provider is enabled for Azure Event Grid Domain
Identity and access management	Identity and access management	N/A	CKV_AZURE_195	Ensure that Azure Event Grid Domain local Authentication is disabled
General security	General security	N/A	CKV_AZURE_196	Ensure that SignalR uses a Paid Sku for its SLA
Networking	Networking	N/A	CKV_AZURE_197	Ensure the Azure CDN disables the HTTP endpoin
Networking	Networking	N/A	CKV_AZURE_198	Ensure the Azure CDN enables the HTTPS endpoin
Encryption	Encryption and data protection	N/A	CKV_AZURE_199	Ensure that Azure Service Bus uses double encryption
Networking	Networking	N/A	CKV_AZURE_200	Ensure the Azure CDN endpoint is using the latest version of TLS encryptio
Encryption	Encryption and data protection	N/A	CKV_AZURE_201	Ensure that Azure Service Bus uses a customer-managed key to encrypt data
Identity and access management	Identity and access management	N/A	CKV_AZURE_202	Ensure that Managed identity provider is enabled for Azure Service Bus
Identity and access management	Identity and access management	N/A	CKV_AZURE_203	Ensure Azure Service Bus Local Authentication is disabled
Networking	Networking	N/A	CKV_AZURE_204	Ensure 'public network access enabled' is set to 'False' for Azure Service Bu
Networking	Networking	N/A	CKV_AZURE_205	Ensure Azure Service Bus is using the latest version of TLS encryptio
Backup and recovery convention	Backup	N/A	CKV_AZURE_206	Ensure that Storage Accounts use replication
Identity and access management	Identity and access management	N/A	CKV_AZURE_207	Ensure Azure Cognitive Search service uses managed identities to access Azure resources
General security	General security	N/A	CKV_AZURE_208	Ensure that Azure Cognitive Search maintains SLA for index updates
General security	General security	N/A	CKV_AZURE_209	Ensure that Azure Cognitive Search maintains SLA for search index queries
Networking	Networking	N/A	CKV_AZURE_210	Ensure Azure Cognitive Search service allowed IPS does not give public Acces
General security	General security	N/A	CKV_AZURE_211	Ensure App Service plan suitable for production use
General security	General security	N/A	CKV_AZURE_212	Ensure App Service has a minimum number of instances for failover
Networking	Networking	N/A	CKV_AZURE_213	Ensure that App Service configures health chec
General security	General security	N/A	CKV_AZURE_214	Ensure App Service is set to be always on
Encryption	Encryption and data protection	N/A	CKV_AZURE_215	Ensure API management backend uses https
Networking	Networking	N/A	CKV_AZURE_216	Ensure DenyIntelMode is set to Deny for Azure Firewall
Encryption	Encryption and data protection	N/A	CKV_AZURE_217	Ensure Azure Application gateways listener that allow connection requests over HTTP
Encryption	Encryption and data protection	N/A	CKV_AZURE_218	Ensure Application Gateway defines secure protocols for in transit communication
Networking	Networking	N/A	CKV_AZURE_219	Ensure Firewall defines a firewall polic
Networking	Networking	N/A	CKV_AZURE_220	Ensure Firewall policy has IDPS mode as den
Networking	Networking	N/A	CKV_AZURE_221	Ensure that Azure Function App public network access is disable
Networking	Networking	N/A	CKV_AZURE_222	Ensure that Azure Web App public network access is disable
Encryption	Encryption and data protection	N/A	CKV_AZURE_223	Ensure Event Hub Namespace uses at least TLS 1.2
Logging	Logging and monitoring	N/A	CKV_AZURE_224	Ensure that the Ledger feature is enabled on database that requires cryptographic proof and nonrepudiation of data integrity
Backup and recovery convention	Backup	N/A	CKV_AZURE_225	Ensure the App Service Plan is zone redundant
Kubernetes	General security	N/A	CKV_AZURE_226	Ensure ephemeral disks are used for OS disks
Kubernetes	General security	N/A	CKV_AZURE_227	Ensure that the AKS cluster encrypt temp disks, caches, and data flows between Compute and Storage resources
Backup and recovery convention	Backup	N/A	CKV_AZURE_228	Ensure the Azure Event Hub Namespace is zone redundant
Backup and recovery convention	Backup	N/A	CKV_AZURE_229	Ensure the Azure SQL Database Namespace is zone redundant
Backup and recovery convention	Backup	N/A	CKV_AZURE_230	Standard Replication should be enabled
Backup and recovery convention	Backup	N/A	CKV_AZURE_231	Ensure App Service Environment is zone redundant
Kubernetes	General security	N/A	CKV_AZURE_232	Ensure that only critical system pods run on system nodes
Backup and recovery convention	Backup	N/A	CKV_AZURE_233	Ensure Azure Container Registry (ACR) is zone redundant
General security	General security	N/A	CKV_AZURE_234	Ensure that Azure Defender for cloud is set to On for Resource Manager
General security	General security	N/A	CKV_AZURE_235	Ensure that Azure container environment variables are configured with secure values only
Networking	Networking	N/A	CKV_AZURE_236	Ensure Storage Account is using the latest version of TLS encryption

D List of Checks in Checkov 2

Old category	New category	Rule name	Rule ID	Rule description
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_1	Ensure storage for critical data are encrypted with Customer Managed Key
General Security	General Security	N/A	CKV2_AZURE_2	Ensure that Vulnerability Assessment (VA) is enabled on a SQL server by setting a Storage Account
General Security	General Security	N/A	CKV2_AZURE_3	Ensure that VA setting Periodic Recurring Scans is enabled on a SQL server
General Security	General Security	N/A	CKV2_AZURE_4	Ensure Azure SQL server ADS VA Send scan reports to is configured
