SANS

A SANS Whitepaper

How to Secure App Pipelines in AWS

Written by **Dave Shackleford**

September 2019

Sponsored by:

AWS Marketplace

We are seeing nothing less than an evolutionary shift as security infrastructure moves to software-defined models that improve speed and scale, and afford enterprise IT more agility and capabilities than ever before. Application development and deployment are driving this shift, and as the pace of development increases, organizations have a real need to ensure application security is embedded in all phases of the development and deployment life cycle, as well as in the cloud during operations.

Much like other areas of security, the responsibility for application security varies in the cloud widely, depending on the model in place. In a software-as-a-service (SaaS) model, the provider is entirely responsible for application security in almost every case. With a platform-as-a-service (PaaS) model, the provider supplies the underlying systems and templates, so it has a significant degree of control and responsibility—although any applications developed by the consumer are necessarily the consumer's own responsibility, and that extends to the security. With an infrastructure-as-a-service (IaaS) model, entire workloads and their contents (including application components) are the responsibility of the consumer.

In this paper, we delve into the changing nature of application development and security as organizations are building and deploying applications for the cloud. We'll cover the various phases of a modern application pipeline and discuss some of the security controls that organizations should consider implementing in each. We'll also touch on a number of other critical areas such as privilege management, containers and orchestration, and automation.

Analyst Program

How the SDLC Is Changing

The software development life cycle (SDLC) has moved to a methodology that prioritizes collaboration and more frequent (yet smaller) updates to application stacks. Standards for code quality and security, as well as application workload configuration, should be defined and published so that all teams have something to measure throughout the entire application life cycle. Ideally, organizations will lock down cloud workloads as much as possible, running a minimum of necessary services. They should also revisit configuration requirements to ensure that any cloud-based infrastructure is resilient.

The SDLC has moved to a methodology that prioritizes collaboration and more frequent (yet smaller) updates to application stacks.

To shift toward a more collaborative culture, security teams need to integrate with the developers responsible for promoting code to cloud-based applications. Security teams can impress upon development and operations that they bring a series of tests and "quality conditions" to bear on any production code push without slowing the process. Security teams should work with quality assurance (QA) and development to define certain parameters and key qualifiers (such as bug count and severity) that need to be met before any code is promoted.

In addition, security teams need to determine which tools they can use to integrate into the application pipeline. They also need to identify areas and controls that may need to be updated or adapted to work in a Continuous Integration and/or Continuous Delivery model (covered in the next section). It is likely that new standards for many security prevention, detection and response capabilities should be revisited, as well. Examples of these areas include encryption, privileged user management, network security access controls, event management, logging policies and incident response strategy.

Once initial processes, policies and standards have been defined and agreed upon, the security team should focus on automation and seamless integration of controls and processes at all stages of the deployment pipeline.

The Modern CI/CD Pipeline

Many organizations are adopting Continuous Integration (CI) and Continuous Delivery (CD) for their cloud application pipelines. CI is often the most feasible part of the application development life cycle to be targeted by a team looking to speed up and implement more collaborative development practices. With CI, all developers have their code regularly integrated into a common mainline code base. This practice helps to prevent isolation of code with individual developers and can also lead to more effective control over code in a central repository.

CD is usually exhibited through small, incremental and frequent code pushes (often to stage or test environments), as opposed to the more traditional way of pushing code as large releases to production every few weeks or months. Modern development practices (e.g., Scrum, Kanban, Crystal, etc.) often release code more frequently than older models (e.g., waterfall) in an SLDC. CD means you deliver code to production in an automated pipeline, which is less common in traditional enterprises.

Modern cloud application pipelines strive for a number of goals and focal areas:

- **Automated provisioning—**The more automated the provisioning of resources and assets is, the more rapidly the SDLC and operations model can operate.
- **No-downtime deployments—**Because cloud services are based on service-oriented costing models, downtime is less acceptable.
- **Monitoring—**Constant monitoring and vigilance of code and operations help to streamline and improve quality immensely.
- **Rapid testing and updates—**The sooner code flaws can be detected, the less impact they'll have in a production environment. Rapid and almost constant testing needs to occur for this to happen.
- **Automated builds and testing—**More automation in the testing and QA processes will help to speed up all activities and improve delivery times.

