Modélisation des processus d'attaques pour l'évaluation des IDS

Mohammed Gadelrab (gad-el-rab@laas.fr)[1]
Anas Abou El Kalam (anas.abouelkalam@enseeiht.fr)[2]
Yves Deswarte (deswarte@laas.fr)[1]

Résumé: Face à l'émergence massive des programmes malveillants (maliciels) et à la multiplication des attaques et de leurs variantes sur l'Internet, des outils de détection d'intrusion (ou IDS pour *Intrusion Detection Systems*), souvent développés empiriquement, ont vu le jour. Néanmoins, ces outils, au même titre que les techniques sur lesquelles ils se basent, ont une efficacité limitée et difficile à évaluer. Pour pouvoir tester les systèmes de détection d'intrusion et les évaluer, il faut tout d'abord comprendre le comportement des attaquants. A cet égard, nous présentons dans cet article une analyse des maliciels automatiques les plus connus, ainsi que de certaines attaques interactives. Cette analyse nous a permis de développer un modèle de processus d'attaques décrit par une machine à état. Le but est de pouvoir instancier ce modèle pour générer de façon systématique et rationnelle des scénarios d'attaques, notamment pour tester et évaluer des IDS. Notre modèle peut aussi servir au test de pénétration, à la spécification de signatures basées sur les exécutions d'attaques, à la corrélation d'événements, à la simulation d'outils de sécurité ainsi qu'à des fins pédagogiques pour l'enseignement de la sécurité.

Mot clès : Attaques, maliciel, modèle, test, évaluation, génération d'attaques, IDS.

1 Introduction

Ces dernières années ont vu une augmentation massive du nombre des virus, vers et autres programmes malveillants (maliciels). Et même si les rapports se contredisent parfois, on parle d'une multiplication par cinq en 2007 par rapport à l'année précédente ! Pour ne citer qu'un exemple représentatif, le très sérieux projet *AV-Test* (15 ans d'expérience dans l'analyse des programmes malveillants) a répertorié en 2007, 5.5 millions d'échantillons de programmes malveillants, contre moins d'un million en 2006 (figure 1) [1]. Ce nombre est non cumulatif, il ne tient pas compte des programmes malveillants précédemment détectés.

Cette analyse est alarmante pour ce qui concerne la détection d'intrusions. En effet, une légère modification d'un programme malveillant connu suffit pour tromper la plupart des mécanismes de détection à base de signature. De plus, le processus de génération de signatures est généralement fastidieux, même pour des variantes de maliciels connus, alors qu'il est facile de le modifier automatiquement. Il n'est donc pas étonnant de voir de nouvelles mutations d'un cheval de Troie chaque jour, voire parfois chaque heure.

De toute façon, la croissance actuelle du nombre de maliciels ne peut que provoquer l'effondrement des performances des outils de sécurité qui ont besoin d'une signature spécifique

[1]LAAS-CNRS, Université de Toulouse, 7 av. du Colonel Roche, 31077 Toulouse cedex 4, France
[2]IRIT, ENSEEIHT, Université de Toulouse, 2 rue Camichel, 31071 Toulouse, France.

par maliciel, que ce soit un nouveau programme malveillant ou une variante d'un maliciel connu. Il est donc important de trouver une solution efficace pour ce problème, plus que jamais d'actualité. Pour aider à le résoudre, le travail présenté ici propose de modéliser les processus d'attaques sous forme de scénarios composés de séquences de primitives d'exécution. La définition de ce modèle repose sur l'analyse d'incidents de sécurité résultant aussi bien d'outils d'attaque automatiques que d'attaques interactives. Ce modèle peut alors servir à générer automatiquement des scénarios d'attaques représentatifs des attaques réelles, et ces scénarios peuvent être utilisés pour tester les outils de détection d'intrusion (*IDS*) et en évaluer l'efficacité.

La suite de cet article est organisée comme suit. La section 2 présente notre approche pour traiter ce problème. Ensuite, la section 3 analyse les attaques les plus connues, essentiellement des vers, pour en extraire des primitives. Dans la section 4, nous décrivons comment modéliser les processus d'attaques. La section 5 analyse l'applicabilité de notre modèle à d'autres classes d'attaques, notamment les attaques interactives. Enfin, nous présentons dans la section 7 d'autres champs potentiels d'application pour notre modèle.

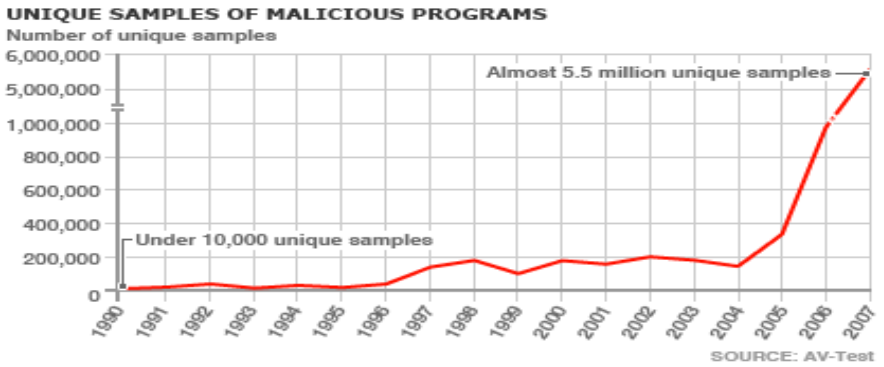

Figure 1. La croissance du nombre de maliciels ces dernières années

2 Approche

Afin de traiter cette question de la multiplication des variantes d'attaques, et contrairement à d'autres travaux qui visent à générer une signature pour chaque variante, notre approche vise plutôt à décrire les attaques par des signatures qui soient moins sensibles au polymorphisme des variantes.

