National Defense ISAC

How to Protect Cloud-native Applications

April 3, 2023

Authors:
Raul Barreras
Terence Ho
Levi Lyons
Waldemar Pabon
Andrew Zuehlke

Reviewers:
Allan Jacob
Will Jimenez
Renee Stegman

About the Authors

Dr. Waldemar Pabon, ND-ISAC Lead & Senior Contributor

Dr. Waldemar Pabon is a Cyber Security Architect with over 28 years of experience in software engineering. Dr. Pabon leads the Application Security Working Group, Software Security Automation, and the COTS Software Assessments Subgroups at the ND-ISAC. Under his leadership, the Software Security Automation Working Group has published four white papers. The papers provide ND-ISAC members and the industry with a roadmap on how to adopt application security best practices while leveraging automation as a catalyst to achieve efficiencies. Dr. Pabon has a Doctor of Science in Cybersecurity degree from Capitol Technology University.

Raul Barreras, ND-ISAC Contributor

Raul is an Information Security Professional with more than 20 years of experience. His multiple roles throughout his career include information security officer, systems administrator, developer, teacher, and occasional pen tester. In recent years, Raul has had the opportunity to use the accumulated experience in a new role: application security. This role has allowed him to discover how much he enjoys being a developer advocate and how fun and useful it is to build a community of security champions while improving the quality of the organization's software. This is Raul’s second opportunity to contribute to application security papers at the ND-ISAC.

Terence Ho, ND-ISAC Contributor

Terence Ho is a Cloud Application Cybersecurity Specialist with 4 years of experience in the Information Security field with a primary focus on developing automation, integrating security tooling, testing, and standardizing innovative solutions to enhance the security of applications in the Enterprise. He joined ND-ISAC in December 2022 and became a member of the Application Security working group to learn more from the Information Security community and contribute his expertise to the group. Terence graduated from the University of Washington with a Bachelor’s Degree in Computer Science and Software Engineering with a focus on Information Assurance and Cybersecurity.

Levi Lyons, ND-ISAC Contributor

Levi serves as the lead engineer of Attack Surface Management which comprises of web application scanning, vulnerability scanning, external attack surface monitoring, and security assessment orchestration. Levi started as a system administrator before moving to the security team where he took on the role of vulnerability scanner administrator and eventually internal subject matter expert on vulnerability management. Levi Lyons graduated in 2014 with a Bachelor of Science Degree in Information Systems and Cybersecurity.

Andrew Zuehlke, ND-ISAC Senior Contributor

Andrew Zuehlke is a Cyber Security Architect with six years of experience in the information security field. Andrew graduated from Appalachian State University in 2017 with a Bachelor of

Science Degree in Computer Science and Computational Mathematics. In his former role, Andrew served as the lead administrator of SIEM and EDR solutions. In early 2020, Andrew moved to his current role as Cyber Security Architect, primarily supporting Research & Development and Information Technology. Since joining the ND-ISAC in December 2020, Andrew has become a member of the Application Security working group as well as the Cloud Security and Architecture Working Group. Andrew has co-authored multiple white papers with ND-ISAC's Application Security Working Group.

Executive Summary

Cloud-native applications encompass a new approach on how software is built, deployed, and managed in cloud computing environments (Amazon, n.d.). Cloud-native applications provide scalability and efficiency to meet customer demands. With the increased adoption of cloud technologies, many organizations have begun moving applications to the cloud and creating cloud-native applications. While Software Development Lifecycle (SDLC) security controls can help with securing cloud applications, these controls neglect the infrastructure layer of cloud-native applications and the unique risks presented by applications deployed in the cloud. The cloud architecture for applications, Open Worldwide Application Security Project (OWASP) top 10 cloud-native risks, and DevSecOps, help to understand and address these unique security concerns.

1) Cloud Architecture Overview for Applications - the use of microservices, containers, service meshes, declarative Application Programming Interfaces (API), and immutable infrastructure-enabled building blocks of scalable and resilient capabilities. Cloud offerings such as Software as a Service (SaaS), Platform as a service (PaaS), Infrastructure as a service (IaaS) applications. Cloud application implementations such as Cloud-enabled applications, containerized applications, microservices and the Zero Trust Model adapted for the cloud environment.
2) OWASP Top 10 Cloud-native Risks - most common application vulnerabilities that occur in cloud-native applications.

3) DevSecOps Security Concerns - shifting security left to establish additional security guardrails for cloud-native applications through the use of:
    - Security controls - Infrastructure as Code (IaC) scanning. Implementing traditional methods such as Static Application Security Testing (SAST) and Software Composition Analysis (SCA) to scan for vulnerabilities in code deployed within cloud-native services. Enabling workload monitoring and integrity checks for workloads.
    - Security Standardization - Utilizing security standardization frameworks such as CIS, ISO and NIST to implement the best security practices and develop applications in line with the latest application security standards.

      Security Automation - Leveraging automation to enforce policies in a cloud environment, disabling unwanted configurations, and detecting configuration drift to meet cloud security standardization frameworks recommendations.

The threat landscape is always changing, and cloud technologies are constantly evolving, so it is vital to continuously improve existing security controls such as the use of SAST and SCA in the SDLC and apply additional security safeguards like IaC scanning and additional security automations as cloud-native applications become more widely adopted. Minimizing risk in environments must be achieved through strategic planning and automation throughout the organization. This paper will address the main challenges faced when protecting cloud-native applications.

Table of Content

Introduction

Objective

Cloud technology has opened a world of opportunities when implementing applications. The ability to quickly respond to customer demands for highly efficient features has made cloud environments a perfect ecosystem to meet such demands. The capabilities provided by cloud architectures enable the organization to accelerate the time to market solutions while at the same time having the flexibility to respond to increasing resource needs.

Despite all these benefits, security is still a major area of concern when moving application implementations to the cloud. The vast availability of resources and cloud-native services requires careful consideration when implementing the solutions in cloud environments. Because the infrastructure now relies on services developed and deployed by cloud providers, the shared responsibility model is an important component to define the security implementation strategy.

To provide visibility of the multiple layers of security concerns, the OWASP has developed the “OWASP Cloud-Native Application Security Top 10” to assist organizations to securely deploy applications in the cloud (n.d.). Security misconfigurations, resource integrity, and traditional Software Development Lifecycle (SDLC) concerns are among the wide range of considerations for any organization looking to secure their implementations end-to-end in a cloud environment.

With those concerns in mind, this paper will address the main challenges faced when protecting cloud-native applications. The term cloud-native expands beyond the traditional realm of SDLC practices. It involves not only the application, but new technology components (such as containers) and configuration requirements. Cloud-native applications encompass a new approach to how software is built, deployed, and managed in cloud computing environments (Amazon, n.d.). The need to be able to continuously repeat a secure deployment in an environment where the security responsibility is shared with the cloud provider requires standardization and automation to be at the forefront of the discussion on how to secure cloud-native applications.

Security controls such as Static Application Security Testing (SAST), Software Composition Analysis (SCA), Infrastructure as Code (IaC), and other SDLC security controls are discussed as they provide the perfect continuity of shift-left adoption in cloud computing. This paper will cover anti-tampering protection along with workload monitoring and alerting to complete the broad spectrum of security concerns.

Audience

The audience of this white paper includes cloud architects, security engineers, lead software engineers, product managers, senior managers, and senior executives responsible for the implementation of software security in cloud environments, as well as those managing the risk associated with threat vectors in cloud-native applications.

