université de Caen
Basse-Normandie

Université de Caen

Département d'Informatique

Master 1

Projet annuel

Développement d'un ver informatique grâce à une faille du système et étude de sa propagation au sein d'un réseau de machines

"It's not a bug, it's a feature"
- GNU about Shellshock bug

Projet réalisé par Mathieu Valois

Encadrant : M. Alain Bretto

Relecteur : None

Année 2014-2015

Résumé

Ce projet consiste en la réalisation d'un ver informatique, capable de se déplacer dans un réseau de manière autonome, rampant de machine en machine grâce à une glsfaille des systèmes qu'il infecte. Une fois réalisé, il faudra étudier sa progression dans un réseau depuis une première machine, et éventuellement améliorer ses capacités de propagation et de furtivité en conséquence. Par ailleurs, on pourra aussi le doter d'une charge virale et l'améliorer.

1 Introduction

1.1 Aaaah, un ver ! Quelle horreur !

En informatique, on appelle un ver, un programme capable de s'auto-répliquer. Contrairement aux virus, il ne possède pas de programme hôte, cela implique qu'il doive se déplacer seul. En effet, le virus lui, se déplace avec le programme qu'il a infecté (sur une clé USB, par e-mail, ...), alors que le ver doit progresser de machine en machine seul. C'est pour ça que les vers utilisent en général une faille du système.

On appelle charge virale d'un ver ou virus le but final de ce programme. Par exemple : surveiller le système, détruire des fichiers, attaquer un serveur, envoyer des milliers d'e-mails ...

Généralement, un ver se décompose en 3 fonctions distinctes :

- une fonction de propagation, permettant d'infecter un nouveau système
- une fonction de persistance qui permet au virus de résister aux redémarrages de la machine
- une fonction d'attaque exécutant sa charge virale

1.2 Avantages/inconvénients face au virus

De part sa nature, le ver informatique possède plusieurs avantages mais aussi des inconvénients face au virus. Premièrement, le fait de ne pas requérir de programme hôte lui permet de se déplacer à une vitesse bien supérieure au virus pour deux raisons :

- il n'attend pas pour aller de machine en machine
- est lancé directement dès qu'il arrive sur une nouvelle machine cible

Deuxièmement, il est généralement plus furtif, car il s'introduit de manière active sur un système : aucune action d'un utilisateur n'est nécessaire. Cela implique qu'un utilisateur même averti [1] aurait du mal à détecter la présence d'un nouveau ver sur son système.

Cependant, il réside des inconvénients de taille de part sa manière d'agir :

- une fois les failles qu'il utilise bouchées, il ne peut plus se propager sur un système patché [2]. Il est donc très dépendant des failles qu'il exploite
- il suffit de couper le réseau pour arrêter sa propagation

1.3 Court historique

Le premier ver informatique recensé est le ver *Morris*, apparu en 1988. Il utilisait 2 failles présentes dans les systèmes UNIX de l'époque, une dans *sendmail*[3] et l'autre dans *fingerd*[4]. On estime à 6000 (soit environ 10% du réseau Internet de l'époque) le nombre de machines infectées par ce ver.

Depuis qu'Internet existe, beaucoup de vers informatiques ont vu le jour et ils sont même devenus une pratique courante d'attaque informatique à des fins militaires [5].

1. qui lui passerait un coup d'antivirus sur un fichier suspect
2. dont la faille a été corrigée
3. la commande UNIX permettant d'envoyer un e-mail
4. commande qui permettait d'obtenir des informations sur la personne derrière une adresse mail
5. Stuxnet (découvert en 2010), ver destiné à espionner les réacteurs nucléaires de Siemens en Iran, endommageait les centrifugeuses en augmentant et en baissant leurs vitesses sans que les techniciens s'en aperçoive

2 Contenu du projet

2.1 Description détaillée

Dans ce projet, je développerai un ver informatique, basé sur une faille système. Le langage de programmation utilisé m'est libre de choix, ainsi que la/les faille(s) à exploiter (le type de système aussi, mais il est lié aux failles). Je réalisera les 3 fonctions essentielles d'un ver, c'est à dire la fonction de propagation, celle de persistance ainsi que la fonction d'attaque. Je veillerai à ce que le ver reste le plus furtif possible sur les systèmes infectés afin de pouvoir se propager sur un maximum de machines.

Par la suite, il faudra l'essayer sur un réseau (coupé du monde, l'Internet n'est pas un réseau pour y tester de tels programmes). Des machines virtuelles feront l'affaire, en espérant qu'elles soient assez nombreuses (idylliquement un parc de 40/50 machines, mais une dizaine suffiront dans un premier temps).

