

3 weeks to start AppSec - The simplest plan for improving your security posture

04 January 2021, 01:00

Nick Boucart
Tatiana Galibus

'Shift security left' is a popular IT industry paradigm which is very easy to understand but not so obvious to implement. Adopting this statement requires more than just use of technology: it is a shift in culture, integrated approach to application security and continuous learning process. Most start-ups are eager to adopt it in theory, but discover obstacles when applying it in practice.

Perhaps, the existing strategies are tailor-made for companies with a defined security posture, but there are no common guidelines on how to start application security, especially if a start-up has limited resources and expertise. What are the basic actions to take, in order to have a clean security maturity improvement plan and be ready to answer the customers' questions on trust and security?

Being driven by the passion to find a pragmatic solution for start-ups that really works, we spent almost three months on research and brainstorming at Sirris. Here is a result: *the simplest 3-week plan for improving security posture*. It is inspired by *DevSecOps* philosophy and *OWASP* standards.



In a previous article we explained the first week's actions. So now, what should be done on a second week of the simplest security improvement plan?

Second week: define quality gates and integrate static code analysis tools into your build chain

Second week in practice is dedicated to your first steps towards the security automation. There are hundreds of open-source and commercial tools for the application/website security analysis. Among those, the tools analysing the code and its dependencies, not the running application, are referred to as **Static Application Security Testing (SAST) tools**. There are several categories of such tools:

1. Code scanners

Scanners verify your code against built-in coding rules and notify you about the code quality issues or security vulnerabilities. These tools may have thousands of rules for each programming language and different scanning algorithms. Some of them are security-oriented, i.e. provide deep analysis of security vulnerabilities. Others are code-quality-oriented and notify you of possible coding problems or inaccuracies. We recommend you to start from one of these:

SonarScanner	Scans for both security and code quality issues. I both open-source and commercial version and ca integrated as a plug-in into your development environment, which makes it easy to use.
Fortify	Security-oriented open-source tool. It is easily int into the development environment and CI/CD pip

2. Software Composition Analysis tools (SCA)

SCA tools scan code dependencies and 3rd-party components for security bugs and vulnerabilities. Indeed, even if your code fits the security requirements, a call to insecure functions, such as Java Random.NextInt() can be a source of vulnerability. Using secure APIs and libraries is something we may often overlook. We recommend the following dependency scan tools:

OWASP DependencyCheck	Scans for insecure dependencies, integrated with SonarQube
Fortify	Scans for insecure dependencies and libraries, a Fortify code scan plug-in
Retire.js	Checks the use of insecure libraries and components in Java
Bundler-audit	Checks for vulnerable versions of Ruby components
Gemnasium/GitLab security scan	Monitors component versions in different programming language and notifies about the changes. Currently, this tool is integrated into GitLab CI/CD automation pipeline.

3. Secret detection tools

Often, developers unintentionally commit secrets, API tokens and credentials to the remote repositories. In this case, the sensitive information easily becomes exposed and can be used by adversaries to steal the identity and get unauthorised access to specific resources. Secret detection tools scan the code and detect the presence of keys, tokens and credentials. We recommend the following tools (most are integrated into the corresponding CI/CD environment):

GitGuardian	Automatically integrated into GitLab repository environment
GittyLeaks	Automatically integrated into GitHub repository environment
Git-secrets	Prevents developer from committing secrets to G repository
Truffle Hog	This tool scans repository including all its branch sensitive information

4. Build verification tools (BVT) and container analysis tools

BVT or smoke tests reveal simple failures that are severe enough to reject a software release. Docker container scan tools are crucial to know what libraries might be vulnerable in your

container. These tools detect the insecurities and bugs in Docker images before the code is deployed. We recommend the following tools for smoke tests and container scanning:

Clair-scanner	Container scan tool integrated with GitLab CI/CD pipeline
Trivy	Container vulnerability scanner, detecting problem OS packages and application dependencies
Twistlock	Provides full-cycle container security, integrated v cloud-based vulnerability management system

