Virus Bénéfiques
Théorie et pratique

Michel Dubois
myshell.dubois@neuf.fr
http://vaccin.sourceforge.net

8 août 2006

Le 22 mars 1991, le record mondial de vitesse de calcul par un ordinateur est battu par une société du Massachusetts spécalisée dans le calcul parallèle[1]. Le record précédent était détenu par un virus écrit et distribué sur l'Internet par Robert Morris, alors étudiant à l'université de Cornell.

C'est par ce constat que Fred Cohen commence son article sur les virus bénéfiques [1]. Il démontre par la suite, que les technologies virales peuvent être utilisées à des fins bénéfiques. En effet, si un virus exécute un certain nombre de calculs sur les ordinateurs qu'il infecte et qu'il peut ensuite coordonner les résultats de ces calculs, il met en place une grille d'ordinateurs effectuant des calculs en parallèle et ceci à l'échelle mondiale.

1 Caractéristiques d'un virus bénéfique

Plusieurs spécialistes de la virologie informatique ont étudié la possibilité d'utiliser les virus à des fins bénéfiques. C'est à Vesselin Bontchev que revient le mérite d'avoir fixé les critères permettant de différencier un virus bénéfique d'un autre virus [2].

Dans le cadre d'une enquête réalisée sur le forum de news *Virus-L/comp.virus*, Vesselin Bontchev a demandé aux internautes de décrire les critères qui, selon eux, faisaient que les virus étaient perçus comme nocifs. Il a ensuite rassemblé et classé les réponses et en a déduit les critères définissant un virus bénéfique.

1.1 Les critères techniques

1.1.1 Le contrôle

Le premier critère technique concerne le contrôle de la propagation du virus. Par défaut, un virus se propage d'ordinateurs en ordinateurs de façon aléatoire. Ainsi, il n'est à priori pas possible de déterminer à l'avance si un ordinateur va être infecté ou pas. Un virus bénéfique doit pouvoir être contrôlé et les ordinateurs cibles fixés à l'avance.

Une solution consiste à mettre en place un système d'invitation. Avant d'infecter un ordinateur, le virus vérifie s'il y est invité et c'est seulement en cas de réponse positive qu'il s'installe sur l'ordinateur.

Vesselin Bontchev imagine alors le scénario suivant : soit une compagnie spécialisée dans la production de virus bénéfiques. Lors de la création d'un nouveau virus, elle le rend disponible sur l'un de ses sites de dépôts. Elle envoie ensuite à ses abonnés, un message contenant la clé publique du virus. L'administrateur d'un réseau désirant utiliser ce virus contrôle l'authenticité du message et envoie une invitation signée électroniquement. Cette dernière précise alors le type de virus désiré, l'adresse du réseau concerné et le mode d'entrée dans les systèmes (par exemple un numéro de port spécifique). Une fois le message reçu, le dépôt de virus bénéfiques envoie le virus choisi sur le réseau cible. Une fois sur place, le virus s'authentifie et commence à se reproduire.

[1] Depuis le 30 juin 2005, ce record est détenu par l'équipe OASIS qui rassemble des chercheurs de l'INRIA Sophia Antipolis, du laboratoire CNRS I3S et de l'Université de Nice Sophia-Antipolis. Ces chercheurs sont les premiers au monde à avoir calculé le nombre de façons de déposer 25 reines sur un échiquier de taille 25 x 25 de sorte que les reines ne puissent être en prise. Ce nombre est égal à 2 207 893 435 808 352. Le calcul s'est exécuté sur une grille de PCs de bureau à l'INRIA Sophia Antipolis.

1.1.2 L'identification

Le deuxième critère technique concerne l'identification du virus bénéfique par les antivirus. Les programmes antivirus détectent les virus nocifs soit à partir de leur signature, soit en contrôlant les modifications effectuées sur les programmes et documents, soit en détectant les processus et actions suspectes.

Une fois qu'un virus bénéfique est authentifié sur un ordinateur, il est possible de spécifier à l'antivirus de considérer ce virus comme un programme sûr. Ce mécanisme inhibe la détection par signature. Concernant les autres modes de détections, le virus bénéfique ne doit pas effectuer de modifications sur les programmes et documents du système. Par conséquent, un virus bénéfique ne peut être qu'un ver se propageant uniquement par les réseaux.

1.1.3 La gestion des ressources

De nombreux virus nocifs, provoquent une surcharge du système infecté pouvant aller jusqu'au déni de service.

Pour éviter ce problème, un virus bénéfique doit être conçu afin de consommer le minimum de mémoire, d'espace disque ou de temps processeur. En fait, la charge système occasionnée par le virus doit être négligeable par rapport aux bénéfices que l'utilisateur en retire. Concrètement cela implique, entre autres, qu'il n'y est qu'une seule instance du virus sur le système.

1.1.4 Les bogues

Comme tout programme informatique, un virus contient des bogues. Dans certains cas, comme avec le RTM WORM, ces bogues ont entraîné la perte de contrôle du virus.

Une fois qu'un bogue, découvert dans le code d'un virus bénéfique, est corrigé, la version patchée du virus doit être fournie avec les mêmes règles d'authentification. En reprenant le scénario évoqué plus haut, la nouvelle version du virus pourrait être diffusée de la même manière que la version originale : envoi d'un message contenant la clé publique du virus patché, puis authentification par le système et mise à jour des instances du virus présentes sur le réseau.

De plus, la possibilité de provoquer l'arrêt instantané du virus doit être proposée aux utilisateurs. En effet, si un administrateur constate un fonctionnement anormal sur un virus bénéfique, il doit pouvoir imposer l'arrêt de toutes les instances du virus sur son réseau.

1.2 Les critères éthiques et légaux

1.2.1 Modification des données

La plupart des législations interdisent la modification des données personnelles sans l'autorisation du propriétaire. Un virus bénéfique ne doit donc modifier aucune donnée. Le scénario envisagé permet de vérifier ce critère. Les actions effectuées par le virus bénéfique sont connues à l'avance et l'utilisateur "invite" lui-même le virus sur son système.

1.2.2 Détournement du virus

Un virus bénéfique peut être utilisé par un pirate comme vecteur de transport pour pénétrer dans un réseau. Afin de supprimer ce risque, un virus bénéfique doit mettre en œuvre des techniques cryptographiques fortes pour s'authentifier dans les réseaux qu'il colonise. Une telle authentification interdisant toute modification du virus par un éventuel pirate.

Enfin, un pirate pourrait analyser un virus bénéfique afin de s'approprier son code et de développer un virus nocif similaire. Cependant, étant donné qu'un virus bénéfique tel que nous l'avons décrit, ne pénètre pas de lui-même dans un système mais attend d'y être invité, le pirate devra convaincre un utilisateur ou l'administrateur réseau du bien fondé de son virus.

1.2.3 La responsabilité

La question de la responsabilité en cas d'atteinte à l'intégrité, la disponibilité ou la confidentialité d'un système d'information est un problème complexe. Les administrateurs réseaux sont par nature très méfiants

quand à l'utilisation de programmes autoreproducteurs sur leur système d'information.