Structure of the paper

This paper introduces the importance of securing cloud-native applications to minimize risk to the organization. First, a discussion of cloud architecture is covered, followed by the presentation of the most common threat vectors impacting cloud-native applications as defined by OWASP. Next, the security controls needed to ensure cloud-native applications enforce a strong security posture are introduced and discussed. Strategies on how to leverage standardization and automation are finally discussed to ensure a consistent, repeatable, and secure deployment of applications in cloud environments to ensure security and compliance requirements are met.

How to protect cloud-native applications?

Securing cloud-native applications requires a comprehensive approach considering not only the traditional security controls needed to secure applications but also securing resources and configurations in cloud environments. This section will provide a high-level understanding of cloud architectures, and how OWASP is classifying the most common threat vectors for applications in cloud environments, followed by a discussion of standardization and automation to enforce DevSecOps in the cloud.

Cloud Architecture Overview for Applications

A cloud-native application is a software application designed and developed from the start to run in cloud computing environments. These applications are built using a set of principles and practices that maximize the benefits of cloud computing, such as scalability, elasticity, security, and resilience.

While traditional applications are developed using a 3-tier, monolithic client-server architecture comprising the presentation, application, and database layers, cloud-native applications will use microservices, containers, service meshes, declarative Application Programming Interfaces, and immutable infrastructure, a loosely coupled collection of building blocks designed to be scalable, resilient, and manageable.

By breaking down an application into smaller, independently deployable microservices, developers can create more agile and flexible applications that can be scaled up or down depending on the demand. Declarative APIs and immutable infrastructure ensure that applications are built using a consistent and repeatable process, making deployment and management easier. Containers, which are lightweight and portable, make deploying and managing microservices easier. At the same time, service meshes provide advanced networking capabilities that allow microservices to communicate with each other securely and reliably. Figure 1 provides an overview of new technologies in cloud environments supporting application implementations.

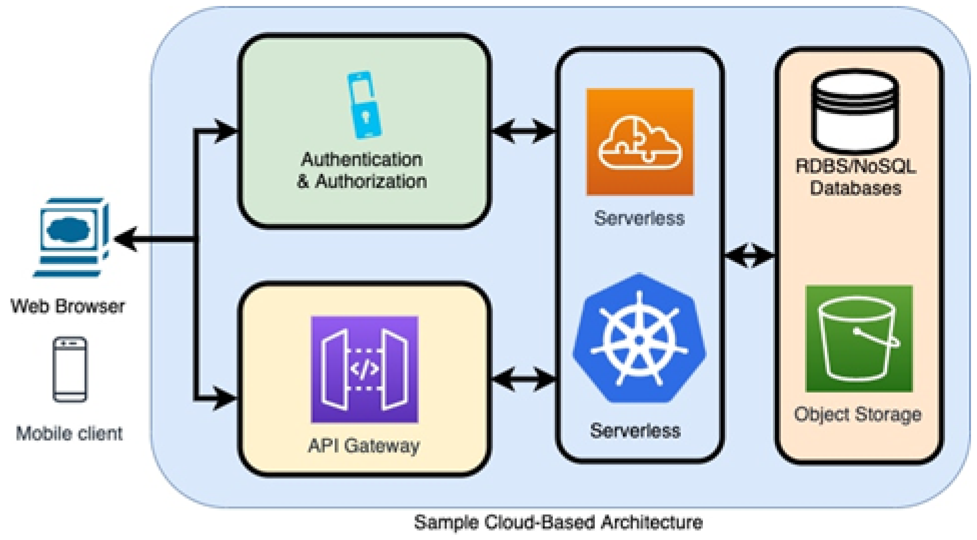


Figure 1 New technologies used in cloud environments to support application implementations. From "Four Architecture Choices for Application Development in the Digital Age", by Saraswathi, 2020, [Digital Image]. https://www.ibm.com/cloud/blog/four-architecture-choices-for-application-development

Shared Responsibility Model

There are three main cloud service models:

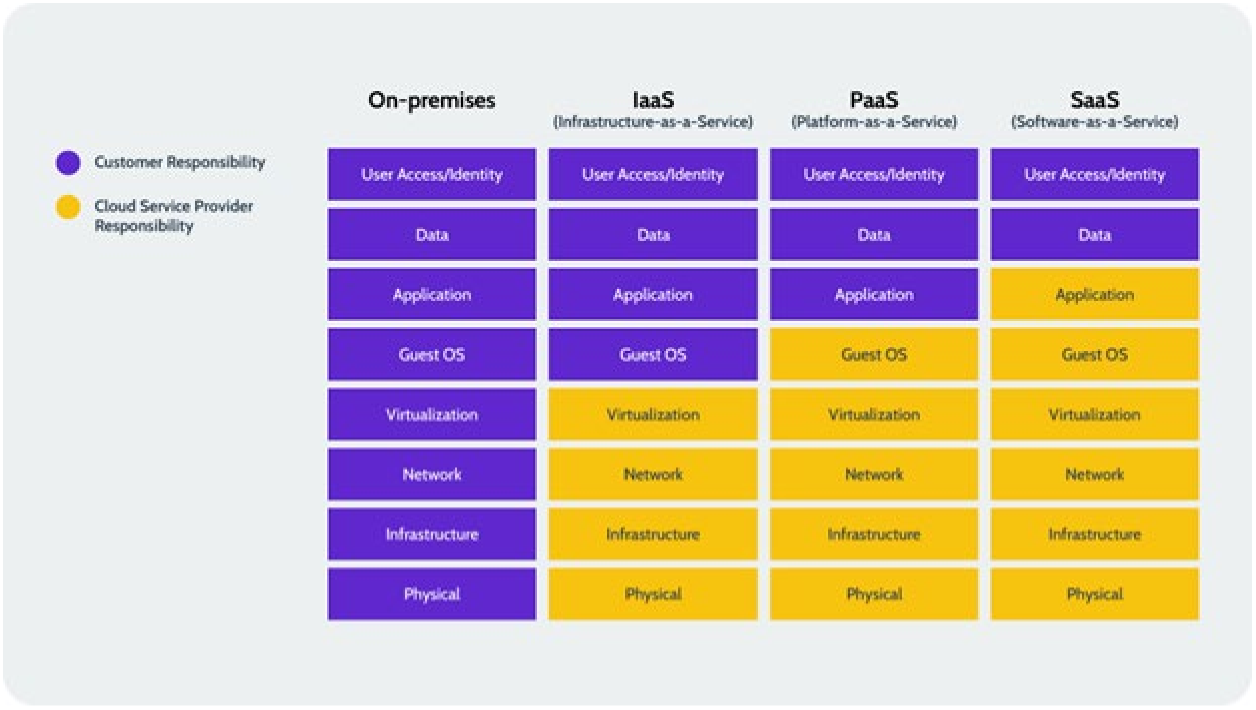
- Software as a service (SaaS)
- Platform as a service (PaaS)
- Infrastructure as a service (IaaS)

Software as a service (SaaS) is a cloud computing model where software applications are provided over the internet. The software provider hosts and maintains the application, manages the underlying infrastructure and handles software updates and upgrades. They are responsible for the application's security, while the user is responsible for its data and granting proper access. Examples of SaaS applications include email, customer relationship management (CRM) software, project management tools, and accounting software.

Platform as a service (PaaS) is a cloud computing model where users can rent and access a complete platform for developing, running, and managing their applications over the internet without worrying about the underlying infrastructure. PaaS platforms typically provide features such as automatic scaling, load balancing, and application monitoring, which can help improve applications' availability, performance, and scalability. In this mode, users are responsible for the application running on the platform and their data. Examples of PaaS are AWS (Amazon Web Services) Elastic Beanstalk, Heroku, and Red Hat OpenShift.

