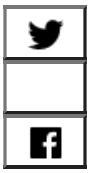
Effective DevSecOps

medium.com/@fabiojose/effective-devsecops-f22dd023c5cd

26 July 2018



Security isn’t a brochure with lots of rules and policies that developers and operators must follow. Security isn’t people arguing that they are the cyber security team that nobody knows what’s happening with projects or nobody is helping the development and operations to be secure.

Sometimes the security team of our company performs the enforcement of the rules and policies, through meetings, asking by e-mail or chatting to know how the security is, or worse, after a security leak performs a witch hunting to find the guilty. ***This is a typical way to get hacked.***

> Security is made all the time, everywhere by everyone.

Designed by Freepik

Let’s Take Effective

Concentrating the security of a company to a single team is a big mistake. Security is the responsibility of all people, every single person that is directly or indirectly involved with our company must be concerned.

This article is resulted after I saw misunderstandings about which DevSecOps are applied in practice. Here I show how to be effective with DevSecOps and runaway of wrong approaches.

Rules and Policies Enforcement

Ok, we need brochures to know what we need to do, but the enforcement cannot be made reactively, during a war room or during a coffee break. Again, delegating the entire security to a unique team will not work. The enforcement must to be made by software-defined stuff and embedded within our DevOps Pipeline.

To enforce our rules and policies, we have many tools and approaches. In this article, I will show how to be effective when starting a DevSecOps initiative and how to get results.

To meet the objectives we should employ the following steps:

1. **Define the target**
2. **Create the blue print**
3. **Broadcast**
4. **Enforce**
5. **Measure**

I'm assuming that the company has already reached a mature DevOps level.

Define the target

To know where we want to go, we must to define the objectives, our targets. Without them we will be like drifting ship.

Let's take some examples and use them as targets for this article.

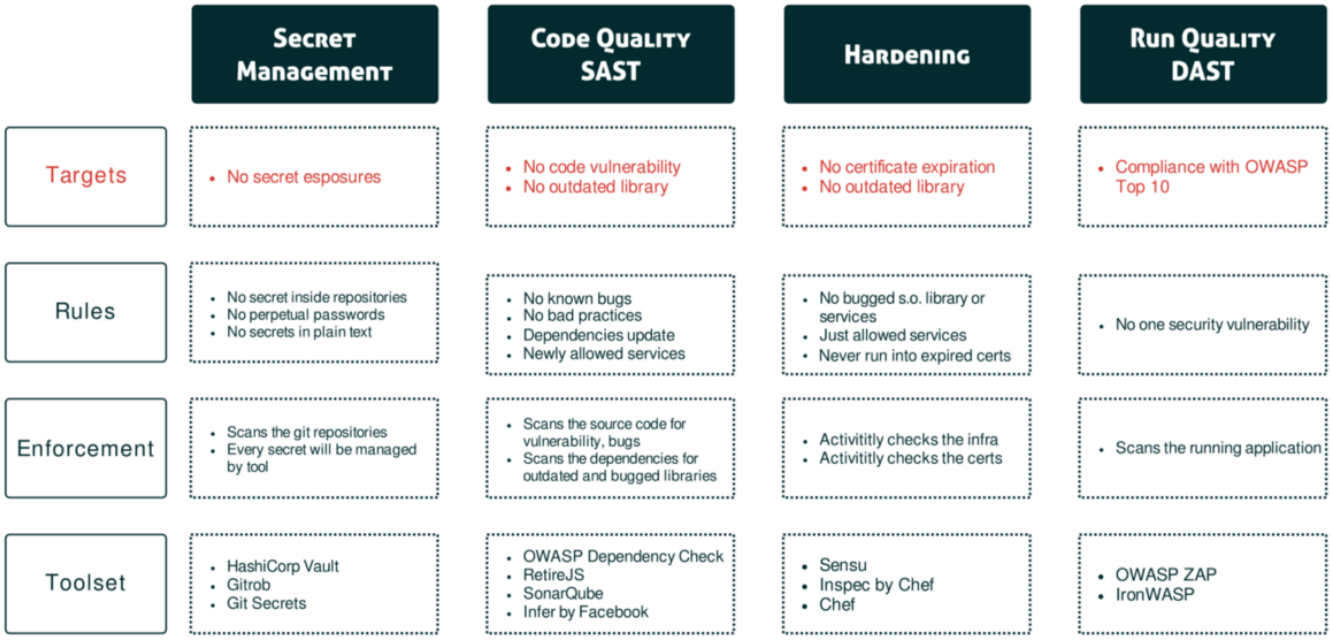
- **No secret exposures**: works to guarantee that no one password, passphrase, certificate chain, private key will be leaked.
- **No certificate expiration**: no more expired TLS certificate.
- **No outdated library**: scans the projects dependencies, o.s. libraries, o.s. services and o.s. utilities.
- **No code vulnerability**: statically scans the source code for vulnerabilities and bugs.
- **Compliance with OWASP Top 10**: dynamically scans the running app for vulnerabilities.