Protection for application workloads requires a dedicated commitment to security at many levels of any organization. A sound governance model that includes collaborative discussions about code quality, system builds, architecture and network controls, identity and access management, and data security is critically important to developing the standards for controls and security posture (mentioned earlier).

Ideally, the following types of roles will be a part of any cloud application security and development model:

- Application development teams
- Cloud architecture and engineering teams
- Security architecture and operations teams
- IT in infrastructure teams (server engineering, database management and more)
- Compliance and legal teams (where appropriate)
- Business unit management (where appropriate)

Make sure that your security teams discuss:

- **Standard and planned coding and release cycles—**If the development teams plan on doing CI, how will the code be centrally stored and managed? Security teams should focus on code scrutiny and auditing the code storage/management platform and tools.
- **Tools in use for development, testing and deployment—**Automated testing suites are ideal, but security teams need to understand the tools the development teams plan to use so that they can become familiar with platform security, logging and privilege/credential management.

A sound governance model that includes collaborative discussions about code quality, system builds, architecture and network controls, identity and access management, and data security is critically important to developing the standards for controls and security posture.

- **How security can best integrate with the teams—**Ideally, security teams will have some understanding of development practices, and will know how to write test scripts and infrastructure-as-code templates where applicable.
- **Expected standards and behaviors—**If there are no standards to adhere to, what will the team seek to enforce? Think about standards for secure coding, configuration benchmarks (like CIS and others) and vulnerability scan results (what is acceptable to be released).

In addition, security teams should define policies for components, networks and architecture where they can. In other words, they should ask: Where can security create policies that are embedded and applied automatically? Examples might include:

- Configurations for instances and images used in development and production
- App deployment and automation security
- Expected and accepted standards (What does a successful and secure component or deployment look like? Start with the end in mind to ensure you have a target goal.)

One additional area of IT that will likely need to adapt is change management. In traditional IT environments, change requests are often created for weekly or biweekly change windows, where IT staff make changes during the scheduled times (usually off-hours). In a fast-moving cloud application environment, much more rapid changes will need to be allowed. Teams will usually need to adapt by deciding ahead of time which severity of changes will be allowed to occur without prior approval or review versus those that will need more attention. Collaboration platforms can also be useful for enabling more rapid discussions about proposed changes as needed.

Security in the CI/CD World

When integrating into a cloud-focused application development model, security teams need to focus on the following:

- **Code security—**How is code being scanned for vulnerabilities?
- **Code repositories—**How is code being checked in and checked out, and by whom?
- **Automation tools—**What tools are in use to automate builds, deployments, etc.? How can security integrate with these?
- **Orchestration platforms—**How are orchestration tools being used to coordinate and automate infrastructure and cloud components?
- **Gateways and network connectivity—**How can the teams ensure secure connectivity to the cloud for deployments?

Authentication/authorization and privileged user monitoring and management are critical, too. While this sounds obvious, cloud application development pipelines tend to include high-privilege users doing lots of activities, and overallocation of privileges can quickly become an issue without oversight and planning.

When planning for cloud application development, security teams first need to work with application development groups to perform threat modeling and risk assessment for the deployment types that they envision. By performing a threat modeling exercise, security and development teams can better understand the types and sensitivity levels of the assets they protect, how to manage and monitor them in the cloud, and the most likely threat vectors for those assets. The type of data that is stored, transmitted and processed makes a difference when assessing the risk of systems and applications in the cloud. Some data types dictate specific security controls, as well as provisioning into compliant cloud provider environments.

When planning for cloud application development, security teams first need to work with application development groups to perform threat modeling and risk assessment for the deployment types that they envision.