Infrastructure as a service (IaaS) is a cloud computing model where users can rent and access computing infrastructure resources, such as virtual machines, storage, networking, and other computing resources, on a pay-as-you-go basis. The cloud provider hosts and manages the underlying physical infrastructure, such as servers, storage devices, and networking equipment. Users control those virtual machines' operating systems, applications, and other software. Some IaaS providers include Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and Oracle Cloud.

The following figure shows how the Customer and the Cloud Service Provider (CSP) share the responsibilities, comparing the previous models with the on-premises model. CSPs are responsible for the *Security of the Cloud* and customers are responsible for *Security in the Cloud*. It is essential to highlight that the client is always responsible for protecting their data. Figure 2 provides an overview of the responsibility breakdown based on the shared responsibility model.



🟪 Customer Responsibility

🟨 Cloud Service Provider Responsibility

On-premises	IaaS (Infrastructure-as-a-Service)	PaaS (Platform-as-a-Service)	SaaS (Software-as-a-Service)
🟪 User Access/Identity	🟪 User Access/Identity	🟪 User Access/Identity	🟪 User Access/Identity
🟪 Data	🟪 Data	🟪 Data	🟪 Data
🟪 Application	🟪 Application	🟪 Application	🟨 Application
🟪 Guest OS	🟪 Guest OS	🟨 Guest OS	🟨 Guest OS
🟪 Virtualization	🟨 Virtualization	🟨 Virtualization	🟨 Virtualization
🟪 Network	🟨 Network	🟨 Network	🟨 Network
🟪 Infrastructure	🟨 Infrastructure	🟨 Infrastructure	🟨 Infrastructure
🟪 Physical	🟨 Physical	🟨 Physical	🟨 Physical

Figure 2 Shared Responsibility Model per type of cloud architecture. From "The Shared Responsibility Model and SaaS, Explained", by Ciesielski, 2023, [Digital Image]. https://rewind.com/blog/shared-responsibility-model-saas-explained/

Sometimes, the Cloud Service Provider (CSP) and the customer share security responsibilities. For instance, the CSP will provide an Identity and Access Management service where the customer is responsible for creating the groups the application needs and assigning proper permissions.

Types of Implementations

Cloud-enabled applications

Traditional applications can be deployed on the cloud, using IaaS which performs the function of the on-premises computing center. These applications are called *cloud-enabled applications*. Traditional apps and services typically require a virtual machine to run (IBM, n.d.).

Containerized applications

Developers can use containers to encapsulate and distribute the application's architectural blocks, such as the application servers, database servers, and web servers. Containers will provide a high level of isolation between the application and the host system, which helps to ensure that the application runs consistently on any host with a container runtime.

A container runtime, such as Docker, is a piece of software responsible for creating and managing the lifecycle of containers, including starting, stopping, pausing, and restarting containers as needed. The runtime also provides isolation between containers, ensuring each container has its dedicated resources and cannot interfere with other containers running on the same host system.

Container runtimes are part of a container engine, also known as a container platform or container management system. Container engines provide tools and services for building, deploying, and managing containerized applications and a variety of other services and tools, such as:

- Image registries for storing and distributing container images.
- Container orchestration tools for managing and scaling containerized applications across multiple hosts.
- Networking and storage plugins for managing container networking and storage resources.
- Security and monitoring tools for ensuring the security and performance of containerized applications.

Examples of container engines include Docker, Kubernetes, and Amazon Elastic Container Service (ECS).

Microservices

Another architectural solution is to arrange the application as a collection of loosely coupled, fine-grained services called microservices. These microservices communicate with each other through lightweight protocols. With this architectural model, developers can develop and deploy their services independent of other services, reducing dependencies and making it easier to make changes: a change to a small part of the application only requires rebuilding and redeploying only one or a small number of services (Fowler, & Lewis, 2014).

Kubernetes is a popular platform for building and deploying applications based on microservices architecture. It is an open-source container orchestration system that automates containerized applications' deployment, scaling, and management across multiple hosts. Kubernetes has emerged as the industry standard for container orchestration and continues to gain widespread adoption as more organizations embrace cloud-native application development practices. To cater to the growing demand, major cloud providers such as Amazon, Microsoft, Google, and IBM offer their Kubernetes implementations as a service, including Amazon Elastic Kubernetes Service (EKS), Microsoft Azure Kubernetes Service, Google Kubernetes Engine, and IBM Cloud Kubernetes Service. These services allow organizations to leverage the power of Kubernetes without managing the underlying infrastructure, making it easier to deploy and manage applications based on microservices at scale.

Zero Trust

The conventional method of securing a network assumes that a user, device, or application that has been given access to the network can be trusted to access any resource on that network based on its physical or network location. However, this approach is no longer effective in protecting against modern cyber threats. Zero Trust is an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources (National Institute of Standards and Technology, 2018).

In a Zero Trust model, all users, devices, and applications must undergo verification and authentication before accessing any resource: "never trust, always verify". The core principles of Zero Trust are (Microsoft, n.d.):

- Verify explicitly: Always authenticate and authorize based on all available data points. Use strong authentication (MFA) and verify that every device meets security requirements.
- Use least privilege access: Limit user access with Just-In-Time and Just-Enough-Access (JIT/JEA), risk-based adaptive policies, and data protection.
- Assume breach: Minimize blast radius and segment access. Verify end-to-end encryption and use analytics to get visibility, drive threat detection, and improve defenses. Encrypt your data wherever it resides.

OWASP Top 10 Cloud-native Risks

The Open Worldwide Application Security Project, often abbreviated OWASP, is a nonprofit foundation led by community open-source software projects and members. The foundation helps developers and security professionals across all industries to improve software security. OWASP initially gained popularity due to their "Top 10 Web Application Security Risks," a list of the most common application vulnerabilities, highlighting areas on which developers should focus on. In 2021, OWASP established a new top ten list, "Cloud-Native Application Security Top 10," (n.d.) focused on cloud-native applications. This targeted list is intended for organizations looking to implement a new—or mature and existing— secure cloud-native application strategy.

OWASP's "Cloud-Native Application Security Top 10" continues to be updated as needed; as a result, the list may change from year to year, either in the type of risk or the order in which the risks appear. To help address these risks being so prevalent, multiple vendors and open-source projects have developed cloud security scanners that can scan configuration files to identify these risks before they are deployed into production. The following are the top 10 risks as of March 2023, with a high-level summary of each below:

- Insecure cloud, container, or orchestration configuration
- Injection flaws (app layer, cloud events, cloud services)

- Improper authentication & authorization
- CI/CD pipeline & software supply chain flaws
- Insecure secrets storage
- Over-permissive or insecure network policies
- Using components with known vulnerabilities
- Improper assets management
- Inadequate ‘compute’ resource quota limits
- Ineffective logging & monitoring (e.g., runtime activity).

Insecure cloud, container, or orchestration configuration

The first and most prominent risk for cloud-native applications is insecure configurations. Insecure configurations can be present in a variety of situations, ranging from containers running as a super “root” user to the unintentional exposure of data by utilizing public instead of private storage settings.

Injection flaws (app layer, cloud events, cloud services)

Just as standard applications are susceptible to a multitude of injection-related vulnerabilities, so too are cloud-native applications. Services that are publicly exposed are more vulnerable to attacks such as SQL injection. Injection flaws like OS Command injection can allow an attacker to execute arbitrary operating system commands on the underlying server running the application.