Si les conditions d'implémentation d'un virus bénéfique telles que nous les avons décrites sont respectées, la responsabilité, en cas d'incident, ne repose plus intégralement sur l'administrateur. En effet, dans le cas du scénario exposé ci-dessus, il existe nécessairement un contrat signé entre l'entreprise fournissant les virus bénéfiques et l'entreprise achetant ces virus. C'est donc au niveau du contrat que les responsabilités s'établissent au même titre que lorsque la même entreprise achète un système d'exploitation ou un logiciel de comptabilité.

1.3 Les critères psychologiques

1.3.1 La confiance

Souvent, l'utilisateur d'un ordinateur s'imagine qu'il exerce un contrôle total sur ce qui se passe dans sa machine. En fait, l'ordinateur étant un outil très sophistiqué, la plupart des utilisateurs ne saisissent pas très bien les rouages de son fonctionnement interne. Cette incertitude suscite de la méfiance vis à vis de la machine et seul le sentiment de contrôler les réactions de l'ordinateur peut aider l'utilisateur à surmonter son appréhension.

C'est pourquoi, l'implémentation d'un virus bénéfique dans un réseau d'entreprise doit se faire de manière à ce que l'utilisateur continue de croire qu'il maîtrise le fonctionnement de son ordinateur. Ceci peut s'obtenir, par exemple, en lançant une campagne de communication interne. Dans ce cas, il s'agira d'expliquer le mode de fonctionnement du virus, en insistant sur son utilité, ses avantages et surtout la possibilité de le désactiver en cas de problème.

1.3.2 La vision négative des virus

Les médias ont grandement contribué à créer une vision néfaste des virus en les présentant comme étant intrinsèquement mauvais. Et de fait, la plupart des utilisateurs suspecte les virus dès qu'ils ont un problème sur leur ordinateur.

Un administrateur désirant utiliser un virus bénéfique sur son système d'information, ne doit pas l'appeller *virus*, s'il veut que l'action bénéfique de son virus soit reconnue.

En résumé, pour qu'un virus puisse être considéré comme bénéfique, il doit répondre aux impératifs suivants :

- Attendre de recevoir une invitation avant de coloniser un ordinateur (présence sur le système cible d'un fichier spécifique par exemple) ;
- Utiliser un mécanisme de chiffrement fort pour s'authentifier auprès du système (utilisation d'une connection *SSH* par exemple) ;
- Se suffire à lui-même et ne pas modifier d'autres programmes (être un ver réseau) ;
- Ne pas être considéré comme un virus ;

2 Le virus KOH

Un certain nombres de virus bénéfiques ont déjà vu le jour, cependant, aucun d'entre eux ne répond à l'ensemble des critères que nous venons d'évoquer. Nous pouvons citer notamment :

- des virus antivirus :
  - DEN ZUK qui enleve le virus BRAIN ;
  - les dernières versions de YANKEE_DOODLE qui suppriment les versions précédentes du virus ;
  - NEUROQUILA qui désactivent un certain nombre d'antivirus et suppriment certains virus ;
- des virus de compression de fichiers comme CRUNCHER. Le concept de ce virus est de faire gagner de la place sur le disque dur en compressant et en décompressant les fichiers en fonction de leur utilisation ;
- des virus pour la maintenance des ordinateurs comme le virus MAINTENANCE décrit par Fred Cohen [3] et qui permet d'effacer les fichiers temporarires, d'arrêter les processus zombies ou encore de vérifier les permissions sur les fichiers ;
- des virus de chiffrement de disque dur comme KOH ;

À titre d'exemple, nous allons détailler le fonctionnement du virus KOH. Ce virus, écrit par Mark Ludwig [4], permet de chiffrer l'ensemble d'un disque dur.

2.1 Justification

2.1.1 virus de démarrage résident

Si l'on souhaite chiffrer l'intégralité d'un disque dur, les technologies virales semblent les plus appropriées. En effet, un logiciel standard de chiffrement est dépendant du système d'exploitation et doit donc attendre que celui-ci soit chargé pour pouvoir fonctionner. Cette contrainte implique que le secteur de démarrage du disque dur ne soit pas chiffré, ce qui représente une vulnérabilité non négligeable.

L'utilisation d'un virus de démarrage permet de s'affranchir du système d'exploitation et de chiffrer l'intégralité du disque dur. De plus, le virus est résident, ce qui offre plusieurs avantages. Tout d'abord, le virus est exécuté et chargé en mémoire à chaque démarrage de l'ordinateur, ensuite, le virus exécute les opérations de chiffrement et de déchiffrement en toute transparence.

Au final, nous avons un virus qui se lance automatiquement, dès le démarrage de l'ordinateur, et qui gère les opérations de chiffrement et de déchiffrement de la totalité du disque dur de façon transparente pour l'utilisateur.

2.1.2 Autoreproduction

À l'époque ou KOH a été écrit, l'utilisation des disquettes était généralisée. En tant que programme autoreproducteur, KOH présente alors plusieurs avantages :

- il suffit d'insérer une disquette dans le lecteur pour qu'aussitôt, le virus se recopie dessus et la chiffre. L'utilisateur n'a plus à se soucier de savoir si l'un de ses supports de stockage est chiffré ou non ;
- l'utilisateur n'a pas besoin d'emmener avec lui un logiciel de déchiffrement pour pouvoir lire ses informations sur d'autres ordinateurs. Le virus étant sur la disquette, l'insertion de celle-ci dans un autre ordinateur entraîne l'activation du virus sur ce dernier ;

Un tel outil de chiffrement - déchiffrement des supports de stockage est un réel atout pour une entreprise ou une administration. En effet, il est difficile de contrôler les flux entrant et sortant des disquettes. KOH utilise des clés de chiffrement différentes pour les disquettes et les disques durs. L'administrateur réseau peut très bien distribuer les clés de chiffrement des disques durs mais pas celles des disquettes. Ainsi, l'utilisateur indélicat ne peux pas lire ses disquettes sur un ordinateur autre que ceux de son lieu de travail.

2.2 Modes d'action

KOH est un virus de démarrage, résident et non furtif, ayant comme unique action d'assurer le chiffrement et le déchiffrement des supports de stockage. Les sources du virus comprennent cinq fichiers :

- *KOH.ASM* qui constitue le fichier principal. Il contient, entre autre, le *loader*, le secteur de démarrage, les gestionnaires d'interruptions et le code de chiffrement du disque dur ;
- *KOHIDEA.ASM* est un fichier *include* qui implémente l'algorithme de chiffrement *IDEA* ;
- *FATMAN.ASM* est un fichier *include* qui permet de gérer les routines de la FAT ;
- *PASS.ASM* est un fichier *include* qui contient la routine de saisie de la phrase clé ;
- *RAND.ASM* est un fichier *include* qui contient le générateur de nombres pseudo-aléatoires ;

2.2.1 Colonisation

KOH se copie sur les disquettes et disques durs en remplaçant le secteur de démarrage original par le sien, puis en copiant le reste de son code dans une zone inoccupée du disque. Cette zone est protégée en marquant, dans la *FAT*, les secteurs correspondants comme défectueux.