Risk assessment and analysis practices should be updated to continually review the following:

- Cloud provider security controls, capabilities and compliance status
- Internal development and orchestration tools and platforms
- Operations management and monitoring tools
- Security tools and controls, both on premises and in the cloud

After risk reviews, and keeping the shared responsibility model in mind (meaning cloud providers and consumers share responsibility for security at different layers of the stack), security teams should have a better understanding of what controls they currently have, what controls they need to modify to successfully operate in the cloud, and what the most pressing concerns are (as they change). It's almost a guarantee that some security controls—tools, processes, policies, etc.—won't operate the way they did on premises, or won't be available in cloud service provider environments in the same format or with the same capabilities.

Security for the CI/CD Pipeline

In the modern CI/CD pipeline for cloud application development and deployment, one of the most pressing needs for all teams is automation, far beyond what we've traditionally seen in enterprise data centers. With cloud deployment moving faster than ever, security and development teams need to automate static code security scans, dynamic platform build and QA application and vulnerability tests. They also need to automate most (if not all) configuration and operations tasks, including web application firewall (WAF) deployments and network access controls (NACs).

For cloud deployments, all application development teams, as well as security teams, also need to embrace API integration/use. Providers like Amazon Web Services (AWS)[1] operate a completely software-based infrastructure that may offer sophisticated APIs for creating workloads, adding security controls around those workloads, updating and

[1] This paper mentions the names of products and services to provide real-life examples of how security tools can be used. The use of these examples is not an endorsement of any product or service.

integrating new code and images for containers, and much more. In keeping with the theme of automation, scripted and programmatic methods of automating deployments need to make heavy use of provider APIs.

Security teams have a number of security controls and areas of emphasis to consider for all phases of the application development and deployment pipelines, as shown in Figure 1 and discussed in the following sections.

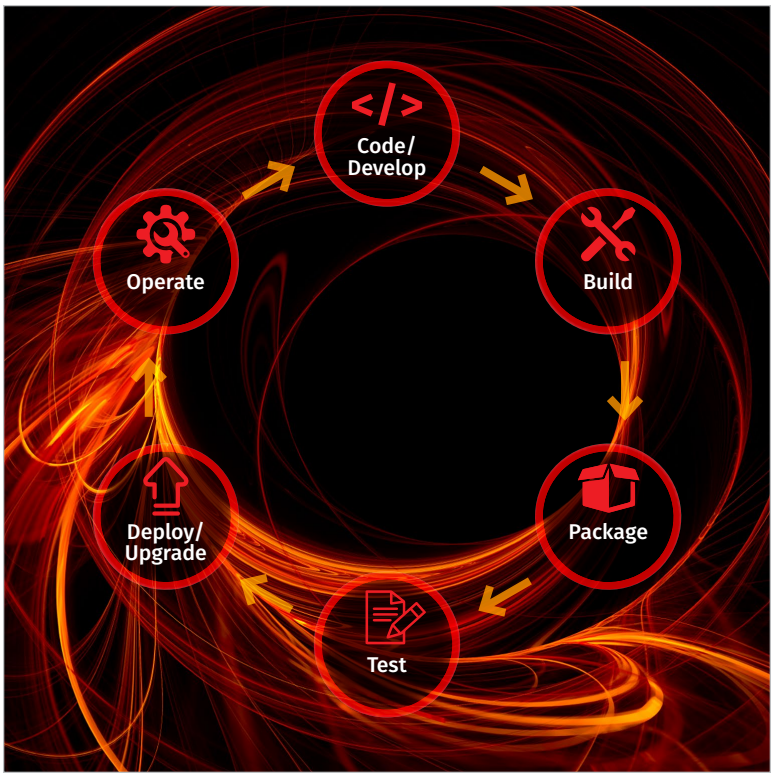

Figure 1. Phases of Application Development and Deployment Pipelines

</> Code/Develop

Ideally, your organization already follows secure coding practices. Security and development teams need to discuss standards for languages and frameworks to make sure risk is acceptable before deployment. This objective can be a tall order, and secure coding and development practices are still not all that commonplace today. Look into static code analysis tools, and ensure the code is secured within repositories:

- Are check-in and check-out procedures defined?
- Do solid role-based access controls exist?

