



SANS Institute

Sponsored by:



AppSec/DevSecOps Best Practices in AWS

Compiled from works completed by
Nathan Getty | **Shaun McCullough** | **Dave Shackelford**
with an introduction by **John Pescatore**

December 2019

Table of Contents

3	Introduction
4	How to Protect a Modern Web Application in AWS
5	Introduction
5	A Threat Modeling Primer
6	Threat Modeling Process and Frameworks
6	Risk Assessment and Prioritization
8	DevOps with Security
9	Threat Modeling a Web Application
9	Risk of Web Application Attacks
9	Use Case: Spoofing an Identity
12	Use Case: SQL Injection Attack
12	Threat Modeling the DevSecOps Platform
13	Use Case: Credential Disclosure
14	Use Case: Software Vulnerability to Denial of Service
15	Summary
18	JumpStart Guide for Application Security in AWS
19	Introduction
19	Understanding Your Needs
21	Implementation Options in AWS
21	Cloud-Native Services
21	Open Source and Custom Solutions
22	Consulting Partner Private Offers
22	Needs and Capabilities: The Business Case for Application Security in the Cloud
23	General AWS Web Application Security Considerations
25	AWS Implementation Considerations
26	Making the Choice
26	Evaluate Your Organization's Current Deployment Process
26	Define a Plan and Implement
27	Conclusion
29	How to Secure App Pipelines in AWS
30	How the SDLC Is Changing
31	The Modern CI/CD Pipeline
33	Security in the CI/CD World
34	Security for the CI/CD Pipeline
35	Code/Develop
35	Build
36	Package
36	Test
37	Deploy/Upgrade
38	Operate
38	Best Practices
39	Additional Development Security Concepts for Cloud
39	Secrets Management
39	API Security
40	Privilege Management and IAM
40	Containers and Container Management/Orchestration
40	Serverless Applications and Security
41	Use Case
41	Summary
43	Next Steps

“Avoiding—versus constantly patching—vulnerabilities in applications is the most effective and efficient way to lower risk and avoid security incidents.”

Today, the first impression a customer has of a company and its products and services usually starts with software: either the company’s website or its mobile applications. Because businesses have learned that to be successful they need to continually refresh their products and be first to market with new offerings, the pace of change for websites and applications has continued to increase. As a result, a number of software development methodologies have evolved to meet the need for speed: Lean, Agile, DevOps, etc. A common factor across these approaches is the idea of building in quality: Focus on the user needs early, involve development and operations continually, break projects into smaller pieces and iterate through more frequent build-a-little/test-a-little cycles versus the traditional long cycle waterfall model. The availability of infrastructure-as-a-service (IaaS) offerings has greatly accelerated this movement.

It turns out that today’s focus on digital business also means software is vulnerable to hackers and cybercriminals. Efforts at moving to secure development lifecycle (SDL) approaches have often focused on the long phases of the traditional waterfall model. As DevOps principles have been increasingly adopted, traditional security testing approaches have often been too slow and are frequently bypassed.

However, many security organizations have been able to work cooperatively with IT and app development to make sure “building quality in” includes “building security in.” Avoiding—versus constantly patching—vulnerabilities in applications is the most effective and efficient way to lower risk and avoid security incidents. Doing so successfully requires security teams to understand how DevOps works across hybrid IT architectures and to evolve security processes, architectures and controls by embedding them in DevOps methodologies.

The papers that follow describe best practices and techniques for securing web applications and app pipelines in Amazon Web Services (AWS):

- ***How to Protect a Modern Web Application in AWS***, written by Shaun McCullough, provides a use case of modeling the threats against a web application server and how to address those risks in a cloud environment. This paper covers the complete web application stack, including the web server, application code and DevOps pipelines to manage it, and provides a tutorial on threat modeling concepts and frameworks.
- Nathan Getty’s ***JumpStart Guide for Application Security in AWS*** provides details on the application deployment and security options in AWS. It defines a methodology that details the business, technical and operational considerations needed to develop the optimal application security approach to avoiding and mitigating vulnerabilities in applications that run on AWS.
- In ***How to Secure App Pipelines in AWS***, Dave Shackelford provides detail on how Continuous Integration/Continuous Delivery (CI/CD) pipelines are typically implemented for cloud-based application development. This paper highlights the key areas security needs to integrate into the CI/CD pipeline, including code security, code repositories, automation tools, orchestration platforms and gateways, and network connectivity.

Adapting a strong threat modeling process; understanding the business, technical and operational considerations associated with application security; and automating security controls throughout the entire development and deployment life cycle is critical to securing cloud-based apps.



How to Protect a Modern Web Application in AWS

Written by **Shaun McCullough**

April 2019

Sponsored by:

AWS Marketplace

Webcast

You can access the associated webcast at:

<https://pages.awscloud.com/How-to-Secure-a-Web-App-in-AWS.html>

Introduction

As businesses move more assets to the cloud, having a security plan is essential, but nobody has the time or resources to do everything that is needed from the start. Instead, organizations need to prioritize their security plans based on the risks to which they are exposed. Too often, organizations start with securing the service they know best or have read about in a blog, or they try to buy their way out of the risks with multiple, expensive security appliances.

While the team is knee-deep in transitioning core services, security takes a back seat. It's confusing to understand where the cloud service provider's responsibility ends and the customer's responsibility begins, or how best to secure the services and leverage new tools properly.

Prioritizing the risks, and hence determining what should be secured first, can be simplified through *threat modeling*—the process of identifying and prioritizing the risks to infrastructure, applications and the services they provide. A proper threat model allows organizations to identify applicable risks, prioritize those risks and evaluate how to manage changes in risks over time.

Implementing threat modeling in the cloud is similar to implementing for a traditional infrastructure, but the cloud services, risk priority levels and potential solutions can be vastly different. A threat against a web application stack will be the same in the cloud as it is when deployed on premises. However, cloud providers offer new tools to address the risks. Security teams can bring together cloud-native services, centralized logging, new identity access management processes and easy-to-implement third-party services to make applications and infrastructures safer.

This paper is a use case of modeling the threats against a web application server and how to address those risks in a cloud environment. We will cover the web app stack, including the web server, the application code, and the DevOps pipelines to manage it. Database threats will be covered in future papers in this series. We'll examine the tools and services that cloud providers offer to operate web applications at scale and integrate security services. The paper also breaks down the DevOps process, explains how it can be threat-modeled, and describes common security risks and improvements over traditional workflows.

A Threat Modeling Primer

As defined in a special publication by the National Institute of Standards and Technology (NIST), threat modeling is “a form of risk assessment that models aspects of the attack and defense sides of a particular logical entity.”¹ By implementing a threat modeling process, organizations can improve their security posture, identify unrealized risks and provide their leadership with the proper tools to prioritize which risks to focus on first.

Threat Modeling Process and Frameworks

Most threat models start in one of two ways:

- Identifying a set of attacker techniques the organization is at risk from
- Identifying a set of deployed assets that are at risk

Organizations need to pick the approach that works best for them, but asset-focused threat modeling is usually the most straightforward.

Threat modeling is a process, not a one-time whiteboard session on a Monday afternoon. As the threats evolve, so do an organization's risk appetite and security implementations, along with the experience of the team. Organizations must create a culture of threat modeling, where the model is evaluated, implemented, tested, reviewed and re-evaluated regularly.

The first threat model an organization builds could take time and even be painful. As the team gains experience, the process becomes more natural and standardized. Security teams should hold quarterly reviews to make updates, question assumptions and adjust risks.

Teams should also perform a yearly re-evaluation of the whole threat model, with all the experts available. Regular reviews of the threat model help organizations understand whether the risk-reduction plans are working.

Among the various threat modeling frameworks, the DREAD risk assessment model works well. Used at OpenStack, DREAD helps teams evaluate the potential results of an attack. DREAD helps the team walk through how a system is at risk, what the attack vector looks like, how likely the attack is to occur and how to prioritize which risks to focus on.