Le virus n'infecte les disquettes que si l'indicateur de condition *FD_INFECT* est égal à un. Dans ce cas, il commence par chiffrer le contenu de la disquette de sorte que si le processus de contamination échoue (manque d'espace libre), la disquette peut être utilisée normalement.

Lorsqu'un ordinateur démarre à partir d'une disquette contenant KOH, celui-ci demande s'il faut chiffrer le disque dur (voir FIG. 2). En cas de réponse positive, le disque dur est chiffré puis le système démarre normalement. En cas de réponse négative, le virus ne fait rien, par contre, il repose la question à chaque redémarrage de l'ordinateur.

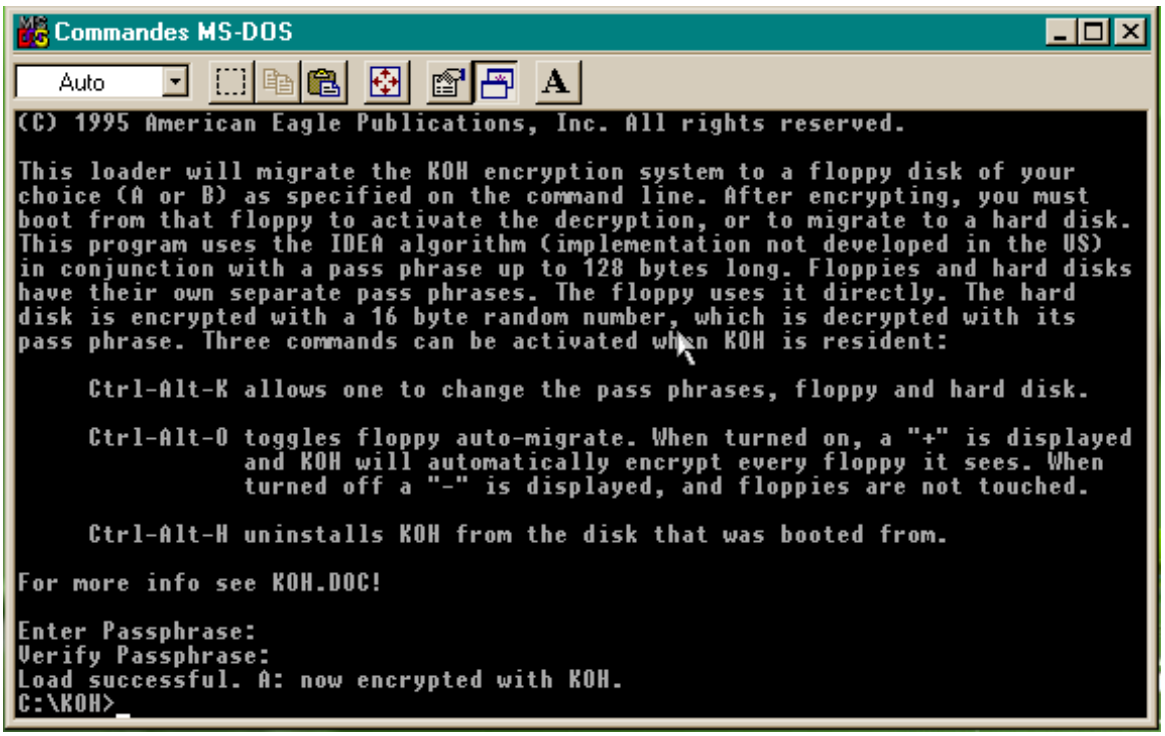

FIG. 1 – Création d'une disquette de boot contenant KOH

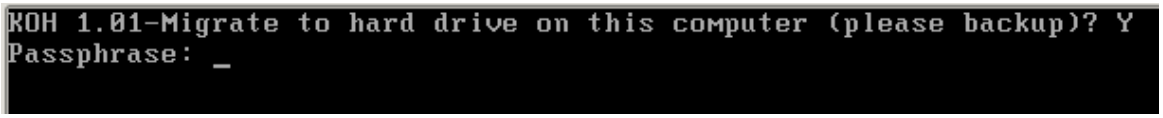

FIG. 2 – Chiffrement d'un disque dur par KOH pendant la séquence de boot

2.2.2 Chiffrement

KOH utilise l'*International Data Encryption Algorithm (IDEA)*[2] pour chiffrer et déchiffrer les données. *IDEA* chiffre des blocs de 64 bits avec des clés de 128 bits. Les processus de chiffrement et de déchiffrement sont similaires.

KOH conserve trois clés de 128 bits stockées dans *HD_KEY*, *HD_HPP* et *FD_HPP*[3].

2.2.3 Détournement d'interruption

KOH détourne les interruptions *13H* pour le disque dur et *09H* pour le clavier.

Le *DOS* utilise l'interruption *13H* pour accéder au disque. En interceptant tous les appels à l'interruption *13H*, KOH peut effectuer les opérations de chiffrement - déchiffrement de façon totalement transparente. Ainsi, lorsque le *DOS* cherche à ouvrir un fichier sur le disque, le virus intercepte l'appel, déchiffre le fichier et l'envoie au système d'exploitation. Lors de l'enregistrement d'un fichier, l'opération inverse se déroule.

L'interruption *09H* est utilisée par le *DOS* pour la gestion du clavier. Le détournement de cette interruption permet à KOH de mettre en place des raccourcis clavier. Ainsi, la combinaison de touches <*CTRL*> <*ALT*> <*K*> permet de changer de phrase clé, <*CTRL*> <*ALT*> <*O*> permet de dire à KOH de chiffrer automatiquement ou non les disquettes et <*CTRL*> <*ALT*> <*H*> désinstalle le virus.

[2] *IDEA* est un algorithme de chiffrement par bloc mis au point, en 1991, par Xuejia Lai et James Massey de l'Institut Fédéral de Technologie Suisse. Initialement cet algorithme a été crée pour emplacer le *Data Encryption Standard (DES)*.

[3] *HPP* est l'acronyme de Hashed PassPhrase.

2.3 KOH virus bénéfique ?

Le virus KOH répond-il aux critères d'un virus bénéfique tels que nous les avons définis ci-dessus ?

Concernant les critères techniques, le contrôle de la propagation du virus n'est pas assuré. Certes, la demande d'autorisation de chiffrement du disque dur est effective, mais lorsque la question est posée, le virus est déjà dans l'ordinateur. Concernant l'identification, KOH est considéré, par les logiciels antivirus, comme étant un virus de la famille des virus STEALTH_BOOT. Il ne répond donc pas au critère. Concernant le critère de gestion des ressources, KOH est optimisé pour effectuer du chiffrement de disque dur à la volée. En dehors du chiffrement initial du disque, son action est transparente pour l'utilisateur. Il répond donc correctement à ce critère. Enfin pour le critère de gestion des bogues, il n'existe aucune infrastructure permettant de mettre à jour le virus. Cependant, lors de son lancement, KOH affiche à l'écran un avertissement (voir FIG. 2) demandant de faire une sauvegarde du système avant d'effectuer le chiffrement. Donc en cas de bogue dans le code du virus, l'utilisateur dispose normalement d'une sauvegarde de ses données.

