Journal : Vulnérabilités multiples dans sudo-rs

12 nov. 2025

16

Posté par **Faya** le 12 novembre 2025 à 13:00. **Licence CC By-SA.**
Étiquettes : **sudo**, **faille**, **rust**

Un grand sage a dit un jour : *« le fait qu'il y ait un bug dans sudo ne garantit pas qu'il n'y ait pas de bug dans sudo-rs. »*

Et bien il avait raison et on a pas attendu longtemps pour en trouver, des bugs. Ils sont gratinés

> Two security issues were discovered in sudo-rs, a Rust-based implemention of sudo (and su), which could result in the local disclosure of partially typed passwords or an authentication bypass in some targetpw/rootpw
> configurations.

Bref, memory-safe ne veut pas dire bug-free et *rewrite everything in Rust* ne fait pas repousser les cheveux ni revenir l'être aimé.

(10 commentaires). Markdown EPUB

Petite question à ceux qui "baignent" encore dans le C

Posté par **totof2000** le 12 novembre 2025 à 14:16.
Évalué à 4 (+2/-0).

> Est-ce qu'il serait possiblede faire évoluer le C pour qu'il integre un système de contrôle de mémoire "à la rust", tout en gardant une certaine compatibilité avec le code actuellement écrit ? Est-ce qu'il y a des discussions/recherches/évolutions potentielles dans ce sens ? Est-ce que ça vaudrait le coup ?

Répondre

[^] # Re: Petite question à ceux qui "baignent" encore dans le C

Posté par **Tonton Th** (**site web personnel**, **Mastodon**) le 12 novembre 2025 à 14:40. Évalué à 3 (+1/-0).

> Est-ce qu'il serait possiblede faire évoluer le C pour qu'il integre un système de contrôle de mémoire
>
> Je pratique le C depuis les années 80, et c'est une idée qui refait surface régulièrement. Jusqu'à présent il n'y a pas eu de proposition universelement adoptée.
>
> > Est-ce qu'il y a des discussions/recherches/évolutions potentielles dans ce sens ?
>
> Oh que oui, un bon gazillion. Une des plus récentes, c'est https://fil-c.org/ qui annonce :
>
> *Fil-C is a fanatically compatible memory-safe implementation of C and C++. Lots of software compiles and runs with Fil-C with zero or minimal changes. All memory safety errors are caught as Fil-C panics.*
>
> Après, il faut voir ce que ça donne à l'usage...

Répondre

[^] # Re: Petite question à ceux qui "baignent" encore dans le C

Posté par **thoasm** le 12 novembre 2025 à 15:03.
Évalué à 3 (+0/-0).

> C'est pas vraiment "à la rust", le principe c'est de planter à l'exécution, au prix de moins de perf (et d'un gc dans certains cas, j'ai cru comprendre).
>
> Discussion là dessus sur un site de Rustacés

Répondre

[^] # Re: Petite question à ceux qui "baignent" encore dans le C

Posté par **serge_sans_paille** (**site web personnel**) le 12 novembre 2025 à 14:42. Évalué à 3 (+1/-0).

> Il y a des tentatives en cours (basées sur des dialectes):
>
> - https://clang.llvm.org/docs/L
> - https://clang.llvm.org/docs/A capture-by (et ses sœurs)

Répondre

[^] # Re: Petite question à ceux qui "baignent" encore dans le C

Posté par **thoasm** le 12 novembre 2025 à 14:49.
Évalué à 4 (+1/-0).