The IANS Pragmatic Threat Modeling Toolkit is a spreadsheet that helps organizations walk through the DREAD framework. Users can identify assets at risk, work through DREAD rankings and graph results for easier understanding.²

Drivers of Threat Prioritization

Prioritizing threats is often tricky and likely influenced by the expertise or culture of the organization. If the network team is seasoned, runs a stable environment and has the time to research new threats, it can create the most detailed plan for reducing security risks in the team's responsibility area. In contrast, a host team caught in the middle of a complicated operating system upgrade has no time to think of next week's risks, much less next year's. The organizational culture, workloads, expertise and maturity drive how organizations respond to threats. A threat model process helps level the playing field by giving the appropriate team members the space, tools and support to think about risks and threats across the organization.

Threat modeling is a process, not a one-time whiteboard session on a Monday afternoon.

Risk Assessment and Prioritization

Every risk in an environment is addressed in one of four ways, as illustrated in Figure 1.

Mitigate—Putting a firewall in front of your web server will mitigate some attacks, but not all of them. Most security controls focus on mitigating risks.

Eliminate—Eliminating a risk will likely require changing the nature of the asset at risk in such a way that the risk fundamentally goes away. A firewall cannot eliminate all scripting attacks against a web application,

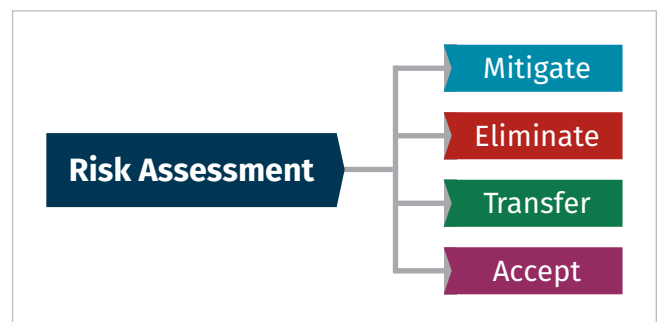


Figure 1. Risk Management Strategies

¹ Draft NIST SP 800-154, Guide to Data-Centric System Threat Modeling, <https://csrc.nist.gov/publications/detail/sp/800-154/draft>

² IANS Pragmatic Threat Modeling Toolkit, https://portal.iansresearch.com/media/739278/ians_pragmatic_threat_modeling_toolkit.xlsm

but removing all data entry fields and making the website completely static will certainly eliminate whole categories of attacks. Eliminating risks is ideal, but difficult—and usually means re-architecting.

Transfer—When an organization decides to move on-premises infrastructure to a cloud provider, it is effectively transferring asset risks to the service provider. The organization is making a business decision to pay for the provider to manage, secure, provision or operate the service. Cloud providers operate on a shared responsibility model. From a security perspective, that means that parts of the infrastructure stack have been transferred to the cloud provider. It is now responsible for operating, security and managing the assets.

Serverless technology is a good example of transferring risk and taking advantage of this shared responsibility model. A customer could spin up virtual machines in the cloud, managing the full stack from operating system to application. The customer is responsible for the patching, configuration and security monitoring of that virtual machine operating system, while the cloud provider is responsible for the virtualization infrastructure, storage and network. Serverless offerings allow the customer to execute a bundle of code, yet have no direct interaction with the executing operating system. The service provider manages the servers in a serverless offering. The risk of operating system vulnerabilities is now transferred to the cloud provider.

Accept—If an organization is unable to mitigate, eliminate or transfer the risk, then it is accepting that risk. It might be a temporary acceptance to be re-evaluated later. In the threat model process, it is healthy for the organization to understand that accepting risk is a valid option that frees it to plan, prioritize, and dive into the other risks.

As an organization gets more comfortable with its threat model process, it should start incorporating the model into the beginning of the development cycle, helping to identify risks that need to be mitigated or eliminated before the organization has invested the time in creating and deploying it. Security teams that work separately from those who create the systems are fighting an uphill battle that will impair effectiveness while raising costs. Include the whole team when modeling a set of services. The developers likely can suggest and implement ways to significantly reduce the risk scores.

Building threat models for IT-operated application services will help with prioritizing and accepting risks.

Building threat models for IT-operated application services will help with prioritizing and accepting risks. Cloud services offer new opportunities for customers to mitigate, eliminate or transfer those risks for traditional IT service applications and to establish new workflows for developing and deploying those systems through DevOps.

DevOps with Security

DevOps is a process that enables close coordination between development and operation teams.³ That integration enables organizations to develop and quickly deploy new services with zero downtime and improved reliability. The process is especially beneficial for organizations that deploy new versions of software multiple times a day.

To incorporate DevOps, organizations rework testing and deployment processes to be safe, automated and executable at any time. Continuous Integration is the process by which software changes from multiple developers are integrated into a single stack, likely multiple times a day. With *Continuous Integration*, security teams can avoid the big end-of-a-sprint integration sessions that cause delays and waste resources. *Continuous Deployment* is the process of building software to be releasable into production at any time, with an easy push of the button.

Continuous Integration and Continuous Deployment (CI/CD) require organizations to rethink their planning, development and deployment pipelines to be highly automated. See Figure 2.

With CI/CD, every evaluation, decision, configuration or security test that can be automated is automated. If these processes cannot be automated, then the development team must rework the architecture.

DevSecOps takes the DevOps process and builds in automated security evaluation gates. The “Sec” of DevSecOps requires the organization to establish security policies for the product before development starts, implementing them in the testing and deployment pipelines. Automated tests are security policies that become reality, not just words in a binder. The best CI/CD processes incorporating DevSecOps give developers the tools to test the security of their code at their workstations—at the beginning of the process rather than waiting until the end of development and being surprised.⁴

CI/CD is usually focused on deploying applications automatically and continuously. However, the cloud opens a whole new area, allowing the automatic provisioning and deployment of core infrastructure itself. The cloud provides APIs, development kits and specialized services that let customers control every aspect of the infrastructure with DevOps-like processes and tooling.

Imagine creating an infrastructure pipeline where a configuration file is used to build a web application stack. And say that a new version of the web server is released with a software patch, and you want to deploy it. After testing it locally, the team updates the configuration file and checks it into version control, and a CI/CD pipeline kicks in and replaces all deployed web servers with the updated versions—automatically.

Companies using on-premises environments have been leveraging DevOps processes to create close coordination between the developers, who create new applications, and operations, which provides the virtual machines they run on. The cloud brings a whole host of services to automate all aspects of the infrastructure deployment and management that on-premises services are unable to match.

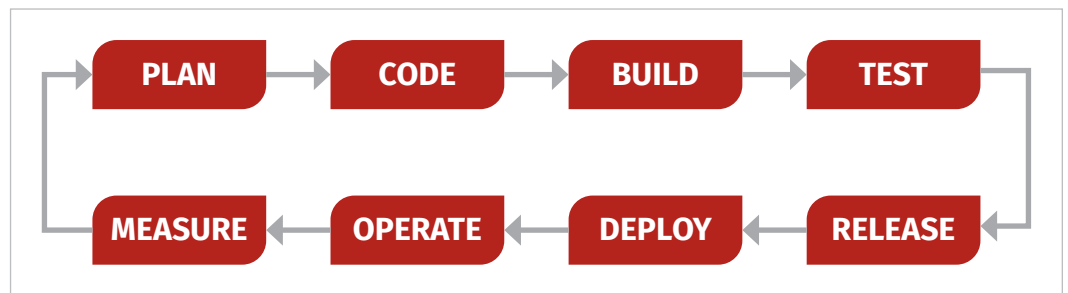


Figure 2. Continuous Integration and Continuous Deployment (CI/CD)

³ NIST SP800-190, <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-190.pdf>

CI/CD comes with risks, however. Automating processes traditionally done by humans can reduce errors, but it also hides unforeseen problems. The platforms that implement DevSecOps and CI/CD pipelines are new attack vectors. The CI/CD platform must become part of the threat modeling process for an organization to ensure that the entire infrastructure is evaluated.

Threat Modeling a Web Application

As previously discussed, the threat model process starts with identifying deployed assets that are at risk—assets that are well understood and vital to the business. As part of our use case, let's model the threat to the web application itself and investigate a threat model for the web application.