General Security	General Security	N/A	CKV2_AZURE_5	Ensure that VA setting 'Also send email notifications to admins and subscription owners' is set for a SQL server
General Security	General Security	N/A	CKV2_AZURE_6	Ensure 'Allow access to Azure services' for PostgreSQL Database Server is disabled
General Security	General Security	N/A	CKV2_AZURE_7	Ensure that Azure Active Directory Admin is configured
Logging	Logging and monitoring	N/A	CKV2_AZURE_8	Ensure the storage container storing the activity logs is not publicly accessible
General Security	General Security	N/A	CKV2_AZURE_9	Ensure Virtual Machines are utilizing Managed Disks
General Security	General Security	N/A	CKV2_AZURE_10	Ensure that Microsoft Antimalware is configured to automatically updates for Virtual Machines
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_11	Ensure that Azure Data Explorer encryption at rest uses a customer-managed key
Backup and Recovery Convention	Backup	N/A	CKV2_AZURE_12	Ensure that virtual machines are backed up using Azure Backup
General Security	General Security	N/A	CKV2_AZURE_13	Ensure that sql servers enables data security policy
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_14	Ensure that Unattached disks are encrypted
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_15	Ensure that Azure data factories are encrypted with a customer-managed key
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_16	Ensure that MySQL server enables customer-managed key for encryption
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_17	Ensure that PostgreSQL server enables customer-managed key for encryption
Networking	Networking	N/A	CKV2_AZURE_19	Ensure that Azure Synapse workspaces have no IP firewall rules attached
Logging	Logging and monitoring	N/A	CKV2_AZURE_20	Ensure Storage logging is enabled for Table service for read requests
Logging	Logging and monitoring	N/A	CKV2_AZURE_21	Ensure Storage logging is enabled for Blob service for read requests
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_22	Ensure that Cognitive Services enables customer-managed key for encryption
Networking	Networking	N/A	CKV2_AZURE_23	Ensure Azure spring cloud is configured with Virtual network (Vnet)
General Security	General Security	N/A	CKV2_AZURE_24	Ensure Azure automation account does NOT have overly permissive network access
General Security	General Security	N/A	CKV2_AZURE_25	Ensure Azure SQL database Transparent Data Encryption (TDE) is enabled
General Security	General Security	N/A	CKV2_AZURE_26	Ensure Azure PostgreSQL Flexible server is not configured with overly permissive network access
General Security	General Security	N/A	CKV2_AZURE_27	Ensure Azure AD authentication is enabled for Azure SQL (MSSQL)
General Security	General Security	N/A	CKV2_AZURE_28	Ensure Container Instance is configured with managed identity
General Security	General Security	N/A	CKV2_AZURE_29	Ensure AKS cluster has Azure CNI networking enabled
General Security	General Security	N/A	CKV2_AZURE_30	Ensure Azure Container Registry (ACR) has HTTPS enabled for webhook
General Security	General Security	N/A	CKV2_AZURE_31	Ensure VNET subnet is configured with a Network Security Group (NSG)
General Security	General Security	N/A	CKV2_AZURE_32	Ensure private endpoint is configured to key vault
General Security	General Security	N/A	CKV2_AZURE_33	Ensure storage account is configured with private endpoint
Networking	Networking	N/A	CKV2_AZURE_34	Ensure Azure SQL server firewall is not overly permissive
Identity and Access Management	Identity and Access Management	N/A	CKV2_AZURE_35	Ensure Azure recovery services vault is configured with managed identity
Identity and Access Management	Identity and Access Management	N/A	CKV2_AZURE_36	Ensure Azure automation account is configured with managed identity
Encryption	Encryption and Data Protection	N/A	CKV2_AZURE_37	Ensure Azure MariaDB server is using latest TLS (1.2)
General Security	General Security	N/A	CKV2_AZURE_38	Ensure soft-delete is enabled on Azure storage account
Networking	Networking	N/A	CKV2_AZURE_39	Ensure Azure VM is not configured with public IP and serial console access
Identity and Access Management	Identity and Access Management	N/A	CKV2_AZURE_40	Ensure storage account is not configured with Shared Key authorization
Identity and Access Management	Identity and Access Management	N/A	CKV2_AZURE_41	Ensure storage account is configured with SAS expiration policy
General Security	General Security	N/A	CKV2_AZURE_42	Ensure Azure PostgreSQL server is configured with private endpoint
General Security	General Security	N/A	CKV2_AZURE_43	Ensure Azure MariaDB server is configured with private endpoint
General Security	General Security	N/A	CKV2_AZURE_44	Ensure Azure MySQL server is configured with private endpoint
General Security	General Security	N/A	CKV2_AZURE_45	Ensure Microsoft SQL server is configured with private endpoint
General Security	General Security	N/A	CKV2_AZURE_46	Ensure that Azure Synapse Workspace vulnerability assessment is enabled
Identity and Access Management	Identity and Access Management	N/A	CKV2_AZURE_47	Ensure storage account is configured without blob anonymous access

E List of Checks in Terrascan

Old category	New category	Reference ID	Rule ID	Rule description
Infrastructure Security	Networking asdf	accurics.azure.NS.361	AC_AZURE_0131	Ensure ‘Enforce SSL connection’ is set to ‘ENABLED’ for MySQL Database Server.