> C'est pas une question nouvelle Ce fil reddit d'il y a quelques années l'évoque, je lie une des réponses
>
> Si c'est possible de faire ça, et ça demanderait quand même sans doute des évolution pas neutre dans le langage parce que Rust s'appuie sur le système de type pour faire ça et des indications que le compilateur doit comprendre, et si tu veux garder une compatibilité, ça poserait le même type de problème que la distinction code "safe / unsafe" en rust.
>
> Le code "unsafe" en Rust permet d'utiliser des pointeurs sans restrictions "à la C", mais l'approche de Rust est de limiter au maximum les sections "unsafe" dés le développement initial, pour avoir le maximum de trucs "tout bon" du point de vue sûreté mémoire dés le début. Si tu pars d'une base de code "toute unsafe" pour essayer de la rendre "safe" (en gardant la compatibilité C au max) tu te retrouves, intuitivement, à avoir le même problème que craignent certains mainteneurs du noyau Linux j'ai cru comprendre et à avoir à réécrire les API ou de grosses parties du code existant parce que les restrictions vont se propager dans le code dont il dépend, à la manière de la propagation des "async / await" dans du code qu'on veut rendre asynchrone dans certains langages comme le js. En plus velu parce que ça va rendre impossible l'utilisation de certaines constructions et peut être obliger à des changements d'architectures du code ou du gros refactoring.
>
> Une autre approche serait d'utiliser des analyseurs statiques de code velus qui tentent de démontrer pleins de propriétés et ne pas laisser passer du code qui n'a pas ces propriétés. Mais ça pose sans doute le même type de pb de faire évoluer du code pré-existant vers plus de sûreté avec ce genre de techniques.

Répondre

[^] # Re: Petite question à ceux qui "baignent" encore dans le C

Posté par **pulkomandy** (**site web personnel**, **Mastodon**) le 12 novembre 2025 à 15:06. Évalué à 4 (+1/-0).

> > Si tu pars d'une base de code "toute unsafe" pour essayer de la rendre "safe" (en gardant la compatibilité C au max) tu te retrouves, intuitivement, à avoir le même problème que craignent certains mainteneurs du noyau Linux j'ai cru comprendre et à avoir à réécrire les API ou de grosses parties du code existant parce que les restrictions vont se pro-

> pager dans le code dont il dépend, à la manière de la propagation des "async / await" dans du code qu'on veut rendre asynchrone dans certains langages comme le js. En plus velu parce que ça va rendre impossible l'utilisation de certaines constructions et peut être obliger à des changements d'architectures du code ou du gros refactoring.

J'ai du mal à voir en quoi c'est une mauvaise chose. Le code Rust obligerait les interfaces à être propres et bien définies, et mettrait alors en évidences les endroits où ce n'est pas le cas et où il y a du code à retravailler.

Répondre

[^] # Re: Petite question à ceux qui "baignent" encore dans le C

Posté par **thoasm** le 12 novembre 2025 à 15:30.
Évalué à 3 (+0/-0).

> Je sais pas si c'est une bonne ou une mauvaise chose mais j'ai l'impression que les mainteneurs n'aiment généralement pas les gros patchs sur du code existant et stable … Et là si ça se propage ça risquerait (je met un point d'interrogation, si ça se trouve c'est pas si pire et mitigeable) dans le pire des cas de se propager un peu partout d'un seul coup. Et ça certains mainteneurs, je pense, n'aiment pas du tout ça.
>
> En tout cas c'est ce que j'avais retenu de l'affaire Christoph Hellwig , en particulier ce post mais à la relecture je suis pas sûr que ce soit spécifiquement dû à la gestion mémoire de Rust et il aurait peut être la même objection pour n'importe quel base de code multilingue.

Répondre

[^] # Re: Petite question à ceux qui "baignent" encore dans le C

Posté par **thoasm** le 12 novembre 2025 à 15:56.
Évalué à 3 (+0/-0).

> Oups trompé dans le premier lien, je voulais lier https://lwn.net/Articles/1006805/ l'article de lwn.

Répondre

titre

Posté par **Maderios** le 12 novembre 2025 à 14:41.
Évalué à 6 (+4/-0).

> Le titre devrait être: "Vulnérabilités multiples dans les anciennes versions de sudo-rs"
> La récente 2.10 n'est pas concernée
> https://archlinux.org/packages/extra/x86_64/sudo-rs/

Répondre

[^] # Re: titre

Posté par **Faya** le 12 novembre 2025 à 15:40.
Évalué à 4 (+2/-0).

> Ancienne = 2 jours donc…
>
> Arch est "cutting edge" mais ça n'est pas le cas de toutes les distros. Par exemple Ubuntu qui vient d'adopter sudo-rs par défaut est visiblement en 2.8

Répondre

Envoyer un commentaire

Suivre le flux des commentaires

Note : les commentaires appartiennent à celles et ceux qui les ont postés. Nous n'en sommes pas responsables.

Revenir en haut de page