Sécurité - CVE 2018–10933

Cyril Dussert - Emilien Mottet

`https://www.cvedetails.com/cve/CVE-2018-10933/`

22 novembre 2018

1 Service compromis

La faille de sécurité etudiée (CVE 2018–10933) concerne la librairie `libssh` côté serveur dans les versions inférieures à 0.7.6 et 0.8.4. Néanmoins, des systèmes Linux embarquent la librairie dans leurs dépôts. Ces systèmes deviennent ainsi aussi vulnérables. Il s'agit d'Ubuntu pour ses versions LTS (14.04, 16.04 et 18.04), ainsi que Debian Stretch (9.0). Le risque est un contournement de la politique de sécurité.

Le type de cette faille est principalement `Authentication Issues`.

Cette faille de sécurité affecte les applications qui utilisent `libssh` pour implémenter un serveur SSH. Le client SSH n'est pas concerné. Cette faille ne concerne pas `libssh2` ni `openssh`.

2 Description de la vulnérabilité

Cette faille permet une authentification non autorisée sur un serveur. Au cours de l'échange lors de l'authentification d'un client aurpès d'un serveur, la simple présentation d'un message `SSH2_MSG_USERAUTH_SUCCESS` de la part du client, à la place d'un `SSH2_MSG_USERAUTH_REQUEST` (normalement attendu). permet à l'attaquant de s'authentifier correctement sans aucun besoin d'identifiants.

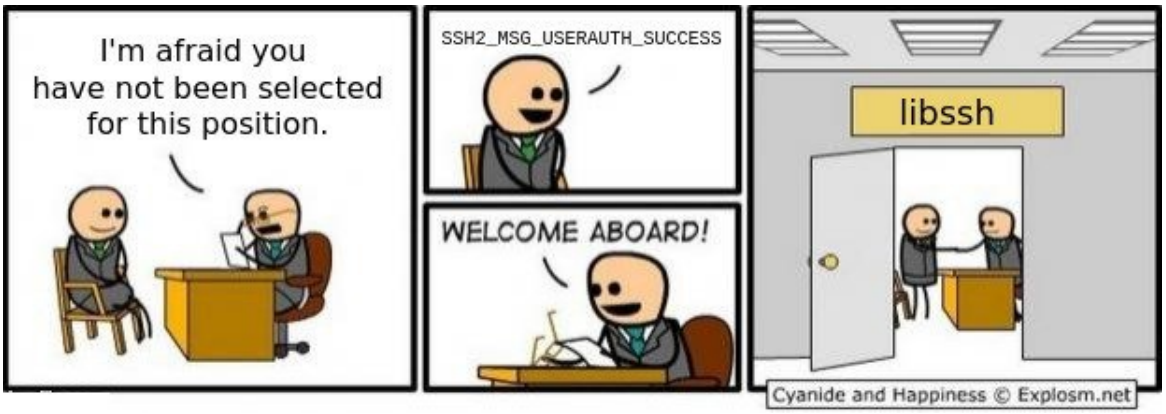

Figure 1 – CVE-2018-10933 in real world

3 Cibles de la faille

Comme décrit ci-dessus, les cibles concernées par cette faille sont les serveurs utilisant la librairie `libssh` comme serveur SSH.

3.1 Versions concernées

— Versions supérieures à 0.6
— Version inférieures à 0.7.6
— Version inférieures à 0.8.4

3.2 Trouver des victimes

Grâce à shodan (google du hacker) nous pouvons trouver une liste de serveur utilisant une librairie potentiellement. Screenshot disponible dans le README.md
`https://www.shodan.io/search?query=product%3A%22libssh%22+version%3A%220.6.3%22`

4 Exemple d'architecture

La dangerosité de cette faille est qu'il n'y a pas besoin d'une architecture spécifique pour l'exploiter. Il suffit d'un serveur, accessible sur un réseau (par exemple, internet), utilisant `libssh` pour son serveur SSH et d'un client malicieux.

5 Préconisations de sécurité

Etant donné qu'il n'y à pas de workaround directement accessible pour palier cette faille, la meilleure solution pour s'en protéger est de mettre à jour **régulièrement** les librairies présentes sur le serveur. Peu de temps après la découverte de la faille, un patch correctif (versions 0.7.6 et 0.8.4) était disponible.

Un bon moyen de suivre les différents problèmes de sécurité importants est de regarder régulièrement les bugs les plus critiques déclarés par les différents systèmes d'exploitation (par exemple, Ubuntu security notices : `https://usn.ubuntu.com/`). Il va de soi que le mieux est d'avoir une infrastructure homogène (mêmes bases logicielles sur différents serveurs), afin de pouvoir réagir efficacement (un seul patch à appliquer).

Dans le cas des systèmes RedHat Enterprise Linux, les notifications par mail des nouvelles failles sont très efficaces. De plus, il ne faut pas hésiter à faire appel au support en cas de doute sur n'importe quel point en rapport avec le système d'exploitation.

5.1 Bonnes pratiques à mettre en oeuvre

Voici une liste de bonnes pratiques qui pourraient vous sauver la vie dans le cadre de cette faille :

— Mettre à jour le système régulièrement
— Ne pas exposer son serveur SSH directement sur internet (bien configurer son iptables)
— Changer le port d'écoute par défaut de SSH (2222 par exemple)
— Interdire à root de se connecter au serveur via SSH
— Utiliser un framework de prévention d'intrusions (ex : Fail2ban)

6 PSSI ?

7 Expérimentation

7.1 Le serveur

Afin de rendre facile cette expérimentation, nous vous avons préparé une image Docker contenant un seveur SSH utilisant `libssh 0.7.4` Tout cela se trouve dans le dossier `libssh-7.4-ssh_server_fork` et un simple `docker build -t unsecure-libssh .` au sein du dossier permettra de construire l'image.

Ensuite, il vous suffit de démarrer le serveur sur le port de votre choix (ici 2222) en utilisant `docker run -it -p 2222:22 unsecure-libssh`.

7.2 Le client malicieux

Côté client, un script pour l'exploitation de cette faille est disponible dans le dossier `script`. Vous avez juste besoin d'une version récente de Python (par exemple 3.6) pour l'exécuter. N'oubliez pas d'installer les librairies tierces requises en exécutant `pip install -r requirements.txt` au sein du dossier.

Ensuite, le script fonctionne comme suit : `python exploit.py <host> <port> <command>`. Exécutez-le de manière à contacter le conteneur Docker préalablement lancé, par exemple : `python exploit.py localhost 2222 id` exécutera la commande `id` au sein de l'image Docker et affichera la sortie sur le terminal. Vous devriez normalement voir une sortie du genre :

```
(venv) $ python script/exploit.py localhost 2222 id
INFO:paramiko.transport:Connected (version 2.0, client libssh_0.7.4)
uid=0(root) gid=0(root) groups=0(root)
```

On remarque que le client n'a eu besoin d'aucune authentification et a pu exécuter une commande en tant que root...