Logging and Monitoring	logging_and_monitoring	accurics.azure.LOG.357	AC_AZURE_0136	Ensure that ‘Auditing’ Retention is ‘greater than 90 days’ for MSSQL servers.
Logging and Monitoring	logging_and_monitoring	accurics.azure.MON.355	AC_AZURE_0137	Ensure that ‘Auditing’ is set to ‘On’ for MSSQL servers
Data Protection	encryption_and_data_protection	accurics.azure.EKM.156	AC_AZURE_0143	Ensure that ‘Unattached disks’ are encrypted in Azure Managed Disk
Infrastructure Security	Networking	accurics.azure.NS.382	AC_AZURE_0158	Ensure AKS cluster has Network Policy configured.
Infrastructure Security	general_security	accurics.azure.NS.383	AC_AZURE_0161	Ensure Kube Dashboard is disabled
Data Protection	encryption_and_data_protection	accurics.azure.EKM.26	AC_AZURE_0163	Ensure that the expiration date is set on all secrets
Data Protection	encryption_and_data_protection	accurics.azure.EKM.25	AC_AZURE_0164	Ensure that the expiration date is set on all keys
Logging and Monitoring	logging_and_monitoring	accurics.azure.EKM.20	AC_AZURE_0169	Ensure that logging for Azure KeyVault is ‘Enabled’
Data Protection	encryption_and_data_protection	accurics.azure.EKM.164*	AC_AZURE_0170	Ensure the key vault is recoverable - enable “Soft Delete” setting for a Key Vault
Infrastructure Security	Networking	accurics.azure.NS.32	AC_AZURE_0184	Ensure to filter source Ips for Cosmos DB Account
Resilience	backup	accurics.azure.AKS.3	AC_AZURE_0185	Ensure Container Registry has locks
Identity and Access Management	identity_and_access_management	accurics.azure.EKM.164*	AC_AZURE_0186	Ensure that admin user is disabled for Container Registry
Infrastructure Security	Networking	accurics.azure.NS.147	AC_AZURE_0189	Ensure Azure Application Gateway Web application firewall (WAF) is enabled
Infrastructure Security	Networking	AC_AZURE_0270	AC_AZURE_0270	Ensure CIFS / SMB (Tcp:3020) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0271	AC_AZURE_0271	Ensure CIFS / SMB (Tcp:3020) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0272	AC_AZURE_0272	Ensure CIFS / SMB (Tcp:3020) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0273	AC_AZURE_0273	Ensure Cassandra (Tcp:7001) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0274	AC_AZURE_0274	Ensure Cassandra (Tcp:7001) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0275	AC_AZURE_0275	Ensure Cassandra (Tcp:7001) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0276	AC_AZURE_0276	Ensure Cassandra OpsCenter (Tcp:61621) is not exposed to entire internet for Azure Network Security Rule
Compliance Validation	general_security	accurics.azure.CAM.162	AC_AZURE_0277	Ensure that Cosmos DB Account has an associated tag
Identity and Access Management	identity_and_access_management	accurics.azure.NS.169	AC_AZURE_0280	Restrict Azure SQL Server accessibility to a minimal address range
Infrastructure Security	Networking	AC_AZURE_0285	AC_AZURE_0285	Ensure SSH (Tcp:22) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0286	AC_AZURE_0286	Ensure SSH (Tcp:22) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0287	AC_AZURE_0287	Ensure SSH (Tcp:22) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	accurics.azure.NS.370	AC_AZURE_0309	Ensure default network access rule for Storage Accounts is set to deny.