Cloud providers often have options available for code storage and management that include authentication with strong identity management and robust logging/tracking of activity. AWS CodeCommit is a fully managed source control service that hosts secure Git-based repositories that encrypts all files both in transit and at rest, integrates with AWS Identity and Access Management (IAM) for controlling privileges and access to code stores, and logs all activity in AWS CloudTrail. Additionally, AWS CodeCommit has a wide range of APIs available that can enable automation and integration with third-party static code analysis tools for code analysis and review by security teams. Code can be automatically scanned upon check-in, and bug/vulnerability reports can be sent automatically to the appropriate teams.

Security and development teams need to discuss standards for languages and frameworks to make sure risk is acceptable before deployment.

Build

Building code and workload stacks for cloud applications should incorporate automated and intelligent security controls as well. This stage should include:

- Validated code
- An approved build architecture and controls
- Automated build testing for compiled code

Above and beyond the aforementioned automation and security controls and processes, we need automated reporting that goes to the proper parties for review. This is what will ultimately contribute to a more effective vulnerability management program across

the environment. Much like the previous phase of development (code/develop), the build phase can often be securely implemented within cloud provider environments. AWS CodeBuild is a fully managed CI service that compiles source code, runs tests and produces software packages that are ready to deploy. Managing encryption of build artifacts is critical, and AWS CodeBuild integrates with AWS Key Management Service (KMS). AWS CodeBuild also integrates with AWS IAM for control over privileges to builds and compiled code, and all activity is also logged to AWS CloudTrail.

Package

Packaging is the phase of application development when the build is updated with additional software packages, some of which may be open source or from in-house repositories. It is important for development and security teams to audit open source modules for flaws, then discuss methods to protect code repositories automatically. A regular schedule for threat and vulnerability updates with the development and operations teams should be decided upon and incorporated into defined processes.

Some traditional vulnerability scanning vendors have adapted their products to work within cloud provider environments, often relying on APIs to avoid manual requests to perform more intrusive scans on a scheduled or ad hoc basis. Another option is to rely on host-based agents that can scan their respective virtual machines continually or as needed. Ideally, systems will be scanned on a continuous basis, with reporting of any vulnerabilities noted in real or near real time. AWS Systems Manager can be used to manage package repositories and secure build images with up-to-date patches and libraries. Tools like Trend Micro Deep Security can help to automate application protection and package validation for workloads, too.

Test

The testing phase is one that can be highly automated. Consider both static and dynamic tools, depending on builds. Keys for security teams during the testing phase are:

- Run security testing that's as seamless as possible (avoid interfering with QA if you can help it).
- Define test cases and tools.
- Define acceptable outcomes that meet policy.
- Automate tools and teach developers/QA engineers to run them.

The last point is a crucial one—security teams need to hand off tools to the application developers wherever possible and not insert themselves into every process. Involvement is key, but running test tools is something the application teams can do. Security should only perform pen tests and continuous monitoring activities regularly once policies and standards are defined.

Using open source build testing tools like Test Kitchen and Vagrant can simplify internal policy validation before you push them, and also in an ongoing fashion.

To coordinate penetration tests and routine checks to validate policies' effectiveness, ask:

- Are only required ports open?
- Are credentials secured?
- Are encryption keys secured?
- Are privileges assigned properly?

Really, any specific elements of your configuration standard or expected posture should be continually validated and assessed using automated orchestration tools and platforms. Many third-party dynamic application scanning and pen testing service providers have fully integrated into the cloud. These tests can be run upon build check-in, image update or manually as needed, with fully automated reporting sent to the right teams.