Improper authentication & authorization

With the complexity that comes with cloud identity management, it is not uncommon for access and authorizations to be improperly assigned. A mistake such as an overly permissive cloud IAM role could allow a user to have access to resources they otherwise should not. An unauthenticated API could allow an attacker access to data without having to provide any credentials.

CI/CD pipeline & software supply chain flaws

Flaws in the continuous integration and continuous development (CI/CD) pipeline can be the entry point for an attacker. For example, a lack of proper authentication and authorization on the pipeline applications may allow an attacker to manipulate code and push it into production without proper checks and balances.

Insecure secrets storage

Secrets can hold the keys to an application; consequently, the improper and insecure storage of keys can lead to secrets being viewed by an attacker. The more common scenarios in which secrets are unintentionally exposed are secrets, keys, or passwords that are stored unencrypted or hardcoded into the application’s code.

Over-permissive or insecure network policies

The importance of secure network polices for a cloud-native application cannot be overstressed. The risk of not monitoring or blocking potentially malicious domains could lead to malicious traffic hitting an application. However, internal network policies are equally critical to securing an application; inadequate segmentation could allow an attacker to access resources intended to be internally accessible only while unencrypted communication channels could allow an attacker to gather potentially sensitive information or even perform a man-in-the-middle attack.

Using components with known vulnerabilities

An application is only as secure as its weakest link. Just as with standard applications, cloud-native applications often fall victim to vulnerabilities that exist in third-party packages that are imported. Standard code scanning tools in the Software Development Life Cycle (SDLC) can often catch risks in this category.

Improper assets management

Knowing what exists in an environment is one of the essential aspects of securing an environment. Improper asset management—or a lack of asset management all together—can lead to undocumented APIs and services.

Inadequate 'compute' resource quota limits

While dynamic resource scaling is a critical component of cloud computing, applications without controls in place to limit this scaling could lead to excessive resource usage. Similarly, not having enough resources available for scaling could allow an attacker to perform a denial-of-service (DOS) attack by overloading the system.

Ineffective logging & monitoring (e.g., runtime activity)

Proper logging not only enables an organization to proactively monitor activity and performance of an application to prevent issues from happening, but it can also be critical to a successful forensics' investigation in a scenario where suspicious activity was detected.

DevSecOps Security Concerns

The adoption of a shift-left approach along with standardization and automation capabilities provides a comprehensive roadmap toward the enforcement of secure implementations of cloud-native applications. The need to establish security guardrails will inevitably require the implementation of traditional security controls to ensure early detection of potential risks. As cloud-native applications are deployed within the purview of cloud services, securing the configuration of the application and the cloud services becomes a key combination of any successful implementation.

Having the ability to successfully repeat a consistent secure implementation will require the addition of Infrastructure as Code (IaC) along with automation to continuously improve and achieve efficiencies during the deployment and management of cloud-native applications. Policy and compliance guardrails will require the adoption of monitoring and integrity checks to ensure business regulatory requirements are met in a standard fashion. The next sections in the paper provide a broad discussion on how to integrate all these components to secure cloud-native applications. Specifically, the paper will focus the discussion on three key areas:

- Security controls
- Standardization
- Automation

Security controls available to protect cloud-native apps

Cloud-native applications operate in a different technology landscape than traditional applications. In a traditional sense, the organization creating software has full control of the code to be deployed. When dealing with cloud-native applications, that is not necessarily the case. Cloud providers have different native services intended to support the implementation of features without the need to create any component by the organization. This reality brings to the forefront the discussion of shared responsibility model, which is discussed as part of the section Cloud Architecture Overview for Applications. In essence, there are several areas of interest when securing cloud-native services:

- Scanning IaC code
- Implementing traditional SAST/SCA to scan for vulnerabilities in code deployed in cloud-native services
- Workload monitoring
- Integrity checks for workloads

Scanning IaC code

In a shared responsibility model, both the organization and the cloud provider will need to ensure the implementation of native services leveraged by organizations is secure. From a traditional security controls perspective in the Software Development Lifecycle (SDLC), testing tools such as Static Application Security Testing (SAST), Software Composition

Analysis (SCA), etc. are not going to come into play for the organization from a responsibility perspective in terms of the cloud-native services created by the cloud providers. Figure 3 provides a summary of the security controls traditionally needed to secure an application. As they implement the code that will support the cloud-native services, the cloud providers will have to ensure SAST, SCA, etc. are implemented as part of their secure Software Development Lifecycle (SDLC) to enforce a shift-left and minimize the risk of potential software weaponization.

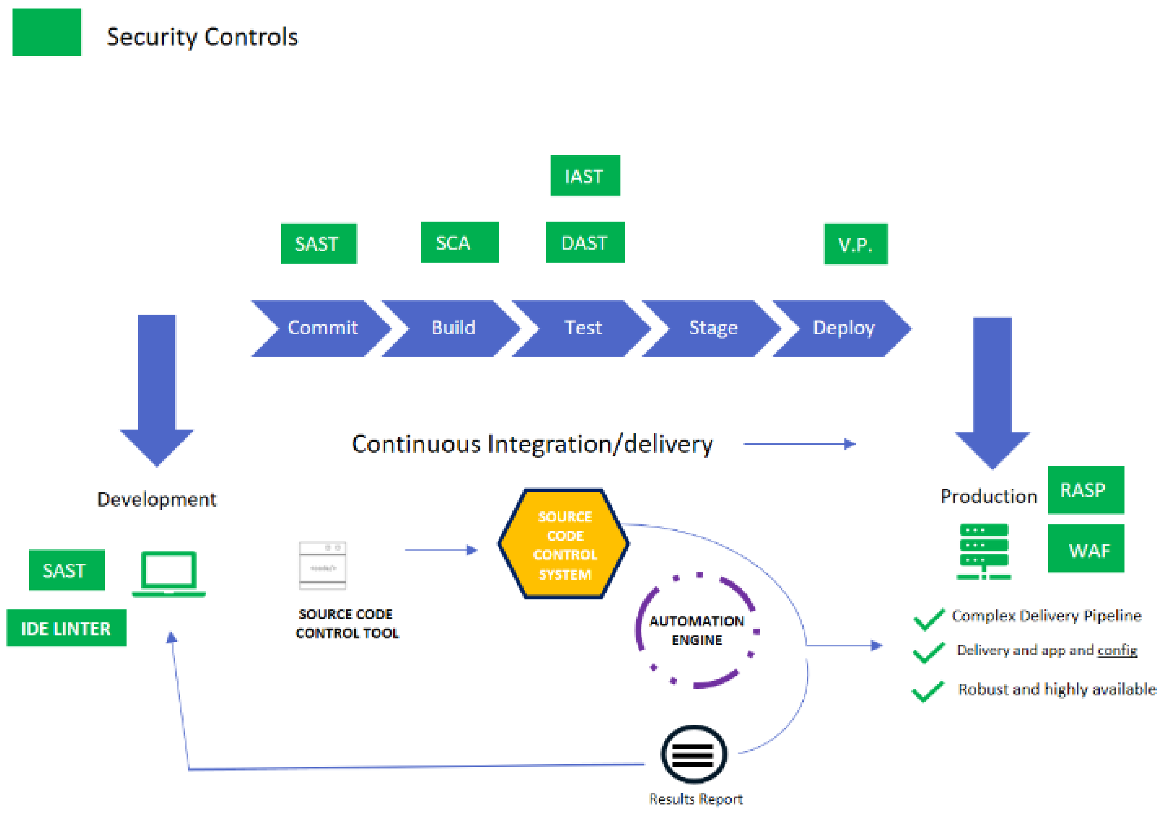


Figure 3 Security controls needed in the SDLC. From "Software Security Automation: A Roadmap toward Efficiency and Security", by Heim, Keim, Munsch & Pabon, 2020, [Digital Image]. https://ndisac.org/wp-content/uploads/ndisac-security-automation-white-paper.pdf

This leads the discussion to the configuration needed to implement the solution in the cloud environment. The configuration of the cloud-native services is the responsibility of the organization. Since the configuration very often will need to ensure all potential scenarios are covered, cloud providers tend to start with a wide-open stance with the understanding that organizations will harden the configuration as needed. That is where the risk lies; the lack of understanding on how to securely configure those services could expose the organization to unnecessary risk. Consider a scenario where a cloud provider provides a Network Firewall as a service. The Network Firewall is intended to protect part of network traffic but is configured "out-of-the-box" to enable wide access to the application(s) and is only effective if the organization provides the right Access Control rules. Therefore, the organization becomes responsible for enforcing the rules needed to secure the network.