Infrastructure Security	Networking	AC_AZURE_0342	AC_AZURE_0342	Ensure that RDP access is restricted from the internet for Azure Network Security Rule
Infrastructure Security	Networking	accurics.azure.NS.161	AC_AZURE_0356	Ensure that Azure Virtual Network subnet is configured with a Network Security Group
Infrastructure Security	Networking	AC_AZURE_0357	AC_AZURE_0357	Ensure that request initiated from all ports () for all destination ports () is restricted from the internet for Azure Network Security Rule
Identity and Access Management	identity_and_access_management	accurics.azure.IAM.368	AC_AZURE_0366	Anonymous, public read access to a container and its blobs can be enabled in Azure Blob storage. This is only recommended if absolutely necessary.
Infrastructure Security	Networking	accurics.azure.NS.4	AC_AZURE_0370	Ensure default network access rule for Storage Accounts is not open to public
Infrastructure Security	general_security	accurics.azure.NS.2	AC_AZURE_0371	Ensure ‘Trusted Microsoft Services’ is enabled for Storage Account access
Data Protection	encryption_and_data_protection	accurics.azure.EKM.7	AC_AZURE_0373	Ensure that ‘Secure transfer required’ is enabled for Storage Accounts
Compliance Validation	general_security	accurics.azure.LOG.356	AC_AZURE_0375	Ensure that ‘Auditing’ Retention is ‘greater than 90 days’ for SQL servers.
Logging and Monitoring	logging_and_monitoring	accurics.azure.MON.354	AC_AZURE_0376	Ensure that ‘Auditing’ is set to ‘On’ for SQL servers
Compliance Validation	general_security	accurics.azure.IAM.138	AC_AZURE_0377	Avoid using names like ‘Admin’ for an Azure SQL Server admin account login
Identity and Access Management	identity_and_access_management	accurics.azure.IAM.10	AC_AZURE_0378	Ensure that Azure Active Directory Admin is configured for SQL Server
Infrastructure Security	Networking	accurics.azure.NS.21	AC_AZURE_0380	Ensure that no SQL Server allows ingress from 0.0.0.0/0 (ANY IP)
Identity and Access Management	identity_and_access_management	accurics.azure.NS.5	AC_AZURE_0381	Ensure entire Azure infrastructure doesn’t have access to Azure SQL ServerEnsure entire Azure infrastructure doesn’t have access to Azure SQL Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.MON.157	AC_AZURE_0383	Ensure that ‘Threat Detection’ is enabled for Azure SQL Database
Compliance Validation	general_security	accurics.azure.IAM.137	AC_AZURE_0384	Avoid using names like ‘Admin’ for an Azure SQL Server Active Directory Administrator account
Security Best Practices	general_security	accurics.azure.OPS.349	AC_AZURE_0385	Ensure that standard pricing tiers are selected
Logging and Monitoring	logging_and_monitoring	accurics.azure.MON.353	AC_AZURE_0386	Ensure that ‘Send email notification for high severity alerts’ is set to ‘On’
Identity and Access Management	identity_and_access_management	accurics.azure.IAM.388	AC_AZURE_0388	Ensure that there are no guest users
Identity and Access Management	identity_and_access_management	accurics.azure.NS.272	AC_AZURE_0389	Ensure that Azure Resource Group has resource lock enabled
Identity and Access Management	identity_and_access_management	accurics.azure.NS.166	AC_AZURE_0390	Ensure there are no firewall rules allowing Redis Cache access for a large number of source IPs
Identity and Access Management	identity_and_access_management	accurics.azure.NS.31	AC_AZURE_0391	Ensure there are no firewall rules allowing unrestricted access to Redis from other Azure sources
Identity and Access Management	identity_and_access_management	accurics.azure.NS.30	AC_AZURE_0392	Ensure there are no firewall rules allowing unrestricted access to Redis from the Internet
Security Best Practices	general_security	accurics.azure.NS.13	AC_AZURE_0393	Ensure that Redis is updated regularly with security and operational updates.Note this feature is only available to Premium tier Redis Caches.
Infrastructure Security	Networking	accurics.azure.EKM.23	AC_AZURE_0394	Ensure that the Redis Cache accepts only SSL connections
Resilience	backup	accurics.azure.BDR.163	AC_AZURE_0407	Ensure that Geo Redundant Backups is enabled on PostgreSQL
Identity and Access Management	identity_and_access_management	accurics.azure.EKM.1	AC_AZURE_0408	Ensure ‘Enforce SSL connection’ is set to ‘ENABLED’ for PostgreSQL Database Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.LOG.364	AC_AZURE_0409	Ensure server parameter ‘log_checkpoints’ is set to ‘ON’ for PostgreSQL Database Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.LOG.155	AC_AZURE_0410	Ensure server parameter ‘log_retention_days’ is greater than 3 days for PostgreSQL Database Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.LOG.154	AC_AZURE_0411	Ensure server parameter ‘log_duration’ is set to ‘ON’ for PostgreSQL Database Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.LOG.153	AC_AZURE_0412	Ensure server parameter ‘log_disconnections’ is set to ‘ON’ for PostgreSQL Database Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.LOG.152	AC_AZURE_0413	Ensure server parameter ‘log_connections’ is set to ‘ON’ for PostgreSQL Database Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.LOG.151	AC_AZURE_0414	Ensure server parameter ‘connection_throttling’ is set to ‘ON’ for PostgreSQL Database Server
Logging and Monitoring	logging_and_monitoring	accurics.azure.NS.11	AC_AZURE_0418	Enable Network Watcher for Azure subscriptions. Network diagnostic and visualization tools available with Network Watcher help users understand, diagnose, and gain insights to the network in Azure.

Resilience	backup	accurics.azure.NS.342	AC_AZURE_0419	Ensure that Network Security Group Flow Log retention period is 'greater than 90 days' for Azure Network Watcher Flow Log
Infrastructure Security	Networking	AC_AZURE_0421	AC_AZURE_0421	Ensure server is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0422	AC_AZURE_0422	Ensure VNC Server (Tcp:5900) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0423	AC_AZURE_0423	Ensure VNC Server (Tcp:5900) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0424	AC_AZURE_0424	Ensure VNC Server (Tcp:5900) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0425	AC_AZURE_0425	Ensure VNC Listener (Tcp:5500) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0426	AC_AZURE_0426	Ensure VNC Listener (Tcp:5500) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0427	AC_AZURE_0427	Ensure VNC Listener (Tcp:5500) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0428	AC_AZURE_0428	Ensure Telnet (Tcp:23) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0429	AC_AZURE_0429	Ensure Telnet (Tcp:23) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0430	AC_AZURE_0430	Ensure Telnet (Tcp:23) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0431	AC_AZURE_0431	Ensure SaltStack Master (Tcp:4506) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0432	AC_AZURE_0432	Ensure SaltStack Master (Tcp:4506) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0433	AC_AZURE_0433	Ensure SaltStack Master (Tcp:4506) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0434	AC_AZURE_0434	Ensure SaltStack Master (Tcp:4505) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0435	AC_AZURE_0435	Ensure SaltStack Master (Tcp:4505) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0436	AC_AZURE_0436	Ensure SaltStack Master (Tcp:4505) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0437	AC_AZURE_0437	Ensure SQL Server Analysis (Tcp:2383) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0438	AC_AZURE_0438	Ensure SQL Server Analysis (Tcp:2383) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0439	AC_AZURE_0439	Ensure SQL Server Analysis (Tcp:2383) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0440	AC_AZURE_0440	Ensure SQL Server Analysis (Tcp:2382) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0441	AC_AZURE_0441	Ensure SQL Server Analysis (Tcp:2382) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0442	AC_AZURE_0442	Ensure SQL Server Analysis (Tcp:2382) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0443	AC_AZURE_0443	Ensure SNMP (Udp:161) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0444	AC_AZURE_0444	Ensure SNMP (Udp:161) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0445	AC_AZURE_0445	Ensure SNMP (Udp:161) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0446	AC_AZURE_0446	Ensure SMTP (Tcp:25) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0447	AC_AZURE_0447	Ensure SMTP (Tcp:25) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0448	AC_AZURE_0448	Ensure SMTP (Tcp:25) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0449	AC_AZURE_0449	Ensure Puppet Master (Tcp:8140) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0450	AC_AZURE_0450	Ensure Puppet Master (Tcp:8140) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0451	AC_AZURE_0451	Ensure Puppet Master (Tcp:8140) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0452	AC_AZURE_0452	Ensure Prevalent known internal port (Tcp:3000) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0453	AC_AZURE_0453	Ensure Prevalent known internal port (Tcp:3000) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0454	AC_AZURE_0454	Ensure Prevalent known internal port (Tcp:3000) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0455	AC_AZURE_0455	Ensure PostgreSQL (Udp:5432) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0456	AC_AZURE_0456	Ensure PostgreSQL (Udp:5432) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0457	AC_AZURE_0457	Ensure PostgreSQL (Udp:5432) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0458	AC_AZURE_0458	Ensure PostgreSQL (Tcp:5432) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0459	AC_AZURE_0459	Ensure PostgreSQL (Tcp:5432) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0460	AC_AZURE_0460	Ensure PostgreSQL (Tcp:5432) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0461	AC_AZURE_0461	Ensure POP3 (Tcp:110) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0462	AC_AZURE_0462	Ensure POP3 (Tcp:110) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0463	AC_AZURE_0463	Ensure POP3 (Tcp:110) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0464	AC_AZURE_0464	Ensure Oracle DB SSL (Udp:2484) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0465	AC_AZURE_0465	Ensure Oracle DB SSL (Udp:2484) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0466	AC_AZURE_0466	Ensure Oracle DB SSL (Udp:2484) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0467	AC_AZURE_0467	Ensure Oracle DB SSL (Tcp:2484) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0468	AC_AZURE_0468	Ensure Oracle DB SSL (Tcp:2484) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0469	AC_AZURE_0469	Ensure Oracle DB SSL (Tcp:2484) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0470	AC_AZURE_0470	Ensure NetBIOS Session Service (Udp:139) is not exposed to private hosts more than 32 for Azure Network Security Rule

Infrastructure Security	Networking	AC_AZURE_0471	AC_AZURE_0471	Ensure NetBIOS Session Service (Udp:139) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0472	AC_AZURE_0472	Ensure NetBIOS Session Service (Udp:139) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0473	AC_AZURE_0473	Ensure NetBIOS Session Service (Tcp:139) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0474	AC_AZURE_0474	Ensure NetBIOS Session Service (Tcp:139) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0475	AC_AZURE_0475	Ensure NetBIOS Session Service (Tcp:139) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0476	AC_AZURE_0476	Ensure NetBIOS Datagram Service (Udp:138) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0477	AC_AZURE_0477	Ensure NetBIOS Datagram Service (Udp:138) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0478	AC_AZURE_0478	Ensure NetBIOS Datagram Service (Udp:138) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0479	AC_AZURE_0479	Ensure NetBIOS Datagram Service (Tcp:138) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0480	AC_AZURE_0480	Ensure NetBIOS Datagram Service (Tcp:138) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0481	AC_AZURE_0481	Ensure NetBIOS Datagram Service (Tcp:138) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0482	AC_AZURE_0482	Ensure NetBIOS Name Service (Udp:137) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0483	AC_AZURE_0483	Ensure NetBIOS Name Service (Udp:137) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0484	AC_AZURE_0484	Ensure NetBIOS Name Service (Udp:137) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0485	AC_AZURE_0485	Ensure NetBIOS Name Service (Tcp:137) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0486	AC_AZURE_0486	Ensure NetBIOS Name Service (Tcp:137) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0487	AC_AZURE_0487	Ensure NetBIOS Name Service (Tcp:137) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0488	AC_AZURE_0488	Ensure MySQL (Tcp:3306) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0489	AC_AZURE_0489	Ensure MySQL (Tcp:3306) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0490	AC_AZURE_0490	Ensure MySQL (Tcp:3306) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0491	AC_AZURE_0491	Ensure Mongo Web Portal (Tcp:27018) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0492	AC_AZURE_0492	Ensure Mongo Web Portal (Tcp:27018) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0493	AC_AZURE_0493	Ensure Mongo Web Portal (Tcp:27018) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0494	AC_AZURE_0494	Ensure Microsoft-DS (Tcp:445) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0495	AC_AZURE_0495	Ensure Microsoft-DS (Tcp:445) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0496	AC_AZURE_0496	Ensure Microsoft-DS (Tcp:445) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0497	AC_AZURE_0497	Ensure Memcached SSL (Udp:11215) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0498	AC_AZURE_0498	Ensure Memcached SSL (Udp:11215) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0499	AC_AZURE_0499	Ensure Memcached SSL (Udp:11215) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0500	AC_AZURE_0500	Ensure Memcached SSL (Udp:11214) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0501	AC_AZURE_0501	Ensure Memcached SSL (Udp:11214) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0502	AC_AZURE_0502	Ensure Memcached SSL (Udp:11214) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0503	AC_AZURE_0503	Ensure Memcached SSL (Tcp:11215) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0504	AC_AZURE_0504	Ensure Memcached SSL (Tcp:11215) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0505	AC_AZURE_0505	Ensure Memcached SSL (Tcp:11215) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0506	AC_AZURE_0506	Ensure Memcached SSL (Tcp:11214) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0507	AC_AZURE_0507	Ensure Memcached SSL (Tcp:11214) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0508	AC_AZURE_0508	Ensure Memcached SSL (Tcp:11214) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0509	AC_AZURE_0509	Ensure MSSQL Server (Tcp:1433) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0510	AC_AZURE_0510	Ensure MSSQL Server (Tcp:1433) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0511	AC_AZURE_0511	Ensure MSSQL Server (Tcp:1433) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0512	AC_AZURE_0512	Ensure MSSQL Debugger (Tcp:135) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0513	AC_AZURE_0513	Ensure MSSQL Debugger (Tcp:135) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0514	AC_AZURE_0514	Ensure MSSQL Debugger (Tcp:135) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0515	AC_AZURE_0515	Ensure MSSQL Browser (Udp:1434) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0516	AC_AZURE_0516	Ensure MSSQL Browser (Udp:1434) is not exposed to public for Azure Network Security Rule

Infrastructure Security	Networking	AC_AZURE_0517	AC_AZURE_0517	Ensure MSSQL Browser (Udp:1434) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0518	AC_AZURE_0518	Ensure MSSQL Admin (Tcp:1434) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0519	AC_AZURE_0519	Ensure MSSQL Admin (Tcp:1434) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0520	AC_AZURE_0520	Ensure MSSQL Admin (Tcp:1434) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0521	AC_AZURE_0521	Ensure LDAP SSL (Tcp:636) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0522	AC_AZURE_0522	Ensure LDAP SSL (Tcp:636) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0523	AC_AZURE_0523	Ensure LDAP SSL (Tcp:636) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0524	AC_AZURE_0524	Ensure Known internal web port (Tcp:8080) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0525	AC_AZURE_0525	Ensure Known internal web port (Tcp:8080) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0526	AC_AZURE_0526	Ensure Known internal web port (Tcp:8080) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0527	AC_AZURE_0527	Ensure Known internal web port (Tcp:8000) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0528	AC_AZURE_0528	Ensure Known internal web port (Tcp:8000) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0529	AC_AZURE_0529	Ensure Known internal web port (Tcp:8000) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0530	AC_AZURE_0530	Ensure Hadoop Name Node (Tcp:9000) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0531	AC_AZURE_0531	Ensure Hadoop Name Node (Tcp:9000) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0532	AC_AZURE_0532	Ensure Hadoop Name Node (Tcp:9000) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0533	AC_AZURE_0533	Ensure DNS (Udp:53) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0534	AC_AZURE_0534	Ensure DNS (Udp:53) is not exposed to public for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0535	AC_AZURE_0535	Ensure DNS (Udp:53) is not exposed to entire internet for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0536	AC_AZURE_0536	Ensure Cassandra OpsCenter (Tcp:61621) is not exposed to private hosts more than 32 for Azure Network Security Rule
Infrastructure Security	Networking	AC_AZURE_0537	AC_AZURE_0537	Ensure Cassandra OpsCenter (Tcp:61621) is not exposed to public for Azure Network Security Rule

F List of Checks in tfsec

Old category	New categories	rule name	rule ID	rule description
N/A	Identity and Access Management	azure-appservice-require-client-cert	AVD-AZU-0001	Web App accepts incoming client certificate
N/A	Identity and Access Management	azure-appservice-account-identity-registered	AVD-AZU-0002	Web App has registration with AD enabled
N/A	Identity and Access Management	azure-appservice-authentication-enabled	AVD-AZU-0003	App Service authentication is activated
N/A	Networking	azure-appservice-enforce-https	AVD-AZU-0004	Ensure the Function App can only be accessed via HTTPS. The default is false.
N/A	Networking	azure-appservice-enable-http2	AVD-AZU-0005	Web App uses the latest HTTP version
N/A	Networking	azure-appservice-use-secure-tls-policy	AVD-AZU-0006	Web App uses latest TLS version
N/A	Identity and Access Management	azure-storage-no-public-access	AVD-AZU-0007	Storage containers in blob storage mode should not have public access
N/A	Networking	azure-storage-enforce-https	AVD-AZU-0008	Storage accounts should be configured to only accept transfers that are over secure connections
N/A	Logging and Monitoring	azure-storage-queue-services-logging-enabled	AVD-AZU-0009	When using Queue Services for a storage account, logging should be enabled.