Deploy/Upgrade

In this phase, security teams are focused on:

- **Documentation—**Note any bugs that are outstanding; document plans to fix and when.
- **Communication—**Coordinate with development and operations teams to instantiate any controls needed for remediation or stopgaps.
- **Life cycle—**Ensure an approved policy for bug remediation is in place and monitored for future release cycles.

Even though you'll still have bugs, make sure to fix any of a certain severity before you push applications and systems out the door.

Deployment involves more on the operations side. Ideally, controlled and automated deployments will be coordinated and controlled by operations with input from the application development teams involved. Where does security fit?

- Nothing new is added/changed once approved builds are ready.
- Deployment is done to the appropriate location/endpoints.
- Deployment is performed over a secure channel for cloud (TLS/SSH).
- Checks exist to ensure a failed deployment rolls back.

It is critical for security teams to be invested and involved in the development stage. Secure network channels should be established for any deployment activities, which likely involves the use of dedicated circuits like AWS Direct Connect, VPN tunnels using IPSec and/or secure certificate-based HTTPS with strong cryptographic TLS implementations. Image validation—which will heavily rely on automation and a combination of vulnerability scanning and host-based agents that can validate all libraries, binaries and configuration elements used in the application workloads—should also take place at this phase. Orchestration engines are useful for some of these tasks, as are cloud-native tools like AWS OpsWorks that can reliably and securely handle the configuration and assessment of application images.

Operate

This final stage primarily focuses on protection of applications with tools like NACs and WAFs, as well as monitoring, logging and alerting. Define security use cases for production operations by answering the following questions:

- What events should trigger alerts?
- What events should trigger automated remediation?
- What event severities should be in place?
- What controls are needed to properly secure the environment?

For starters, teams should define deployment attributes that can be monitored continuously. Examples of quick wins for monitoring include the following:

- Types of instances allowed to be deployed (size and build)
- Image expected for deployment
- Location/source of deployment (such as IP address or account/subscription)
- IAM or other user invoked in operations

These attributes should all be known and relatively inflexible, and can easily be used as simple trigger points for alerting or even automated rollback or preventative actions. For example, if an instance type of `m1.small` is deployed, and the only approved type is `t2.micro`, this trigger could cause the workload to shut down entirely. Cloud-native or third-party web application firewalls like AWS WAF can easily be set up to block malicious application attacks like SQL injection, cross-site scripting (XSS) and others. In addition, they can perform manual or automated blocking of IP addresses based on threat intelligence that incorporates reputation analysis. WAFs can generate detailed logs, too, which security teams can then stream back to a central analysis engine like a SIEM platform.

Best Practices

To summarize, Table 1 describes the key security areas of focus in the modern cloud application development pipeline.

Table 1. Considerations for Key Security Phases

Phase	Focus
Code/Develop	Look for static code analysis tools that are in place and performing (ideally) automated code scans for checked-in code. Reports from these scans should be sent to stakeholders that include security teams and/or application developers.
Build	Tools like Jenkins can be used to create builds, and they often have many plug-ins and local controls that should be tuned. What types of builds are allowed, and where are the images stored? A secure location where image security and integrity are controlled is paramount for this phase.
Package	Code will need to be packaged for installation on builds, and this should be done through automated tools that also have the appropriate permissions and access controls (keys to check out code, for example).
Test	The test phase should include Dynamic Application Security Testing (DAST) tools, as well as (possibly) traditional network vulnerability scans and various flavors of pen tests.
Deploy/Upgrade	Only approved builds with packages/software that passes testing should be deployed over a secure channel.
Operate	Now we're in operations, where we should have "guardrails" set up like the appropriate account/subscription separation, IAM policies, network controls and logging/monitoring.

Additional Development Security Concepts for Cloud

Along with core security controls and practices in each major phase of a modern development pipeline, some additional topics and concepts should be in place. Think of these as overarching concepts that apply throughout the entire life cycle. Figure 2 illustrates these concepts, which we cover in the following sections.