Il faudra après cela essayer d'améliorer ce ver en fonction des résultats observés sur le réseau de test. Si la propagation et la furtivité sont suffisamment satisfaisantes, alors on pourra améliorer les capacités de la charge virale afin d'étendre les privilèges par exemple, ou de surveiller le système de manière discrète pour soutirer des informations sur ses utilisateurs comme des mots de passe, informations personnelles, ...

2.2 Exemples d'utilisations

Voici quelques exemples qui pourraient découler de ce projet :

- puisqu'un ver n'est par définition pas obligatoirement malveillant, pourquoi pas s'en servir de façon bénéfique ? Par exemple, on pourrait se servir de ce ver pour se propager au sein des systèmes vulnérables à la faille qu'il utilise afin de prévenir les administrateurs qu'une faille est présente et qu'il faut mettre à jour pour la corriger.
- si le ver est capable d'étendre ses privilèges [6], alors on pourrait s'en servir pour faire la même chose que l'exemple précédent mais en patchant directement la faille sans prévenir l'utilisateur toute de suite, mais seulement après coup. En entreprise par exemple, au lieu de mettre à jour les machines une à une ce qui peu être lent et exhaustif, on pourrait lâcher un tel ver à condition qu'il ait été analysé par l'administrateur réseau de l'entreprise auparavant (et dans l'idéal, certifié par la communauté).
- a des fins statistiques, on peut s'en servir pour comptabiliser le nombre de machines vulnérables, dans le monde, aux failles qu'il exploite. Le ver, une fois sur une machine vulnérable, envoie sur un serveur prédéfini l'information que le système sur lequel il est est vulnérable. Anonymement si le programmeur est bienveillant, sinon en donnant le maximum d'informations.
- et bien sûr, il est possible d'en faire un usage malveillant : infiltration de systèmes sans autorisation, surveillance, modification, keylogger, backdoor, ...

Ces exemples sont donnés à titre indicatif. Cependant, la terreur que propage le terme "ver" rend ces utilisations difficiles à mettre en œuvre. Car pour la plupart des personnes, les mots "virus" et "ver" sont toujours péjoratifs, alors que la définition ne parle en aucun cas d'un quelconque but malveillant. C'est donc difficile de lancer un ver dans une bonne intention sans être vu comme un malfaisant [7].

6. vers des privilèges root

7. peut-être que la loi de certains états est différente de celle de la France, ici une fois infiltré dans le système le programmeur est coupable

2.3 Contexte

Étant plutôt intéressé par la sécurité informatique (système et réseau) dans son ensemble, je suis assez à la page au niveau des nouvelles failles de sécurité découvertes et publiées. Il s'avère qu'au moment où nous devions choisir nos projets, une faille venait tout juste d'être dévoilée, mettant en péril beaucoup de machines tournant sous un système type UNIX. Je me suis donc mis au travail en cherchant un moyen rapide d'exploiter cette faille, et c'est là que j'ai mis au point le prototype d'un ver. M. Alain Bretto avait fait quelques propositions concernant des projets qu'il souhaitait réaliser avec des étudiants, et la conception d'un ver en faisait partie. J'ai donc sauté sur l'occasion. De plus, j'ai lus quelques ouvrages [2] concernant les virus et les vers. Ce qui m'a permis d'en connaître suffisamment pour en développer.

2.4 Solution

Premièrement, ce ver sera écrit en Perl. Tout simplement parce que Perl est un langage facile d'accès [8], très utilisé pour faire de la configuration système UNIX, et le top du top, il est présent sur la plus grande partie des distributions UNIX. Car il faut quand même que le système cible soit capable d'exécuter le ver ! Cela nous offre donc une quasi certitude que le ver ne sera pas bloqué sur un système parce que ce dernier ne peut pas l'exécuter.

2.4.1 La faille

La première faille utilisée par le ver que je vais développer s'appelle Shellshock. Shellshock est une faille découverte le 24 septembre 2014, touchant le shell bash des système UNIX. D'après certains experts, c'est une des plus grosses failles connues [9] sur de tels systèmes jamais trouvée. Voici quelques détails techniques : bash permet d'exporter des fonctions par une variable d'environnement à des processus enfants.