This reality creates an inherent risk to the organization. If multiple configurations are needed for multiple network firewalls, the organization will need to establish a common configuration that can serve as a standard secure baseline. Performing this configuration process manually over and over again would not only create severe overhead but risk. Organizations should consider the use of templates as an approach to standardize configurations and facilitate automation by leveraging Infrastructure as Code (IaC). IaC enables organizations to deploy infrastructure using configuration files which facilitates a standard baseline across the board for all implementations (National Institute of Standards and Technology, 2023).

To properly secure code used associated with IaC, a security control similar to the SAST will have to be deployed as close as possible to the developers responsible for creating the code. This security control can evaluate the IaC code and determine the existence of security misconfiguration that could pose a risk to the implementation. Just like SAST, this security control can be embedded in the Integrated Development Environment (IDE) as a plugin as well as work in a DevOps pipeline through either a plugin or a Command Line Interface (CLI). In a DevOps pipeline scenario, the security control can break the execution of the stages in the DevOps pipeline and report back the findings identified as part of the scanning. Automation is a key component of the IaC security control in DevOps. For details about automation, please see the section Security Automation which provides a broader discussion.

Implementing traditional SAST/SCA to scan for vulnerabilities in code deployed in cloud-native services

Moving the discussion away from IaC and focusing on native cloud services (for example, serverless functions) used by applications, some cloud providers will allow adding code and libraries or frameworks to support the implementation. In such cases, the code as well as the dependencies used with the native cloud services by the organization would need to follow a traditional approach of leveraging common security controls such as SAST and SCA. The SAST tool will help identify any potential threat vector included in code snippets. The

SCA tool will help understand any risk associated with any dependency included as part of the organization's implementation of native cloud services.

With the use of SAST and SCA for the code deployed into cloud-native services, we must remember that the same principles applied to the SDLC will apply. Security vulnerabilities identified by security controls will need to be recorded as part of the issue tracker. A remediation workflow must be established to ensure proper planning and remediation are exercised. Finally, security testing becomes important to ensure security vulnerabilities previously identified are remediated. Regardless of whether we are dealing with IaC or cloud-native services, security controls are going to be an important component when securing applications in cloud environments.

Workload Monitoring

Cloud environments also offer special workload capabilities such as containers. The use of container images will require the verification of the dependencies with SCA, and the workload runtime will need to ensure that security is embedded using a shift-left approach. Establishing policies to validate workloads are following security best practices (for example, workloads not intended to be public are configured to allow public IP address access) must be enforced through continuous monitoring of security misconfiguration.

The second area of concern is associated with monitoring the activity executed against workloads. Ensuring logs are configured, activity is monitored, and runtime security policies are enforced will enable the organization to be proactive in the response to potential malicious behaviors in the application implementation. Cloud providers offer services to not only log data but also alert when special instances of deviations are detected.

Integrity checks for workloads

Just like in on-premises implementations, organizations need to ensure attackers are not able to tamper with the application implementation. Countering this concern will require the implementation of anti-tampering controls. Organizations need to pay special attention to how they verify the integrity of workloads and the means of operation for an application/implementation in the cloud. Utilizing code signing with their artifacts will provide a much-needed integrity check which will enforce a non-repudiation mechanism.

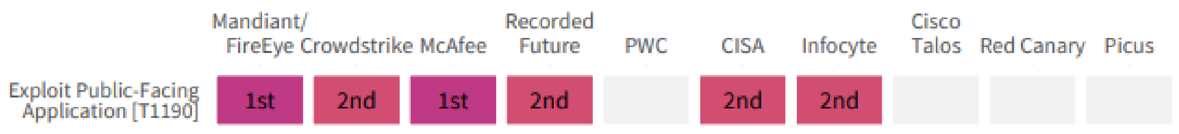
Security Standardization

According to a 2021 report from F5 Labs (2021):

- 56% of the largest incidents in the previous 5 years were associated with a web application security issue.
- The average time-to-discovery for incidents involving web application exploits is 254 days.
- Web application attacks were the #1 way leading to a data breach incident.

- Exploit Public-Facing Application is the #1 or #2 reported technique for Initial Access among varying security vendors.

Figure 4 provides a breakdown of the top initial access techniques associated with exploits in web applications.



	Mandiant/ FireEye	Crowdstrike	McAfee	Recorded Future	PWC	CISA	Infocyte	Cisco Talos	Red Canary	Picus
Exploit Public-Facing Application [T1190]	1st	2nd	1st	2nd		2nd	2nd			

Figure 4 Top Initial Access Techniques According to Multiple Sources. From "The State of the State of Application Exploits in Security Incidents", by F5 Labs, 2021, [Digital Image]. https://www.f5.com/content/dam/f5-labs-v2/article/pdfs/The-State-of-the-State-of-Application-Exploits-in-Security-Incident-F5Labs-rev22JUL21.pdf

A different report from security firm Rapid7 (2021) suggests that 50% of the vulnerabilities they monitored were exploited within seven days of public disclosure and the average time to known exploit was 12 days. Whereas only 30% were exploited within seven days of public disclosure and the average time to known exploits was 42 days the year prior, which is a 71% decrease in time. With adversarial temptation being relatively high due to the public-facing nature of many web applications, paired with increasingly shortened time from public disclosure to active exploit, organizations need to consider adopting a security standardization framework to help implement best application security practices and develop applications that are in line with the latest application security standards to ensure maximum security and protection of users.

Today there are several standardization frameworks which are created with organizations in mind to help identify and remove application security vulnerabilities in complex software systems. Some organizations such as CIS, ISO, and NIST offer up Application Security recommendations as part of a broader set of controls that govern IT security. Other organizations such as OWASP are popular because they have a framework specifically designed for web applications.

The Center for Internet Security (CIS) is a nonprofit organization that helps develop, validate, and promote timely best-practice solutions that help people, businesses, and governments protect themselves against pervasive cyber threats. The CIS Application Software Security Control spans 14 safeguards (Center for Internet Security, n.d.):

- 16.1: Establish and Maintain a Secure Application Development Process
- 16.2: Establish and Maintain a Process to Accept and Address Software Vulnerabilities
- 16.3: Perform Root Cause Analysis on Security Vulnerabilities
- 16.4: Establish and Manage an Inventory of Third-Party Software Components
- 16.5: Use Up-to-Date and Trusted Third-Party Software Components
- 16.6: Establish and Maintain a Severity Rating System and Process for Application Vulnerabilities

- 16.7: Use Standard Hardening Configuration Templates for Application Infrastructure
- 16.8: Separate Production and Non-Production Systems
- 16.9: Train Developers in Application Security Concepts and Secure Coding
- 16.10: Apply Secure Design Principles in Application Architectures
- 16.11: Leverage Vetted Modules or Services for Application Security Components
- 16.12: Implement Code-Level Security Checks
- 16.13: Conduct Application Penetration Testing
- 16.14: Conduct Threat Modeling

The Application Normative Framework under International Organization for Standardization (ISO) 27034 is a list of application security controls which are applied with a targeted level of trust in mind, which ISO defines as "a set of Application Security Controls deemed necessary by the application owner to lower the risk associated with a specific application to an acceptable (or tolerable) level" (International Organization for Standardization, n.d.).

