Dix pistes d'action pour sécuriser l'open source

Combien faut-il investir pour sécuriser l'*open source* ? Environ 150 millions de dollars sur deux ans. C'est l'estimation de l'OpenSSF (Open Source Software Security Foundation). En tout cas en support du « plan de mobilisation » qu'elle a présenté la semaine dernière à Washington – avec le soutien, notamment, du gouvernement américain et de la Fondation Linux. Amazon, Ericsson, Google, Intel, Microsoft et VMware, entre autres, se sont engagés à apporter une contribution financière.

Le programme comporte trois grands axes :

– Sécuriser la production de l'*open source*
– Améliorer la découverte des vulnérabilités et la réponse
– Raccourcir les délais de correction

L'ensemble est décliné en dix points, résumés dans le tableau ci-dessous. Il s'agit, admet l'OpenSSF, d'un « premier jet »*, amené à évoluer.

1 – Éduquer au développement de logiciels sécurisés

D'après l'OpenSSF, cet objectif nécessiterait au moins 10 heures de formation auprès des publics concernés ; idéalement, 40 à 50 heures. La fondation dispose de modules sur le sujet. Elle entend en faire la base de la démarche. Et les pousser par divers moyens : partenariats académiques, mise en avant lors de conférences majeures type SCALE ou FOSDEM, canaux de diffusion de type podcast et webinaire…
Pour inciter les porteurs de projets *open source* importants à se former et à se faire certifier, l'OpenSSF envisage des aides financières de 1000 $. Et des accords avec les principales plates-formes – GitHub et GitLab en tête de liste – pour mettre en avant les certifications.

2 – Évaluer et publier les niveaux de risque

Là non plus, on ne part pas de rien. Au cœur de l'initiative, il y aurait la plate-forme LFX Security and Insights, que la Fondation Linux exploite pour analyser les projets qu'elle héberge. Objectif : tirer parti de son architecture extensible pour y connecter, en particulier, les Security Scorecards et les Criticality Scores de l'OpenSSF. Mais aussi des résultats d'analyse de dépendances et de composition logicielle.
L'idée est d'avoir une évaluation continue (scoring au minimum hebdomadaire) du niveau de risque des 1000 projets *open source* qu'on aura catégorisés comme les plus critiques. Le tout serait réalisé au niveau de la plate-forme. Une solution plus onéreuse que de déployer des outils sur les SCM, mais moins contraignante.

3 – Systématiser la signature du code

Il s'agit là de pouvoir attester de l'intégrité du code sur tout son cycle de vie. Et non seulement, comme cela tend à être le cas actuellement, sur les ultimes étapes. L'OpenSSF entend s'appuyer sur le projet sigstore, lancé en 2020. D'abord en accompagnant son passage en phase opérationnelle. Puis en aidant à son implémentation dans les écosystèmes *open source* les plus critiques, et en étendant son périmètre (protocoles, algorithmes, systèmes pris en charge).

4 – Passer à des langages qui gèrent la sécurité de la mémoire

Ici, la base serait Prossimo. Ce projet de l'ISRG (Internet Security Research Group) a pour but d'identifier les projets qui gagneraient à être réécrits dans des langages qui évitent une classe de failles très répandue : celles liées à la mauvaise gestion de la mémoire. En tête de liste, C, que l'OpenSSF suggère de remplacer par Rust. Les premières cibles sont grosso modo celles de Prossimo : le noyau Linux et des bibliothèques liées aux protocoles TLS, DNS et NTP. Il s'agira aussi d'améliorer les compilateurs.

5 – Aider à répondre aux failles

L'OpenSSF estime qu'il lui faudra constituer une équipe de 30 à 40 experts pour son OSS-SIRT (centre de réponse aux incidents de sécurité). Experts qui pourront alors se rendre disponibles à plein temps pour les projets concernés. Autant pour négocier la publication des failles qu'écrire les correctifs et en coordonner la diffusion.

6 – Améliorer la capacité à détecter les vulnérabilités

Objectif : constituer une plate-forme d'outils d'analyse et de monitoring (SAST, DAST, *fuzzing*…) que les porteurs de projets pourraient exploiter avec l'appui d'experts. À terme, il s'agirait de couvrir les 10 000 principaux composants *open source*. Levier envisagé : le projet Alpha-Omega de l'OpenSSF.

7 – Réaliser des audits de projets critiques

Dans le viseur : 50 projets critiques sur l'année 1, puis 100 à 200 par an ensuite. Le moyen : un effort coordonné à l'échelle du secteur pour réaliser des audits dont les résultats seraient publics.

8 – Encourager le partage des informations

Cette initiative se traduirait par un framework et des accords multilatéraux entre une poignée d'acteurs. Lesquels s'engageraient à fournir des données anonymes en vue d'une agrégation au

bénéfice des chercheurs en sécurité. L'OpenSSF compte sur la participation de :

– La Fondation Linux (qui apporterait son *data lake*)
– Les 3 ou 4 principaux éditeurs d'OS Linux
– Les 5 ou 7 principaux CSP
– Et les 3 ou 4 principaux fournisseurs de solutions d'analyse de composition logicielle

Dans un premier temps, on pourrait accéder aux données en SQL ou sous forme d'interfaces BI « légères ».

9 – Standardiser et démocratiser le SBOM

Un constat : difficile de savoir à quels risques on s'expose si on n'a pas recensé son patrimoine *open source*. Une solution, tout du moins pour l'OpenSSF : activer le « réflexe SBOM » (Software Bill of Materials). C'est-à-dire précisément la démarche de recensement des actifs logiciels.
L'idéal visé ? Que ces SBOM soient produits automatiquement à chaque étape de conception et de distribution. Cela impliquera une intégration au sein des outils de *build*, des gestionnaires de paquets, des orchestrateurs, etc. Mais aussi, en parallèle, une standardisation à l'appui de schémas lisibles par la machine.

10 – Renforcer la sécurité des principaux systèmes de build et de distribution

Dans la continuité du point précédent, l'OpenSSF cherche à harmoniser la sécurité à l'échelle de ces composants critiques. Notamment en trouvant des consensus sur les attributs, les outils et les contrôles de sécurité à utiliser.

** Le budget global communiqué tient compte d'une marge d'erreur de 50 %.*

Illustration principale © monsitj – Adobe Stock