Par exemple [1] :

```
1 $ env mafonction='() { echo "bonjour vous"; }' bash -c 'mafonction;'
```

Jusqu'ici rien d'anormal : on défini juste la fonction "mafonction", et on demande à un processus fils de l'exécuter. Cependant, si l'on place du code **APRÈS** la fin de la fonction :

```
1 $ env mafonction='() { echo "bonjour vous"; }; echo "voici shellshock" '
    bash -c "mafonction;"
```

retournera

```
1 voici shellshock
2 bonjour vous
```

Ici, lorsque le processus défini la fonction "mafonction" [10], il ne se contente pas d'exécuter "echo "bonjour vous"" mais exécute aussi "echo "voici shellshock"" (le code souligné dans l'exemple). On remarque que le délimiteur ' n'est pas placé à la fin de la fonction mais bien après le code injecté.

On peut dès à présent imaginer les possibilités qu'offre cette faille. Par exemple, il est possible de récupérer les hashs des mots de passe des utilisateurs du système, grâce à cette commande :

```
1 $ env mafonction='() { :; }; cat /etc/passwd' bash -c "echo"
```

8. il faut entendre "à prototypage rapide". Quand on connaît Python, on est pas perdu avec Perl
9. grosse dans le sens qui peut faire le plus de dégâts
10. le fils n'a même pas besoin d'exécuter la fonction, le code situé après est exécuté quoi que le fils fasse

On peut supposer que l'attaquant envoi la réponse de cette commande sur un serveur où il possède les droits afin de la récupérer.

De plus, Bash étant très pratique pour de l'administration système, beaucoup d'utilisateurs s'en sont servis pour écrire des scripts web (CGI). Il alors possible via des requêtes bien précises, d'exécuter du code sur des machines distantes sans aucun privilèges sur celle-ci. Entre autres, via les en-têtes HTTP, il est possible d'y écrire le code source d'un programme afin qu'il se copie dans un fichier sur la machine distante. Nous avons là un ver.

2.4.2 Target aquired, et maintenant ?

Jusque là, nous sommes capable d'infecter une nouvelle machine à condition qu'elle soit vulnérable. Cependant, si elle redémarre, le ver sera tué et ne se relancera pas au redémarrage. Ce qui est gênant car nous souhaitons que le ver persiste et survive aux redémarrages.

Dans un premier temps, nous n'aurons accès qu'à très peu de fichiers dans l'arborescence du système, car nous aurons les droits de l'utilisateur qui exécute le script CGI [11] grâce auquel le ver à infecté la machine. Il nous faudra donc trouver un moyen d'être relancé par un utilisateur non root de la machine, ou au pire des cas, par www-data [12] lui-même.

Dans un deuxième temps, si les droits root sont obtenus, alors il est plus facile de résister aux redémarrages. Le ver pourrait par exemple écrire dans le fichier rc.local [13] ou mieux, dans le dossier init.d [14] en tant que service afin d'être plus discret. On veillera à donner un nom qui ressemble à celui d'un service commun aux systèmes UNIX afin de ne pas éveiller les soupçons.

2.4.3 Méchant ver !

Notre ver résiste aux redémarrages ! Dans l'idéal, il est furtif et ressemble à un processus légitime du système. On peut désormais essayer d'appliquer une charge virale à notre création ! C'est ici que l'imagination entre en jeu, nous pourrions :

- espionner les utilisateurs du système en surveillant les dossiers de leurs home [15]
- tenter de récupérer les mots de passe des utilisateurs en surchargeant la commande sudo [16]
- altérer le système (supprimer des fichiers ...)
- et plein d'autres choses selon l'imagination

2.4.4 Ah, petits problèmes ...

On va me dire, si ce ne sont que de petits problèmes, c'est pas très grave alors ... Eh bien quand même ! Effectivement, quelques problèmes apparaissent :

- exploiter la faille Shellshock nécessite l'accès à un script CGI exécuté avec bash ... Rien ne nous dit que toutes les pages d'un serveur web sont générées avec du CGI+bash !
- et Windows dans tout ça ? Quand est-ce qu'on l'attaque ?

Bon pour Windows, on va laisser tomber [17] ...