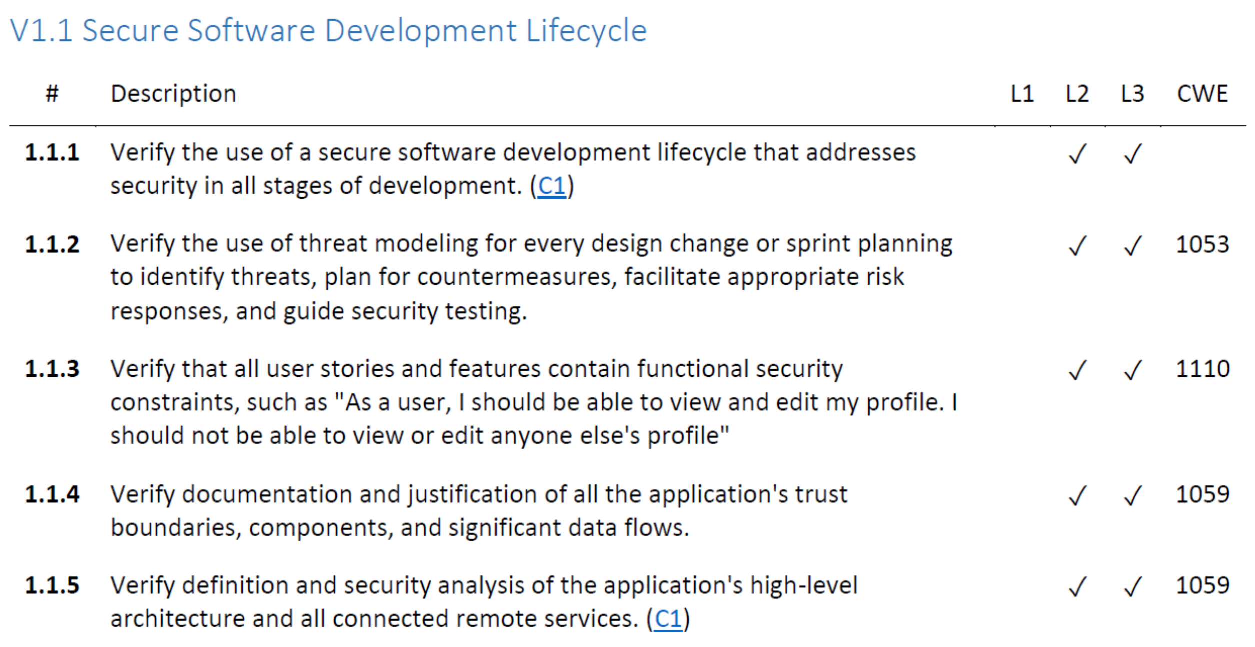
The National Institute of Standards and Technologies (NIST) provides various standards and frameworks for cybersecurity, including NIST Special Publication (SP) 800–218: Secure Software Development Framework (SSDF) Version 1.1. NIST Special Publication (SP) 800–

218 can also help with the creation of high-level objectives and covers the following safeguards (National Institute of Standards and Technology, 2022):

- Organizations should ensure that their people, processes, and technology are prepared to perform secure software development.
- Organizations should protect all components of their software from tampering and unauthorized access.
- Organizations should produce well-secured software with minimal security vulnerabilities in its releases.
- Organizations should identify residual vulnerabilities in their software releases and respond appropriately to address those vulnerabilities and prevent similar ones from occurring in the future.

The Open Worldwide Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software and may be considered one of the more popular options due to its focus specifically on web applications. The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and provides developers with a list of requirements for secure development along the way (OWASP, 2021).

The OWASP Application Security Verification Standard (ASVS) spans 14 safeguards with 3 Application Security Verification Levels with a required checklist to ensure that key actions are completed to receive a verification level. Figure 5 provides an example of the list of Application Security Verification Levels.



V1.1 Secure Software Development Lifecycle

#	Description	L1	L2	L3	CWE
1.1.1	Verify the use of a secure software development lifecycle that addresses security in all stages of development. (C1)		✓	✓	
1.1.2	Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing.		✓	✓	1053
1.1.3	Verify that all user stories and features contain functional security constraints, such as "As a user, I should be able to view and edit my profile. I should not be able to view or edit anyone else's profile"		✓	✓	1110
1.1.4	Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.		✓	✓	1059
1.1.5	Verify definition and security analysis of the application's high-level architecture and all connected remote services. (C1)		✓	✓	1059

Figure 5 Checklist example. From "Application Security Verification Standard 4.0.3", by OWASP, 2021, [Digital Image]. https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

Verification Levels:

- ASVS Level 1 is for low assurance levels and is completely penetration testable - An application achieves ASVS Level 1 if it adequately defends against application security vulnerabilities that are easy to discover and included in the OWASP Top 10 and other similar checklists.

- ASVS Level 2 is for applications that contain sensitive data, which requires protection and is the recommended level for most apps.
- An application achieves ASVS Level 2 (or Standard) if it adequately defends against most of the risks associated with software today.
- ASVS Level 3 is for the most critical applications - applications that perform high-value transactions, contain sensitive medical data, or any application that requires the highest level of trust - This level is typically reserved for applications that require significant levels of security verification, such as those that may be found within areas of military, health and safety, critical infrastructure, etc.

Safeguards:

- Architecture, Design, and Threat Modeling
- Authentication
- Session Management
- Access Control
- Validation, Sanitization, and Encoding
- Stored Cryptography
- Error Handling and Logging
- Data Protection
- Communication
- Malicious Code

- Business Logic
- Files and Resources
- API and Web Service
- Configuration

Many organizations can benefit from adopting a standardization framework as a means of creating and maintaining secure software.

Security Automation

Maintaining a consistent and compliant security posture in cloud environments and on-premises requires the enforcement of a set of standard configurations and security policies. Statistics show many organizations are still enforcing a standard configuration using manual execution without leveraging automation (Wallgren, 2017). This approach minimizes the efficiency gains organizations can achieve by leveraging technological innovations. To complicate the security landscape, cloud environments tend to have an open stance configuration by default. For example, when a cloud tenant (IaaS, PaaS) gets created, all the resources defined are not assuming a default deny stance and expose resources without public and private considerations.

To be compliant and secure, organizations will need to adopt a proactive approach toward standardizing how resources are deployed, configured, and tested in a cloud environment.

Employing security automation leveraging standardization will guarantee a consistent, secure, and compliant configuration will be enforced every time applications are deployed. When automating security, organizations will need to capture all potential scenarios to properly define what to look for:

- internet-facing resources.
- private resources.

This breakdown will enable organizations to differentiate when resources used by an application need to enforce a more restrictive approach because they are private than when they are intended to be used by anyone on the internet. Let’s consider a scenario to understand the importance of security automation.

Cloud providers offer several capabilities to restrict how resources are used and protected. Security groups, Network Access Control Lists, and API Gateways are just a few examples of resources available to organizations when enforcing a strong security stance. As an example, any resource deemed internal will have to ensure a quad zero (0.0.0.0/0) configuration is never included in Security Groups. Quad zero configurations enable any resource to be accessible from anywhere (assuming there are no other compensating controls in place).