Concernant les critères éthiques et légaux, KOH ne modifie pas les données personnelles de l'utilisateur : une fois le chiffrement terminé, l'utilisateur retrouve l'ensemble de ses informations. Par contre, le code source du virus étant ouvert, un développeur mal intentionné peut rajouter des routines nocives au virus et le faire passer pour l'original. Il n'y a aucun contrôle de l'authenticité du virus lors de son utilisation. Enfin, le problème de la responsabilité reste entièrement du domaine de l'administrateur ou de l'utilisateur.

Concernant les critères psychologiques, KOH met en confiance son utilisateur : les messages qu'il affiche et la documentation fournie par son auteur sont clairs et précis. Malheureusement, KOH est considéré comme un virus, et à ce titre un administrateur sera naturellement méfiant avant de l'implanter sur son réseau.

En conclusion, KOH ne satisfait pas pleinement aux critères d'un virus bénéfique. Cependant, il n'existe pas actuellement de virus bénéfique répondant à l'ensemble des critères techniques, éthiques, légaux et psychologiques. KOH est, à ma connaissance, celui qui s'en rapproche le plus.

Nous venons de détailler les critères permettant de différencier un virus bénéfique d'un virus nocif. Nous allons maintenant nous attacher à mettre en œuvre un virus bénéfique. L'objectif de ce développement est de fournir un outil d'aide à l'administration réseau basé sur les technologies virales.

3 Objectifs et code

Le scénario retenu est le suivant. Le système d'information d'une entreprise X s'appuie sur un réseau d'ordinateurs sous *GNU/Linux*. L'administrateur du réseau souhaite mettre en place un système flexible de suivi de l'ensemble des ordinateurs. Pour cela, il décide d'utiliser les technologies virales et installe le ver bénéfique VACCIN sur son réseau.

3.1 Diagramme fonctionnel

Le mode de fonctionnement de VACCIN s'articule selon le diagramme de la figure 3.

Tout d'abord, VACCIN contrôle son niveau d'accès au système. S'il est exécuté par un autre utilisateur que *root*, il s'arrête. Le test suivant permet à VACCIN de déterminer à partir de quel ordinateur il est lancé. Soit le ver est exécuté à partir de l'ordinateur de l'administrateur réseau et dans ce cas il lance ses routines de colonisation, soit il est exécuté sur un autre ordinateur et dans ce cas il lance ses routines de collecte de données.

3.1.1 Colonisation

La routine de colonisation de VACCIN commence par un scan du réseau. Celui-ci consiste, à partir de l'adresse IP du poste de l'administrateur et du masque réseau, à déterminer la plage d'adresses IP des ordinateurs. Ensuite, une requête est envoyée à chacune de ces adresses sur le port *SSH*, cette routine permet de définir une liste des ordinateurs effectivements connectés au réseau et faisant tourner un serveur *SSH*.

Une fois la liste des ordinateurs connectés au réseau construite, VACCIN la parcourt et établit une connexion *SSH* avec chacune des machines dont l'adresse IP y est référencée. Ensuite, le ver se recopie sur les ordinateurs et s'y exécute.

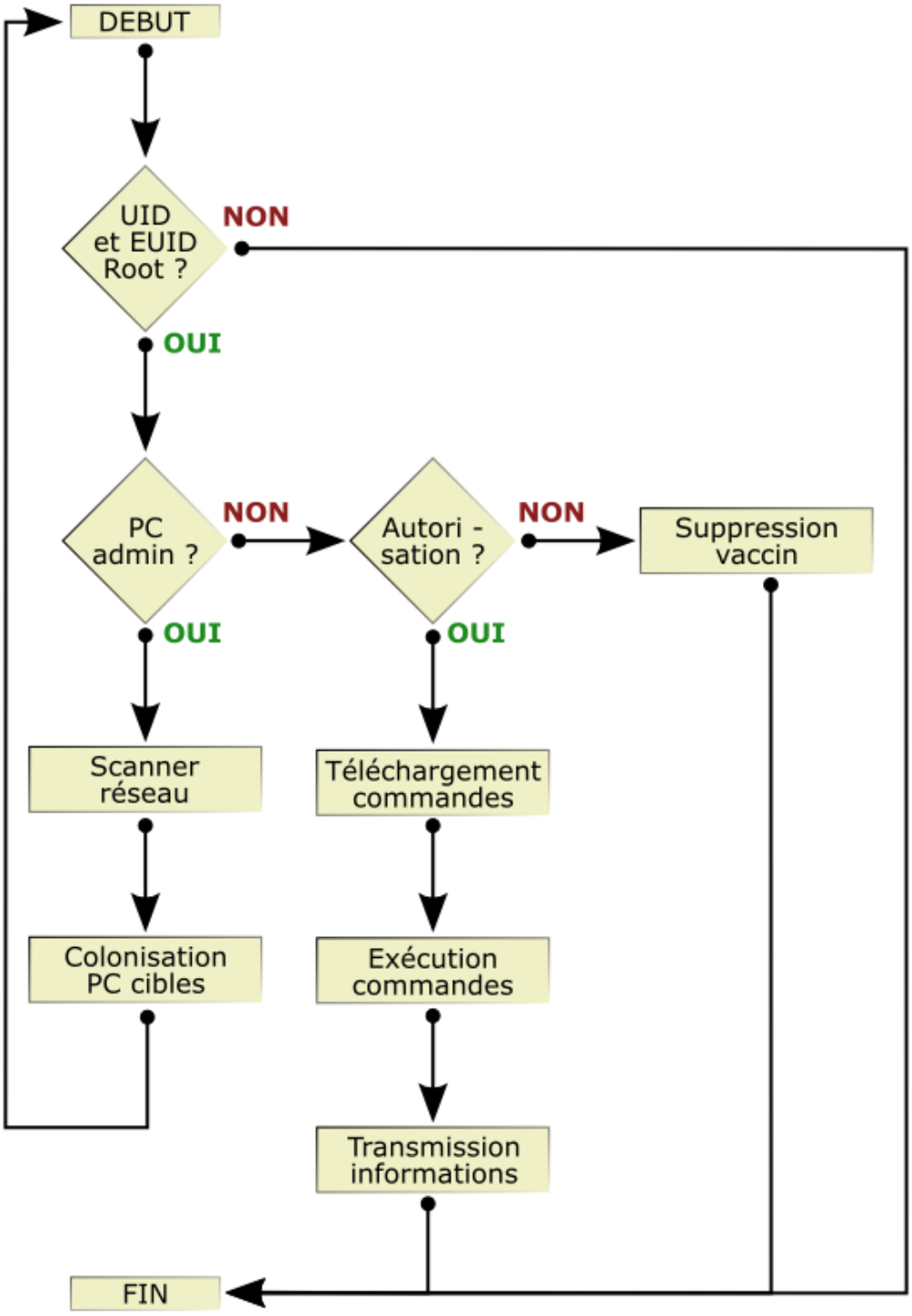

FIG. 3 – Diagramme fonctionnel de VACCIN

3.1.2 Collecte de données

Lorsque VACCIN est lancé sur un ordinateur du réseau autre que celui de l'administrateur, il commence par vérifier qu'il y est autorisé. Si c'est le cas, il télécharge sur le poste de l'administrateur un fichier de commandes qu'il exécute ensuite et il renvoie le résultat sur l'ordinateur de l'aministrateur sous forme de fichier texte. Si VACCIN n'est pas autorisé à s'exécuter il s'efface automatiquement de l'ordinateur concerné.

