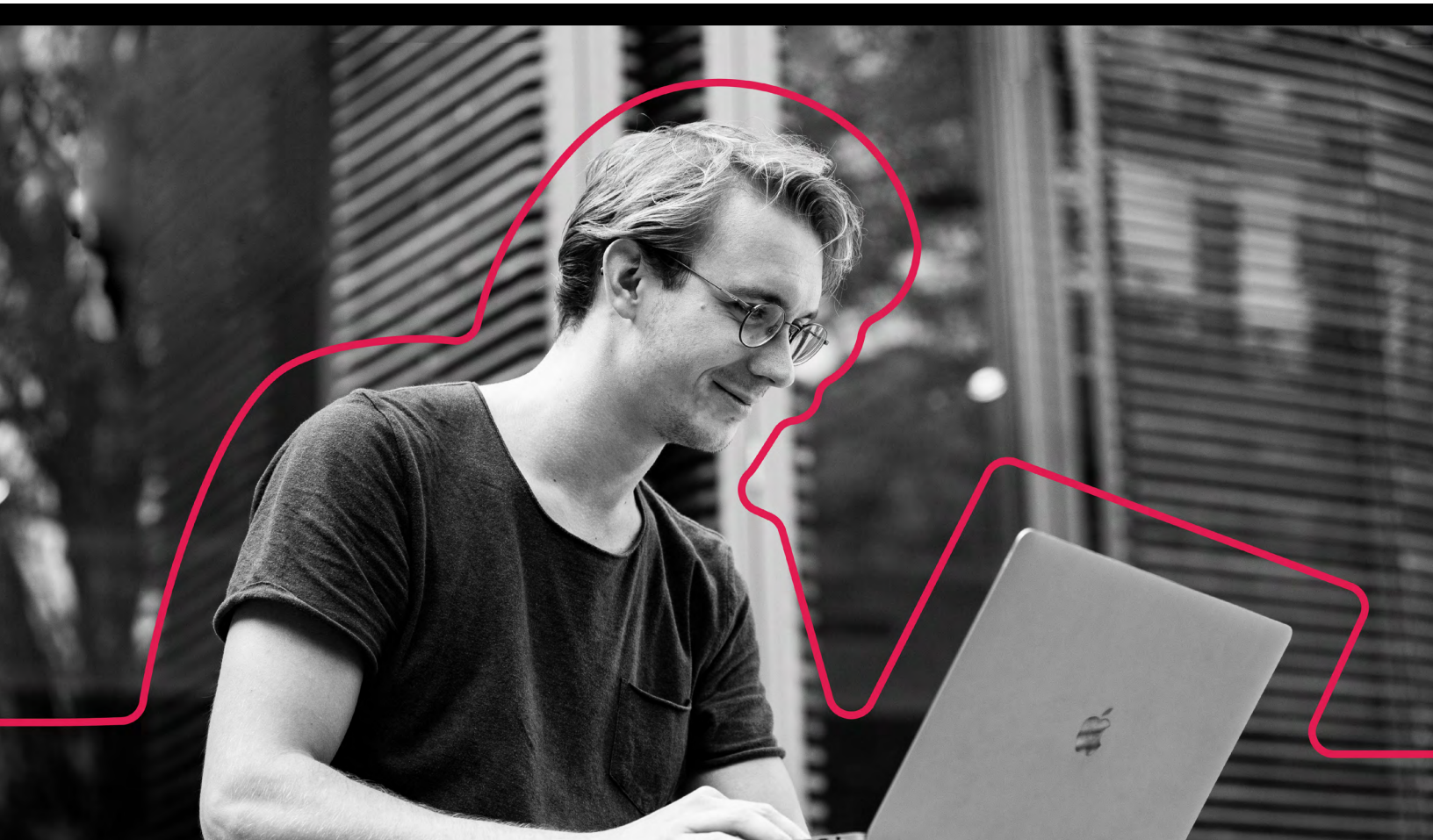