In a scenario where a database is used as part of an application, not paying attention to the quad-zero configuration could expose the data to risk. Per security best practices, databases should never be exposed directly to the internet. Having a quad-zero configuration for a database could potentially expose it to the Internet, enabling attackers to attempt to compromise it. To enforce a strong security posture, teams working with the implementation will have to ensure the configuration is always set to prevent internet exposure for database assets. In this scenario, it will be expected for any team implementing an application that is using a database to always ensure the default quad-zero configuration is removed.

Because of human nature, performing a manual process in every single implementation to prevent a data compromise could lead to a potential compromise. There are many scenarios under which relying upon a manual configuration to be consistently repeated could face challenges. Some common scenarios include:

- Members of a team with the required knowledge to protect may leave.
- Subject Matter Experts (SMEs) may forget to adjust the configuration.
- New members of the implementation team may not be security-savvy.
- Default behaviors of cloud resource provisioning may be overlooked.

To minimize the potential impact of overlooking the configuration needed to enforce a strong security posture, organizations need to combine standardization with automation. There are two main trends in terms of how to leverage technology innovation in the automation space to achieve standardization. First, cloud providers offer native services to evaluate the configuration of resources to ensure they are following security best practices. All major cloud providers (AWS, Azure, Google Cloud, etc.) provide services that can leverage tags to determine if resources are not following a compliant deployment configuration. These services can automatically adjust the configuration to ensure a standard deployment across the cloud landscape.

Using the previous database example, such a capability can be used to automate checking that resources tagged as private do not have a quad-zero configuration present. This means that such services can adjust the configuration automatically to always enforce a consistent implementation without any human intervention. Whether the team lost the SMEs or lacked the knowledge to secure the implementation becomes irrelevant; as the environment itself is enforcing a secure posture through the automation of required security configurations for native cloud services. Because of the powerful capabilities provided by hosting services in cloud environments, the organization needs to establish a Change Control Board (CCB) to evaluate not only the initial implementation but also any future changes that could potentially impact how security is enforced in the cloud. This evaluation will ensure automation activities are performing sound and compliant security practices.

The other option available to organizations to enforce standardization through automation is the use of Infrastructure as Code (IaC). The use of IaC has been recognized as a key component of not only the provisioning of resources in technology environments but also a core capability for DevSecOps. Figure 6 provides an overview of the configuration provisioning using IaC to control a standard configuration.

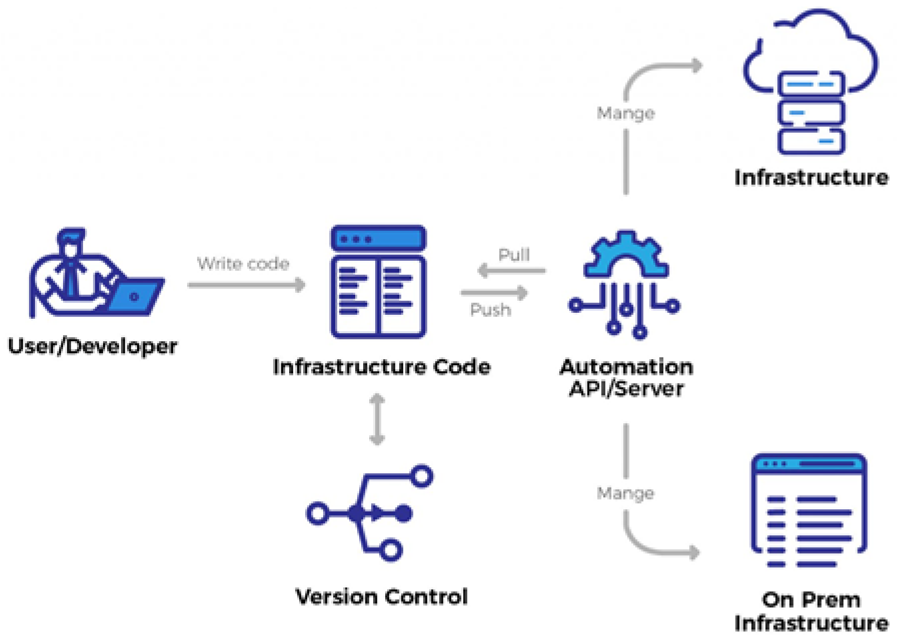


Figure 6 IaC enforcing a standard secure configuration. From “The best Infrastructure as Code tools for 2022” by Valdes, 2022, https://www.clickittech.com/devops/infrastructure-as-code-tools/

The NIST recognizes the importance of IaC as a mechanism to achieve repeatable and consistent configurations (National Institute of Standards and Technology, 2022, p. 19). The use of IaC with automation would ensure in our database example that resources tagged as private are always provisioned without the quad-zero configuration. Just like with the native services, a CCB will be required to review and approve any changes to the IaC code, and to verify any enhancements adhere to the organization's compliance and security requirements.

Now, automation can be as good as the definition that is embedded as part of the IaC code. Therefore, it becomes extremely important to automate the scanning of security misconfigurations in IaC code before including those changes as part of the standard baseline to be utilized when deploying resources for an application implementation in a cloud environment. There are many solutions provided in the industry to scan for security misconfigurations.

The process provided by these tools follows a similar approach to the one associated with Static Application Security Testing (SAST) tools. You can embed the security scanning of IaC as part of the Integrated Development Environment (IDE) to enforce a "shift-left approach" and detect risk as soon as the developer is creating the IaC code. Figure 7 provides an overview of embedding IaC security check through automation. There is also the option of embedding the security checks as part of the DevOps pipeline. Teams can use plugins and Command Line Interfaces (CLIs) to trigger scans as part of a stage in a DevOps pipeline, identify a security risk, and even stop the execution to prevent the potential rollout of the risk to a production environment.