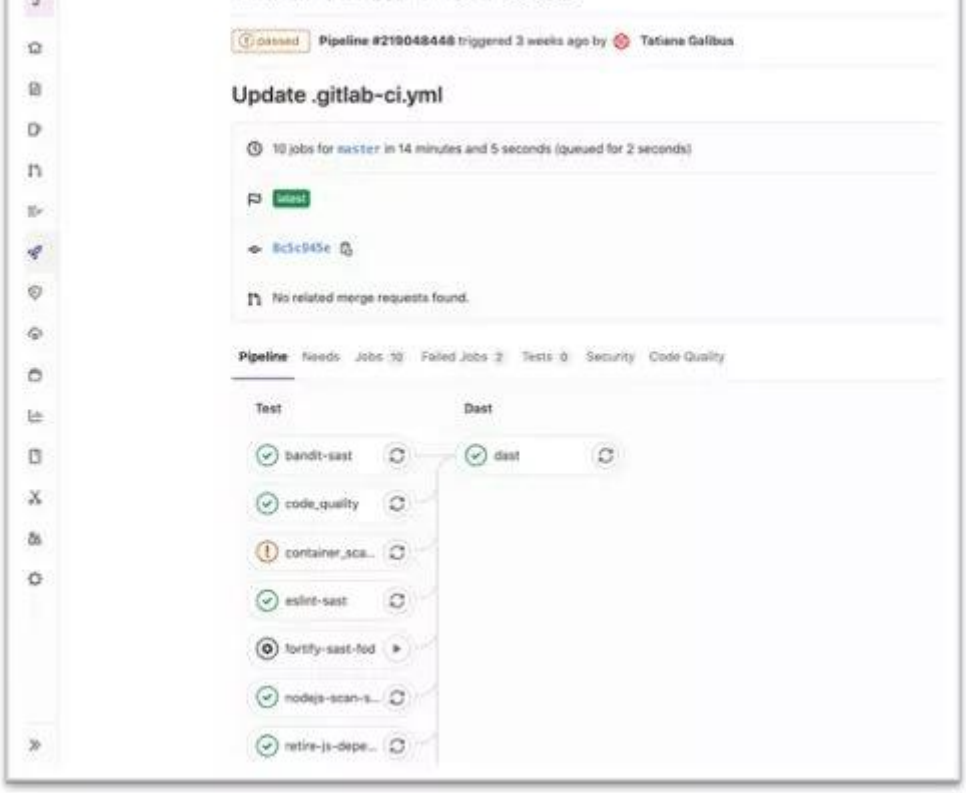
5. Vulnerability managers

Vulnerability management is an essential part of continuous integration, as in simplifies the continuous monitoring, identifying, and preventing risks to the code, dependencies, containers, images, and hosts in their environment. Often, SAST tools are integrated with vulnerability managers, helping to have deep insights into the security issues. We recommend the following vulnerability managers:

SonarQube server	Displays all detected issues classified by severity type in a project dashboard. Allows to track, visua analyse the vulnerabilities
DefectDojo	A vulnerability management platform allowing to integrate and orchestrate the output of multiple s scanners
Tenable.io	Vulnerability management tool integrated with Gi Provides automatic dashboard and security bugs

One of the key strengths of SAST tools is the broad coverage of programming languages and development platforms. They are relatively easy to adopt and integrate. In spite of a high number of false positives, SAST tools allow to identify nearly 50 per cent of bugs and are specifically efficient for improving code quality and detecting recurring security issues.

SAST tools are the foundation of application security pipeline. AppSec pipeline is a chain of essential security tools integrated in all stages of DevSecOps software development life cycle. SAST tools provide continuous security at the stages of design, code, build and validate.



Example of GitLab pipeline, SAST tools and testing stage

Curious to know more about this and other steps ? Do you have any other practical question on application security? Reach out to us on security@sirris.be.



]]>

Authors



Nick Boucart



Tatiana Galibus