A Tale of Best Frenemies: How to Embed Security in the DevOps Mindset

Okta Inc.
100 First Street
San Francisco, CA 94105

info@okta.com
1-888-722-7871

Table of Contents

Introduction 3

Exploring the DevOps mindset 4

The role of system passwords in DevOps security 5

Other considerations for a secure DevOps methodology 6

What is one thing that businesses need in order to stand out against their peers? Speed to market.

Responding to this key concept, the DevOps mindset was born as an opportunity to enhance how products are designed and deployed. When focusing on speed and agility, embedding important security measures across the lifecycle can feel like a necessary evil. But in order to move quickly while protecting the business from potential vulnerabilities, DevOps and security need to be friends—or at least, frenemies.

The reason many companies have embraced the DevOps model is simple: by aligning their developer and operational goals, they're able to keep pace with modern innovation by designing and delivering products quickly and seamlessly. But at many organizations, teams forget security is a major component of operations and all too often it's not being included in these critical collaborations, which can lead to disastrous results.

When security is an afterthought in the DevOps process, it's either tacked on at the end of the development lifecycle, or worse, a product is shipped without adequate protections in place. Ultimately, this approach to security is costly and time-consuming, and compromises the integrity of end-user data.

While DevOps is disrupting traditional forms of software development—transforming operations, driving productivity gains, and bringing out the best of engineering capability—it's time security was brought into the fold. This whitepaper will address where security fits into DevOps, and outline how teams can adopt security-first practices at every stage of development.

Exploring the DevOps mindset

First, it's necessary to clearly define DevOps in context, as the term can mean different things to different stakeholders. For the purposes of this whitepaper, DevOps focuses on IT automation in backend infrastructure, where operations teams implement code to help facilitate the developer team's business-critical tasks. This in turn allows software builders to streamline the delivery of products, from planning to production and beyond.

In a DevOps mindset, speed is typically the first priority, and system design is the means to achieve it. If there's any point in the software development lifecycle that currently requires human intervention and decision-making, the DevOps mindset asks whether it should be an automated process instead.

However, while it's clear that security is integral to DevOps, it can often fall to the wayside as DevOps teams focus on other priorities, such as speed. Facebook's popular slogan "move fast and break things"—which has been adopted by many technology companies—is often taken too liberally, without the nuance of service stability. And with security on the back burner, breaking things can actually pose a significant threat to your business.

A robust system design requires solutions such as automated provisioning, service reliability, and rapid error recovery in order to be truly effective. These precautions have to be built-in to prevent lengthy and expensive problems later on—and the sooner security can "shift left" in the product lifecycle, the better the outcomes will be.

Here are a few examples of the automation and system design features that DevOps focuses on, and how security is a core consideration in each:

- **Infrastructure as code:** Operations teams take critical infrastructure such as load balancers, servers, and DNS records and put them into code form, in order to improve their reliability, repeatability, and in certain cases, their modularity. But infrastructure operations and software development are a continuous cycle, so in order to build secure products, security has to be brought as close to the design phase as possible and injected into this code from the outset.

- **Configuration management:** Just as we use code to orchestrate managing infrastructure for our systems, we also want to use code to configure those resources., This can include everything from Linux OS configuration files and Secure Shell (SSH) user accounts to Java code env properties or Apache conf files. Tools such as Chef, Puppet, Ansible, and SaltStack can help manage all of these systems in repeatable code. Don't forget that all of these configs can be accessed and changed by a bad actor if you don't monitor and secure them with an identity and access management solution. In short: lock down your access then monitor that lock.

- **Continuous integration and continuous deployment pipelines:** Due to their capacity to fully automate a software's lifecycle from test to push, CI/CD pipelines are likely considered the holy grail of DevOps. However, to maintain a secure environment, it's crucial to include security tests in your pipelines in order to catch as many vulnerabilities as you can to prevent any real incidents.

Too often, engineers and developers focus nearly all of their attention on setting up systems and making functional changes and not enough attention on managing access and permissions. The same way organizations can see a buildup of tech debt in their infrastructure when they don't evolve with the times, security becomes a burden when it isn't proactively incorporated into DevOps workflows.

The role of system passwords in DevOps security

To put it simply, automation should not proceed without security, but security systems can be deep and complex. To exemplify this, we'll take a look at system passwords and how they offer a nuanced and multifaceted method for securing an organization's data. Here are some of the questions DevOps teams should keep top of mind when reviewing their approach to system passwords.