Secrets Management

A critical aspect of managing security in a cloud environment is to carefully limit and control the accounts and privileges assigned to resources. All users, groups, roles and privileges should be carefully discussed and designated to resources on a need-to-know basis. The best practice of assigning the least privilege model of access should also be applied whenever possible. Any privileged accounts (such as root and the local administrator accounts) should be monitored closely—if not disabled completely or used only in break-glass procedures.

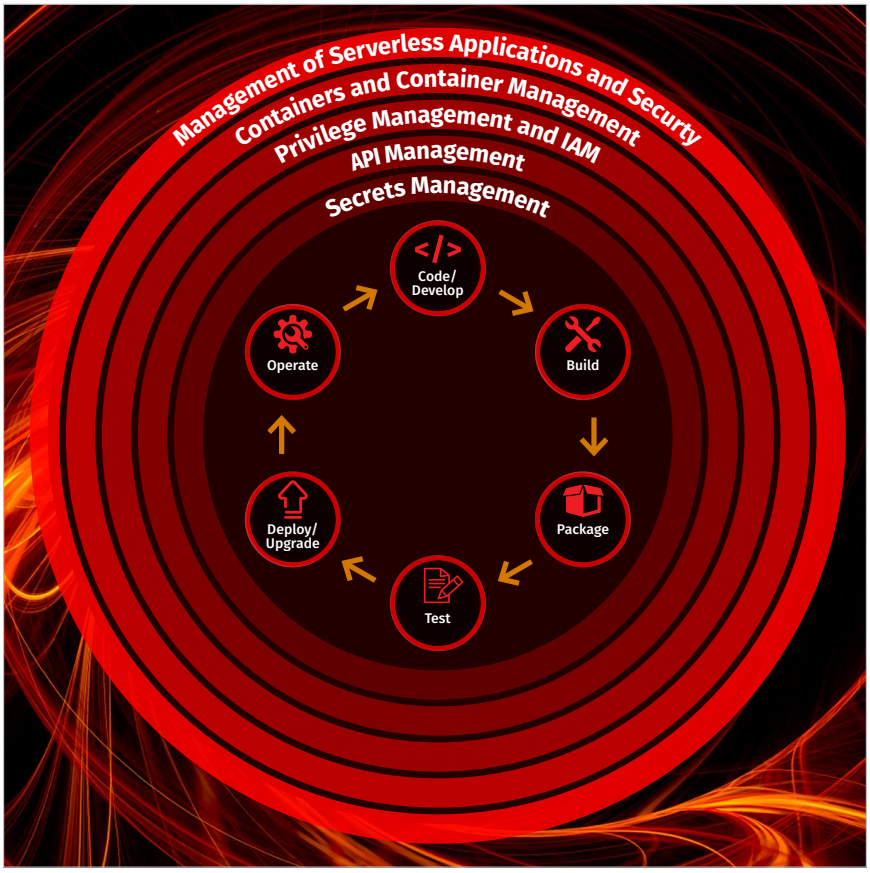

Figure 2. Additional Security Considerations Throughout the Life Cycle

In addition to privilege management in configuration definitions, application development teams need to ensure no sensitive material like encryption keys or credentials is stored in definition files, on systems that are exposed or in code that could be exposed. As encryption and data protection strategies are increasingly automated along with other development activities, it's critical to make sure the proverbial keys to the kingdom are protected at all times. In the cloud, this can be easily accomplished with a variety of tools like AWS Key Management Service (KMS) and AWS Secrets Manager.

API Security

As mentioned earlier, APIs are integral to building a robust and automated development pipeline. The security posture of APIs should be documented by providers, and all APIs should be strongly controlled through IAM policies. Use of APIs should be carefully monitored, too, with full logging to AWS CloudTrail and other logging engines.

Application development teams need to ensure no sensitive material like encryption keys or credentials are stored in definition files, on systems that are exposed or in code that could be exposed.