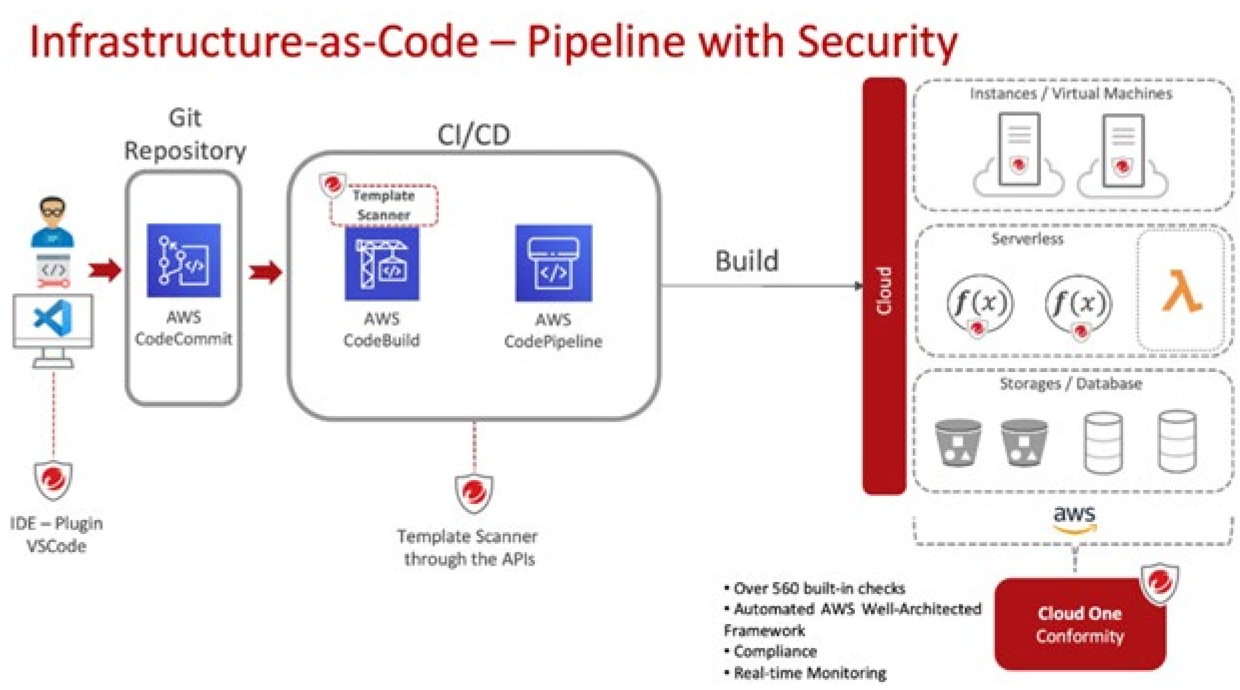


Figure 7 Embedding IaC security checks through automation. From " Building IaC Pipeline on AWS with Security Fully Integrate ", by Cardoso, 2020,]. https://fernando0stc.medium.com/building-iac-pipeline-on-aws-with-security-fully-integrate-48952c3435b7

Security automation can also be extended to the feedback loop. Once a security misconfiguration is detected in the DevOps pipeline by a security scan plugin or a CLI, the automation can leverage the information provided by the scanning tools to upload the information to the issue tracker for proper planning. This level of automation will ensure consistent ingestion of security misconfigurations for each release of code is consistently available to be evaluated as part of each build cycle planning.

Despite human capital challenges faced by teams, automation through standardization ensures a consistent and repeatable secure, and compliant implementation all the time. Enforcing a consistent configuration reduces manual overhead while improving the efficiency of the delivery of a standard security posture. Organizations evaluating the cost-benefit associated with the implementation of security automation in their resource provisioning will quickly realize not only the efficiency gains associated with adopting automation but also a stronger security posture. These benefits will empower decision-makers to start adopting a “shift-left” approach to standardize secure configuration and deployment of applications in cloud environments.

Conclusion

Mitigating risk associated with cyberattacks against cloud-native applications requires a continuous improvement process that includes careful consideration of the following key concerns:

- Who is responsible for which component of the cloud infrastructure?
- How can organizations repeat a consistently secure and compliant deployment?

- How can organizations detect and protect proactively against the risk associated with code, dependencies, workloads, and configuration?
- How can organizations implement efficiencies while maintaining a strong security posture in cloud-native applications?

The answer to these questions relies on a combination of security guardrails designed to leverage continuous innovation with security embedded at every step of the process. Cloud providers enable multiple services to allow an organization to define its security posture. To ensure the achievement of consistent implementation of security guardrails and compliant configurations organizations must leverage standardization through the use of IaC. However, IaC alone will not minimize risk.

The code associated with IaC must be scanned for security vulnerabilities just like we scan for SAST and SCA in the SDLC. As workloads are implemented and code gets deployed to cloud-native services, scanning for dependencies and containers becomes a key risk mitigation strategy. Leveraging SCA and container security scanning will help with proactive detection. Embedding these scans in a DevOps pipeline will help the organization achieve efficiencies and security gates throughout the process of deploying applications to the cloud.

Once organizations have embedded security scanning in IaC, automation through DevSecOps must be implemented to ensure they achieve efficiency gains while security is enforced at the forefront of the implementation. Monitoring workloads and ensuring compliance with established security policies will help minimize risks across every single implementation. In summary, securing cloud-native applications requires covering concerns for:

- Security misconfigurations
- Threat vectors in dependencies
- Lack of proper authentication and authorization
- CI/CD and software supply chain risks
- Inadequate resource utilization controls
- Ineffective logging and monitoring

Effectively protecting cloud-native applications involves a complex mix of traditional controls for new technology innovation elements provided by cloud providers. Minimizing risk in such environments must be planned and executed with a strategic mindset by the organization.

References

Amazon. (n.d.). What is Cloud-native? Amazon. https://aws.amazon.com/what-is/cloud-native/

Cardoso, F. (2020, November 2). Building IaC Pipeline on AWS with Security Fully Integrate. [Digital Image]. https://fernando0stc.medium.com/building-iac-pipeline-on-aws-with-security-fully-integrate-48952c3435b7

Ciesielski, J. (2023, January 16). The Shared Responsibility Model and SaaS, Explained. [Digital Image]. https://rewind.com/blog/shared-responsibility-model-saas-explained/

Center for Internet Security. (n.d.). CIS Critical Security Control 16: Application Software Security. CISSecurity. https://www.cisecurity.org/controls/application-software-security

F5. (2021). The State of the State of Application Exploits in Security Incidents. The-State-of-the-State-of-Application-Exploits-in-Security-Incident-F5Labs-rev22JUL21.pdf

Fowler, M., Lewis, J. (2014, March 25). Microservices. Martin Fowler. https://martinfowler.com/articles/microservices.html

Heim, M., Keim, P., Munsch, J., Pabon, W. (2020). Software Security Automation: A Roadmap toward Efficiency and Security. https://ndisac.org/wp-content/uploads/ndisac-security-automation-white-paper.pdf

IBM. (n.d.). Cloud-native. IBM. https://www.ibm.com/topics/cloud-native

International Organization for Standardization. (n.d.). Information technology — Security techniques — Application security — Part 1: Overview and concepts. ISO. https://www.iso.org/obp/ui/#iso:std:iso-iec:27034:-1:ed-1:v1:en

Microsoft. (n.d.). Zero Trust implementation guidance | Microsoft Learn. Microsoft. https://learn.microsoft.com/en-us/security/zero-trust/zero-trust-overview

National Institute of Standards and Technology. (2018). Zero trust architecture. (Department of Commerce D.C.), Special Publication 800-207. https://doi.org/10.6028/NIST.SP.800-207

National Institute of Standards and Technology. (2022). Implementation of DevSecOps for a Microservices-based Application with Service Mesh. (Department of Commerce D.C.), Special Publication 800-204C. https://doi.org/10.6028/NIST.SP.800-204C

National Institute of Standards and Technology. (2022). Secure Software Development Framework (SSDF) Version 1.1. (Department of Commerce D.C.), Special Publication 800-218. https://doi.org/10.6028/NIST.SP.800-218

National Institute of Standards and Technology. (2023). Glossary. National Institute of Standards and Technology. https://csrc.nist.gov/glossary/term/infrastructure_as_code

OWASP. (n.d.). OWASP Cloud-Native Application Security Top 10. https://owasp.org/www-project-cloud-native-application-security-top-10/

OWASP. (2021). Application Security Verification Standard 4.0.3. OWASP. https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

Rapid7. (2021). The 2021 Vulnerability Intelligence Report.
https://www.rapid7.com/products/insightvm/vulnerability-report-hub-page/

Saraswathi, R. (2020, January 6). Four Architecture Choices for Application Development.
https://www.ibm.com/cloud/blog/four-architecture-choices-for-application-development

Valdes, A. (2021, April 20). The best Infrastructure as Code tools for 2022. [Digital Image].
https://www.clickittech.com/devops/infrastructure-as-code-tools/

Wallgren, A. (2017, March 17). Are you Actually Doing DevOps?. DevOpsDigest.
https://devopsdigest.com/are-you-actually-doing-devops