3.2 Le code

3.2.1 Remarques préliminaires

VACCIN est un ver écrit en langage C.

Afin de clarifier le code, les routines de gestion des erreurs n'ont pas été implémentées. Par conséquent, en cas d'erreur d'ouverture d'une *socket*, d'un fichier ou encore d'exécution d'une commande *system* le programme s'arrête sans afficher de message.

```
#define SOURCE_HOST_IP "192.168.184.1"
#define MASK_NETWORK "255.255.255.0"
#define CONTROL_FILE "/root/datav"
#define COMMAND_FILE "/root/commandes.sh"
#define VER_FILE "/root/vaccin"
```

CODE 3.1: Les variables définies par VACCIN

Pour pouvoir fonctionner, VACCIN définit cinq variables (voir CODE 3.1). *SOURCE _HOST _IP* et *MASK _NETWORK* définissent respectivement l'adresse IP de l'ordinateur de l'administrateur et le masque réseau utilisé. Enfin, *CONTROL_FILE*, *COMMAND_FILE* et *VER_FILE* définissent respectivement le nom complet du fichier de contrôle, le nom complet du fichier de commandes et le nom complet du fichier dans lequel VACCIN sera copié lors de sa colonisation.

3.2.2 La fonction *main*

Comme tout programme écrit en C, VACCIN commence par exécuter la procédure *main*.

```
openlog("vaccin", LOG_PID, LOG_USER);
...
syslog(LOG_NOTICE, "## Lancement de vaccin par root.\n");
...
closelog();
```

CODE 3.2: Initialisation, écriture et fermeture des opérations *syslog*

VACCIN est un ver il se doit donc d'être le plus possible furtif. C'est pour cette raison qu'il n'affiche aucun message sur la console par le biais de *stdout* ou de *stderr*. Cependant, l'administrateur, qui l'utilise, doit pouvoir suivre son fonctionnement, c'est pourquoi les messages d'état de VACCIN sont transmis au système par le biais du démon *syslogd*. La procédure *main* débute donc par l'ouverture d'une session *syslog* (voir CODE 3.2) avec la commande *openlog()*. Tout au long de son exécution les messages sont envoyés au démon *syslogd* par la commande *syslog()*. Cette dernière prend comme premier argument une constante qui permet de définir l'urgence du message. Les deux niveaux d'urgences utilisés par VACCIN sont présentés dans la figure 4. Enfin, VACCIN se termine par la cloture de la session *syslog* grâce à la commande *closelog()*.

Une fois la session *syslog* initialisée, VACCIN appel la fonction *isRoot()*. Celle-ci permet de déterminer que c'est bien l'utilisateur *root* qui a lancé le programme. En cas de réponse négative, VACCIN s'arrête et envoi un message de niveau critique au démon *syslogd*. Dans le cas contraire, VACCIN appel la fonction *isSourceHost()*. Cette dernière détermine l'adresse IP de l'ordinateur sur lequel le ver s'exécute. S'il s'agit de l'ordinateur de l'administrateur, VACCIN crée la liste des machines connectées au réseau grâce à la fonction *scanNetwork()* et les colonise en appellant la fonction *colonise()*. Si, au contraire l'adresse IP correspond a

NOM	SIGNIFICATION
LOG_NOTICE	Information importante, mais fonctionnement normal.
LOG_CRIT	Des conditions critiques se présentent, pouvant nécessiter une intervention.

FIG. 4 – Les deux niveaux d'alertes de VACCIN

une machine du réseau, VACCIN lance la fonction *isAutorise()* pour déterminer s'il est, oui ou non autorisé à s'exécuter sur son hôte. En cas de réponse positive, le ver exécute la fonction *recupInfos()* et en cas de réponse négative VACCIN s'efface du système en exécutant la fonction *effaceVaccin()*.

3.2.3 La fonction *isRoot*

```
int isRoot()
{
  int result;
  if( ( getuid() != 0 ) || ( geteuid() != 0 ) )
    result = 0;
  else
    result = 1;
  return(result);
}
```

CODE 3.3: La fonction *isRoot*

La fonction *isRoot()* utilise les fonctions système *uid_t getuid (void)* et *uid_t geteuid (void)* pour identifier l'utilisateur qui exécute le ver. Si l'*UID* réel ou effectif de l'utilisateur est différent de zéro, *isRoot()* retourne O sinon elle retourne 1.

3.2.4 La fonction *isSourceHost*

À partir des *IOCTLs*, la fonction *isSourceHost* construit la liste des interfaces réseaux disponibles sur l'ordinateur hôte. Cette liste est ensuite parcourut et l'adresse IP de chaque interface est comparée à celle de l'ordinateur de l'administrateur définit dans *SOURCE_HOST_IP*. S'il y a correspondance, le ver tourne sur l'ordinateur de l'administrateur et *isSourceHost* renvoie 1. Dans le cas contraire, le ver tourne sur l'une des machines du réseau et *isSourceHost* renvoie 0.

3.2.5 La fonction *scanNetwork*

La fonction *scanNetwork* exécute deux opérations. Tout d'abord (voir CODE 3.4) elle détermine, à partir du masque réseau définit dans *MASK_NETWORK*, la plage d'adresse des ordinateurs du réseau. De cette information *scanNetwork* déduit le nombre de machines adressables et initialise la variable *nbrComputer*.

La deuxième opération réalisée par *scanNetwork* (voir CODE 3.5) consiste à effectuer une tentative de connexion sur l'ensemble de la plage d'adresses IP précédemment déterminée, en utilisant le port *SSH* (port numéro 22). Cette routine permet de construire une liste des ordinateurs connectés au réseau et ayant un serveur *SSH fonctionnel*.

3.2.6 La fonction *colonise*

C'est la fonction *colonise* qui réalise la copie du ver sur les machines distantes (voir CODE 3.6). En s'appuyant sur la commande *system*, la colonisation d'un ordinateur par VACCIN se déroule en trois étapes :

1. Le ver ouvre une connexion SSH sur l'ordinateur distant et, grâce à la commande *scp*, se copie dessus sous le nom fixé par la variable *VER_FILE* ;
2. Ensuite, le ver rajoute une ligne dans la *crontab* de l'utilisateur *root* (sur l'ordinateur distant) pour que le ver s'exécute automatiquement toute les heures ;

```
inet_aton(MASK_NETWORK, &mask);
inet_aton("255.255.255.255", &broadcast);

nbrComputer = ntohl(broadcast.s_addr ^ mask.s_addr);
```