Pour cela, nous avons d'abord étudié les classifications d'attaques existantes, même si elles ont pour la plupart été développées à d'autres fins [2, 3, 4, 5, 6]. Notre analyse nous permet de dire que, malheureusement, la plupart de ces travaux sont soit trop peu précis, ce qui dans notre contexte produirait un grand nombre de fausses alarmes, soit focalisés au niveau d'une méta-détection (par exemple, pour rechercher les intentions des attaquants afin d'aider les forces de l'ordre à les identifier et à analyser les dégâts qu'ils commettent), ce qui n'est pas adapté à notre étude.

Nous avons également analysé les modèles d'attaques [7, 8, 9, 10, 11] qui décrivent le comportement des utilisateurs et programmes (potentiellement malveillants) accédant et/ou s'exécutant sur un système. Mais nous avons constaté que ces modèles sont généralement

spécifiques de l'environnement d'exécution, et nécessitent donc une connaissance précise et détaillée de l'architecture, de la topologie et des vulnérabilités du réseau et du système considérés. De plus, ces modèles se basent essentiellement sur les vulnérabilités connues et ignorent les attaques susceptibles d'exploiter des vulnérabilités encore inconnues.

Dans notre contexte, ceci constituerait une limite sérieuse, dans la mesure où la robustesse des IDS dépend également des vulnérabilités inconnues et des nouvelles attaques. De plus, le modèle qui répondrait à nos attentes devrait être facilement adaptable et extensible notamment lors de l'ajout ou de la suppression d'un utilisateur ou d'une machine, lors de l'installation ou la mise à jour d'un logiciel, ou encore lors de l'application de rustines (*patches*) pour corriger ses vulnérabilités.

Pour pallier ces limites, nous proposons dans ce travail un modèle suffisamment abstrait pour couvrir un maximum de classes ou de types d'attaques, et qui soit le plus possible indépendant de l'environnement.

Pour établir un tel modèle, il s'avère nécessaire d'analyser un nombre suffisant de données sur les attaques réelles, ce qui constitue en soi un problème car les données disponibles pour la communauté scientifique sont limitées, voir parfois biaisées et non représentatives.

C'est pour cela que nous avons basé notre analyse préliminaire sur les attaques de type virus et vers les plus répandus. En effet, étant donné que les vers sont autonomes, ils doivent comporter toutes les étapes d'un processus d'attaque. De plus, les virus comme les vers peuvent être vus comme une classe d'attaques automatiques développées par des attaquants habiles; cela peut donc aider à comprendre comment les attaques interactives peuvent être menées.

Dans la section suivante, nous présentons l'analyse d'environ 40 virus, vers, chevaux de Troie ainsi que d'autres incidents liés à des attaques, dans le but d'identifier les types et séquences d'actions qu'un attaquant exécute.

3 Analyse de maliciels

Nous avons analysé les primitives d'exécution des 39 maliciels de la liste CME (*Mitre's Common Malware Enumeration list*) [12], qui sont représentatifs des attaques les plus dangereuses et les plus répandues. Nous avons également utilisé d'autres données intéressantes disponibles sur des sites spécialisés comme `http://research.eeye.com`, et `http://www.viruslist.com`.

Le premier résultat surprenant que nous avons pu constater est que, malgré la diversité de ces maliciels, les étapes suivies pour ces attaques peuvent être classées en seulement 8 primitives. Nous avons identifié chaque primitive par un symbole, comme indiqué ci-dessous :

- *R*: Reconnaissance
- *VB*: Exploration de la machine / réseau de la victime (*Victim Browsing*)
- *EP*: Exécution de programme (*Execute Program*)
- *GA*: Gain d'accès (*Gain Access*)
- *IMC*: Implantation de code malveillant (*Implant Malicious Code*)

- *CDI*: Compromission de l'intégrité (*Compromise Data Integrity*)
- *DoS*: Déni de service (*Denial of Service*)
- *HT*: Effacer les traces (*Hide Traces*)

Notons qu'au lieu d'analyser les détails des commandes et des instructions de bas niveau, nous nous sommes plutôt intéressés au processus d'attaque dans sa globalité. Cela nous permet d'identifier les primitives communes aux différents types d'attaques et être le plus indépendant possible de la plateforme ou de l'environnement.

D'ailleurs, pour ne pas biaiser l'étude et afin d'avoir des résultats plus généraux, nous prêtons peu d'attention à l'aspect propagation, caractéristique plutôt spécifique aux vers. En réalité, nous considérons l'étape de propagation comme un gain d'accès (GA) ou une implantation de code malveillants (IMC).

De plus, étant donné que plusieurs étapes d'attaques peuvent être considérées, de manière globale, comme une exécution de programme (EP), nous préférons les différencier en considérant une relation d'ordre partielle (dénoté par >) ; le but étant ainsi de déterminer (et ne considérer que) la primitive qui exprime et reflète le mieux l'étape de l'attaque en cours. Ainsi, nous considérons que :

- *IMC > CDI > EP*
- *HT > CDI > EP*
- *[R|VB] > EP*
- *HT > DoS*

Ces relations peuvent être interprétées de la manière suivante :

- Si l'étape d'attaque exécutée contribue à l'installation d'un code malveillant, elle est classifiée comme IMC. Sinon, si elle modifie le système de fichiers, les fichiers de configuration, les clefs d'enregistrement (*windows* registry), ou les variables d'environnement, on la considère plutôt comme un CDI. Autrement, on la considère comme EP.

