Infrastructure as Code Security Cheatsheet

Introduction

Infrastructure as code (IaC) also known as software-defined infrastructure, allows the configuration and deployment of infrastructure components faster with consistency by allowing them to be defined as a code and also enables repeatable deployments across environments.

Security best practices

Here are some of the security best practices for IaC that can be easily integrated to the Software Development Lifecycle:

Develop and Distribute

- IDE plugins - Leverage standard security plug-ins in the integrated development environment (IDE) which helps in the early detection of potential risks and drastically reduces the time to address any issues later in the development cycle. Plugins such as TFLint, Checkov, Docker Linter, docker-vulnerability-extension, Security Scan, Contrast Security etc, help in the security assessment of the IaC
- Threat modelling - Build the threat modelling landscape earlier in the development cycle to ensure there is enough visibility of the high-risk, high-volume aspects of the code and flexibility to include security throughout to ensure the assets are safely managed
- Managing secrets - The secrets usually are confidential data and information such as application tokens required for authentication, passwords, and SSH (Secure Shell Keys). The problem is not the secrets, but where you store them. If you are using a simple text or word file or SCMs like Git, then the secrets can be easily exposed. Use vaults for storing all your secrets and refer them inside configuration files instead of the secrets. Opensource tools such as truffleHog, git-sercrets, GitGaurdian and similar can be utilized to detect such vulnerable management of secrets
- Version control - Version control is the practice of tracking and managing changes to software code. Ensure all the changes to the IaC is tracked with the right set of information that helps in any revert operation. The important part is that you're checking in those changes alongside the features they support and not separately. A feature's infrastructure requirements should be a part of a feature's branch or merge request. Opensource tools such as Git, GitHub, Bitbucket etc. can be used as the source code version control system
- Principle of least privilege - define the access management policies based on the principle of least privilege with the following priority items: Defining who is and is not authorized to create/update/run/delete the scripts and inventory.

- Limiting the permissions of authorized IaC users to what is necessary to perform their tasks. The IaC scripts should ensure that the permissions granted to the various resources it creates are limited to what is required for them to perform their work.
- Static analysis - Analyzes code in isolation, identifying risks, misconfigurations, and compliance faults only relevant to the IaC itself. Tools such as kubescan, Snyk, Coverity etc, can be leveraged for static analysis of IaC
- Open Source dependency check - Analyzes the open source dependencies such as OS packages, libraries etc to identify potential risks. Tools such as BlackDuck, Snyk, WhiteSource Bolt for GitHub, and similar can be leveraged for open source dependency analysis of IaC
- Container image scan - Image scanning refers to the process of analyzing the contents and the build process of a container image in order to detect security issues, vulnerabilities or potential risks. Opensource tools such as Dagda, Clair, Trivy, Anchore etc can be leveraged for container image analysis CI/CD pipeline and Consolidated reporting - enabling the security checks to be made available in the CI/CD pipeline enables the analysis of each of the code changes, excludes the need for manual intervention, enables maintaining the history of compliance. Along with consolidated reporting, these integrations enhance the speed of development of a secure IaC codebase. Opensource tools such as Jenkins etc can be leveraged to build the CI/CD pipelines and defect dojo, glue can help in tying the checks together and visualizing the check results in a single dashboard
- Artifact signing - Digital signing of artifacts at build time and validation of the signed data before use protects that artifacts from tampering between build and runtime, thus ensuring the integrity and provenance of an artifact. Opensource tools such as TUF helps in the digital signing of artifacts

Deploy

- Inventory management:
  - Commissioning - whenever a resource is deployed, ensure the resource is labeled, tracked and logged as part of the inventory management
  - Decommissioning - whenever a resource deletion is initiated, ensure the underlying configurations are erased, data is securely deleted and resource is completely removed from the runtime as well as from the inventory management
  - Tagging - It is essential to tag cloud assets properly. During IaC operations, untagged assets are most likely to result in the ghost resources that make it difficult to detect, visualize, and gain observability within the cloud environment and can affect the posture causing a drift. These ghost resources can add to the billing, make it difficult to maintain, and affect the reliability. The only solution to this is careful tagging and monitoring for untagged resources.

- Dynamic analysis - Dynamic analysis helps in evaluating any existing environments and services that it will interoperate with or run on. This helps in uncovering potential risks due to the interoperability. Opensource tools such as ZAP, Burp, GVM etc, can be leveraged for dynamic analysis

Runtime

- Immutability of infrastructure - The idea behind immutable infrastructure is to build the infrastructure components to an exact set of specifications. No deviation, no changes. If a change to a specification is required, then a whole new set of infrastructure is provisioned based on the updated requirements, and the previous infrastructure is taken out of service as it is obsolete.
- Logging - Keeping a record is a critical aspect to keep an eye on risks. You should enable the logging - both security logs and audit logs while provisioning infrastructure as they help assess the security risks related to the sensitive assets. It also assists in analyzing the root cause of the incidents. It also helps in identifying potential threats. Opensource tools such as ELK etc, can be leveraged for static analysis of IaC
- Monitoring - Continuous monitoring assists in looking out for any security and compliance violations, helps in identifying attacks and also provides alerts upon such incidents. Certain solutions also incorporate new technologies like AI to identify potential threats early. Opensource tools such as Prometheus, Grafana etc. can be leveraged for static analysis of IaC
- Runtime threat detection: Implementing a runtime threat detection solution helps in recognizing unexpected application behavior and alerts on threats at runtime. Opensource tools such as Falco etc. can be leveraged for runtime threat detection. Certain application such as Contrast (contrast-community-edition) can also detect OWASP Top 10 attacks on the application during runtime and help block them inorder to protect and secure the application.

References

- Securing Infrastructure as code: https://www.opcito.com/blogs/securing-infrastructure-as-code
- Infrastructure as code security: https://dzone.com/articles/infrastructure-as-code-security
- Shifting cloud security left with infrastructure as code: https://securityboulevard.com/2020/04/shifting-cloud-security-left-with-infrastructure-as-code/