Copilot : l'IA de GitHub produit-elle du code sécurisé ?

Copilot, à utiliser avec précaution ? GitHub ne l'a pas caché : cet outil de complétion de code présente encore bien des limites. En premier lieu, sur la qualité des suggestions.

À l'origine du modèle d'apprentissage automatique sur lequel repose Copilot*, OpenAI pointe un autre aspect : la sécurité du code généré. L'entreprise a abordé la question dans un rapport technique. Avec des observations néanmoins restreintes. En l'occurrence, à deux aspects cryptographiques : la création de clés RSA trop courtes et l'usage d'AES en mode ECB.

Répondant à l'appel d'OpenAI, une étude en provenance de l'université de New York est finalement venue compléter cette première approche. Et la mettre en chiffres. Avec le top 25 CWE comme référentiel, elle s'est étendue sur trois axes :

- Propension à générer, dans un scénario donné, du code susceptible de présenter une faille
- Réaction à la modification d'éléments de contexte
- Comportement en fonction des langages et des paradigmes de programmation

Dans les grandes lignes, la démarche a consisté à créer, sur chacun de ces volets, plusieurs scénarios susceptibles d'engendrer telle ou telle CWE. Et à laisser Copilot faire ses suggestions – jusqu'à 25 par scénario. Au global, **sur les 1 692 programmes corrects générés, 40,48 % se sont révélés vulnérables**.

Les chercheurs reconnaissent que leur démonstration a des limites. Entre autres, celles de l'outil CodeQL, utilisé pour réaliser une part importante des tests de sécurité. Mais aussi l'inconstance de la « boîte noire » que constitue Copilot : ses suggestions sur un même scénario peuvent varier avec le temps.

Copilot mord à l'hameçon…

Sur le premier axe, l'analyse a couvert 18 CWE. Avec, pour chacune, trois scénarios. Parfois créés sur mesure ; éventuellement tirés de la base MITRE ou issus du catalogue CodeQL. Des 1087 programmes valides, 477 (43,88 %) contenaient la CWE pour laquelle les scénarios associés avaient été conçus. Dans 44,44 % des scénarios, la suggestion principale de Copilot s'est avérée vulnérable.

Le taux de vulnérabilité apparaît plus important en C (50 % des programmes) qu'en Python (38,4 %). De manière générale, il aurait pu, soulignent les chercheurs, être plus élevé si on n'avait écarté les suggestions incomplètes mais probablement sujettes à faille. Ou encore si on avait tenu compte des programmes susceptibles d'abriter d'autres CWE que celles ciblées (par exemple, dans une fonction de login, injection SQL et divulgation de secrets).

Par CWE, les taux les plus élevés se constatent sur :

- Injection de commande
- Traversement de répertoire
- Dépassement d'entier
- Déréférencement de pointeur nul
- Protection d'identifiants insuffisante

… sous conditions

Pour le deuxième axe, l'étude s'est focalisée sur une CWE : l'injection SQL. Elle s'est restreinte au langage Python, exclusivement avec des modèles issus de CodeQL et des tests de sécurité manuels. Sur 407 programmes générés pour 17 scénarios, 152 (37,35 %) se sont révélés vulnérables. Les chercheurs ont joué sur trois dimensions :

- Métaéléments
  Entre autres constats, insérer le nom d'un développeur connu sur GitHub fait baisser le taux de vulnérabilités. À l'inverse, l'indentation par tabulations à la place des espaces augmente le nombre de suggestions problématiques… et la confiance que leur porte Copilot.
- Commentaires/documentation
  Ajouter ou supprimer une ligne de commentaire peut tout changer, souvent en mal. Remplacer un mot par un autre aussi (ici, « remove » substitué à « delete »).
- Code
  Le taux de vulnérabilités tend à augmenter quand on supplante des constantes par des variables. Les changements de bibliothèques (par exemple postgres à la place de MySQLdb) sont moins significatifs. Effet positif, en revanche, lorsqu'on ajoute des fonctions indépendantes non vulnérables.

Le troisième axe s'est incarné à travers la génération de code RTL (description d'architectures électroniques) en Verilog. Avec comme référence une classe particulière de CWE : celles qui touchent au hardware. À défaut de top 25, les chercheurs en ont sélectionné six, avec trois scénarios chacune. Sur 198 programmes, 56 (28,28 %) ont présenté des vulnérabilités. Sur 7 scénarios (38,89 %), la principale suggestion était problématique.

** Copilot s'appuie sur Codex, une famille de modèles elle-même fondée sur GPT-3. Il hérite de certains de ses attributs, dont le fait de générer la suggestion la plus « logique » vis-à-vis de ses données d'entraînement – et non pas la plus optimisée. GitHub a récemment élargi son périmètre d'expérimentation, sur invitation.*

Photo d'illustration © maciek905 – Adobe Stock