

Three weeks to start AppSec, the simplest plan for improving your security posture (week 3)

26 April 2021, 02:00

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, an 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. Sirris offers you a solution: 'the simplest 3-week plan for improving security posture'.

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 the 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 about trust and security?

Being driven by the passion to find 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 previous articles we explained the first and second week actions. What are the final steps of the simplest security improvement plan?

Third week: perform manual code reviews for authentication and crypto modules

We propose to dedicate the third week to manual code review. The most pragmatic application security standard, OWASP ASVS directly states that level 2 security maturity cannot be achieved without manual review. Level 2 requirements are recommended for all applications processing any kind of personal or sensitive data, in other words, applications collecting customer data of any kind. In short: most probably, your start-up has to do it too!

Indeed, static code analysers, such as SonarQube (see our previous blog article) are able to find or notify of the majority of standard code issues, but the modules and libraries processing sensitive data require special attention. Organising such review for a complete code repository is not easy, as in most cases special security expertise is necessary and time-consuming as well. The amount of work can be reduced by providing the review of the most critical security modules in the first hand. We propose to start from the *authentication* and *crypto modules* in the third week of security improvement plan. Even without special security expertise it is possible to comply with the security requirements and good practices. Even if you are not an expert in cryptography, the important fact is that all industrially standardised algorithms and approaches are well-known and limited. Although the specification and terminology might look scary for a non-expert, the choice is very simple and straightforward, it is not difficult to be able to verify correct algorithms, parameters and values. Here are the steps we propose:

1. Choose your sensitive code snippets. OWASP Security Knowledge Framework helps to automate this process. The snippets should be short and correspond to specific authentication/cryptography requirement:

• Sanitize both input and output in your own area of responsibility.	`/* Security middleware */`
	`app.use(helmet.noSniff())`
	`app.use(helmet.frameguard())`
	`// app.use(helmet.xssFilter());`
	`app.disable('x-powered-by')`
	`app.use(helmet.featurePolicy({ features: {`
	`payment: ["'self'"]`
		XSS Filter disabled

• Check dependencies	`exports.isAuthorized = () => expressJwt({`
	`secret: this.publicKey }) exports.denyAll`
	`= () => expressJwt({ secret: '' +`
	`Math.random() })`
	`exports.sanitizeHtml = html =>`
	`sanitizeHtml(html) exports.sanitizeLegacy`
	`= (input = '') =>`
	`input.replace(/<(?:\w+)\W+?[\w]/gi, '')`
		Insecure library

*2. Verify that the code snippet indeed corresponds to the requirement. For the specification of requirements, you can refer to [OWASP cheat sheet series](): *



Cheat sheets contain short and condensed data and advise on best practices, secure algorithms, parameters and specifics to keep an eye on. They are up-to-date, extremely informative and often, are the only data source needed for a developer.

3. Do your corrections. If necessary, iterate the check. You can enhance the manual code check by reviewing security hotspot notifications from SonarQube or secret detection scanners to check the presence of exposed credentials in the code (see our previous blog article).

This 3-step procedure will allow you to keep an eye on the most critical security modules in your code. Having snippets and requirements at hand helps to speed up the review every time the new version is released, and your code will continue to stay up to standard security requirements! Simple, isn`t it?

Authors



Nick Boucart



Tatiana Galibus