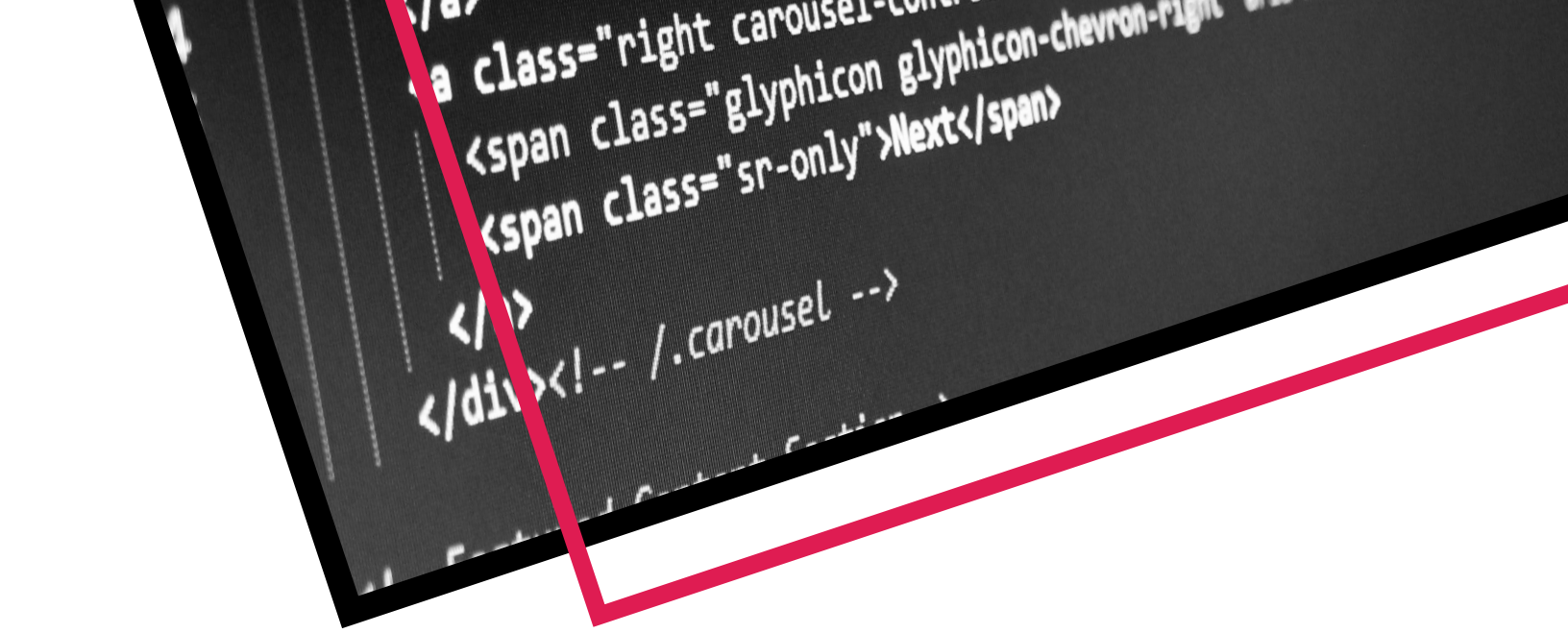
cprime

DevSecOps PLAYBOOK

A Guide by Carlos “Kami” Maldonado

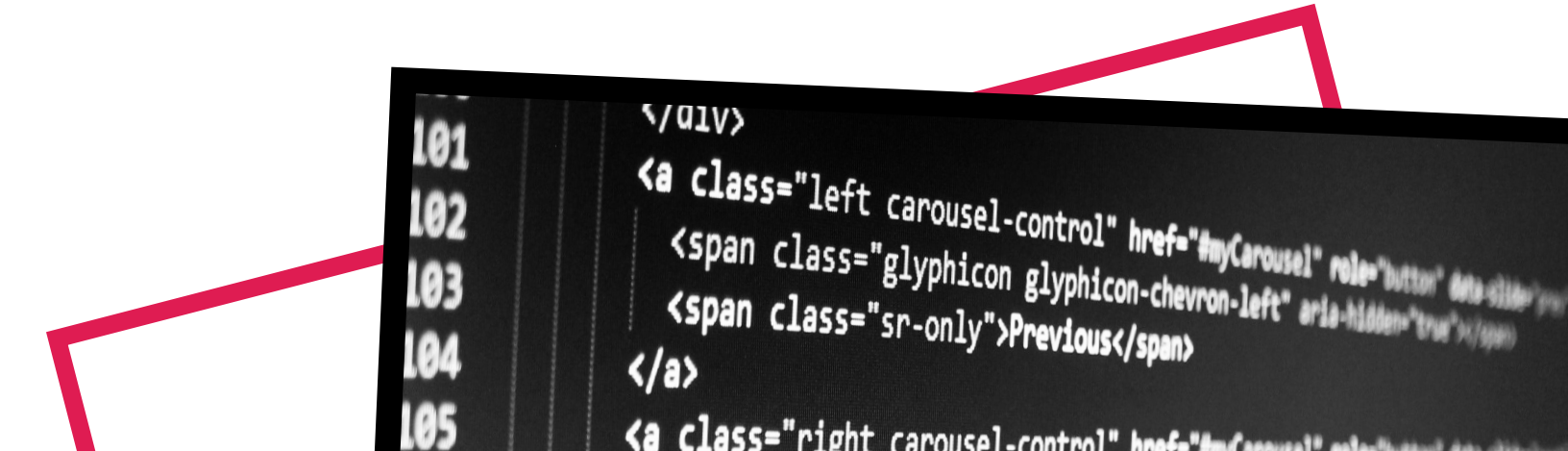




In the IT industry, practicing DevOps has become a predominant way to improve the efficiency of teams. DevOps isn't just a trendy buzzword; it's a cultural movement. Adopting its methodologies will help your organization in terms of both performance and outcomes. Its real, demonstrable results make it understandable why DevOps has taken the world by storm.

But is DevOps missing a key element?

Yes—in this paper, we'll argue it is missing something. Security is that missing piece. And because security is so important, we'll first explain why you should make it a primary consideration. But you'll need the "how" once you know the "why." So we'll then give you a thorough road map, showing you to path to DevSecOps. This will allow you to enjoy all the benefits of a DevOps transition without leaving out the vital aspect of security.



SECURITY IN THE SOFTWARE DEVELOPMENT LIFE CYCLE

Companies undergoing a DevOps transformation often underestimate the security aspects behind software creation. Security checks traditionally happen at the end of the development process, and they're a gateway before a product goes live. Such behavior is not compatible with the DevOps principle of "deploy often." Your manual security checks get in the way of continuous integration and continuous deployments. It creates a gap; DevOps increases business velocity but security is left behind (which we'll talk more about in the next section).

DevSecOps, as opposed to just DevOps, changes this way of working. Whereas security was previously just an afterthought before deploying to production, DevSecOps makes security an integral part of your development and operations practices.

The Proportion of Security Experts to Developers

According to Sonatype's 2018 DevSecOps Community Survey, there's only one security expert for every 100 developers in an organization. That's a staggering disproportion, and no matter how fast a security team is, they won't be able to keep up with the velocity resulting from DevOps practices.

In the DORA's 2018 Accelerate: State of DevOps report, James Wickett, head of research at Signal Sciences, estimated that there's one InfoSec person per 10 infrastructure people per 100 developers in large companies. Wickett also observed that security is more of an afterthought at companies, involved at the end of the delivery life cycle. And he rightly points out that security improvements at that stage aren't just agonizing for the employees involved—they're also expensive for the business.

Meanwhile, all findings in DevOps research emphasize one thing: everyone needs the "sec" in DevSecOps, and far too few companies prioritize security enough to prevent disaster later.

Why It Makes Sense to Automate Security

DevSecOps means embedding security checks in every phase of your software development life cycle. But deploying code frequently makes it harder to implement preventive and reactive security measures. The logical path, then, is automating your security procedures. This way, there won't be a person acting as the gateway but rather an automatic check with precise requirements.

YOUR ROADMAP TO DEVSECOPS

Hopefully at this point, you realize how important security is to consider in a DevOps operation—or at least how punishing it can be if you neglect security. So, what comes next? How do you begin to turn your DevOps organization into a true DevSecOps organization? Well, you're in luck: the rest of what we discuss here will serve as your roadmap to DevSecOps.

First, Learn and Promote Core Concepts

As is the case with every cultural change, it's important that you can count on support from management. And support means more than a single email blast from the CEO talking about how exciting she finds the new transition to DevSecOps. Organizations need to align budget and strategic decisions with this new way of doing things. People need the means to learn through books, training, internal meetups, and conferences.

What's more, managers and technical leads need to carry a consistent and coherent message about the new change. They should make it easy for teams to learn about their new DevSecOps ways. And of course, managers should let their reports ask questions and express concerns in private, without any worry of being judged.

If you're working against a particularly stubborn culture, seek out early adopters in each of your developer teams and convince them of the importance of DevSecOps. Though conventional wisdom says change comes from above, these early-adopter developers could be your allies, triggering a groundswell of change from an unexpected part of the organization.

Use the Concept of "Security As Code" As a Stepping Stone

Following suit from "infrastructure as code" in DevOps, "security as code" will be your stepping stone to achieve DevSecOps. Every security check needs to be reproducible and available in the form of code to all interested parties in the organization. And it needs to come from a single source of truth, like a Git repository.

Implementing security as code will enable developers to:

- Learn the nature of security checks
- Know earlier about security requirements
- Start thinking like an attacker, when the situation calls for it

And security experts will benefit from:

- Decreasing the execution time of their assessments
- Increasing visibility and usability of their tools for the rest of the teams
- Sharing their source of checks instead of relying on other team members

YOUR ROADMAP TO DEVSECOPS CONTINUED

Automate Security Testing

Having security checks automated in your software delivery pipeline means that everyone triggers them each time something happens—like on every commit or every merge request. It will enable developers to know if any new code they've added complies with their current security policies. This is a way to reduce friction between security professionals and developers. That's because checks are automatically performed by machines, not by people who could be perceived as gatekeepers who hinder you from completing work.

Self-Service Security Assessments

When DORA asked organizations about the characteristics that impacted software delivery performance, forty-six percent included implementing on-demand self-service of computing resources as one of the top five. Now, let's apply this lesson to the security field. If people need to submit a ticket in order for a security check to happen, they'll avoid checking their compliance. Developers and operators need to have the independence to perform security reviews on their own work and see their results with transparency.

Shift Left to Improve Early Discovery

Shift left. It's a well-known pattern from DevOps practices. In fact, a 2002 report from the National Institute of Standards and Technology supports the idea that shifting left can help any organization. Numbers in that report point out that the relative cost to repair a defect (in person-hours) increases by a factor of five to thirty when you discover bugs later in the software development life cycle.

As you can see in the table below, the earlier you find a defect, the cheaper it will be to fix. A software defect found after a product release could cost you up to thirty times more resources to fix.