N/A	General Security	azure-storage-allow-microsoft-service-bypass	AVD-AZU-0010	Trusted Microsoft Services should have bypass access to Storage accounts
N/A	Networking	azure-storage-use-secure-tls-policy	AVD-AZU-0011	The minimum TLS version for Storage Accounts should be TLS1_2
N/A	Networking	azure-storage-default-action-deny	AVD-AZU-0012	The default action on Storage account network rules should be set to deny
N/A	Identity and Access Management	azure-keyvault-specify-network-acl	AVD-AZU-0013	Key vault should have the network acl block specified
N/A	Identity and Access Management	azure-keyvault-ensure-key-expiry	AVD-AZU-0014	Ensure that the expiration date is set on all keys
N/A	Identity and Access Management	azure-keyvault-content-type-for-secret	AVD-AZU-0015	Key vault Secret should have a content type set
N/A	Identity and Access Management	azure-keyvault-no-purge	AVD-AZU-0016	Key vault should have purge protection enabled
N/A	Identity and Access Management	azure-keyvault-ensure-secret-expiry	AVD-AZU-0017	Key Vault Secret should have an expiration date set
N/A	General Security	azure-database-threat-alert-email-set	AVD-AZU-0018	At least one email address is set for threat alerts
N/A	Logging and Monitoring	azure-database-postgres-configuration-log-connections	AVD-AZU-0019	Ensure server parameter 'log_connections' is set to 'ON' for PostgreSQL Database Server
N/A	Identity and Access Management	azure-database-enable-ssl-enforcement	AVD-AZU-0020	SSL should be enforced on database connections where applicable
N/A	General Security	azure-database-postgres-configuration-connection-throttling	AVD-AZU-0021	Ensure server parameter 'connection_throttling' is set to 'ON' for PostgreSQL Database Server
N/A	Identity and Access Management	azure-database-no-public-access	AVD-AZU-0022	Ensure databases are not publicly accessible
N/A	General Security	azure-database-threat-alert-email-to-owner	AVD-AZU-0023	Security threat alerts go to subcription owners and co-administrators
N/A	Logging and Monitoring	azure-database-postgres-configuration-log-checkpoints	AVD-AZU-0024	Ensure server parameter 'log_checkpoints' is set to 'ON' for PostgreSQL Database Server
N/A	Logging and Monitoring	azure-database-retention-period-set	AVD-AZU-0025	Database auditing rentention period should be longer than 90 days
N/A	Networking	azure-database-secure-tls-policy	AVD-AZU-0026	Databases should have the minimum TLS set for connections
N/A	Logging and Monitoring	azure-database-enable-audit	AVD-AZU-0027	Auditing should be enabled on Azure SQL Databases
N/A	General Security	azure-database-all-threat-alerts-enabled	AVD-AZU-0028	No threat detections are set
N/A	Identity and Access Management	azure-database-no-public-firewall-access	AVD-AZU-0029	Ensure database firewalls do not permit public access
N/A	Identity and Access Management	azure-authorization-limit-role-actions	AVD-AZU-0030	Roles limited to the required actions
N/A	Logging and Monitoring	azure-monitor-activity-log-retention-set	AVD-AZU-0031	Ensure the activity retention log is set to at least a year
N/A	Logging and Monitoring	azure-monitor-capture-all-regions	AVD-AZU-0032	Ensure activitys are captured for all locations
N/A	Logging and Monitoring	azure-monitor-capture-all-activities	AVD-AZU-0033	Ensure log profile captures all activities
N/A	Networking	azure-synapse-virtual-network-enabled	AVD-AZU-0034	Synapse Workspace should have managed virtual network enabled, the default is disabled.
N/A	Identity and Access Management	azure-datafactory-no-public-access	AVD-AZU-0035	Data Factory should have public access disabled, the default is enabled.
N/A	Encryption and Data Protection	azure-datalake-enable-at-rest-encryption	AVD-AZU-0036	Unencrypted data lake storage.
N/A	Secrets	azure-compute-no-secrets-in-custom-data	AVD-AZU-0037	Ensure that no sensitive credentials are exposed in VM custom_data
N/A	Encryption and Data Protection	azure-compute-enable-disk-encryption	AVD-AZU-0038	Enable disk encryption on managed disk
N/A	Identity and Access Management	azure-compute-disable-password-authentication	AVD-AZU-0039	Password authentication should be disabled on Azure virtual machines
N/A	Logging and Monitoring	azure-container-logging	AVD-AZU-0040	Ensure AKS logging to Azure Monitoring is Configured
N/A	Networking	azure-container-limit-authorized-ips	AVD-AZU-0041	Ensure AKS has an API Server Authorized IP Ranges enabled
N/A	Identity and Access Management	azure-container-use-rbac-permissions	AVD-AZU-0042	Ensure RBAC is enabled on AKS clusters
N/A	Networking	azure-container-configured-network-policy	AVD-AZU-0043	Ensure AKS cluster has Network Policy configured
N/A	Logging and Monitoring	azure-security-center-alert-on-severe-notifications	AVD-AZU-0044	Send notification emails for high severity alerts
N/A	General Security	azure-security-center-enable-standard-subscription	AVD-AZU-0045	Enable the standard security center subscription tier
N/A	General Security	azure-security-center-set-required-contact-details	AVD-AZU-0046	The required contact details should be set for security center
N/A	Networking	azure-network-no-public-ingress	AVD-AZU-0047	An inbound network security rule allows traffic from /0.
N/A	Identity and Access Management	azure-network-disable-rdp-from-internet	AVD-AZU-0048	RDP access should not be accessible from the Internet, should be blocked on port 3389
N/A	Logging and Monitoring	azure-network-retention-policy-set	AVD-AZU-0049	Retention policy for flow logs should be enabled and set to greater than 90 days
N/A	Identity and Access Management	azure-network-ssh-blocked-from-internet	AVD-AZU-0050	SSH access should not be accessible from the Internet, should be blocked on port 22
N/A	Networking	azure-network-no-public-egress	AVD-AZU-0051	An outbound network security rule allows traffic to /0.
N/A	Secrets	general-secrets-no-plaintext-exposure	N/A	Sensitive data should not be exposed in plaintext.