There's a lot of targets, let's see more examples: *no authorized process*, *no authorized ports*, *no bugged library*, *no authorized repositories*, *end-of-life management*, *no authorized account*, *compliance with Common Vulnerabilities and Exposures — CVE*, *vulnerability management*,*TLS everywhere*, etc.

Create the blue print

A blue print is a critical step, because we will define the "how" the development and operations, together, are concerned about security. In this blue print we will define the rules and how they will be enforced, preparing the entire company for the next steps.

Here is a diagram to understand the blue print for our defined targets.



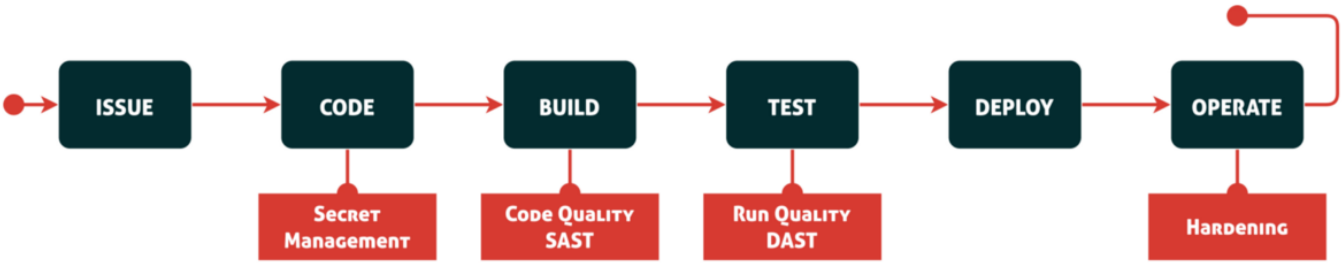
Four building blocks and targets

In the diagram above, we have four building blocks to map our targets: Secret Management, Code Quality, Hardening and Run Quality. But for your DevSecOps initiative, many others are necessary to map you own targets.

Side note: I defined few targets and building blocks to write a non exhaustive article.

Now create a space in your wiki and write clear and concise topics explaining every detail expressed in the blue print, highlighting the following topics:

- Link the blueprint to targets
- Detail the building blocks
- How they will be enforced
- What's the tool set
- Security first
- Security mindset
- Special exceptions
- How to stay in compliance
- The road map



Mapping the Building Blocks to our Pipeline

As you can see, we must map each building block to our pipeline steps. This will drive us to enforce how to collect metrics and how employ in our projects.

Broadcast

> People must know.

Now it’s time to market this new initiative. The best blueprint adds nothing without people knowing it and adopting it.

First, take the best sponsors ever. They will buy the blueprint and help to adoption growth. Never perform a big bang as this is will not work.

People should know and understand the benefits, and how important to be in compliance to the DevSecOps Blueprint because everyone performs a critical role. To broadcast successfully, our company should do this:

- Communicate the roadmap. Put it everywhere: banners, intranet, videos and digital sign.
- Make gifts: t-shirts, bottles, stickers, hats
- Look for early adopters
- Workshops
- Internal meetups
- Show the benefits
- Help teams play Sec

A good solution has good communication and marketing. Nothing happens if people do not know how to play Sec, they must know what is right and what is wrong when they develop software or provision and configure infrastructure.

Enforcement

Now let’s enforce the rules and policies. Everyone knows how to play Sec and they are prepared to implement the policies and follow the rules.

If no human task is able to answer the demand, then we must implement the enforcement in our pipeline, either for software and infrastructure. We will employ tools and automations to perform this task for each target we have defined. Here’s a toolset for each one.

I will not show a proprietary tool, but you will find paid alternatives in sites like stackshare.io.

No secret exposures

Secrets are passwords, passphrases, private keys, db account, tokens.

To have no secret exposures, first we do not have secrets saved within flat files, hardcoded, within git repository and no perpetual passwords. To solve almost all of these problems we must employ a vault tool to manage every single one secrets.