- Si l'étape d'attaque exécutée cache des informations ou bloque l'accès aux informations qui montrent l'existence d'un code malveillant, elle est considérée comme HT. Sinon, si elle modifie le système de fichiers, les fichiers de configuration, les clefs d'enregistrement (*windows* registry), ou les variables d'environnement, sans cacher des informations liées à l'attaque, on la considère comme un CDI. Autrement, nous la considérons comme EP.

- La recherche à distance des informations reliées aux victimes potentielles est une étape de reconnaissance (R). Si l'attaquant cherche des informations localement stockées sur la victime on l'identifie comme une étape d'exploration (VB).

- Si l'étape d'attaque conduit à bloquer/arrêter/compromettre l'accès aux services qui fournissent des informations reliées aux activités malveillantes, il s'agit en fait d'une étape d'effacement des traces (HT). Sinon, si le service bloqué/arrêté/compromis ne cache pas des informations sur les activités malveillantes, nous le considérons comme une étape de déni de service (DOS).

Dans la suite de cette section, nous construirons progressivement un modèle de processus d'attaques.

3.1 CodeRed-I

Le ver *Code Red-I* commence par chercher un serveur web *IIS* vulnérable (figure 2). Ensuite, il exécute une requête "*HTTP Get*" avec un code approprié pour exploiter la vulnérabilité *ida* [13, 14]. En fait, cette vulnérabilité permet au ver de s'auto-installer en implantant un code malveillant dans la mémoire. Il s'agit donc d'un "gain d'accès" puis d'une "implantation de code malveillant", notés respectivement par *GA* et *IMC*. Le code malveillant initialise ensuite les variables du ver et localise les adresses des fonctions avec les *dll* (fichiers de bibliothèques dynamiques) chargés en mémoire. Cette étape sera donc considérée comme *EP* (exécution de programme). À l'étape suivante, le ver crée 99 fils d'exécution (*threads*) pour propager le ver en infectant d'autres serveurs Web (*EP*). Le 100ème fil vérifie la langue du système d'exploitation (*Windows*) de la machine victime (il s'agit donc d'une exploration, *VB*). Si c'est un *Windows NT*/2000 en anglais, le ver défigure (*defacing*) les pages Web du système cible (compromission de l'intégrité, *CDI*) ; si *Windows* est dans une autre langue, le ver continue à se propager (*EP*). Par la suite, chaque fil recherche le fichier `C:\notworm` (*VB*) ; s'il existe, le ver reste dormant, sinon il continue à infecter le système (*EP*). De plus, le ver vérifie la date (exploration, *VB*), si elle n'est pas entre le 1er et le 19 du mois, il attaque le site de la maison blanche `www.whitehouse.gov` (déni de service, *DoS*).

On peut donc conclure que le processus d'attaque de CodeRed-I est : *GA, IMC, EP, EP, VB, CDI, EP, VB, EP, VB, DoS*.

3.2 CodeRed-II

CodeRed-II exploite pratiquement les mêmes vulnérabilités que *CodeRed-I*, à la différence près qu'il utilise d'autres mécanismes pour propager l'infection et installer un cheval de Troie [15]. Les deux premières étapes sont similaires (*GA* et *IMC*) à celles de *CodeRed-I* (figure 3). Le ver identifie ensuite les adresses IP locales (*VB*), en déduit les masques de sous-réseau, qu'il utilisera ultérieurement pour la propagation. Le ver identifie également le système d'exploitation de la cible (*VB*). De plus, il cherche s'il s'est déjà exécuté auparavant sur la même machine (*VB*), et si c'est le cas, il commence la phase de propagation (*EP*). Ensuite, il crée 600 fils d'exécution pour les systèmes chinois et 300 pour les autres. À l'étape suivante, le ver recherche le répertoire natif du système (typiquement, `C:\winnt\system32`) (*VB*). Il copie ensuite le programme `cmd.exe` dans `inetpub/script/root.exe` et dans `program~1/common~1/msadc/root.exe` (compromission de l'intégrité, *CDI*), puis implante un binaire malveillant dans `explorer.exe` (*IMC*). Enfin, il reboote le système (*EP*).

Après le redémarrage du système infecté, le cheval de Troie `explorer.exe` opère en modifiant les clés de registre Windows pour désactiver les protections du système et pour créer des chemins Web virtuels (*CDI*). Ainsi, tant qu'`explorer.exe` s'exécute, l'attaquant peut accéder à distance au serveur infecté.

On peut donc déduire que le processus d'attaque de *CodeRed-II* est: *GA*, *IMC*, *VB*, *VB*, *VB*, EP, *VB*, *CDI*, *IMC*, *EP*, *CDI*.

Quand l'attaquant souhaite se reconnecter à un système infecté, il lui suffit d'envoyer une requête du type "`http://IpAddress/c/winnt/system32/cmd.exe?/c+dir`", où `dir` peut être remplacé par n'importe quelle commande à exécuter à distance sur la machine cible. Ceci correspond ainsi à *GA* suivi d'un *EP*.