CODE 3.4: La routine de calcul du nombre d'ordinateurs adressables

```
for (i = 1; i < nbrComputer ; i++)
{
  hostIP.s_addr = htonl(ntohl( hostMask ) + i);
  if( strcmp( inet_ntoa( hostIP ), SOURCE_HOST_IP ) == 0)
    continue; /* l'ordinateur de l'administrateur n'entre pas dans la liste */
  bzero(&sockHostIP, sizeof(sockHostIP));
  sock = socket(AF_INET, SOCK_STREAM, 0);
  sockHostIP.sin_family = AF_INET;
  sockHostIP.sin_port = htons (22);
  sockHostIP.sin_addr=hostIP;
  if ( connect (sock, (struct sockaddr *)&sockHostIP, sizeof(sockHostIP)) == 0)
  {
    resultIP = (struct in_addr *) realloc( resultIP, (compteur+1) * sizeof(struct in_addr
        ));
    resultIP[compteur] = hostIP;
    compteur++;
  }
  syslog(LOG_NOTICE, "## %s\n", inet_ntoa(hostIP));
}
```

CODE 3.5: La routine de scan du réseau

```
/* Copie du ver sur l'ordinateur distant */
syslog(LOG_NOTICE, "## copie du ver sur %s.\n", inet_ntoa(adresse));
sprintf(commande, "/usr/bin/scp %s %s:%s", nom, destinataire, VER_FILE );
system( commande );

/* Rajout d'une ligne dans la crontab */
syslog(LOG_NOTICE, "## configuration du lancement automatique pour %s.\n", inet_ntoa(
    adresse));
sprintf(commande, "/usr/bin/ssh %s \"echo 0 \\* \\* \\* \\* %s >> /var/spool/cron/tabs/
    root\" ", destinataire, VER_FILE);
system( commande );

/* Premiere execution du ver */
syslog(LOG_NOTICE, "## Premiere execution du ver sur %s.\n", inet_ntoa(adresse));
sprintf(commande, "/usr/bin/ssh %s \"%s\" ", destinataire, VER_FILE);
system( commande );
```

CODE 3.6: La routine de colonisation

3. Enfin, le ver se lance sur l'ordinateur qu'il vient de coloniser ;

3.2.7 La fonction *isAutorise*

Lorsque VACCIN s'exécute sur un ordinateur autre que celui de l'administrateur, il lance, en premier lieu, la fonction *isAutorise*. Le principe de cette routine consiste à déterminer l'existence du fichier dont le nom est fixé dans la constante *CONTROL_FILE*. Pour effectuer cette opération, *isAutorise* tente d'ouvrir *CONTROL_FILE* à l'aide de la fonction *fopen*. Cette fonction présente l'avantage de ne pas créer le fichier s'il n'existe pas, donc si la tentative d'ouverture échoue c'est que le fichier n'a pas été créé par l'administrateur. VACCIN en conclu qu'il n'est pas autorisé à s'exécuter sur ce poste.

La fonction *isAutorise* retourne 1 si le fichier *CONTROL_FILE* existe et 0 sinon.

3.2.8 La fonction *effaceVaccin*

```
syslog(LOG_NOTICE, "## Suppression du vaccin.\n");
unlink(VER_FILE);

syslog(LOG_NOTICE, "## Reinitialisation de la crontab.\n");
sprintf(commande, "perl -ne 'print unless /0 * * * */' /var/spool/cron/tabs/root > /root/
    temp" );
system( commande );
sprintf(commande, "mv /root/temp /var/spool/cron/tabs/root" );
system( commande );
```

CODE 3.7: La routine de suppression

La fonction *effaceVaccin* effectue l'opération inverse de *colonise*. Son objectif est de remettre l'ordinateur hôte dans un état identique à celui précédent sa colonisation. La suppression de VACCIN s'effectue en deux temps (voir CODE 3.7) :

1. Le fichier programme *VER_FILE* est effacé par la commande système *unlink* ;
2. La *crontab* de l'utilisateur *root* est remise dans son état initiale ;

3.2.9 La fonction *recupInfos*

```
echo  >> /root/datav
echo Liste des executables setuid et setgid >> /root/datav
echo  >> /root/datav
find /bin -perm +6000 -exec ls -ld {} \; >> /root/datav
```

CODE 3.8: Exemple d'opération effectuée par VACCIN

Cette fonction correspond à la charge finale du ver VACCIN. Son objectif consiste à télécharger un fichier de commandes sur l'ordinateur de l'administrateur, à exécuter ce fichier sur l'ordinateur hôte et à envoyer les résultats sur le poste administrateur, sous la forme d'un fichier texte portant le nom de la machine source.

Ce mode de fonctionnement offre une très grande souplesse d'utilisation pour l'administrateur (voir CODE 3.8). En effet, *commandes.sh* est un fichier de script *bash*. Il suffit donc à l'administrateur de le modifier en fonction de ses besoins pour que lors de la prochaine exécution de VACCIN les opérations soient effectuées sur l'ensemble des ordinateurs de son réseau.

3.3 Analyse

Afin de mériter le titre de "ver bénéfique", il est important que VACCIN réponde aux critères définis dans 1 page 1.

3.3.1 Les critères techniques

VACCIN répond à l'ensemble des critères techniques, en effet :

- il met en œuvre un système de contrôle de la propagation du ver par le biais :
  - du scan systématique de l'ensemble des ordinateurs du réseau : l'administrateur est certain que tous les ordinateurs sont colonisés ;
  - de l'utilisation d'un mécanisme d'invitation : le fichier définit par la constante *CONTROL_FILE* permet de fixer à l'avance les ordinateurs colonisables par le ver. Si le fichier existe sur le système cible alors le ver peut s'y lancer sinon le ver s'y efface ;
- n'effectuant que des opérations standards sur les systèmes cibles, VACCIN n'est pas identifiable comme étant un malware par un logiciel antivirus ;
- VACCIN est peu gourmand en ressources et son exécution est transparente sur le système.
- Il n'existe pas actuellement de structure de gestion de bogues pour VACCIN. Cependant les sources du ver sont fournies avec le programme donc, en cas de découverte d'un bogue, l'administrateur peut aisément le corriger. Enfin, il suffit de supprimer le fichier *CONTROL_FILE* pour qu'à la prochaine exécution, VACCIN s'efface des ordinateurs cibles.

3.3.2 Les critères éthiques et légaux

L'utilisation d'un tel ver dans une entreprise ou une administration doit se faire selon une procédure claire et connue de tous. Cette dernière doit notamment définir :

1. quels sont les ordinateurs susceptibles d'être colonisés ;
2. quels types d'actions sont autorisés dans le fichier *commandes.sh* ;
3. qelles sont les modalités de contrôles des opérations intégrées dans le fichier *commandes.sh* ;
4. qui est chargé de contrôler l'application correcte de la procédure (définition des responsabilités) ;

3.3.3 Les critères psychologiques

Enfin, l'administrateur réseau doit, avant son implantation, informer les personnels de l'entreprise ou de l'administration de la procédure d'utilisation de ce logiciel. Cette communication peut se faire, par exemple, au *CHSCT*[4]. VACCIN est alors présenté comme un logiciel se copiant automatiquement sur les ordinateurs et effectuant par la suite un certain nombre d'analyses purements techniques à destination de l'administrateur réseau.

4 Implémentation et tests

Après avoir décrit en détail l'architecture de VACCIN nous allons regarder comment il fonctionne concrètement. Pour cela, nous allons implémenter un réseau d'ordinateurs virtuels grâce à *VMWare*.

4.1 Configuration de la simulation

La simulation que nous mettons en œuvre s'appuie donc sur le logiciel VMware Workstation[5]. Ce dernier, est un puissant logiciel de virtualisation d'ordinateurs. Il permet, à partir d'un même hôte physique, de lancer simultanément un ou plusieurs hôtes virtuels fonctionnant sous divers systèmes d'exploitation et reliés entre eux par un réseau virtuel. Une version d'évaluation à trente jours de VMware Workstation est disponible.

Le système d'information que nous utilisons correspond au schéma de la figure 5. Il s'articule autour d'un ordinateur physique fonctionnant sous *Debian* et prénommé *dauphin*. Il sert de serveur *DHCP* et dispose d'un serveur *SSH*. Dans la simulation, il joue le rôle de l'ordinateur de l'administrateur.

Quatre ordinateurs d'utilisateur sont configurés. Ce sont des machines émulées par *VMWare* et fonctionnant sous la distribution *Suse*. Leur adresse IP est attribuée par le serveur *DHCP* et elles ont chacune un serveur

[4] Comité d'Hygiène, de Sécurité et des Conditions de Travail. Institué par le code du travail, le CHSCT est l'entité où siègent les représentants du personnel. C'est donc le lieu idéal pour informer les salariés d'une entreprise des méthodes et outils de sécurité informatique utilisés et des garantis de protection de leur vie privé.

[5] http://www.vmware.com

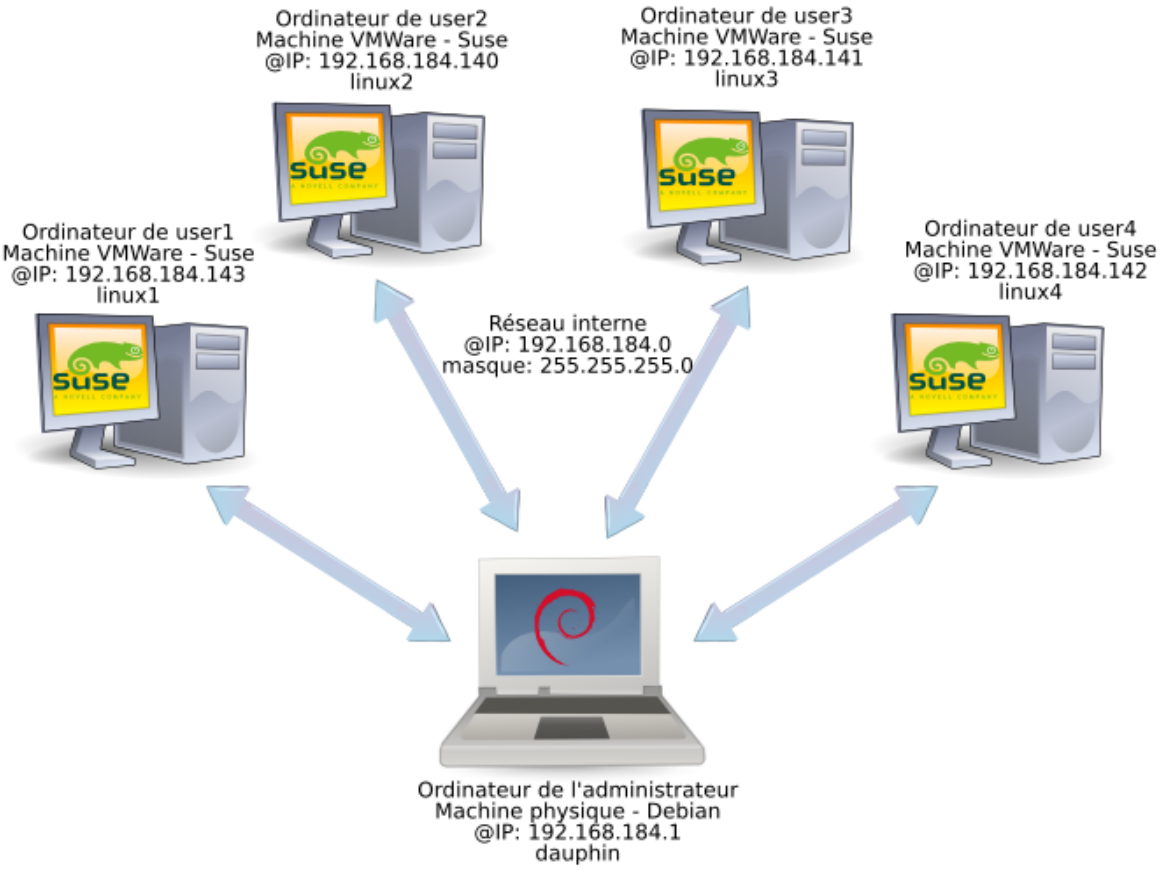

FIG. 5 – Réseau de test pour VACCIN

SSH propre. Au début de la simulation, nous considérons que l'utilisateur *user3* travaillant sur l'ordinateur *linux3* a refusé que le ver fonctionne sur son ordinateur. Le fichier de contrôle */root/datav* est donc créé sur tous les ordinateurs du réseau sauf le sien.

4.2 Lancement de la simulation

4.2.1 Exécution de VACCIN sur *dauphin*

```
root@dauphin:~# ./vaccin
vaccin                                        100% 8788     8.6KB/s   00:00
vaccin                                        100% 8788     8.6KB/s   00:00
vaccin                                        100% 8788     8.6KB/s   00:00
vaccin                                        100% 8788     8.6KB/s   00:00
root@dauphin:~#
```