Risk of Web Application Attacks

Web applications are usually at risk—they live on the internet, with the sole purpose of capturing and providing information to all their users living on untrusted networks. Complex web applications with user access controls, database-backed pages and free-form input fields are notorious for their vulnerabilities.

The Open Web Application Security Project (OWASP) Top 10⁵ is the best starting place when analyzing threats against web applications. Top attack techniques are prioritized, researched and documented, with details of how the attack works and suggested best practices for stopping the attacks.

Cross-site scripting (XSS) is a common attack on web applications that the OWASP Top 10 – 2017 report describes:

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.⁶

Use Case: Spoofing an Identity

Web applications require data inputs and dynamically display information back to users. XSS could result in many different threat categories. For this use case, an XSS attack that exposes other users' browser session credentials can be used to spoof an identity.

⁴ *Accelerate: Building and Scaling High Performing Technology Organizations*, by Nicole Forsgren, Jez Humble and Gene Kim (IT Revolution, 2018)

⁵ OWASP Top Ten Project, www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

⁶ The 10 Most Critical Web Application Security Risks, [www.owasp.org/images/7/72/OWASP_Top_10-2017_\(en\).pdf.pdf](http://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf.pdf)

After categorizing the threat, a team can evaluate the risk using the DREAD model. Each DREAD risk-rating category is given a value from 1 to 10. Figure 2 describes the ratings.

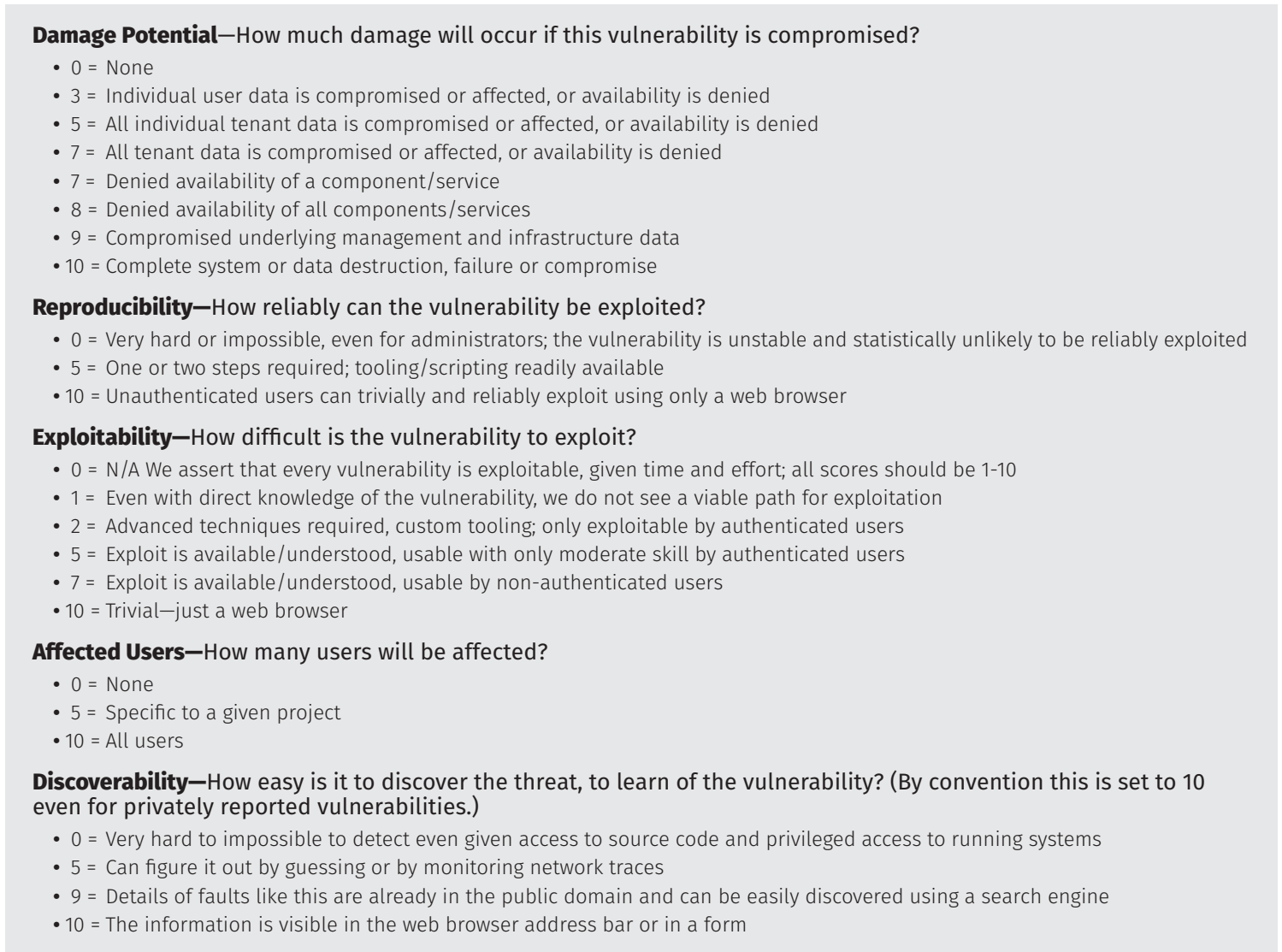


Figure 2. DREAD Risk Ratings⁷

The rating of a single threat does not provide a full picture of the organization's vulnerable landscape. DREAD ratings of multiple risks should be viewed in tandem to get a complete picture of the risks that need to be prioritized. While informed by the DREAD rating guidance, organizations will arrive at their final rating number/prioritization through a combination of the ratings and their own experiences, knowledge and biases. Table 1 on the next page shows the DREAD rating for our use case.

⁷ Adapted from DREAD Rating, <https://wiki.openstack.org/wiki/Security/OSSA-Metrics#DREAD>

Table 1. DREAD Rating for Web Application

Category	Rating	Spoofing Identity
Damage Potential	2	The business unit is a significant driver of the risk rating for an application. What data does the application hold? How far-reaching would the attack be? How important is the asset itself? In this example, an XSS attack to gain credentials does not do any damage itself.
Reproducibility	7	Once identified, an XSS attack is easy to reproduce through scripts. Only common application access is necessary, rather than special access privileges.
Exploitability	4	Depending on the vulnerability of the application, an XSS could be easy or hard to exploit. Discoverability rates how easy it is to determine if there is potential for an XSS; however, making the exploit perform the desired identity spoofing can be tricky, so we will rate this lower.
Affected Users	4	An XSS attack affects the users logged into the application at the time of the attack, and potentially any users who view the corrupted data. Some users will be affected, but not all.
Discoverability	7	Entering JavaScript into a webpage and reviewing the results gives an attacker a good idea if there is an XSS vulnerability, even if they cannot complete the exploit.
DREAD Average	4.8	

Because XSS is a well-known and well-researched attack method, security teams have multiple ways to mitigate the risk of an XSS attack on a web server. A popular security control is incorporating a web application firewall (WAF) to monitor and block any suspicious traffic before it reaches the web server.⁸ Large cloud service providers make it easy to implement a WAF right from the console. AWS's WAF service allows you to customize rules and access control lists to fit your business and risk models.

Larger cloud service providers may offer WAF assets that can be integrated into their service offerings. They are easy to set up, are relatively inexpensive, and should be able to block OWASP Top 10 and other common attacks. If the DREAD risk is higher and more protection is needed, the cloud service provider often has a variety of top-tier third-party products with WAF offerings available for installation (for example, Imperva SecureSphere and Fortinet FortiGate).⁹ One way to eliminate the risk of XSS is to remove data entry fields altogether. It requires rethinking the web application architecture and possibly removing functionality for the sake of security. If eliminating the data entry fields is not viable, you can transfer that ownership to a third party. For instance, if the data input fields are for user authentication, leverage a third-party single sign-on service. Eliminating and transferring risks tends to be more costly, but will help decrease DREAD risk scores. The bottom line is that the threat modeling process should drive prioritization of assets and financial commitments.