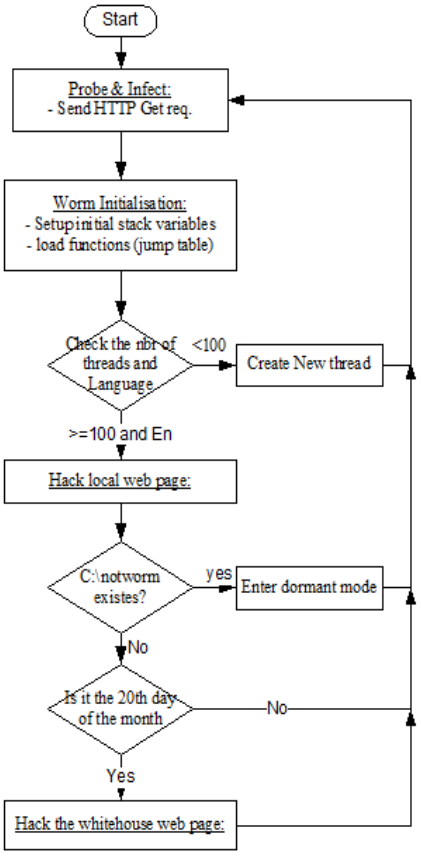

Figure 2. Processus d'attaque de CodeRed-I

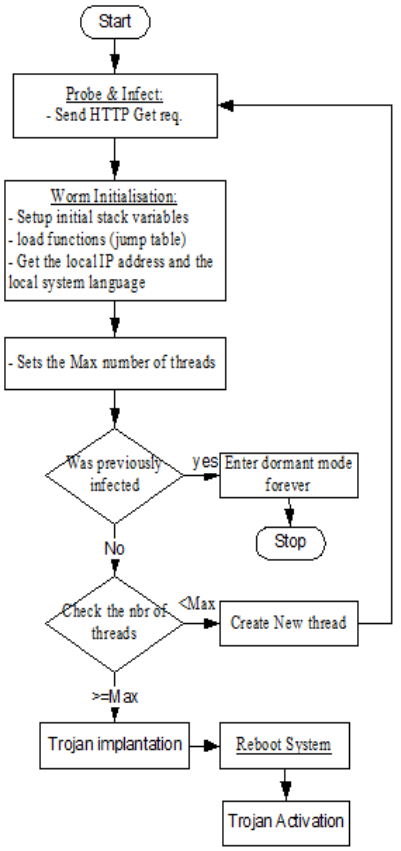

Figure 3. Processus d'attaque de CodeRed-II

3.3 Sasser

Sasser utilise un exploit public des vulnérabilités de la *LSA* (*Local Security Authority*), du service `lsa.exe`, afin d'obtenir un accès au niveau système sur la machine victime [17]. Tout d'abord, il cherche les adresses IP en écoute sur le port TCP-445 (reconnaissance, *R*). Une fois trouvées, il essaye de déterminer la version de *Windows* en analysant la bannière du service *SMB* (*R*). Il envoie ensuite l'exploit LSA qui permet d'établir une connexion `shell` sur le port TCP-9996 (*GA*). L'étape suivante est la création d'un script `ftp` (*IMC*) sur la machine cible. Quand il est exécuté, le script se connecte à la machine de l'attaquant (*EP*) et télécharge le fichier exécutable du ver (*IMC*). Enfin, le ver est exécuté pour démarrer un serveur `ftp` ainsi que 128 fils (*threads*) de propagation (*EP*) avant de supprimer le script `ftp` (*Hide Traces*, *HT*).

Le processus de l'attaque de *Sasser* est donc : *R, R*, *GA*, *IMC*, *EP*, *IMC*, *EP*, *HT*.

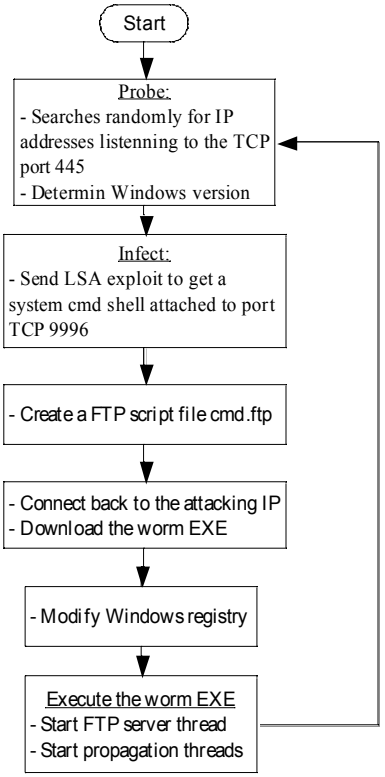

Figure 4. Processus d'attaque de Sasser

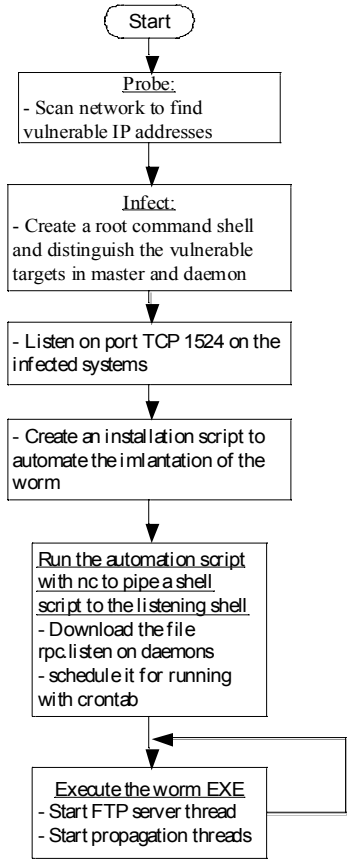

Figure 5. Processus d'attaque de Trinoo

3.4 Trinoo

Trinoo est une attaque par déni de service distribué (*DDoS*) [18] qui commence par parcourir les adresses IP pour trouver celles ayant un service *RPC* vulnérable (*R*). Ensuite, elle infecte ces machines en exploitant les vulnérabilités découvertes pour créer des commandes `shell` en mode `root` (*GA*). De plus, elle démarre (sur la machine infectée) un service (*daemon*) qui écoute sur le port TCP-1524 (*EP*). Un script est ensuite créé pour automatiser l'installation du ver (*IMC*). L'attaque utilise également `netcat` pour associer le script au `shell` et au service en écoute (*EP*); le script exécute la commande `rcp` pour télécharger le fichier `rcp.listen` et l'enregistrer dans le répertoire `/usr/sbin/` (*IMC*); enfin, elle programme l'exécution de `rcp.listen` en arrière plan avec `crontab` (*EP*).

Les machines infectées seront considérées soit comme maîtres, soit comme esclaves. En pratique, l'attaquant contrôle un ou plusieurs serveurs maîtres, et les configurent de manière à contrôler plusieurs clients (*daemons*) esclaves. Ces derniers sont orchestrés pour coordonner une attaque de type déni de service contre une ou plusieurs victimes (*DoS*). Ainsi, le scénario d'attaque de *Trinoo* (figure 5) est : *R*, *GA*, *EP*, *IMC*, *EP*, *IMC*, *EP*, *DoS*.

3.5 SQL-Slammer (Sapphire)

Ce ver exploite plusieurs vulnérabilités de débordement de mémoire (CVE_2002-0649) dans SQL server 2000 [16]. Après débordement du tampon mémoire (*buffer*), le vers prend le contrôle de la machine cible et s'y installe (*GA* puis *IMC*). Il continue par s'auto-propager par UDP sur le port 1434 à des adresses IP générées aléatoirement (*EP*), et ce par une boucle sans fin (*DoS*). Ainsi, le scénario d'attaque de SQL-Slammer est : *GA*, *IMC*, *EP*, *DoS*.

4 Modèle de processus d'attaques

En plus des vers décrits dans la section précédente, nous avons analysé 35 autres maliciels du *CME* (*Common Malware Enumeration*) du *MITRE*; néanmoins, faute d'espace, nous ne pouvons les décrire tous dans cet article. Les détails sont disponibles dans [20].

Cette analyse nous a toutefois permis de construire le modèle décrit par la figure 6. Le premier résultat surprenant que nous pouvons souligner est que, quelque soit la nature de l'attaque et l'expérience des attaquants, ces derniers suivent généralement le même type de processus (étapes) ! Le niveau de l'attaquant se reflète à travers la sophistication et la finesse de l'attaque, la qualité du code, les effets et les dommages causés, etc. En général, les attaquants expérimentés utilisent des techniques plus avancées et des outils plus personnalisés, alors que les débutants (*script kiddies*) utilisent des exploits et des outils souvent développés par d'autres.

Notre modèle distingue les étapes suivantes (figure 6) :

1. *Reconnaissance* : il est logique pour un attaquant de chercher les informations nécessaires sur les victimes potentielles avant de les cibler avec les outils d'attaques les plus appropriés (codes d'exploits, *toolkits*, etc.).

2. *Gain d'accès* (*Gain Access*) : afin d'atteindre leurs objectifs, les attaquants ont généralement besoin d'avoir un accès aux ressources des victimes ; le niveau d'accès requis dépend évidemment de l'attaque. Notons toutefois que certains types d'attaques, comme les attaques en déni de service, n'ont pas besoin d'accès sur la machine victime.

3. *Augmentation de privilèges* (*Privilege Escalation*) : l'accès obtenu initialement par l'attaquant est parfois insuffisant pour réaliser l'attaque ; dans ce cas, l'attaquant essaie d'augmenter ses privilèges pour avoir plus de pouvoir (par exemple, passer du mode utilisateur au mode administrateur pour pouvoir accéder aux ressources systèmes).

4. *Exploration de la machine victime* (*Victim browsing*) : après avoir acquis suffisamment de privilèges, l'attaquant essaie généralement d'explorer la machine ou le réseau cible (par exemple, en fouillant les fichiers et les répertoires), pour rechercher un compte particulier (comme un compte invité ou un compte `ftp` anonyme), pour identifier les composants

matériels, pour identifier les programmes installés, pour rechercher les hôtes de confiance (typiquement, ceux ayant des certificats installés sur la machine de la victime), etc..

5. *Actions principales* (*Principal Actions*) : comme indiqué dans la figure 6, cette étape peut prendre différentes formes ; par exemple, l'attaquant peut exécuter une attaque en déni de service, installer un code malveillant, compromettre l'intégrité des données ou exécuter un programme.

6. *Cacher les traces* (*Hiding Traces*) : les attaquants les plus expérimentés utilisent généralement cette dernière étape pour effacer leurs traces et rendre ainsi la détection plus difficile.

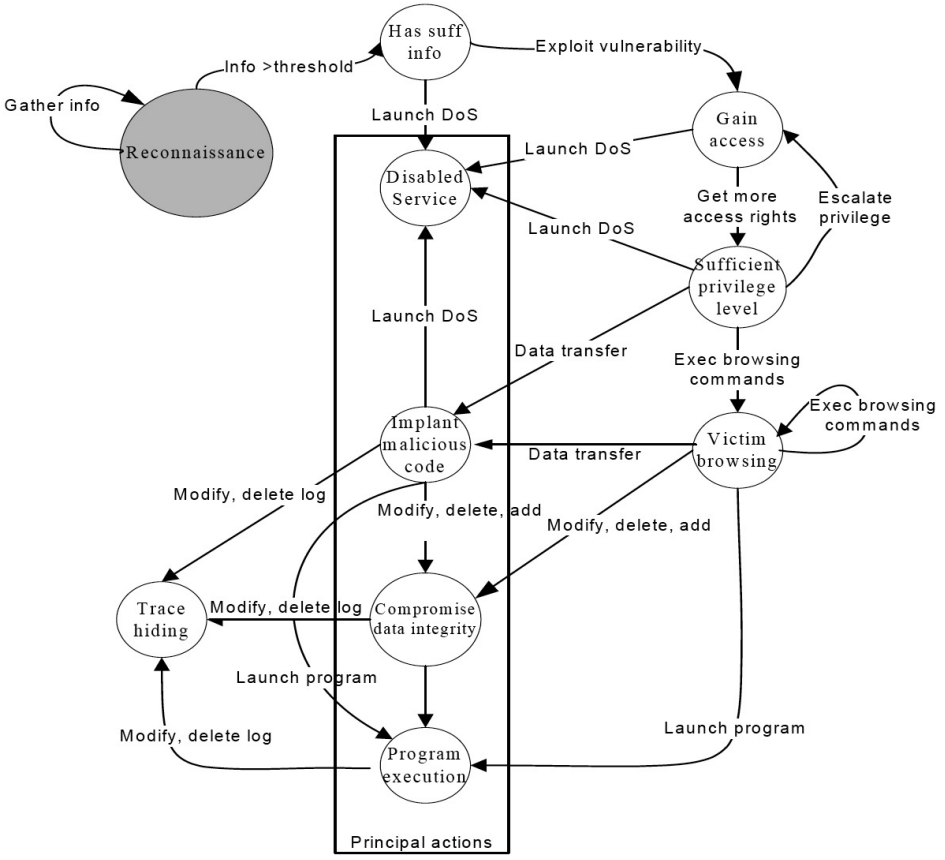

Figure 6. Modèle de processus d'attaque

Il est important de noter que, du point de vue de la détection d'intrusion, le nombre d'étapes qui apparaît dans une certaine session d'un processus d'attaque est arbitraire. En effet, afin d'empêcher la détection, les attaquants peuvent procéder lentement, en plusieurs étapes, sur plusieurs jours, voir même sur plusieurs semaines. Ainsi, quand ils reprennent leur attaque avec les étapes qui suivent, cela pourrait apparaître comme une nouvelle attaque pour l'outil de détection d'intrusion (*IDS*) ; le plus souvent, l'attaquant recommence directement par une

augmentation de privilège ou par exécuter des actions intrusives sans reproduire les étapes précédentes (telles que reconnaissance, exploration, etc.). Par ailleurs, dans bien des cas, l'attaquant peut être un utilisateur interne qui possède un compte valide et qui a déjà suffisamment d'information et de privilèges ; il n'a donc pas besoin de passer par certaines étapes comme la reconnaissance. D'un autre côté, un processus d'attaque peut être interrompu délibérément par une simple décision de l'attaquant, par exemple s'il estime qu'il est difficile de réussir son attaque ou s'il a trouvé une autre cible plus facile ou plus intéressante.

Notons également que même si notre modèle ne tient pas compte directement des attaques dites multi-sauts (*multi-hop attacks*) qui passent par plusieurs victimes, il est tout à fait possible de représenter ces attaques par une concaténation de plusieurs scénarios (c'est-à-dire avec un scénario pour chaque victime).

Dans notre contexte, l'évaluation des IDS, notre modèle de processus d'attaque (figure 6) permet de générer des scénarios d'attaques à un niveau abstrait. Néanmoins, il est clair que cette vue "de haut niveau" n'est pas suffisante pour générer des traces réelles d'attaques. C'est pourquoi nous l'utilisons conjointement avec deux autres modèles [20]: un modèle pour le comportement des attaquants et un autre qui classifie les outils d'attaques (en particulier, nous avons répertorié les outils et commandes qui permettent de réaliser chacune des primitives de la figure 6). L'ensemble de ces trois modèles nous permet d'instancier les scénarios abstraits, et donc de générer des scénarios exécutables réels.

En particulier, la transformation est faite en associant les étapes abstraites avec des commandes concrètes ou des instances d'exécution d'outils qui réalisent et implémentent ces étapes. Par exemple,

- ***R*** est associé avec (c'est-à-dire peut être exécuté de manière concrète par) : nessus, nmap, ping, traceroute, etc.
- ***VB*** peut être exécuté de manière concrète par : ls, ps, uname, etc.
- ***AG*** peut être associé avec : SSH, telnet, exécuter / exploiter une vulnérabilité metasploit, etc.
- ***CDI*** peut être associé avec : cp, rm, mv, éditer un fichier de configuration, changer les variables d'environment, etc.
- ***EP*** peut être associé avec : crontab, lynx, nc, etc.
- ***TH*** peut être associé avec : rm log, kill syslog, tuer un processus antivrirus, etc.
- ***DoS*** peut être associé avec : shutdown -halt, crash system, arrêt d'un service (stop service), etc.
- ***IMC*** peut être associé avec : scp malveillant, ftp malveillant, exécution de metasploit avec une charge utile malveillante, etc.

Il est important de noter que cette approche itérative de génération des scénarios d'attaques nous a permis de pallier le problème de l'explosion combinatoire, problème intrinsèque aux approches classiques de génération des scénarios d'attaques. En effet, dans notre

modèle, le nombre de cas possibles est fortement réduit, grâce notamment à des contraintes sur les boucles et les relations de précédence (déduites à partir de notre graphe) [20].

5 Autres classes d'attaques

Se pose maintenant la question de la validité de notre modèle pour d'autres scénarios d'attaques que les vers. Pour répondre à cette question, nous avons analysé d'autre types d'attaques, et nous présentons dans cette section un échantillon de scénarios d'attaques génériques.

5.1 SYN flooding

L'attaque dite d'inondation SYN (*SYN flooding*) est une forme de déni de service dans laquelle un attaquant envoie une multitude de requêtes TCP SYN à une machine cible, et n'accuse jamais réception des réponses SYN-ACK renvoyées par la cible. La multiplication des connexions semi-ouvertes du côté de la cible consomme des ressources de la victime qui ne sont jamais libérées (sur les machines vulnérables). Selon notre modèle, le scénario d'attaque est tout simplement *DoS*.

5.2 Mitnick

L'attaque de Mitnick est similaire au vol de session TCP (*session hijacking*). Elle procède en deux attaques parallèles contre deux hôtes différents *A* et *B* (les deux parties d'une session TCP en cours), qui se font mutuellement confiance. L'attaquant lance une attaque *SYN flooding* contre *A* (*DoS*), afin de l'empêcher de répondre à *B*. Ensuite, il envoie plusieurs paquets TCP à *B* (la cible) et essaie de prédire le numéro de séquence TCP généré par *B* (*R*). L'attaquant poursuit son attaque en usurpant l'adresse IP de *A* (*IP spoofing*) par l'envoi d'un paquet SYN à B afin d'établir une session TCP entre *A* et *B* (*GA*). *B* répond en envoyant des SYN/ACK à *A*, qui ne répond naturellement pas à ces paquets car sa file d'attente est pleine à cause du déni de service (*SYN Flooding* en cours). L'attaquant, quant à lui, peut répondre aux SYN/ACK de *B* avec le bon numéro de séquence et établir ainsi une connexion TCP avec *B*. Par conséquent, il peut envoyer des données et exécuter des commandes dans le cadre de la connexion établie (*EP*). Ainsi, le scénario d'attaque contre *A* est tout simplement *DoS*, alors que contre *B* le scénario correspond à : *R*, *GA*, *EP*.

5.3 Cross Site Scripting (XSS)

Les attaques de type *Cross-Site Scripting* (notée parfois XSS ou CSS) sont des attaques visant les sites Web affichant des pages dont le contenu peut-être fourni par d'autres utilisateurs (typiquement, des commentaires dans des blogs), sans effectuer de contrôle sur ce contenu. Si ce contenu est malveillant (par exemple, un code HTML ou un script), un utilisateur qui pense lire la page Web de quelqu'un en qui il a confiance, exécutera un code injecté par quelqu'un d'autre.

Prenons un exemple simple : quand un utilisateur (victime) est connecté à son compte bancaire via Internet, un attaquant peut lui envoyer un courriel avec un lien séduisant (*GA*). En réalité, ce lien pointe sur (ou contient) un script malveillant (*IMC*). Quand la victime clique sur ce lien, le script ciblé par (ou enveloppé dans) le lien exécute une action arbitraire avec les

privilèges de l'utilisateur authentifié (*EP*). Les actions (de l'attaquant) sont acceptées en se basant sur les cookies de la session de la victime. Le scénario de l'attaque correspond donc à : *GA*, *IMC*, *EP*.

5.4 Phishing

Le *phishing* est une attaque qui consiste à envoyer un courriel sous la forme et l'image d'une entité de confiance (par exemple, votre banque avec son logo, votre nom, celui de l'expéditeur, etc.) en demandant de manière subtile à la victime ses informations personnelles (par exemple, bancaires). Après cette étape, qui peut être qualifiée d'ingénierie sociale (*social engineering*), l'attaquant peut poursuivre en rassemblant le maximum de garanties (*credentials*) et de privilèges de la victime (*R*); il va ensuite les utiliser pour obtenir des droits (*GA*) et pour effectuer des actions malveillantes à l'insu de la victime (*Victim Browsing, Compromise Data Integrity, Execute Program, etc*.).

Après cette brève présentation de notre analyse des malveillances les plus connues, on peut conclure que les attaques interactives, au même titre que les attaques par des maliciels automatiques (par exemple, les vers), suivent pratiquement le même processus d'attaque et obéissent ainsi à notre modèle. Cela dit, il existe certainement des différences : les vers opèrent souvent au niveau des appels systèmes ou de l'API, alors que les autres opèrent plus fréquemment au niveau applicatif. Mais ces différences n'ont pas une grande importance dans notre contexte, à savoir le test des IDS.

6 Utilisations potentielles de notre modèle

Quand nous avons conçu ce modèle, notre but initial était la génération de scénarios d'attaques dans un cadre global de test des IDS. Toutefois, il est clair que ce modèle peut parfaitement servir à d'autres usages, comme la génération automatique de scénarios d'attaques pour évaluer d'autres types d'outils de sécurité, ou pour aider à la définition de signatures basée sur l'exécution de processus d'attaque, ou pour des tests de pénétration, ou encore pour simuler un environnement d'exécution d'un système ou d'un réseau avant son entrée en opération. Si on prend les tests de pénétration comme exemple, ce domaine manque réellement d'outils automatiques pour la dérivation et la génération de scénarios d'attaques et moyennant quelques modifications mineures, notre modèle peut être utilisé aussi dans ce domaine. De même, il n'y a à ce jour, aucun outil disponible pour évaluer la sécurité dans les réseaux simulés et la plupart des techniques de simulation existantes ne visent pas essentiellement la sécurité, elles s'intéressent plutôt à d'autres problèmes liés aux performances des réseaux. Enfin, notre modèle peut également être utilisé dans des buts pédagogiques, notamment pour la formation des spécialistes de sécurité : ces derniers peuvent ainsi pratiquer la sécurité sans risquer d'endommager les réseaux et les systèmes opérationnels.

