What is SCA – Software Composition Analysis?

Do you know, with all certainty, what lies in your codebase? Today, the software is a chimera of, well, other software and codes and components. Most codebases are a composite of three types of codes – those you created in-house also known as proprietary code, those you bought off the shelf or had developed by a third party, and those your coders simply downloaded off the net, otherwise known as open-source codes. Surveys conducted by different agencies have showered the tech industry with some rather damning stats': 81% of all open-source codes are full of vulnerabilities. And, what's worse, most companies are ignorant of how much of their product's codebase has open-source codes. And it's not just open source code, but the whole triad – one bad line of code and even your carefully supervised source code opens your company to a world of trouble. This is what tools like SCA – software composition analysis – are a necessary evil. In this article, we'll talk about what Software Composition Analysis is and some of its features.

What Is software composition analysis -SCA?

Software composition analysis or SCA is the process of analyzing software to identify its constituent parts and how they are related. In other words, using tools and programs to scan your codebase and give you an audit of all its ingredients – of all the codes and components that make it up.

This includes both static and dynamic analysis. The goal of this type of analysis is to understand how a program's code works, where it came from, how it was developed, and what it

does. Software composition analysis is a critical technique for developers since it helps teams to understand how their product is constructed. Most of the time, the software is constructed in different departments, and under different supervisions. Each company has its own way of developing software. Nevertheless, the truth is that in most cases a product is analogous to a bowl of soup, one being cooked by a lot of different chefs. Each chef has a function to fulfill, a spice to mix up, or a protein to smoke. Each chef cooks their contribution and then tosses it into the cauldron. In an ideal setting, before said ingredients are tossed they would first have to pass through a Gordon Ramsay-like inspection. Some companies have this gatekeeper, but most don't. So, at the end of the day, the soup is cooked and the production team high-fives themselves and prepares for launch. Not really knowing the nutritional value or caloric landmines the soup has.

Software composition analysis software audits that soup, and that codebase, and tells the team what went into the concoction. It gives them a digital signature of each spice, allowing your security team to create a detailed BOM – Bill Of Materials – which they can then leverage to create a detailed risk analysis report.

SCA is basically a technique that provides an overview of the software by providing information about the code, its dependencies, and its complexity. The goal of this technique is to provide insights about how the software was built and to increase the understanding of how it works, what needs to be fixed if there are patches available or updates for some of the open-source codes used, and what threats you're exposing your company, and users to.

It also provides information about which parts are not well-tested or documented, which can be a good starting point for improving both quality and maintainability

Photo by Danial Igdery on Unsplash

How does software composition analysis work?

The goal of this analysis is to analyze the functions and components of a piece of software to determine its structure and identify any potential problems with the software's design. There are many different types of analyses that can be performed on an application, including static program analysis, dynamic program analysis, architectural decomposition analysis, and domain-specific language – DSL – composition analysis.

Software composition analysis provides insight into the overall design and function of a piece of software.

Let's look at some of the capabilities of software composition analysis

Multifactor scanning

SCA scans your whole codebase and gives you digital signatures on all the codes – particular open-source codes – that exist in your software's DNA.

Analysis and visibility

Sometimes open source coding and how it is embedded into your codebase pose a high visibility problem, They are incredibly obscure. For example, a developer might have included several open source packages into their code, unaware that in turn, that package is dependent on other – layers deep – coding.

Remediation advice

SCA offers developers visibility along with accurate vulnerability detection and in many cases fast, easy to implement, patches. In some cases, the creator of the open-source has already published an update – with the fix to that vulnerability – and all your developer needs to do is update the code.

Control

One of the best features of software composition analysis is that it allows developers more control over their products. It gives them actionable insight into what exactly they are creating and how it can be improved. By providing them a deep analysis of their product's genome – along with BOM – developers have better governance and control over their codebase.

Speed

Developers are moving fast – they are always racing to meet deadlines. This means security teams have to constantly keep up with that rampant pace. A pace that is sometimes

overwhelming. SCA, thankfully, offers a lightning-quick analysis of the whole codebase. In a matter of seconds, regardless of a project’s size, security teams can have a detailed blueprint. A blueprint that singles out every component and its vulnerabilities.

Benefits of software composition analysis

Software composition analysis offers teams a multitude of benefits, but above all one huge advantage – visibility. It allows them to properly identify all codes, including third-party and open source ones, that have been integrated into their product. Each of these components has inherent risk attached to them, not just security vulnerability, but out-of-date libraries and versions, redundancy issues that affect a product’s speed and quality, licenses and legal liabilities, etc.