⁸ Web Application Firewall, www.owasp.org/index.php/Web_Application_Firewall

⁹ This paper mentions product names to provide real-life examples of how varying classes of tools can be used. The use of these examples is not an endorsement of any product.

Use Case: SQL Injection Attack

Modern web applications are driven by databases that can contain a wealth of knowledge that attackers want. A SQL injection tricks the database into returning unintended data.¹⁰ One outcome of a SQL injection attack is *information disclosure*. The DREAD rating determines the severity of this attack in the environment. See Table 2.

Table 2. DREAD Rating for Database

Category	Rating	Information Disclosure
Damage Potential	7	A SQL injection, if successful, will likely affect all the data in the database, not just specific users. The actual damage done in information disclosure is another measure that requires the business units to weigh in.
Reproducibility	7	Once a SQL injection attack is identified, it is repeatable.
Exploitability	5	SQL injection (or NoSQL) tends to be easier to accomplish than XSS.
Affected Users	2	Other users may not even notice if a SQL injection attack is happening unless it is damaging the data. For an information disclosure categorized attack, the user effect is nominal.
Discoverability	6	Like XSS, the SQL injection vulnerability is easier to identify than actually to exploit.
DREAD Average	5.4	

The processes for mitigating a SQL injection and XSS attacks are similar. The SQL injection attack comes through the web application itself; thus the WAF is in a position to identify and block potential SQL injection attacks. Not all SQL injection attacks will be detected, and significant research has gone into countering a WAF.¹¹ When deciding on a WAF product, look at the entire threat model process and ensure that the WAF covers all the threats at the same time.

Another option is to leverage secure coding practices to develop safer code that neutralizes invalid text field inputs before being run in the SQL query on the database. Depending on the programming languages, a number of libraries, design patterns and tools can do this. The security team will need to ensure that all code is following these standards or incorporating the right tools. Today, CI/CD platforms provide opportunities to continuously scan, evaluate or test code as it is being developed.

Now that we've looked at modeling the threat to the web application, let us look at the threat to the development and deployment platform that is used in cloud operations.

Threat Modeling the DevSecOps Platform

We have looked at threat models for a well-known architecture like the web application. Now let's walk through a practical threat model of a CI/CD platform. Again, DREAD helps to prioritize the risks.

A CI/CD process is all about safely automating workflows. The Continuous Integration process kicks off when a developer checks code into the designated source code repository. Distributed version control systems (DVCSs) will mirror an entire copy of the codebase, including all history, on every developer's computer.¹² Git is the most

¹⁰ SQL Injection: Modes of Attack, Defence, and Why It Matters, www.sans.org/reading-room/whitepapers/securecode/sql-injection-modes-attack-defence-matters-23

¹¹ SQL Injection Bypassing WAF, www.owasp.org/index.php/SQL_Injection_Bypassing_WAF

¹² Getting Started—About Version Control, <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

popular DVCS in use today, used with a central Git repository management system like GitHub, GitLab or AWS CodeCommit. When developers request to check their code into the designated central repository, the Continuous Integration system kicks off to test the integration to ensure that it does not break the application. See Figure 3.

Use Case: Credential Disclosure

Web applications can make database connections directly to query for data. Many times, the web application connects to the database through credentials stored in

a configuration file on the application’s server. The developers have an instance of the database in their environment for testing, which may include a small copy of production data to test code changes properly.

If that credential file is accidentally checked into the source control system, that configuration file could become visible to unauthorized users—especially with open source software where the DVCS is accessible to the public. Disclosure of credentials can lead to an unauthorized login to the database, called “identity spoofing.” Using the spoofed identity can then lead to additional information disclosure, tampering of data or even denial of service. Identifying each step and categorizing the actions along the way is building up the attack tree.¹³ See Table 3.

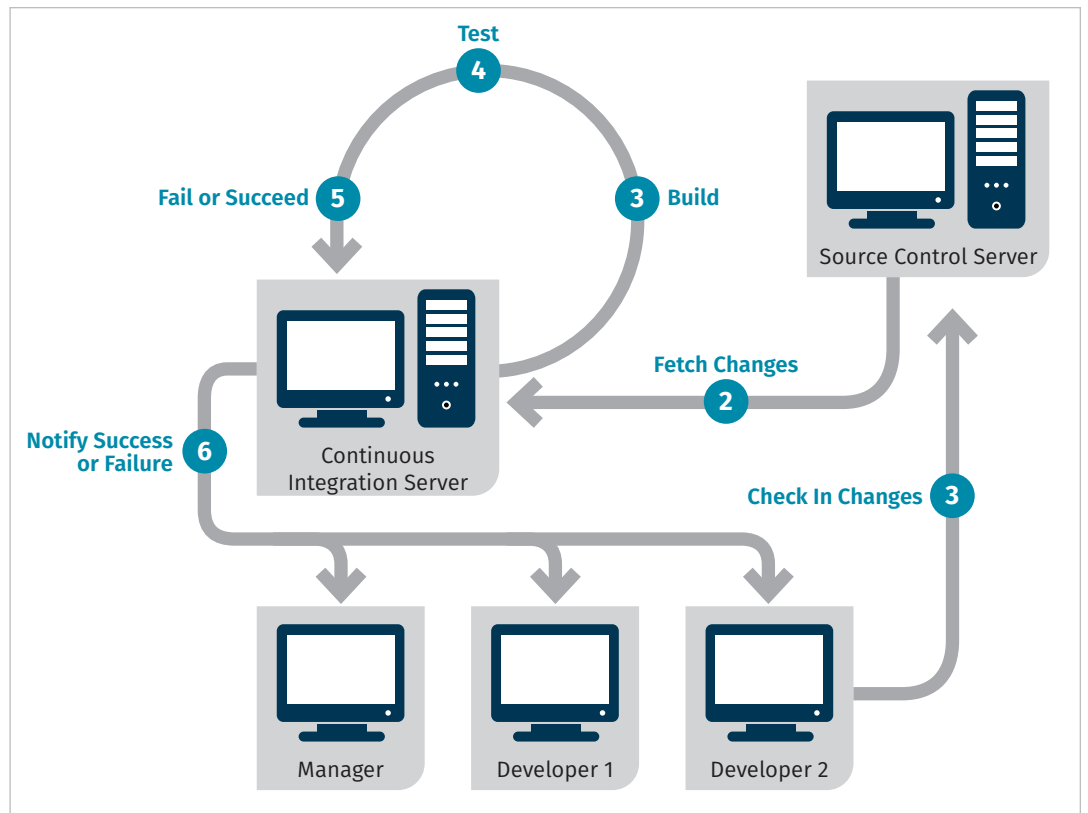


Figure 3. Continuous Integration Process

Table 3. DREAD Rating of Credential Disclosure

Category	Rating	Credential Disclosure
Damage Potential	5	The damage from information disclosure varies depending on the value of the credentials themselves. In this use case, the credentials at risk are for the development environment and reside on the developer’s machine. Because this test database contains a snapshot of production data for testing, customer data is at risk.
Reproducibility	8	The threat exploited is highly reproducible because the attacker can log into the at-risk asset.
Exploitability	8	Logging in with unauthorized credentials is easy when you have the credentials.
Affected Users	5	The database at risk in this particular threat model is a developer’s test environment with limited production data.
Discoverability	9	The software is continuously scanning source code repositories looking for credential-like data, thus discovering the data could take mere minutes.
DREAD Average	7	

¹³ Attack Trees, www.schneier.com/academic/archives/1999/12/attack_trees.html

As the developer is checking in new code in a Continuous Integration process, it is possible that the developer will accidentally check in that credential file and risk disclosure. If undetected, exposure is guaranteed.¹⁴

In CI/CD, the automated test platform could be used to evaluate the code to look for strings that resemble credentials and reject the merge. These tools are inexpensive and are easy to configure and execute; they fit perfectly with the CI/CD process and will mitigate the credential disclosure risks.

To eliminate the risk of credentials being checked in, eliminate the credential file. Secrets management systems, which are available from cloud service providers or through the marketplace, can be used to programmatically store credentials and only provide them to applications that are authorized. Although this risk-reduction will be harder to implement and can cause changes to the asset, eliminating a risk versus mitigating that risk might be worth the cost.

