RFC 8332 : Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 21 mars 2018

Date de publication du RFC : Mars 2018

https://www.bortzmeyer.org/8332.html

Les RFC normalisant le protocole SSH mentionnaient la possibilité pour le serveur d'avoir une clé publique RSA, mais uniquement avec l'algorithme de condensation SHA-1. Celui-ci a subi plusieurs attaques et n'est plus du tout conseillé aujourd'hui. Ce nouveau RFC est la description officielle de l'utilisation de RSA avec SHA-2 dans SSH.

Le RFC officiel du protocole SSH, le RFC 4253 [1], définit l'algorithme `ssh-rsa` comme utilisant "*the SHA-1 hash*" pour la signature et la vérification. Ce n'est plus cohérent avec ce qu'on sait aujourd'hui des faiblesses de SHA-1. Par exemple, le NIST (dont les règles s'imposent aux organismes gouvernementaux états-uniens), interdit SHA-1 <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf> (voir aussi la section 5.1 du RFC).

(Petit point au passage, dans SSH, le format de la clé ne désigne que la clé elle-même, alors que l'algorithme désigne un format de clé **et** des procédures de signature et de vérification, qui impliquent un algorithme de condensation. Ainsi, `ssh-keygen -t rsa` va générer une clé RSA, indépendamment de l'algorithme qui sera utilisé lors d'une session SSH. Le registre IANA <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xml#ssh-parameters-19> indique désormais séparement le format et l'algorithme.)

Notre nouveau RFC définit donc deux nouveaux algorithmes :

— `rsa-sha2-256`, qui utilise SHA-256 pour la condensation, et dont la mise en œuvre est recommandée pour tout programme ayant SSH,
— et `rsa-sha2-512`, qui utilise SHA-512 et dont la mise en œuvre est facultative.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc4253.txt`

Les deux algorithmes sont maintenant dans le registre IANA <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xml#ssh-parameters-19>. Le format, lui, ne change pas, et est toujours qualifié de ssh-rsa (RFC 4253, section 3.) Il n'est donc **pas** nécessaire de changer ses clés RSA. (Si vous utilisez RSA : SSH permet d'autres algorithmes de cryptographie asymétrique, voir également la section 5.1 du RFC.)

Les deux « nouveaux » (ils sont déjà présents dans plusieurs programmes SSH) algorithmes peuvent être utilisés aussi bien pour l'authentification du client que pour celle du serveur. Ici, par exemple, un serveur OpenSSH version 7.6p1 annonce les algorithme qu'il connait (section 3.1 de notre RFC), affichés par un client OpenSSH utilisant l'option -v :

```
debug1: kex_input_ext_info: server-sig-algs=<ssh-ed25519,ssh-rsa,rsa-sha2-256,rsa-sha2-512,ssh-dss,ecdsa-sha
```

On y voit la présence des deux nouveaux algorithmes. ssh-rsa est celui du RFC 4253, avec SHA-1 (non explicitement indiqué).

En parlant d'OpenSSH, la version 7.2 avait une bogue (corrigée en 7.2p2), qui faisait que la signature était étiquetée ssh-rsa quand elle aurait dû l'être avec rsa-sha2-256 ou rsa-sha2-512. Pour compenser cette bogue, le RFC autorise les programmes à accepter ces signatures quand même.

Autre petit piège pratique (et qui a suscité les discussions les plus vives dans le groupe de travail), certains serveurs SSH « punissent » les clients qui essaient de s'authentifier avec des algorithmes que le serveur ne sait pas gèrer, pour diminuer l'efficacité d'éventuelles attaques par repli. Pour éviter d'être pénalisé (serveur qui raccroche brutalement, voire qui vous met en liste noire), le RFC recommande que les serveurs déploient le protocole qui permet de négocier des extensions, protocole normalisé dans le RFC 8308. (L'extension intéressante est server-sig-algs.) Le client prudent peut éviter d'essayer les nouveaux algorithmes utilisant SHA-2, si le serveur n'annonce pas cette extension. L'ancien algorithme, utilisant SHA-1, devrait normalement être abandonné au fur et à mesure que tout le monde migre.

Les nouveaux algorithmes de ce RFC sont présents dans :
— Bitvise <https://www.bitvise.com/> (serveur et client),
— OpenSSH,
— AsyncSSH <https://github.com/ronf/asyncssh>,
— SmartFTP <https://www.smartftp.com/>,
— PKIX-SSH <http://roumenpetrov.info/secsh/index.html> (version 10.1, 25 mars 2017).
Vous pouvez aussi regarder le tableau de comparaison des versions de SSH <http://ssh-comparison.quendi.de/comparison/hostkey.html>. Voici encore un ssh -v vers un serveur dont la clé de machine est RSA et qui utilise l'algorithme RSA-avec-SHA512 :

```
...
debug1: kex: host key algorithm: rsa-sha2-512
...
debug1: Server host key: ssh-rsa SHA256:yiol3GPr1dgVo/SNXPvtFqftLw4UF+nL+ECa1yXAtG0
...
```

Par comparaison avec les SSH récents, voici un ssh -v OpenSSH vers un serveur un peu vieux :

```
debug1: kex_input_ext_info: server-sig-algs=<ssh-ed25519,ssh-rsa,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384
```

On voit que les nouveaux algorithmes manquent (pour RSA ; SHA-2 est utilisable pour ECDSA). Et pour sa clé de machine :

```
...
debug1: kex: host key algorithm: ssh-rsa
...
debug1: Server host key: ssh-rsa SHA256:a6cLkwFRGuEorbmN0oRjvKrXELhIVXdgHRCcbQOM2w8
```

Le serveur utilise le vieil algorithme, `ssh-rsa`, ce qui veut dire SHA-1 (le `SHA256` qui apparait a été généré par le client).