A vault tool is obligatory to manage the secret life cycle, mitigating the vulnerability in case of a leak. A mature tool can manage many different types of credentials like databases, operational systems, cloud providers, etc. and have the capability and ways to consumers got these secrets programmatically, through an API.

- HashiCorp Vault to manage secrets
- Gitrob to scan Git repositories for exposures

If you use secrets in our pipeline, SSH Key, consume it directly from your automation runner (e.g. Jenkins).

No certificate expiration

Nowadays TLS is mandatory and we must manage the certificate expiration to prevent SSL handshake errors.

Employ a tool to monitor each certificate, inside (private domain) and outside (public domain) of our company:

- Sensu to check certificate TTL
- HashiCorp Vault to manage the certificates of our private domain

Sensu does not have this functionality out-of-the-box, but there is a good plugin. With Sensu we can do a lot of things related to monitoring and to help other targets, check this out.

No outdated library

A big company has thousands of projects in development running in parallel. This is prone to use outdated libraries when a team starts to develop a new microservice for example. We can't allow this.

Side note: I wrote "…*when a team starts to develop a new…*" because we must be careful to enforce in projects that reach production or legacies.

Never apply enforcement in legacy just to follow the crowd. Work with the teams and understand the impact of the new rules and avoid the chaos.

The tools available to perform this job are specific to the platform and language that we have in our projects. But these tools must have to help us to identify the outdated and bugged library dependencies.

- OWASP Dependency Check: use for java projects and .net projects.
- RetireJS: use for js projects.

We must create a step in our pipeline to enforce this target in every build. This guarantees that we never pass an unauthorized, outdated or bugged dependency.

To enforce this target in the server farm we can employ tools like Chef and Inspect to manage the configuration and maintain the stability. These tools act in the o.s. level to maintain things right, normally they require an installed agent in the host.

No code vulnerability

Also known as SAST: Static Application Security Testing.

This target hunts bugs, bad practices, potential memory leaks, infinite loops, and anything that can cause a vulnerability. A well-known tool that handles this job is SonarQube, which provides several ways to scan the source files, hunting bugs and bad practices, and is used for many languages.

> Infer by Facebook: this is an option by Facebook and we can scan Java, C/C++ and Object-C.

Compliance with OWASP Top 10

Open Web Application Security Project — OWASP — worldwide not-for-profit charitable organization focused on improving the security of software (source).

OWASP Top 10 is a collection of the top 10 security risks and the latest update was in 2017 (see all of them here). It's a guideline to follow and deliver better security web applications. Many tools follow this guide and apply their own checks, for now I will propose two good options.

- OWASP ZAP: the reference implementation from OWASP
- IronWASP: a good open source option

Measurement

> Know the numbers to understand what happens.

The measurements are critical to understand how much is effective with DevSecOps initiative. Without them we are in a darkened room.

There are good open source options for metrics and for presenting them. I'm a huge fan of Grafana, which gracefully presents the metrics, and to store them I suggest InfluxDB, that can integrate into our pipeline and ingest every single measure. Grafana has a good integration with InfluxDB too.

What can we measure?

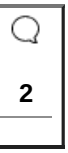
- Secrets found in the Git repositories
- Expired certificates
- Mean time to renew a expired cert
- Outdated dependencies
- Outdated o.s. libraries
- Critical vulnerabilities found in the source code
- Top 10 vulnerabilities found in the APP

That's all folks! This is my view of effective DevSecOps, if you like it, give some claps and share your opinion!

DevSecOps is so hype, but try always to be effective and not just talk, because "talk is cheap!" as Linus said.

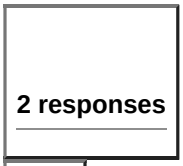
See you in the next articles.

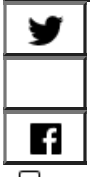
13



- DevOps
- Devsecops
- Security
- Pipeline

13 claps





Written by

Fábio José

Follow

Software Engineer

More From Medium

How to Truly Secure Your Online Accounts—Going Beyond Password Protection

BidiPass in BidiPass

Brown University Paper Shows Research Robot Vulnerability

Synced in SyncedReview

Deciphering the Hill Cipher and Rail Fence Cipher Algorithms

Aman Goyal in The Startup

Why Do We Still Give Away Our Secrets?

Prof Bill Buchanan OBE in ASecuritySite: When Bob Met Alice

Stealing JWTs in localStorage via XSS

David Roccasalva in Privasec RED

Improving Cybersecurity at Home

Peter Jarrett in Data Driven Investor

Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. Watch

Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. Explore

Become a member

Get unlimited access to the best stories on Medium — and support writers while you're at it. Just $5/month. Upgrade