7 Conclusion

Afin de se prémunir contre les attaques, il est naturellement important de cerner le comportement des attaquants. Ceci est également essentiel pour tester les outils de sécurité, et notamment en générant du trafic malveillant basé sur des processus d'attaques bien définis et reflétant le plus possible la réalité. En apportant une contribution à ce domaine complexe et vaste, cet article analyse les primitives d'exécution d'un échantillon représentatif de maliciels

automatiques (les vers) et d'attaques interactives. Nous avons observé que dans la majorité des cas, les attaquants suivent pratiquement les mêmes types d'étapes, et nous avons ainsi construit un méta-modèle (ou modèle de haut niveau) pour décrire les processus d'attaques à un niveau d'abstraction satisfaisant.

Notre modèle peut être vu comme une machine à état où les transitions représentent les actions exécutées par les attaquants, et les états représentent les étapes du processus d'attaque.

Initialement, nous avons instancié notre modèle pour tester les IDS, mais il peut aussi être utile à d'autres fins comme la génération automatique de scénarios d'attaques pour évaluer d'autres outils de sécurité, la définition des signatures à base d'exécution, les tests de pénétration, la simulation de la sécurité, etc.

Actuellement, nous sommes en train de développer un outil qui automatise la génération des attaques dans le contexte décrit précédemment.

Remerciement

Cette étude est menée dans le cadre du réseau d'excellence européen ReSIST (*Resilience for Survivability in IST*) du FP7 (http://www2.laas.fr/RESIST/index.htm) et du projet Airbus ADCN+ (*Aircraft Data Communication Networks*).

Références

[1] AV-Test of Anti-Virus- and Security-Software, http://www.av-test.org

[2] M. Kjaerland, "A classification of computer security incidents based on reported attack data," *Journal of Investigative Psychology and Offender Profiling*, vol. 2, 2005, pp. 105-120.

[3] D. Alessandri, "Attack-Class-Based Analysis of Intrusion Detection Systems," 2004, http://homepage.hispeed.ch/alessandri/Alessandri-Thesis.pdf

[4] K. Killourhy, R. Maxion, et K. Tan, "A defense-centric taxonomy based on attack manifestations," *Dependable Systems and Networks, 2004 International Conference on*, 2004, pp. 102-111.

[5] J.D. Howard, "An analysis of security incidents on the Internet 1989-1995," 1998, http://portal.acm.org/citation.cfm?id=287127

[6] U. Lindqvist et E. Jonsson, "How to systematically classify computer security intrusions," *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, 1997, pp. 154-163.

[7] M. Dacier et Y. Deswarte, "Privilege Graph: an Extension to the Typed Access Matrix Model," *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS '94)*, London, UK: Springer-Verlag, 1994, pp. 319-334.

[8] B. Schneier, "Attack Trees: Modeling Security Threats," *Dr. Dobb's Journal*, Vol. 24, Number 12, déc. 1999, pp. 21-29.

[9] J.S. Templeton et K. Levitt, "A requires/provides model for computer attacks," *Proceedings of the 2000 workshop on New security paradigms (NSPW '00)*, New York, USA: ACM, 2000, pp. 31-38.

[10] O.M. Dahl et S.D. Wolthusen, "Modeling and Execution of Complex Attack Scenarios using Interval Timed Colored Petri Nets," *Proceedings of the Fourth IEEE International Workshop on Information Assurance (IWIA '06)*, Washington, DC, USA: IEEE Computer Society, 2006, pp. 157-168.

[11] U.L. S. Cheung et M. Fong, “Modeling Multistep Cyber Attacks for Scenario Recognition,” *Proceedings of the Third DARPA Inofrmation Survivability Conference and Exposition (DISCEX III)*, Washington, DC, USA, 2003, pp. 284-292; http://citeseer.ist.psu.edu/cheung03modeling.html
[12] *Mitre's Common Malware Enumeration (2008)*, 2008, http://cme.mitre.org/
[13] The ida vulnerability (2007), http://research.eeye.com/html/advisories/published/AD20010618.html
[14] Analysis of CodeRed-I (2007), http://research.eeye.com/html/advisories/published/AL20010717.html
[15] Analysis of CodeRed-II (2007), http://research.eeye.com/html/advisories/published/AL20010804.html
[16] Analysis of Microsoft SQL Server Sapphire Worm (2007), http://research.eeye.com/html/advisories/published/AL20030124.html
[17] Analysis of Sasser worm (2007), http://research.eeye.com/html/advisories/published/AD20040501.html
[18] Trinoo Analysis (2007), http://staff.washington.edu/dittrich/misc/trinoo.analysis
[19] John McHugh, "Testing Intrusion Detection Systems: a Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory", *ACM Transaction on Information System Security*, Vol. 3, pages 262-294, 2000.
[20] “A model-driven approach for attack-scenario generation”, Rapport LAAS N°08198, avril 2008, 26pp.