Use Case: Software Vulnerability to Denial of Service

Humans write software, and humans are experts at making mistakes. Security professionals are continually patching, monitoring and managing software updates. To make matters worse, developers are increasingly reliant on software packages distributed by other developers. Code actually written by the development team may be a small percentage of the entire code base for the application. For this threat model, teams must evaluate the risk of a vulnerable third-party NodeJS module making its way into the software stack.

Node Package Manager (NPM) is the most widely used NodeJS package delivery tool, and is likely what organizations are using for JavaScript-based frameworks. A vulnerable NodeJS module can cause information disclosure, escalation of privileges or denial of service.¹⁵ Let's look at denial of service and rate the DREAD risks, as shown in Table 4.

Table 4. DREAD Rating of Software Vulnerability

Category	Rating	Denial of Service
Damage Potential	7	The amount of damage caused by a denial of service is a business-unit-led decision. Is this a core part of the organization's business? Could it go down for a day and see no real effects? Business drivers are just as important as security risks in the threat model process. Knowing how vital each service is to the business helps define these values. For this use case, the product is a core part of the business and could not go down for any length of time.
Reproducibility	5	Reproducibility can be difficult because the exploit in the NodeJS module could be easy or hard to implement depending on what it is. Predicting future vulnerabilities is impractical. The threat modeling team will have to decide how to handle these ambiguous ratings and be consistent.
Exploitability	5	Similarly, exploitability is hard to assess.
Affected Users	8	The number of affected users can be significant. Denial of service attacks against production systems may slow down or even stop customers from using the application.
Discoverability	3	Because this use case is not an open source application, it will be difficult for an attacker to discover that an application has a particularly vulnerable NodeJS package.
DREAD Average	5.6	

¹⁴ I accidentally pushed sensitive info, <https://github.community/t5/How-to-use-Git-and-GitHub/I-accidentally-pushed-sensitive-info/td-p/225>

¹⁵ NPM security advisories, www.npmjs.com/advisories

It can be difficult to know if a vulnerability exists in any included NodeJS packages. Although the vulnerability may not exist in the packages themselves, each of those packages could rely on other packages, which could be vulnerable. The CI/CD platform must continually analyze deployed modules for vulnerabilities discovered post-deployment.

Some code scanner products are available, usually as scriptable software applications that can be run by any CI/CD platform. Commercial versions provide a wealth of threat intelligence and software analysis and are able to not only identify reported vulnerabilities but also scan deep into the code itself and identify risky functions or statements. The code scanners should be easy to run with the CI/CD platform. When developers integrate their code, third-party vulnerability scanners could scan before acceptance. After deployment, the entire code base should be tested daily for newly discovered vulnerabilities that can flag to the security team.

Expanding on this idea, the entire deployment system can be scanned before deployment. In a cloud service environment, the configuration of the infrastructure itself can be managed by code, using tools such as AWS CloudFormation or HashiCorp's Terraform. When a configuration is changed, a sample virtual machine can be automatically built, then scanned by vulnerability scanning tools to ensure that no known vulnerabilities exist in the packages. Third-party scanners have cloud-ready services that can be initiated by CI/CD in the cloud. The results can be used by the CI/CD to determine if a deployment should continue—all automatically.

The risk model can help inform decision makers on whether to use free or commercial solutions. Investigate what additional services and intelligence the commercial products provide, whether they will be easier to implement and operate, and how they might work in the build process. Remember, the risk scores from the threat modeling process and the priorities they uncover can help direct where to focus time and money.

Summary

Start building a threat model process as part of the security culture of your organization and reap the benefits throughout the life of your infrastructure. Focus on identifying the threats, the risks they pose, and the relative business importance to help the organization prioritize where to focus attention and resources. The automation of the integration and deployment processes of applications means security policies need to be identified and implemented at the beginning of the development cycle, not the end.

Threat modeling is a great process for identifying risks. We recommend that any threat modeling process do the following:

- Prioritize risks so organizations know where to focus investment.
- Produce concrete plans to mitigate, eliminate or transfer any risks that will not be accepted.

- Bring security into the beginning of system development rather than at deployment time.
- Create a repeatable, improvable process that is used to make decisions, not just a checkbox.
- Document not just the plan but also the risk-reduction results. A threat model process can help organizations understand how effective they are in planning, monitoring, addressing and measuring risks.

As your threat model process matures, teams can start to evaluate risks in systems before they are even developed. Architectural decisions to eliminate a risk rather than only mitigate it will improve security and likely reduce overall operating costs. And as automated DevSecOps platforms are brought into the organization's workflow, a whole host of risks can be managed automatically.

Adapt a good threat model process that works for your organization. Constantly re-evaluate, improve and expand the process until the organization can see measured results from planned risk reductions.

About the Author

[Shaun McCullough](#) is a community instructor for the SEC545 Cloud Security Architecture class and gives back to his profession by mentoring and supporting the next generation of cyber professionals. With 25 years of experience as a software engineer, he has been focusing on information security for the past 15 years. Shaun is a consultant with H&A Security Solutions, focusing on secure cloud operations, building DevSecOps pipelines and automating security controls in the cloud. He also served as technical director of red and blue team operations, researched advanced host analytics, and ran threat intelligence on open source platforms in his work with the U.S. Department of Defense.

Sponsor

SANS would like to thank this paper's sponsor:



RETURN TO THE
TABLE OF CONTENTS



JumpStart Guide for Application Security in AWS

Written by **Nathan Getty**

September 2019

Sponsored by:

AWS Marketplace
in conjunction with
Fortinet

Webcast

You can access the associated webcast at:
<https://pages.awscloud.com/JumpStart-AppSec>

Introduction

As organizations begin to transition their applications into cloud environments, security teams must provide application security support and insight during the process.

Today's applications are updated more frequently, and regular release cycles are giving way to more rapid incremental releases. Application development continues to evolve to support a more dynamic release schedule. In response, information security teams must be included in the development process if they are to provide support to development teams. Because organizations plan to deploy applications as soon as they are approved for production, your organization's security team should not be the roadblock.

Because development teams release applications faster than they can be reviewed, it is critical to integrate the skills and guidance of the security team into the development model. Whether the application code is deployed on premises or in a cloud environment, automated security tools provide the information security team with visibility into code as it moves through the developer pipeline. This visibility provides more assurance that security will not be compromised.

This process allows the development teams to remain informed of security concerns for their application as it moves through the pipeline. By embedding security within the build process, your organization can build a strong relationship between the security and development teams. By fostering and developing this relationship, developers and security professionals can work in tandem to deliver secure, timely applications.

According to Forbes, nearly three-quarters of companies are planning to move to a fully software-defined data center within two years. Almost half of businesses are delaying cloud deployment due to a cybersecurity skills gap.¹ This paper seeks to give you a better idea of what your organization needs to successfully plan and execute a secure application transition to, or deployment in, an AWS environment.² We discuss how security teams can best support application development teams, what options you have as a security professional for this support, and how best to guide your development teams as they transition workflows to AWS.

Understanding Your Needs

Historically, application development and security teams did not always work closely together. But given the adoption of rapid release cycles and the transition to cloud services, these teams must build a working relationship that effectively supports rapid deployment of secure applications. How can they do that while best using existing tools and processes in the cloud environment?

1. Understand the applications deployed in your organization.

Security analysts need to be knowledgeable about the applications being deployed, at least to the extent of being aware of their primary purpose and target audience. When they understand the application, the underlying code, and for whom the application is designed, they can run threat modeling assessments and plan accordingly. They can make remediation decisions with confidence, bring attention to specific security vulnerabilities, identify which vulnerabilities and risks are acceptable, and provide feedback to the development team. Encouraging security teams to work closely with development teams and speak their language will build a strong, mutually beneficial relationship.

2. Understand application deployment methods within AWS.

Applications can be deployed through any one of several channels or tools. Knowledge of the tools available to development teams can help information security teams define best security practices within those tools and ease incident response or critical changes to the applications. Through awareness of the underlying development process, an organization can be assured that quality information regarding security concerns is being communicated to the development teams.

3. Understand what options and responsibilities you have in AWS as you prepare for securing the application delivery.

The AWS cloud environment gives organizations access to a large developmental toolset in the form of services that include a number of capabilities. Not every service will be a good fit for your organization, so development and security teams should plan ahead and identify which services they will need to use for their application delivery and the security.

AWS offers various platforms for setting up such services. For example, AWS offers serverless services, which means your organization is not responsible for operating or maintaining the underlying infrastructure. Although AWS takes full responsibility for operating the hardware, networking and patch management of the underlying infrastructure, responsibility for the security of any application built on the platform lies completely with the organization.

¹ "2017 State of Cloud Adoption and Security," www.forbes.com/sites/louiscolombus/2017/04/23/2017-state-of-cloud-adoption-and-security

² This paper mentions product names to provide real-life examples of how firewall tools can be used. The use of these examples is not an endorsement of any product.

Implementation Options in AWS

AWS offers a number of services and options as well as access to third-party services for secure application development and rapid release cycles.

Cloud-Native Services

When applications and security tools work harmoniously, future problems (and the need to fix them) can be avoided more easily. Fortunately, AWS-native services are built to work well with each other. Leveraging native services can ease the speed of deployment and integration of application security tools. AWS Marketplace contains a collection of ready-to-deploy infrastructure components your organization can deploy directly into their Amazon VPC (Virtual Private Cloud). AWS Marketplace offers a variety of software including, but not limited to, operating systems, network and business intelligence tools, machine learning software, security software and development suites.

The ability to find, test, deploy and validate software through AWS Marketplace helps organizations identify which applications work for them, which allows them to procure and deploy solutions much faster than when having to spend time engaging with a variety of vendors. (Although deploying AWS Marketplace products can be quick and fast, you should still engage with your organization's software onboarding team before deploying new solutions within your environment; your organization may have certain software onboarding procedures even when it comes to native AWS services.) Leveraging native services also has the added benefit of pricing consolidation. Because AWS services are billed to your account with detailed information, organizations can use native services to view all of their AWS costs within a single, detailed page.

Open Source and Custom Solutions

Native services offer direct benefit to your organization, but there may be situations where you prefer custom or open source software (OSS) applications. OSS and custom tools can be leveraged within AWS as long as they are compatible with AWS infrastructure (Microsoft Windows- or Linux-based platforms). For example, it is possible to run custom or OSS solutions on Amazon EC2 (Elastic Cloud Compute). The key difference with EC2 (versus native service) is that your organization inherits the full responsibility for any underlying infrastructure. Your organization is responsible for patch management and any security solutions required for the infrastructure (firewall, intrusion detection and other security tools). Refer to the AWS Shared Responsibility Model³ for more information.

³ AWS Shared Responsibility Model, <https://aws.amazon.com/compliance/shared-responsibility-model/>

Consulting Partner Private Offers

Customers can also engage through Consulting Partner Private Offers (CPPO) to work directly with trusted advisors to select and configure Application Security solutions from AWS Marketplace. As organizations build out their cloud and cloud security strategy and plan, they may want to consider working with partners to accelerate their efforts or fill any gaps in knowledge or resources that are identified. All consulting partners may extend AWS Marketplace third-party solutions directly to customers through CPPO.⁴ Not every organization will be able to find resources with deep cloud experience. Even experienced cloud technologists may have experience only with specific industries or cloud vendors. A requirements document could be helpful when approaching prospective consultants.

Needs and Capabilities: The Business Case for Application Security in the Cloud

The benefits of putting applications in the cloud must be balanced by the organization's ability to secure them.



Application Security

The need: Conducting application security assessments and reducing vulnerabilities within the AWS environment

Capabilities



- Increased visibility within the development process and application stack
- Reduced risk and vulnerabilities in the applications before they are deployed
- Automated security assessments with actionable remediation
- A relationship with the development teams

⁴ AWS Marketplace Channel Programs, <https://aws.amazon.com/marketplace/partners/channel-programs>


General AWS Web Application Security Considerations




Regardless of the technology or cloud vendor selected, some general business, technical and operational considerations are associated with implementing application security in the cloud. The following sections highlight many of these considerations.

Business Considerations



	Consideration	Details
	Policies and standards	<p>Organizations must understand their current software development life cycle (SDLC) policy and how it may be affected by a move to a cloud environment. An SDLC policy describes the various stages of application deployment and delivery. These underlying methodologies do not change when moving to a cloud environment but the processes and procedures for application code review, application building, delivery and analysis probably will. Anticipating what changes to the SDLC will be triggered by transitioning to AWS will allow organizations to adopt an SDLC that not only fits the cloud model, but also has tangible benefits for an organization's application delivery within the cloud. Planning and making these changes first will save your organization time should a policy need to be redefined in the future.</p> <p>Organizations should determine the acceptable level of risk for their application(s). Although it would be nice if we could deliver applications without errors or exploitable weaknesses, such a scenario is unfortunately unrealistic. Developers have to release applications within the timelines demanded by their sprints, and they often lack sufficient resources to explore and address all security aspects of their application in the available time. If an organization deploys an application with little or no security validation, it is exposed to a greater risk that the application could be exploited. Organizations must plan ahead and define an acceptable threshold for vulnerabilities within a production-class application. For example: Organization X ships releases for its Acme web app every two weeks. It runs security tests each time the application is built. Its policy states that if those tests find that the application build contains more than three high-risk vulnerabilities or greater than zero critical risk vulnerabilities, Organization X will block application delivery until the issues have been addressed and corrected.</p>
	Licensing options	<p>While AWS operates under the "pay what you use" model, many third-party vendors allow customers to deploy products directly on AWS's infrastructure. Leveraging third-party applications and tools can quickly increase licensing costs for your organization. Take precautions when deploying third-party applications and tools on AWS infrastructure, because your organization will incur both AWS infrastructure usage and software licensing costs. Licensing costs can be charged in a few different ways. They may be billed to the organization on an annual basis or perhaps by the hour. Understanding and planning for expected licensing costs will ensure you are not caught off guard by large invoices from AWS.</p>

Technical Considerations

	Consideration	Details
	Technology deployment	<p>Organizations should plan ways to implement their application security in a repeatable, consumable manner. Security teams can provide guidance in this matter in a variety of ways. Within AWS, applications can be deployed through a fully automated "pipeline"; alternatively, they can be deployed in an ad hoc fashion. An organization would be wise to create small, repeatable security tests as part of the deployment process, and to continuously refine those tests as the application matures. Understanding how your organization deploys its applications will allow the security teams to create and deploy effective security tests that align with the developers' deployment plan.</p> <p>Organizations need to decide if they will allow OSS or unsupported technologies. While it's true that an open source application allows insightful visibility into the application's security, it's also true that open source projects do not come with the luxury of customer support or SLA. If you plan to use open source technology for critical tasks or security assurance, you will need to ensure you have a proper plan in case the tool stops working at some point. On the other hand, OSS tools offer some unique opportunities. Organizations can take advantage of free open source tools and, as their needs outgrow the capabilities, modularity or support level provided by the OSS tooling, they can transition to more professional offerings.</p>




	Consideration	Details
	Application stack	<p>AWS Marketplace offers many tools for securing your organization's applications. Leverage any available open source testing software to get used to integrating security tools into your application development process (and save costs). Static analysis tools (linters) allow you to check your code for programming errors, bugs, stylistic problems and suspicious constructs. Each programming language has its own set of linters, most of which can be installed directly within your developers' preferred integrated development environment (IDE). Having developers use a linter within their IDE saves time in the development process by catching the errors before the application code is pushed. Catching these issues before the application is deployed makes it easier to mitigate them after deployment.</p> <p>Organizations should also consider their application stack and what corresponding Static Analysis Security Testing (SAST) tools might best fit their deployment pipeline. While linters check for bugs, syntactical errors, programmatic errors and code nuances, the purpose of SAST tools is to identify security issues in the application source code (versus during compilation or runtime). As with linters, each language has its own set of SAST tools, so your organization needs to understand the application code being implemented and what the information security teams will need to deploy to validate the codebase.</p>
	Pre-deployment security (inline)	<p>The largest challenge of inline scanning is the time it takes scans to complete. If your organization needs to deploy an application change, your security test should not require a long time to run. Imagine making a small configuration change to your organization's application. You push your code to the development pipeline, and now you have to wait 30 minutes for the security tools to scan your changes. Developers can push these changes many times a day, so waiting for these scans can be frustrating. We recommend that inline scans should not take longer than five minutes (depending on the size of the codebase). Your organization might also want to consider scanning only the changes to the code from the last push (delta scan). This method saves time but may be better suited to more mature organizations. It also makes sense to occasionally scan the entire codebase outside of the pipeline (out-of-band scans).</p> <p>We advise that organizations take small, repeatable, incremental steps in deploying inline scans for application pipelines. It's a good idea for your security team to have its own source code repository where it stores its tests. After a test has been created and validated, it can be stored in the repository. Once the code is in this repository, it may be shared with the developers, and they can include them within their development pipeline. You can work with the developers to ensure that the latest copy of the security test is always referred to when inline scanning. This procedure allows the security team to update the test as it sees fit. Because the development team has the latest copy of the test always being pulled into the pipeline, there should be no additional work when the security tests are updated. Leveraging this approach allows you to continuously test applications, update the tests and keep track of what exactly was changed via revision control.</p>
	Post-deployment security (out of band)	<p>Organizations will need to decide when to implement post-deployment security scanning. We mentioned out-of-band scanning earlier: If scans take too long to complete, they can be scheduled after the application has been deployed. Full scans by Dynamic Application Security Testing (DAST) tools can take hours to run, depending on the application size and scope of the scan. The following are examples of tools that should be run outside of the deployment pipeline:</p> <ul style="list-style-type: none"> • Infrastructure scans—These can take a long time depending on the scope of the resources and security checks the scan performs. • Dynamic application security scans—These require the environment to already be up and running. Like infrastructure scans, these scans can take some time to complete, depending on the organization's scanning scope. • Full web application security scans—Depending on the parameters of the test (credentialed/no-credential/spider/full active scan) and the size of the application, this scan can take a long time to run and should not be used inline. <p>Organizations will need to decide what is necessary to test and ensure application security for applications that have already been deployed. Solutions such as infrastructure security scanning, WAF implementation and DDoS protection should be evaluated.</p>

Operational Considerations

	Consideration	Details
	Processes and procedures	<p>Organizations may need to create or modify processes and procedures for security web applications in AWS. While some existing processes and procedures may work without modification, hosting applications in AWS means different methods of application delivery.</p> <p>Organizations may want to start to include developers and key individuals involved with application delivery in meetings and discussions about application security testing. Security teams might also want to sit in on development meetings and inform discussions when application security concerns arise.</p>
	Resources and deployment synergy	<p>Security in AWS and the applications deployed within the cloud will take dedicated resources to ensure that the proper policies and procedures are followed. Organizations must be cognizant that resources will need to be dedicated in such an effort, and they should plan accordingly.</p> <p>Organizations should consider which approach they would like to take with their cloud application security and the level of responsibility for each team involved within the process. Development and security teams within your organization need to take responsibility for the security and integrity of the application.</p>

AWS Implementation Considerations

Application Security

	Consideration	Details
	Cloud context support	<p>Application deployment leverages many ephemeral resources that support application delivery. Catalog all possible resources used within the deployment process for identifying any issues.</p> <p>Evaluate:</p> <ul style="list-style-type: none"> • The additional cloud context (tags or image IDs, other possible ephemeral resources) captured within the development processes (phoenix servers, artifacts and the like) • Logging and cataloging of the cloud resources for traceability and troubleshooting
	Deployment	<p>Deployment methods for security tools within AWS can vary depending on the development pipeline. Organizations should deploy these tools within the context of the development pipeline.</p> <p>Evaluate:</p> <ul style="list-style-type: none"> • Installation and initial configuration for tools • Possible use of professional services to aid or accelerate tool deployment • Programming tools and languages used in the applications and their corresponding DAST/SAST tools • The availability of managed or SaaS components or preconfigured appliances from AWS Marketplace • Leveraging AWS-native services for security implementation
	Integration	<p>Integration of application security tools into current processes/procedures ensures security teams can respond to risks. Integrating application security tools into the development pipeline allows for visibility, deployment and management. It also provides ease of use for security and development teams.</p> <p>Evaluate:</p> <ul style="list-style-type: none"> • The development pipeline process and how to embed security tools and scans inline within a reasonable time • Tools that integrate with current security solutions (SIEM, SOAR, IT service management) • API support (REST APIs available, SOAP APIs available, other available programmatic APIs) • Use of custom plugins or integrations • Integrations with native AWS services

Making the Choice

To summarize, the key considerations for implementing application security in AWS are:

- Cloud context
- Deployment
- Integration
- Configuration and iteration
- Reporting

Evaluate Your Organization's Current Deployment Process

There are many ways to deploy applications with AWS, and many methods with which to build out your deployment pipeline. When defining your proof of concept, include significant members of the application deployment team and ensure you understand their method of deployment infrastructure (Amazon EC2, Amazon ECS, serverless) and deployment pipelines (AWS CodePipeline, Jenkins, other deployment tools). Once your organization has a strong understanding of the deployment process, it can begin to evaluate its needs and considerations for security tools. Define a proof of concept that meets both your organization's considerations and the developers' current deployment process.

Define a Plan and Implement

By defining and understanding its cloud architecture, risk profile, business requirements and available resources as well as all the possible deployment methods within AWS, an organization should have a clear idea of its road map for application security protection. Understand that defining application security that meets all the discussed considerations is nearly impossible, so define and use what works best for your organization.

The best course of action is to define a proof-of-concept plan based on the considerations and implementation options. Ensure that your organization's development team is included in this process, because they will have a very strong understanding of the application and which security concerns to note. Once you have planned, developed and validated your POC, development and security teams can start defining a repeatable process for integrating app security within the development process. In this stage, your organization should work with the development team to identify the team's current security issues and how the developed POC will help secure the application and reduce the application's risk to meet the organization's risk threshold.

Conclusion

Application security is a crucial step for organizations' cloud security strategy. Having a defined plan and integrating security within the development process allows for greater visibility within the application delivery process, visibility into the security stance of the application and a defined remediation process for application security vulnerabilities.

Work with the development team through each stage of the DevOps life cycle (see Figure 1). Plan with the developers, join meetings when they develop and discuss their applications, ask the developer team for help when writing security tests in the verification stage, add out-of-band security (WAF protections, EDR solutions, DDoS protections and the like) in the release stage, and constantly monitor the security state of the application through your infrastructure monitoring and log analysis. Security tools and checks can be applied to many stages of the development process.

Keep in mind that this process should always be repeatable and easy to use. Start small and build from there. To get started today, consider an evaluation of some of the solutions readily available via the AWS Marketplace. You may also consider leveraging a SaaS solution to jump-start your organization's journey into AWS application security.