1 Can system passwords be changed quickly in the case of an attack?

Any password that remains the same for an extended period of time is subject to inherent risks. System data breaches, brute force attacks, and database leaks could all compromise an organization critically if a long-standing password were discovered by bad actors. Therefore, the question organizations need to ask themselves is how quickly they can change their system passwords in the event of an emergency.

The answer to this question varies across companies, teams, and applications. Broadly speaking, one of the simplest and common strategies for changing a password is to use the SSH protocol and hand edit your passwords on the command line, but there are better ways! For example:

- ClusterSSH (CSSH) allows system administrators to SSH into multiple servers at the same time to speed up their response. This is still prone to human error, however.
- Ansible is powered by SSH and lets users push out remote commands in a fleet, and build playbooks and scripts to perform this programmatically.

Other configuration management tools such as Chef also offer great ways to change system passwords on short notice. And if necessary, you can use targeted resource applies on specific recipes/scripts and minimize your exposure to any unwanted changes.

Beyond configuration management, rollout processes can provide clear, written directions for responding to a compromised password, and define the servers that are the highest priority. But by working together, development, operations, and security professionals can implement proactive measures that prevent breaches long before they're threatened. For example:

- Canary servers can be used in production environments, allowing teams to roll out changes on a smaller subset of production systems before rolling them out to the full fleet.
- Post Deployment Validations can help identify more complicated service-related issues that aren't always immediately apparent or that aren't captured in more specific service health checks.

2 Can system passwords be changed without causing a live service disruption?

It's one thing to be able to update a system password in an urgent situation; it's something else entirely to be able to do it without disrupting the system's users. Updating a password can often lead to unexpected challenges, and in some cases teams may require a maintenance window to address it. It's important to have visibility into how clients are connected to a system, and how they are impacted when a system password is changed. These factors will often have to be assessed on a case by case basis.

A general solution is to do a blue-green style flip of access credentials, in which all clients shift to a new set of credentials before the original credentials are disabled or removed. This extra step should enable a transparent

transition for your clients that doesn't immediately revoke access to the system they are depending on.

3 Are these processes being validated continually?

All of the protective measures outlined in the previous sections are nothing more than theory until the assumptions are validated in practice. In order to do this, DevOps and security need to work together to troubleshoot, revise, and iterate on processes. Effective strategies include:

- Scheduling live demo exercises in testing and QA environments with responsible controls so that additional audiences can participate and bring their ideas, insights, and past experiences to the table.
- Eventually testing and validating these processes in production as well. That's where it's really going to count.

Of course, system passwords are just one small element of a business' security posture—but they can also require some heavy lifting to get right. Automating system password security and effectively embedding it within the DevOps lifecycle doesn't just help businesses avoid threats to their proprietary information and sensitive user data, it's also a great learning exercise and a bridge to exploring more complicated security risks in your organization.

Other considerations for a secure DevOps methodology

> Beyond system passwords, there are a number of other security areas within a typical DevOps lifecycle that require time and effort to get right. Effectively automating these features can help keep businesses secure, and improve user experiences in the process.

1 Account provisioning and deprovisioning

It's surprisingly common for security or operations teams to find active accounts provisioned to users who left their company more than a few months ago—and they still have frighteningly high levels of privilege and access. Problems like this can be mitigated with Okta Lifecycle Management and Workflows, which automate the process of onboarding users to their necessary accounts, and offboarding them as well.

2 SSL certificate revocations

Managing certificates requires vast amounts of knowledge and different tools to be done properly. It's imperative to be able to react should a private key become compromised. Security teams need to know how to quickly revoke a certificate and have the ability to audit the current status of all signed certificates.

3 Disabling lost or stolen MFA devices

Similar to revoking certificates, some authentication factors are contingent upon the user possessing a physical key or token. Measures need to be in place to disable these factors from a user's account immediately if they're lost or stolen.

4 Passwordless authentication

It has been said that the best password to set is no password at all. Many modern companies are maturing in their security

models, and problematic passwords are being superseded by more advanced factors such as biometrics and Okta Verify.

5 Access auditing and monitoring

When a user or system is making changes, it's vital to understand what those changes are, why they were made, and who made them, because it may be the work of a bad actor. Comprehensive monitoring and reporting is mandatory for modern security, and Okta provides this level of visibility.

The DevOps mindset wants everything to be automated, and fortunately, Okta provides the missing piece for most DevOps teams—automated security and identity management. By implementing intelligent measures from the start, your organization can eliminate the security hazards that are traditionally found across the product lifecycle, from design to testing to deployment. In the process, your DevOps team can stop viewing security as their frenemy, and instead see it as a valued collaborator.

For more information about how Okta can support you in your DevOps security journey, get in touch.