Par contre, il est possible de trouver une liste des chemins d'accès les plus courants à des scripts CGI à cette adresse : https://21102894.users.info.unicaen.fr/commoncgipaths.txt [18]

11. en général : "www-data" pour les Debian et "apache" pour les Red-hat
12. l'utilisateur qui exécute le CGI
13. /etc/rc.local : fichier exécuté au démarrage du système avec les droits root
14. /etc/init.d/ : les programmes placés dans ce dossier sont exécutés au démarrage du système, et sont des services, c'est à dire des programmes qui tournent en tâche de fond
15. dossier personnel
16. commande permettant d'exécuter la commande qui suit avec des droits root
17. vous allez rire, mais bash existe pour Windows et il s'avère que certains administrateurs web s'en servent pour leur Windows Server, donc techniquement, il est possible de s'attaquer à ce type de machine ...
18. récupéré depuis https://shellshock.detectify.com/

2.5 Calendrier prévisionnel

Date	fonctions disponibles
30 nov.	le ver est capable d'infecter une nouvelle machine vulnérable en sachant son adresse IP et le chemin d'un script CGI+bash
15 dec.	il peut infecter un grand nombre de machines en récupérant une IP depuis un fichier qui peut être généré aléatoirement ou avec une liste de cibles définies par l'attaquant et en itérant sur la liste des chemins les plus courant aux CGI
31 dec.	il est capable de résister aux redémarrages de la machine grâce à des droits non root
20 jan.	le ver parvient à capturer des mots de passe utilisateurs en surchargeant la commande sudo
10 fev.	avec des mots de passe utilisateurs ayant des droits root, le ver est capable de résister aux redémarrages avec des droits root
25 fev.	désormais, il est capable d'obtenir d'obtenir des informations sur les utilisateurs du système en les surveillant et d'acheminer les données sur un serveur distant

Évidement, il est possible qu'il change au cours du temps, mais je vais essayer de m'y tenir au maximum.

3 Conclusion

On pourrait conclure en disant qu'un tel ver lâché sur le grand réseau qu'est l'Internet pourrait faire des ravages, mais à vrai dire, la plupart des serveurs vulnérables ont été mis à jour depuis. Cependant, il ne faut pas omettre le reste des machines qui tournent sur des systèmes type UNIX et qui ne sont pas des serveurs : ordinateurs de poche [19], routeurs, machin-box [20], caméra-ip, et j'en passe. Le problème de ces appareils est qu'ils ne sont pas tous mis à jour régulièrement (cela dépend en grande partie de la bonne volonté du constructeur). Et la plupart du temps, ils sont connectés à Internet (et avoir une IP privée ne suffit pas à être caché).

Il ne m'est pas demandé de développer une parade à ce ver, mais si le temps me le permet, je mettrai au point un programme capable dans un premier temps de stopper la propagation du ver, pour ensuite désinfecter totalement la machine, puis en patchant le système afin que ce ver ne puisse plus y pénétrer.

Glossary

backdoor porte dérobée en français : programme qui introduit un moyen d'accéder à la machine pour un usage futur. 3

bash bash est le plus répandu des shells disponibles sous UNIX. 4–6

CGI Common Gateway Interface : script utilisé pour générer des pages HTML. 5, 6

charge virale la charge virale est en quelques sortes le but final du ver, ce qu'il fait sur le système à part se dupliquer et s'y loger. 1–3, 5

droit root ce sont les droits les plus élevés des systèmes UNIX. Ils permettent d'écrire n'importe où sur le disque (à quelques conditions près). 5, 6

19. appelés par les médias "smartphones"
20. la boite noire qui vous donne accès à Internet chez vous

en-têtes HTTP les en-têtes HTTP sont des variables du protocole HTTP permettant d'identifier plus précisément le client, tels que le navigateur qu'il utilise, de quel site web il provient,.... 5

faille bug/problème d'un programme du système de la machine permettant un accès non permis à une personne tierce, ou d'exécuter des commandes à distance. 2–5

hashs le hash d'un mot de passe est l'image de ce mot de passe par une fonction non-injective pour laquelle il est difficile de retrouver l'antécédent d'une image. Cela permet de ne pas stocker le mot de passe en clair, tout en étant capable de vérifié qu'un mot de passe donné de façon relativement rapide. 4

keylogger programme tournant en tâche de fond qui enregistre les frappes de touche du clavier. 3

programme hôte programme dans lequel le virus est logé. A chaque fois que ce programme est exécuté, le virus l'est aussi. 2

shell un shell est une interface en ligne de commande permettant d'interagir avec le système. 4

Shellshock faille informatique touchant les systèmes UNIX utilisant le shell bash. 1, 4, 5

UNIX Famille de systèmes d'exploitation. En particulier, les GNU/Linux, MacOSX, sont des systèmes type UNIX. 2, 4–6

Références

[1] Linuxfr Collectif. Une faille nommée "shellshock". 2014. http://linuxfr.org/news/une-faille-nommee-shellshock#explications-techniques.

[2] Eric Filiol. *Les virus informatiques : théorie, pratique et applications*. Springer, 2ème edition, 2009.