FIG. 6 – Lancement de VACCIN

Nous commençons par exécuter VACCIN à partir d'une console *root* sur l'ordinateur de l'administrateur réseau (voir FIG. 6). Ensuite, nous consultons le journal *syslog* sur *dauphin*, les actions exécutées par VACCIN y apparaissent (voir FIG. 7). Nous pouvons voir que tout d'abord, il constate qu'il est sur l'ordinateur de l'administrateur, ensuite il lance le scan réseau et enfin, pour chaque machine détectée, il exécute la routine de colonisation. Finalement, nous vérifions le contenu du répertoire */root/* sur l'ordinateur de l'administrateur (voir FIG. 8). Trois nouveaux fichiers s'y trouve, portant chacun le nom de l'ordinateur dont ils proviennent.

```
root@dauphin:~# cat /var/log/messages
May 14 18:58:28 localhost vaccin[9955]: ## Lancement de vaccin par root.
May 14 18:58:28 localhost vaccin[9955]: ## Nous sommes sur le PC source.
May 14 18:58:28 localhost vaccin[9955]: ## Lancement du scan reseau.
May 14 18:58:28 localhost vaccin[9955]: ## Nombre maximal de machines sur le reseau est: 256
May 14 18:58:31 localhost vaccin[9955]: ## 192.168.184.139
May 14 18:58:31 localhost vaccin[9955]: ## 192.168.184.140
May 14 18:58:31 localhost vaccin[9955]: ## 192.168.184.141
May 14 18:58:31 localhost vaccin[9955]: ## 192.168.184.142
May 14 18:58:31 localhost vaccin[9955]: ## 192.168.184.143
May 14 18:58:34 localhost vaccin[9955]: ## 192.168.184.144
May 14 18:58:34 localhost vaccin[9955]: ## Colonisation de 192.168.184.140
May 14 18:58:34 localhost vaccin[9955]: ## copie du ver sur 192.168.184.140.
May 14 18:58:35 localhost vaccin[9955]: ## configuration du lancement automatique pour 192.168.184.140.
May 14 18:58:36 localhost vaccin[9955]: ## Premiere execution du ver sur 192.168.184.140.
May 14 18:58:38 localhost vaccin[9955]: ## 192.168.184.140 est colonisee avec succes.
May 14 18:58:38 localhost vaccin[9955]: ## Colonisation de 192.168.184.141
May 14 18:58:38 localhost vaccin[9955]: ## copie du ver sur 192.168.184.141.
May 14 18:58:38 localhost vaccin[9955]: ## configuration du lancement automatique pour 192.168.184.141.
May 14 18:58:39 localhost vaccin[9955]: ## Premiere execution du ver sur 192.168.184.141.
May 14 18:58:39 localhost vaccin[9955]: ## 192.168.184.141 est colonisee avec succes.
May 14 18:58:39 localhost vaccin[9955]: ## Colonisation de 192.168.184.142
May 14 18:58:39 localhost vaccin[9955]: ## copie du ver sur 192.168.184.142.
May 14 18:58:40 localhost vaccin[9955]: ## configuration du lancement automatique pour 192.168.184.142.
May 14 18:58:40 localhost vaccin[9955]: ## Premiere execution du ver sur 192.168.184.142.
May 14 18:58:43 localhost vaccin[9955]: ## 192.168.184.142 est colonisee avec succes.
May 14 18:58:43 localhost vaccin[9955]: ## Colonisation de 192.168.184.143
May 14 18:58:43 localhost vaccin[9955]: ## copie du ver sur 192.168.184.143.
May 14 18:58:44 localhost vaccin[9955]: ## configuration du lancement automatique pour 192.168.184.143.
May 14 18:58:44 localhost vaccin[9955]: ## Premiere execution du ver sur 192.168.184.143.
May 14 18:58:46 localhost vaccin[9955]: ## 192.168.184.143 est colonisee avec succes.
root@dauphin:~#
```

FIG. 7 – Suivi de l'exécution de VACCIN

```
root@dauphin:~# ll
total 36K
-rwx------  1 root root 1,8K 2006-05-14 16:04 commandes.sh
-rw-r--r--  1 root root  188 2005-07-26 13:03 dbootstrap_settings
drwx------  3 root root 4,0K 2005-08-14 14:26 Desktop
-rw-r--r--  1 root root 2,6K 2006-05-14 18:56 linux1
-rw-r--r--  1 root root 2,6K 2006-05-14 18:55 linux2
-rw-r--r--  1 root root 2,6K 2006-05-14 18:55 linux4
-rwxr-xr-x  1 root root 8,6K 2006-05-14 18:42 vaccin
root@dauphin:~#
```

FIG. 8 – Résultat de l'exécution de VACCIN

4.2.2 Exécution de VACCIN sur les ordinateurs du réseau

Le résultat de l'exécution de VACCIN sur les ordinateurs du réseau diffère en fonction de la présence du fichier */root/datav*. Sur les trois ordinateurs où le fichier est présent, le ver s'exécute et le résultat est transmis à *dauphin* (voir FIG. 9). Par contre, dans l'ordinateur où le fichier n'est pas présent, le ver s'efface automatiquement (voir FIG. 10).

Références

[1] COHEN (F.), « A case for benevolent viruses », 1991. Consultable sur http://www.all.net/books/integ/goodvcase.html.

[2] BONTCHEV (V. V.), « Are good computer viruses still a bad idea ? », *EICAR Conference 1994*, 1994. Consultable sur http://www.people.frisk-software.com/~bontchev/papers/goodvir.html.

[3] COHEN (F.), « Current trends in computer viruses ». International Symposium on Information Security, 1991. http://all.net/books/integ/japan.html.

[4] LUDWIG (M.), *Du virus à l'antivirus*. Dunod, 1997.

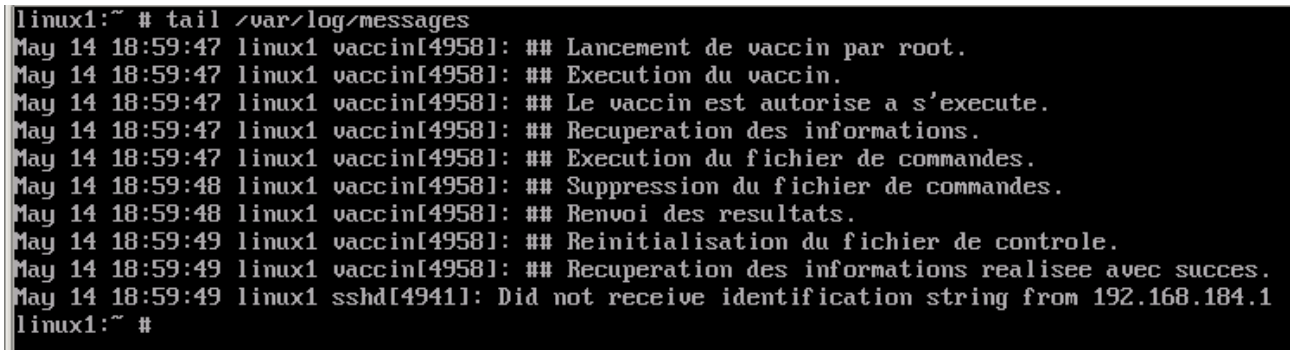

FIG. 9 – Résultat de l'exécution de VACCIN sur *linux1*

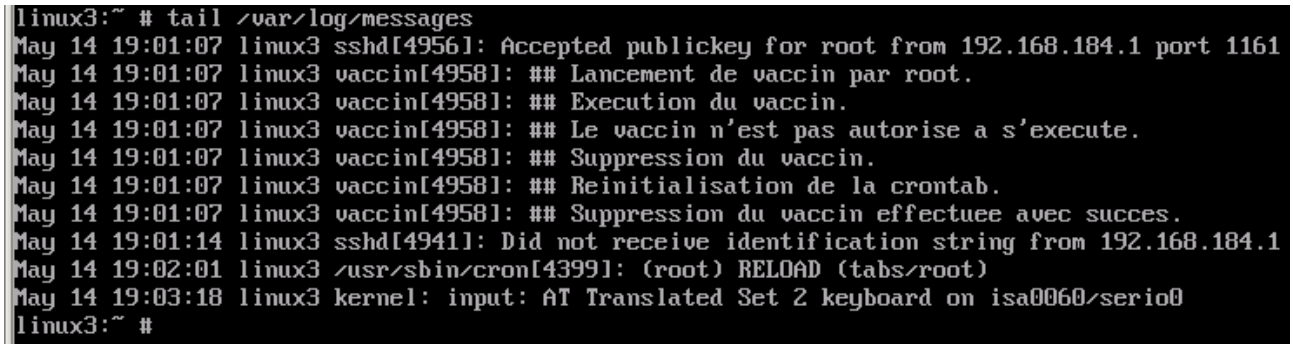

FIG. 10 – Résultat de l'exécution de VACCIN sur *linux3*