

Drie weken om met AppSec van start te gaan, het meest eenvoudige plan om uw beveiligingsprofiel te verbeteren (week 3)

26 april 2021, 02:00

Nick Boucart
Tatiana Galibus

‘Shift security left’ is een populair paradigma in de industriesector, dat makkelijk te begrijpen is maar bij de implementatie niet zo vanzelfsprekend lijkt. Voor de toepassing van deze stelling is meer dan alleen het gebruik van technologie nodig: het impliceert een verschuiving in de cultuur, een geïntegreerde benadering van de applicatiebeveiliging en een permanent leerproces. De meeste start-ups willen de theorie ervan wel overnemen, maar botsen op obstakels bij de praktische toepassing. Sirris biedt u een oplossing: ‘het eenvoudigste 3-wekenplan om uw beveiligingsprofiel te verbeteren’.

Misschien zijn de bestaande strategieën wel op maat gemaakt voor bedrijven met een bepaald beveiligingsprofiel, maar er bestaan geen gemeenschappelijke richtlijnen over de manier om de applicatiebeveiliging aan te pakken, vooral wanneer de start-up over beperkte middelen en expertise beschikt. Welke eenvoudige maatregelen moeten worden getroffen om een duidelijk optimalisatieplan voor de maturiteit van de beveiliging uit te werken en klaar te zijn om de vragen van de klanten over vertrouwen en beveiliging te beantwoorden?

Gedreven door de passie om een pragmatische oplossing voor start-ups te vinden die echt werkt, hebben we bij Sirris bijna drie maanden aan onderzoek en brainstorming gedaan. Het resultaat? *Het eenvoudigste 3-wekenplan om uw beveiligingsprofiel te verbeteren*. Het plan bouwt voort op de



In de vorige artikels lichtten we de acties van de eerste en de tweede week toe. Welke zijn de laatste stappen van het meest eenvoudige plan voor een beter beveiligingsprofiel?

Derde week: een handmatige code review voor authenticatie- en cryptomodules uitvoeren

De derde week wijden we geheel aan de handmatige controle van uw codes. OWASP ASVS, de meest pragmatische standaard voor applicatiebeveiliging, stelt meteen dat het niveau 2 van beveiligingsmaturiteit niet kan worden bereikt zonder handmatige review. De vereisten van niveau 2 zijn aanbevolen voor alle applicaties die persoonlijke of gevoelige gegevens verwerken, met andere woorden, applicaties die klantgegevens van welke aard ook verzamelen. Kortom: de kans is groot dat ook uw start-up dat doet!

Statische codeanalysetools, zoals SonarQube (lees ons vorige blogartikel) zijn in staat om de meeste problemen met standaard codes te achterhalen of te melden, maar de modules en bibliotheken die gevoelige data verwerken, vergen speciale aandacht. Een complete coderepository controleren is niet eenvoudig; in de meeste gevallen is immers een speciale expertise in beveiliging nodig en de procedure is bovendien tijdrovend. De werklast kan worden beperkt door in eerste instantie de meest kritische beveiligingsmodules te reviewen. In de derde week van het plan voor een beter beveiligingsprofiel stellen we voor om te beginnen bij de authenticatie- en cryptomodules. Ook zonder specifieke expertise in beveiliging, is het mogelijk te voldoen aan de veiligheidseisen en goede praktijken. Wat telt is dat u alle industrieel gestandaardiseerde algoritmes kent en beperkt houdt, ook al bent u geen expert in cryptografie. Hoewel de specificaties en terminologie voor een leek eng kunnen lijken, toch is de keuze zeer eenvoudig en rechtlijnig; correcte algoritmes, parameters en waarden controleren, is niet moeilijk. We stellen de volgende stappen voor:

1. Kies uw gevoelige code snippets. Het OWASP Security Knowledge Framework helpt dit proces te automatiseren. De snippets moeten kort zijn en voldoen aan de specifieke vereiste voor authenticatie of cryptografie:

• 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. Ga na of de code snippet effectief aan de vereiste voldoet. Voor de specificatie van de vereisten kunt u de OWASP Cheat Sheet Series raadplegen:



Een cheat sheet bevat korte en bondige gegevens en tips over goede praktijken, veilige algoritmes, parameters en specifieke zaken waar u moet op letten. Ze zijn up-to-date, bijzonder leerrijk en vormen vaak de enige databron die een ontwikkelaar nodig heeft.

3. Voer uw correcties door. Herhaal zo nodig de controle. U kunt de handmatige codecheck verbeteren door de meldingen voor security hotspots van SonarQube of van scanners voor het detecteren van geheimen te controleren, om na te gaan of de code kwetsbare gegevens bevat (lees ons vorige blogartikel).

Met dit 3-stappenplan kunt u de meest kritische beveiligingsmodules van uw code in het oog houden. Hebt u snippets en vereisten bij de hand, dan kunt u de review versnellen telkens een nieuwe versie uitkomt; zo blijft uw code voldoen aan de standaard beveiligingsvereisten! Eenvoudig, toch?

Auteurs



Nick Boucart



Tatiana Galibus