Example of How Much It Could Cost to Repair Defects, Depending on the Stage in Which It's Foun				
Requirements Gathering/ Analysis/ Architectural Design	Coding/Unit Test	Integration and Component/ RAISE System Test	Early Customer Feedback/Beta Test	Post-Product Release
1X	5X	10X	15X	30X

(Table adapted from NIST report)

NEXT, SET EXPECTATIONS FOR YOUR TEAM

If you're following this roadmap to DevSecOps, the next thing you should do is set clear expectations for any new rules to which you'll hold your source code. And don't forget to keep your developers well informed about the new rules of engagement. The Open Web Application Security Project (OSWAP) publishes several documents every year, but there are two particular documents to review. These are:

- The OWASP Application Security Verification Standard Project
- The OWASP Proactive Controls

The first document shows you how to take inventory of your organization's current security status. The second document gathers a comprehensive set of good practices and recommendations to improve security in the software you develop—and the infrastructure in which you run it.

Capture Baseline Data

As a result of analyzing your software development pipelines and environments using OWASP's Application Security Verification Standard, you'll have a better idea of your current security status. Then you'll need to weigh the needs of your business and prioritize tasks to mitigate your weaknesses. But new work shouldn't go exclusively to your operations and security teams; it should be developers' work too. This is the shift in mindset that needs to happen at your company: security needs to become a concern across your whole organization.

Review OWASP Proactive Controls

Now that we've established that security is everyone's concern, it naturally follows that every developer in the organization should be aware that they need to build security into their code. Security experts might be familiar with the OWASP Proactive Control 2018, which is updated every year. These are a set of guidelines organized in order of importance. The updated list for 2018 is:

- Define security requirements
- Leverage security frameworks and libraries
- Secure database access
- Encode and escape data
- Validate all inputs
- Implement digital identity
- Enforce access controls
- Protect data everywhere
- Implement security logging and monitoring
- Handle all errors and exceptions
- Use these criteria to learn best practices so your teams can write secure software that in turn operates in secure environments.

C

PERFORM THREAT MODELING

A third group activity that you'll need is threat modeling. Its purpose is identifying, understanding, and recording threats in order to protect something valuable to your business.

This is when your team should review your software and infrastructure design, mimicking the way that an attacker would analyze it. A threat model usually includes:

- A model of the system you want to protect
- Assumptions to challenge
- Potential threats to the system
- Actions to take against each potential threat
- Methods to validate the model and risks
- Ways to verify your success

You'll need several iterations before you grasp a comprehensive view of your software development cycle and the environments you maintain. It's also a continuous process because every time you add code or change a service version, you're changing your model by adding or removing risk factors.

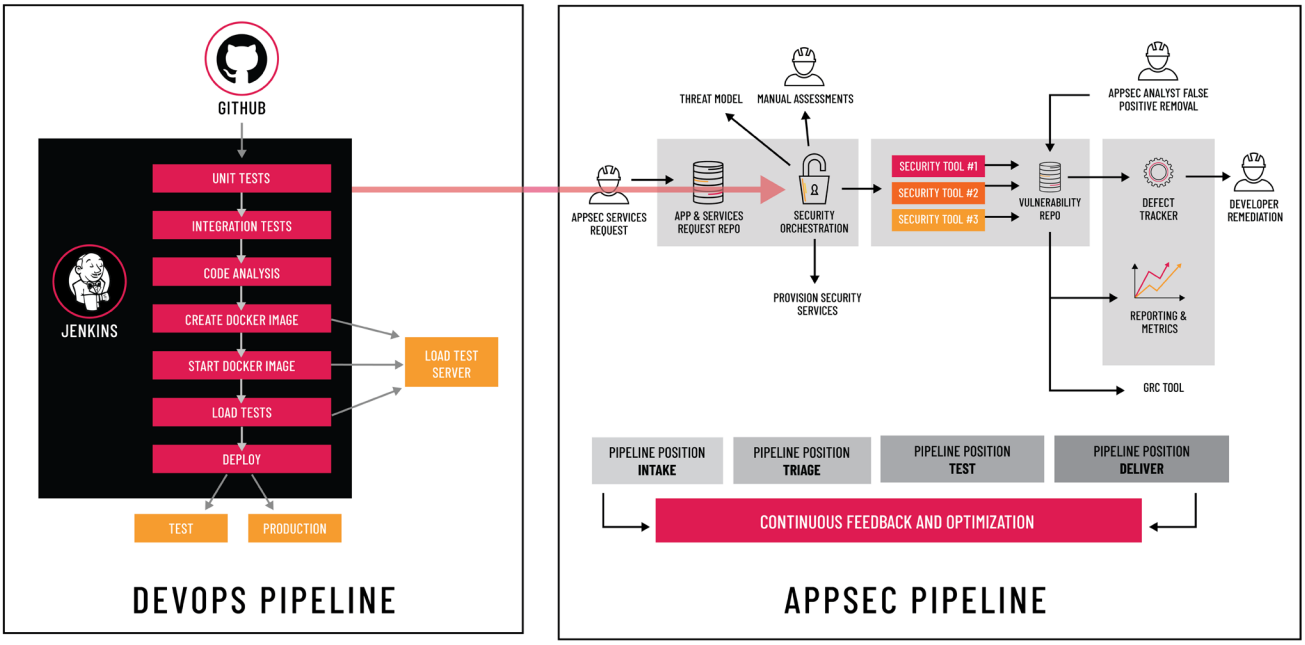
Pick a Reference Architecture

Sonatype has a relevant document with around fifteen different DevSecOps reference architectures from many sources in the industry, ranging from the USA's Department of Defense to Jim Bird's DevSecOps to Accenture to AWS. So that's where you can begin to pick a reference architecture—by researching how others do it. When you're ready to move forward, the next step is to evaluate the size of your company and how your team works. From there, you can design a workflow that will allow you to deliver value without slowing you down.

PERFORM THREAT MODELING CONTINUED

Design Your DevSecOps Workflow

For the sake of simplicity, TeachEra.io's workflow shown below will be used for reference:



From TeachEra.io's DevSecOps Studio (license)

On the left, we have a generic DevOps pipeline, similar to what you might have implemented in your organization. In order to decrease disruption to current processes, my recommendation is to insert AppSec steps, like what's pictured on the right.

Each AppSec step should be relevant to the DevOps phase before it. For example, dependency checks and static application security checks should be done during code analysis—or any previous step in the pipeline. Any penetration testing or simulated attack should happen in a test environment, as similar as possible to production.

SELECT TOOLS FOR EACH PURPOSE

Each phase of the software development life cycle has a component consistent with a stage in your pipeline. I'll describe six different tools, which will give you a place to start.

Tool One: Automated Dependency Checks

Software and its dependencies could be compared to living beings that evolve over time. While developing software, reusing others' libraries is the shortest path to achieving your goals. A modern JavaScript program has a median of fifty-four transitive dependencies. Is your team going to review all these dependencies one by one manually? I should hope not!

In contrast, you could consider reviewing dependencies using a software composition analysis tool from OWASP. This will allow you to execute a dependency check on all your code. You can use it as a Jenkins plugin, a CLI command, a Maven plugin, or a Docker container. It will create comprehensive reports, along with the relevant common vulnerabilities and exposures (CVEs) identifiers for vulnerable frameworks it found implemented in your code.

Tool Two: Static Application Security Testing

These are the kind of checks you perform on applications before they're even executed for the first time. You could perform tests like syntax, style, and lint checks. But there are also ways to check for security vulnerabilities. Choices will vary depending on your programming language. For example, if you program in Ruby, there's Brakeman and RuboCop.

A couple of projects worth mentioning are Find Security Bugs and SonarQube. They're popular in the Java-sphere. SonarQube Scanner for Maven can be configured to analyze source code going automatically through Jenkins so that a "waitForQualityGate" step will pause the pipeline until SonarQube completes its analysis. Then it returns quality gate status to your Jenkins pipeline.

Tool Three: Dynamic Application Security Testing

Dynamic application security testing is also known as "black box" testing. It's performed after your web application has reached some level of compliance to be available in an internal environment. At that point, you can start testing it against cross-site scripting and SQL injection. Creating the payloads to automate this type of testing could be complicated, and what you choose to do will heavily depend on your application functionalities.

I'll provide the names of a couple tools that you can integrate into your pipeline. Both are open-source, but I recommend getting help from someone already familiar with the context to operate them.

- One option is OWASP Zed Attack Proxy as a Jenkins plugin, which can help you automatically find security errors in your web applications while you're developing and testing.
- The other option is w3af, which will allow you to check web applications for 200+ common vulnerabilities, including cross-site scripting, SQL injection, and OS commanding.

It's worth noting that if your organization implements Jira to handle internal requirements, Zed Attack Proxy has a feature that will create a Jira ticket for each security issue that it finds.

Tool Four: Fuzz Testing

Fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as input to your computer program. Our purpose is to monitor the program for events such as crashes and unhandled exceptions.

Fuzz testing usually implies six stages:

- Identification target programs to interfaces to fuzz
- Creation of fuzz data (malformed data)
- Delivery of fuzzed data to the application under test
- Monitoring the software for signs of failure
- Triaging results
- Identifying contributing factors, fixing bugs, and rerunning failures on analyze coverage data (rinse and repeat!)

If you'd like to see more, you can take a look at Daniel Miessler's set of fuzzing data designed for different purposes.

Tool Five: Penetration Testing

Also known as pen testing, this is one of the most sophisticated types of security checks. It simulates an orchestrated attack on an organization, with the organization's permission. The goal is to identify vulnerabilities and security issues so that you can catalog them and slate them for further review. Developers, operations, and security experts should work with these results to implement temporary or permanent fixes.

The quintessential ethical hacking tool—the most recognized name in the space—is Metasploit. It's multi-platform and runs on Linux, macOS, or Windows. What's more, it enjoys abundant documentation. And for those looking to include it in automated custom workflows, the pro edition by rapid7 can be automated through "task chains."

Tool Six: Automated Security Attacks

For your last phase of security compliance in your software development cycle, you should consider a fully automated and reproducible security attack. Let's start by describing how such a test with BDD-Security would work:

- BDD-Security is a Gradle project. That means you can import into a Java IDE, like IntelliJ, NetBeans, or Eclipse.
- These simulated attacks will navigate your website, just like a user on the other end of the browser, through Selenium scripts.
- These checks integrate with either OWASP Zed Attack Proxy or Tenable's Nessus.
- Attacks will run independently as a Gradle JUnit test, from an IDE or inside a Jenkins CI server.
- Their output will be HTML Cucumber reports, JSON Cucumber results, and JUnit results.

If your organization uses techniques like test-driven development, it will be easier for developers to implement BDD-Security as their security testing framework.

LAST IN THE CYCLE: IMPLEMENT AND KEEP ITERATING

By taking a cue from practices in this white paper, you should have at least a basic workflow with comprehensive security checks in each phase of your software development cycle. This will provide you with enough baseline data for your DevSecOps strategy. Vulnerabilities, test coverage, surface tested, and pending bugs to fix are some useful measurements that could help you to see progress after each new iteration.

Train, Implement, Measure, and Compare

As you may know, any digital transformation process is slow. User adoption takes time. But if you leverage training initiatives and show early results to influential engineers and managers, you'll be able to use this as a justification for more budget. From there, you'll be able to train people in these practices and extend the reach of your DevSecOps efforts.

Each new security check will provide new metrics, so you can measure and compare with your security check history. You'll be able to look at past performance, compared with your current performance, and show business stakeholders how your security practices have evolved and security incidents have been reduced.