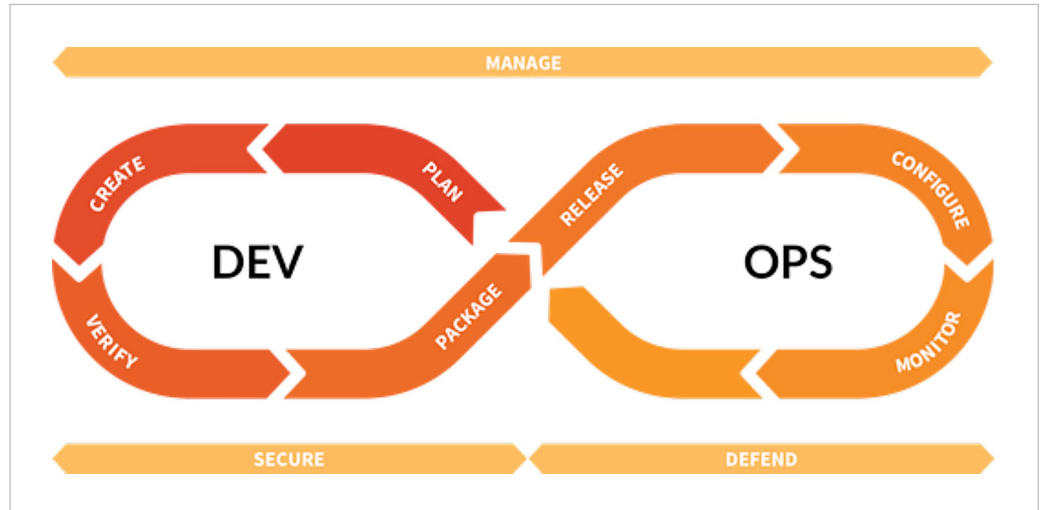


Figure 1. The DevOps Life Cycle

About the Author

Nathan Getty holds the GWAPT and GCIA certifications, and he recently won the SANS Cyber Defense NetWars competition, a defense-focused challenge that tests the ability to solve problems and secure systems from compromise. Nathan currently works in the Canadian insurance industry. In his organization, he focuses on cloud security, including AWS onboarding, and developing best security practices and general security/cloud insights. Nathan also focuses on driving DevOps methodologies in the company's security program, implementing continuous delivery platforms to allow smoother development and improvement of internal security applications.

Sponsor

SANS would like to thank this paper's sponsor:



in conjunction with



About Fortinet

Fortinet breaks down the barriers that inhibit security visibility and management across private, public, and hybrid cloud platforms. The Fortinet Security Fabric for AWS helps organizations maintain consistent security protection in a shared responsibility model, from on-premises to the cloud. It delivers comprehensive and fully programmable multilayered security and threat prevention capabilities for AWS users. At the same time, it streamlines operations, policy management, and visibility for improved security lifecycle management with full automation capabilities. Visit www.fortinet.com/aws for information on Fortinet's application security solutions.

RETURN TO THE
TABLE OF CONTENTS



How to Secure App Pipelines in AWS

Written by **Dave Shackleford**

September 2019

Sponsored by:

AWS Marketplace

Webcast

You can access the associated webcast at:
<https://pages.awscloud.com/HowTo-AppSec>

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.

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.

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.

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

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)

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.

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.
- **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 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.

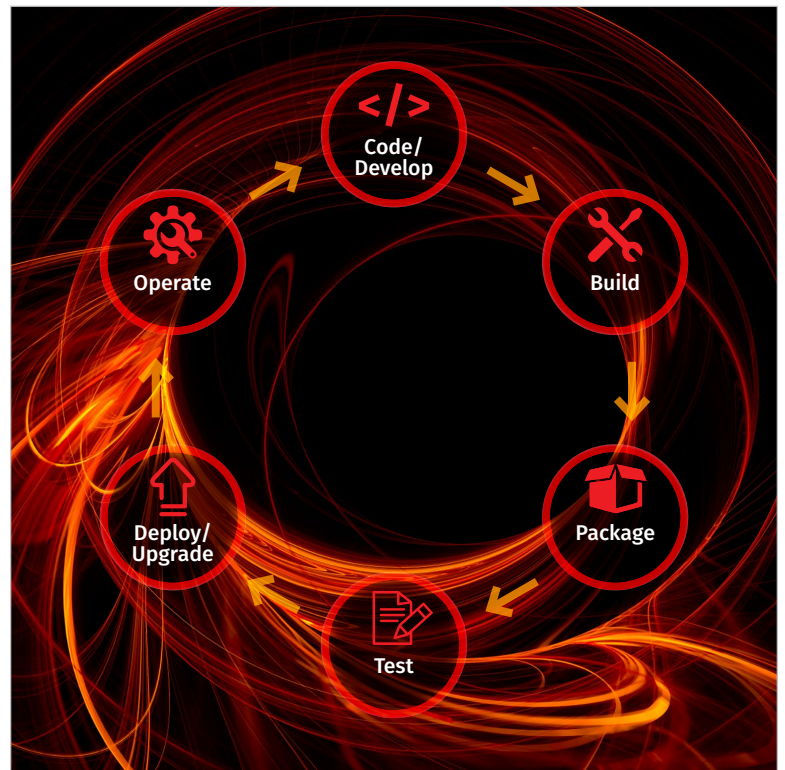


Figure 1. Phases of Application Development and Deployment Pipelines

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.

</> 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.

✂ 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.

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

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.

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.

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.

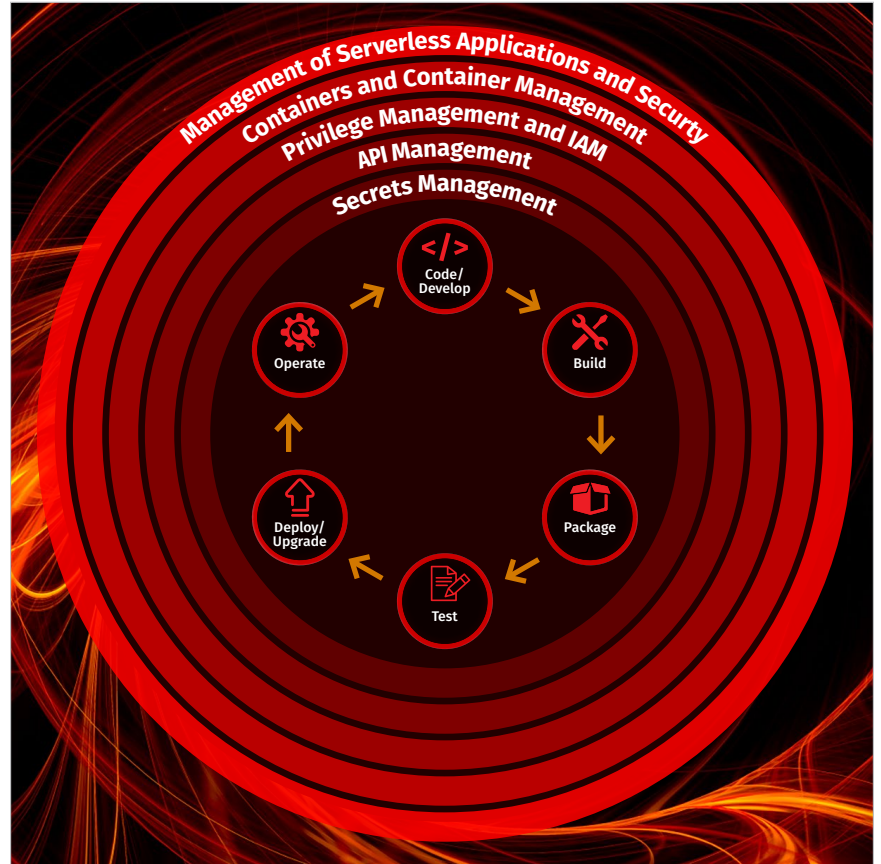


Figure 2. Additional Security Considerations Throughout the Life Cycle

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 Shackelford](#), 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:



RETURN TO THE
TABLE OF CONTENTS

Next Steps

By applying the guidelines in the preceding whitepapers, you have been able to:

- Develop a threat modeling process for your organization’s cloud-based applications
- Understand how DevSecOps and CI/CD processes work and where the key areas for security integration and influence are
- Document an application security plan and architecture, as well as document key changes to existing security processes and controls to ensure security in cloud-based applications

Using these guidelines, the key next steps are to first make sure the security team has the knowledge and skills to understand how the DevOps approach is in use by the IT organization and how best to integrate security controls and processes. Then the CISO or security manager needs to communicate with the app development team and establish working relationships to assure security is included in all sprints or other rapid development cycles. Key to this is communication: Avoiding vulnerabilities needs to be sold as part of building quality in—that really is an easy sell these days.

Improperly implemented DevOps programs at their worst can be an exercise in doing the wrong things faster. However, when application development and security work together and truly implement an integrated DevSecOps approach, organizations can meet the business demand for faster application time to market while ensuring security.

[RETURN TO THE
TABLE OF CONTENTS](#)