Privilege Management and IAM

Strong privilege management is a necessity in fast-moving application pipelines. Integration with secrets management tools and a granular IAM policy engine like AWS IAM is crucial, along with federation capabilities and integration with directory services. Security teams should help to define the appropriate least privilege access models needed for all stages of application development and deployment, and then implement this in a centralized tool/service whenever possible. A fragmented privilege management and IAM implementation strategy often leads to poor operational oversight of users, groups and permissions, so a single policy engine should be used if at all possible.

In addition to these overarching technology concepts, some newer technologies are also being heavily used in application development and deployments today, including containers and serverless applications, discussed next.

Containers and Container Management/Orchestration

Containers are rapidly becoming a common means of quickly deploying application workloads in both internal and cloud environments. Containers are created on a shared OS workload, and both the runtime container image and the underlying OS platform need to be secured and maintained much like other images described earlier. Having a secure repository for container images like Amazon Elastic Container Registry (ECR), as well as orchestration tools that can be used for starting, stopping and managing container deployments securely like Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS), is important for enterprises using containers in the cloud. Encryption and IAM controls for images, as well as strong logging for all activities should be priorities.

Serverless Applications and Security

A final type of technology that many application development teams are employing is serverless, which offloads the entire workload (container and OS instance) to the provider's backplane, allowing developers to create microservices applications that only require application code to be uploaded and operated within the cloud provider environment. Serverless security should involve static code review (numerous third-party providers can integrate into serverless environments like AWS Lambda to scan the code), privilege and permission control over all serverless applications with IAM, and complete logging of all serverless application updates and execution using tools like AWS CloudTrail.

Use Case

For modern hybrid application development pipelines, security needs to be integrated in a number of places. Imagine a fictional organization, ACME Corporation, that needs to integrate security into its hybrid cloud application pipelines with both on-premises resources and those running in AWS. Internal code repositories are synchronized from on-premises code repository tools with AWS CodeCommit across an AWS Direct Connect channel, where all code is encrypted and protected with strong IAM policies that restrict code access and updates to a limited team of developers. All code updates, check-ins and check-outs are logged and recorded in AWS CloudTrail. A third-party static code analysis tool is integrated into AWS and automatically scans all code that is updated and checked in. Reports are automatically sent to security and development team members to review the criticality of bugs discovered for remediation.

AWS CloudFormation templates are used to create builds with approved Amazon Machine Images (AMIs) and container images stored in the Amazon ECR, which is also carefully controlled through IAM policies. In the build and update phases, a dynamic vulnerability scanning platform with agents and network scanning capabilities is integrated to scan all application builds for libraries, binaries and OS configurations to ensure no vulnerabilities are present before deployment. Reports are again automatically generated and sent to team members for review. If the reports show that all images meet pre-approved standards, the images are then pushed into deployment with defined orchestration using Amazon EKS and Amazon EC2 instances with AWS Systems Manager installed for monitoring and administration. Once deployed, AWS WAF is enabled to protect applications from malicious application attacks.

Summary

For modern application pipelines, there are a plethora of tools available from cloud providers and third-party companies to help automate strong security controls through the entire development and deployment process. A strong governance structure is critical to ensure all stakeholders are involved and on board with the new tools and processes needed, and security operations teams will need to help define standards for code and images, as well as build strong protective and detective controls in the cloud environment.

About the Author

Dave Shackleford, a SANS analyst, senior instructor, course author, GIAC technical director and member of the board of directors for the SANS Technology Institute, is the founder and principal consultant with Voodoo Security. He has consulted with hundreds of organizations in the areas of security, regulatory compliance, and network architecture and engineering. A VMware vExpert, Dave has extensive experience designing and configuring secure virtualized infrastructures. He previously worked as chief security officer for Configuresoft and CTO for the Center for Internet Security. Dave currently helps lead the Atlanta chapter of the Cloud Security Alliance.

Sponsor

SANS would like to thank this paper's sponsor: