

Trois semaines pour lancer AppSec, le plan le plus simple pour améliorer votre profil de sécurité (semaine 3)

26 avril 2021, 02:00

Nick Boucart
Tatiana Galibus

'Shift security left' (la sécurité le plus tôt possible) est un paradigme répandu dans le secteur informatique. S'il est très facile à comprendre, il n'est pas évident à mettre en œuvre. Son adoption exige plus que la simple utilisation des technologies : il s'agit d'un changement de culture, d'une approche intégrée de la sécurité des applications et d'un processus d'apprentissage continu. La plupart des start-ups sont désireuses de l'adopter, mais elles tombent sur des obstacles lors de son application dans la pratique. Sirris vous propose une solution : 'le plan le plus simple d'amélioration du profil de sécurité en 3 semaines'.

Les stratégies existantes sont peut-être adaptées aux entreprises ayant un profil de sécurité défini, mais il n'existe pas de lignes directrices communes sur la manière d'aborder la sécurité des applications, surtout si les ressources et l'expertise de la start-up sont limitées. Quelles sont les mesures de base à prendre afin de disposer d'un plan d'amélioration de la maturité de la sécurité clair et d'être prêt à répondre aux questions des clients sur la confiance et la sécurité ?

Animés par la passion de trouver une solution pragmatique qui fonctionne vraiment pour les start-

ups, nous avons mené un travail de recherche et de réflexion de près de trois mois chez Sirris. Résultat : le plan le plus simple d'amélioration du profil de sécurité en 3 semaines. Il s'inspire de la



Dans les articles précédents, nous avons expliqué les actions de la première et de la deuxième semaine. Quelles sont les dernières étapes du plan d'amélioration de la sécurité le plus simple ?

Troisième semaine : réaliser des examens de code manuels pour les modules d'authentification et de cryptographie

Nous proposons de consacrer la troisième semaine à l'examen manuel du code. La norme de sécurité applicative la plus pragmatique, OWASP ASVS, stipule directement que la maturité de sécurité de niveau 2 ne peut pas être atteinte sans examen manuel. Les exigences de niveau 2 sont recommandées pour toutes les applications qui traitent n'importe quel type de données personnelles ou sensibles. En d'autres termes, il s'agit d'applications qui collectent des données clients de quelque nature que ce soit. Bref, il est fort probable que votre start-up doive elle aussi le faire !

En effet, les analyseurs de code statique tels que SonarQube (voir notre article de blog précédent) sont capables de trouver ou de signaler la majorité des problèmes de code standard. Toutefois, les modules et bibliothèques traitant des données sensibles exigent une attention particulière. Il n'est pas facile d'organiser un tel examen pour un référentiel de code complet. Dans la plupart des cas, un savoir-faire spécifique en matière de sécurité requis nécessaire et l'opération prend un temps considérable. La quantité de travail peut être réduite en assurant en premier lieu l'examen des modules de sécurité les plus critiques. Nous proposons de débuter par les modules d'authentification et de cryptographie au cours de la troisième semaine du plan d'amélioration de la sécurité. Même sans savoir-faire particulier en matière de sécurité, il est possible de se conformer aux exigences de sécurité et aux bonnes pratiques. Même si vous n'êtes pas un expert en cryptographie, l'élément important est que tous les algorithmes et approches normalisés industriellement soient connus et limités. Bien que la spécification et la terminologie puissent sembler effrayantes pour un profane, le choix est très simple et direct. Il n'est pas difficile de vérifier

que les algorithmes, paramètres et valeurs sont corrects. Voici les étapes que nous proposons :

1. Choisissez vos fragments de code sensibles. OWASP Security Knowledge Framework permet d'automatiser ce processus. Les fragments doivent être courts et correspondre à une exigence spécifique en matière d'authentification ou de cryptographie :

• 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. Vérifiez que le fragment de code correspond bien à l'exigence. Pour la spécification des exigences, reportez-vous à la série de cheat sheets OWASP :



Les cheat sheets contiennent des données courtes et condensées et des conseils sur les meilleures pratiques, les algorithmes sécurisés, les paramètres et les points spécifiques à tenir à l'œil. Elles sont à jour, extrêmement instructives et constituent souvent la seule source de données dont un développeur a besoin.

3. Apportez les corrections requises. Si nécessaire, répétez la vérification. Vous pouvez améliorer la vérification manuelle du code en examinant les notifications de points chauds de sécurité de SonarQube ou les scanners de détection de secrets, de manière à vérifier la présence d'informations d'identification exposées dans le code (voir notre article de blog précédent).

Cette procédure en 3 étapes vous permettra de tenir à l'œil les modules de sécurité les plus critiques de votre code. Le fait d'avoir des fragments et des exigences à portée de main permet d'accélérer la révision à chaque fois qu'une nouvelle version est publiée. Votre code restera ainsi conforme aux exigences de sécurité standard ! C'est simple, n'est-ce pas ?

Auteurs



Nick Boucart



Tatiana Galibus