More Opportunities for Improvement

Once your DevSecOps workflow is mature enough and you have enough people involved, there will be other activities that you will be able to implement. Here are just some of the things you can have with a DevSecOps organization:

- **Red teams.** These mimic external attackers that want to make system vulnerabilities evident—including those vulnerabilities found in proof-of-concepts—so they don't leak into final products.
- **Blue teams.** These respond to external attackers by implementing temporary or permanent fixes to improve security.
- **An evangelist in each team.** These are volunteers who are enthusiastic about security, so they continuously promote a mindset that values InfoSec.
- **Bug bounties.** This is a way to motivate actors in the wild, under strict disclosure guidelines, so they warn you about vulnerabilities with a certain degree of formality. A good example is the FOSSA bug bounties program that currently exists in Europe.
- **Blameless postmortems.** These are a judgment-free review of actions by your team during an outage incident, and they're a useful learning resource when you have trouble in production.

  Specifically, a blameless postmortem will involve an engineer in charge of the incident reviewing what happened in a storytelling format, with input from every actor. The purpose is to find out what led people to make decisions. As a result, you'll improve your incident response process. With blameless postmortems, I certainly want to stress some key facts:

  - Every incident is an opportunity to learn about your team's incident response capacity.
  - You cannot attribute a single factor as the root cause for an incident. There will always be many contributing factors.
  - Human error can't be a reason behind an incident. You cannot forbid people to make mistakes. System design should be resilient in the face of human error.

LAST RESORTS: GETTING OVER THE INITIAL DEVSECOPS HURDLES

Hopefully the information we've shared here has given you ideas about how to include security in your current DevOps practices. If your organization wants to include security checks in your software development life cycle, you could probably use a hand from external consulting or training to guide your teams in the process. It's an opportunity to evaluate offerings in this field, like Cprime Learning's DevSecOps Boot Camp.

One last note: if your company is too small to hire full-time security experts, consider bringing one in! Don't hesitate to hire an external consulting firm to help you design your DevSecOps strategy, especially as you're first getting started. Cprime is a global consulting firm ready to help your transforming business get in sync. There's no shame in taking guidance from those who know best how to implement DevSecOps.

We wish you the best in your transition to implement DevSecOps. Be patient and flexible, react fast, prioritize with a cool head, and keep moving towards being a secure organization.

DEVSECOPS BOOT CAMP

3 Day Classroom Session | 3 Day Live Online | Custom Onsite

COURSE OVERVIEW

Led by a senior expert, teach your teams how to improve the DevSecOps practice – from guiding principles to daily technical execution.

This DevSecOps training boot camp is the most practical, in-depth educational solution for teams who want to understand, apply and improve their skills on "shifting left" in IT security. This expert-led boot camp focuses on the principles, processes, and technical skills necessary to make security and risk profiling a front-end priority: embracing a "quality first" mindset.

YOU WILL LEARN HOW TO:

- Assess, specify and automate much of the work associated with application security
- Bridge the typical functional silos in IT that prevent proactive security practices.
- Translate common risks into technical use cases and software requirements.
- Apply "security first" engineering and testing practices throughout the entire application pipeline

INDIVIDUAL	CUSTOM ONSITE	
$2450.00	8+ TEAM MEMBERS	Request a quote online or call (877) 753-2760

For more information on our DevOps curriculum, please visit:
www.cprime.com/learning

visit www.cprime.com
or call (877) 753-2760

cprime

ABOUT THE AUTHOR

Carlos “Kami” Maldonado is a Senior System Administrator with BOND Enterprises and a Digital Transformation Trainer with Cprime Learning. Kami has over 16 years of experience in DevSecOps, Site Reliability Engineering, Scrum Master, Technical Writing, and Cloud Architecture. Kami has on the ground experience with production environments, managing thousands of instances running java services, in transition to micro-services.

ABOUT CPRIME

An Alten Company, Cprime is a global consulting firm helping transforming businesses get in sync. Cprime is the partner of choice for Fortune 100 companies looking to achieve value and agility. We help visionary business leaders compose solutions, execute implementations, and exceed against business goals. With our key partnership recognitions, including Atlassian Platinum, AWS Advanced, and SAFe Gold SPCT partner, our industry-leading software and services work in synergy to deliver transformations.

Visit us at **www.cprime.com**