28/06/2025

Une faille de sécurité critique, baptisée "BatBadBut", a été découverte dans la bibliothèque standard Rust, mais elle affecte également Erlang, Go, Haskell, Java, Node.js, PHP, Python et Ruby

Une faille de sécurité critique, baptisée “BatBadBut”, a été découverte dans la bibliothèque standard Rust, mais elle affecte également Erlang, Go, Haskell, Java, Node.js, PHP, Python et Ruby

```
Une faille de sécurité critique, baptisée "BatBadBut", a été découverte
entre autres dans la bibliothèque standard Rust, affectant toutes les
versions antérieures à 1.77.2 sous Windows. La vulnérabilité, identifiée
comme CVE-2024-24576, a un score CVSS de 10.0 et permet à un attaquant
d'exécuter des commandes shell arbitraires en contournant le mécanisme
d'échappement lors de l'invocation de fichiers batch avec l'API Command.
```

La vulnérabilité “BatBadBut” a été découverte par le chercheur en sécurité RyotaK et divulguée de manière responsable à l'équipe de sécurité de Rust, qui a donc expliqué comment l'équipe Rust a géré l'alerte, et patché la faille, ce qui n'est pas encore le cas sur tous les autres langages affectés. Selon le billet de blog de RyotaK, le problème provient des règles complexes d'analyse de cmd.exe , l'invite de commande de Windows, qui est implicitement lancée lors de l'exécution de fichiers batch. La bibliothèque standard Rust ne parvient pas à échapper correctement les arguments de commande pour cmd.exe , ce qui permet l'injection potentielle de commandes. Alors que les API Command::arg et Command::args de Rust garantissent que les arguments seront transmis tels quels au processus créé et ne seront pas évalués par un shell, la mise en uvre sous Windows est plus complexe. L'API Windows ne fournit qu'une seule chaîne de caractères contenant tous les arguments, laissant au processus créé la responsabilité de les diviser. La plupart des programmes utilisent la chaîne d'exécution standard argv en C, ce qui permet d'obtenir un comportement cohérent en matière de division des arguments. Cependant, cmd.exe possède sa propre logique de découpage des arguments, qui nécessite un échappement personnalisé par la bibliothèque standard Rust. “Pour éviter l'exécution inattendue de fichiers batch, vous devriez envisager de déplacer les fichiers batch dans un répertoire qui n'est pas inclus dans la variable d'environnement PATH.”, a noté RyotaK dans son billet de blog. “Dans ce cas, les fichiers batch ne seront pas exécutés à moins que le chemin d'accès complet ne soit spécifié, ce qui permet d'éviter l'exécution inattendue de fichiers batch.” L'équipe de sécurité de Rust a reconnu que la logique d'échappement existante dans la bibliothèque standard était insuffisante, ce qui permettait à des arguments malveillants de contourner l'échappement et d'entraîner une exécution arbitraire de l'interpréteur de commandes. La gravité de la vulnérabilité “BatBadBut” est considérée comme critique si des arguments non fiables sont transmis lors de l'invocation de fichiers batch sous Windows. BatBadBut ne se limite pas à un seul identifiant CVE Alors que l'attention s'est d'abord portée sur le langage de programmation Rust, il est apparu que la vulnérabilité “BatBadBut” ne se limite pas à un seul identifiant CVE. La vulnérabilité

affecte plusieurs langages de programmation et outils, chacun se voyant attribuer un identifiant CVE différent en fonction de la mise en uvre et de l'impact spécifiques. Outre CVE-2024-24576, qui concerne la bibliothèque standard Rust, “BatBadBut” englobe également CVE-2024-1874, CVE-2024-22423 (qui affecte yt-dlp avec un score de risque élevé de 8,3) et CVE-2024-3566 (qui affecte Haskell, Node.js, Rust, PHP et yt-dlp). Cela met en évidence la nature étendue de la vulnérabilité et la nécessité pour les développeurs d'évaluer leurs applications et dépendances à travers différents langages de programmation et outils. Atténuation Pour remédier à la vulnérabilité “BatBadBut”, l'équipe Rust a publié la version 1.77.2, qui inclut un correctif pour le problème. Ce correctif améliore la robustesse du code d'échappement et modifie l'API Command pour qu'elle renvoie une erreur InvalidInput lorsqu'elle ne peut pas échapper un argument en toute sécurité. Cette erreur sera émise lors du lancement du processus. Pour les développeurs qui implémentent leur propre échappement ou qui ne gèrent que des entrées fiables sous Windows, la méthode CommandExt::raw_arg peut être utilisée pour contourner la logique d'échappement de la bibliothèque standard. Il est important de noter que la vulnérabilité n'affecte que le code Rust fonctionnant sous Windows qui exécute des fichiers batch avec des arguments non fiables. Les autres plateformes ou utilisations de Windows ne sont pas concernées. La communauté Rust a exprimé sa gratitude à RyotaK pour avoir divulgué la vulnérabilité de manière responsable, à Simon Sawicki (Grub4K) pour avoir identifié certaines des règles d'échappement adoptées, et aux membres du projet Rust qui ont aidé à développer le correctif, à le réviser et à coordonner le processus de divulgation. Il est fortement conseillé aux développeurs utilisant Rust sur Windows de mettre à jour vers la version 1.77.2 dès que possible afin de réduire le risque d'attaques potentielles par injection de commandes. Il est essentiel de s'assurer que les entrées non fiables sont correctement validées et assainies avant d'être transmises en tant qu'arguments à des fichiers batch. État des lieux sur les autres langages de programmation affectés Sources : Équipe Rust, “BatBadBut: You can't securely execute commands on Windows” (RyotaK, chercheur en sécurité) Et vous ? Quelle lecture faites-vous de cette situation ? Voir aussi : Comment Rust améliore la sécurité de son écosystème : la Rust Foundation publie un rapport sur ce que leur initiative de sécurité a accompli au cours des six derniers mois de 2023 « Choisir Rust est opter pour une meilleure sécurisation des logiciels qu'avec le C, mais une efficacité énergétique et une performance d'exécution que seul le C offre », d'après l'équipe AWS
https://securite.developpez.com/actu/356317/Une-faille-de-securite-critique-baptisee-BatBadBut-a-ete-decouverte-dans-la-bibliotheque-standard-Rust-mais-elle-affecte-egalement-Erlang-Go-Haskell-Java-Node-js-PHP-Python-et-Ruby/