Ecole doctorale : IAEM 0077
Unité et équipe : Loria UMR 7503, département 2 Méthode Formelle.
Nom de l'encadrant : Jean-Yves Marion, Professeur Université de Lorraine et Guillaume Bonfante, maître de conférences habilités
Courriel : Jean-Yves.Marion@loria.fr, Guillaume.Bonfante@loria.fr

Titre de la thèse

Reconstruction du graphe de contrôle de flot des programmes malveillants par des outils combinant analyse statique et dynamique.

Problématique Générale.

Résumé

Ce sujet de thèse a pour objet l'étude, l'analyse et la détection de codes malveillants, encore appelés malwares, comme les virus, les vers, les botnets, etc

L'EPC Carte du centre Nancy Grand-Est et du Loria UMR 7503, dans le cadre du Laboratoire de Haute Sécurité (LHS), a développé une méthode originale, dite par ***analyse morphologique***. Cette méthode permet de détecter des similarités dans des codes. Il est ainsi possible de détecter une fonctionnalité particulière dans un code ou de détecter un code malveillant. Nous avons développé un prototype qui a permis de valider, dans un premier temps, notre méthode. Pour aller plus loin et confirmer nos résultats en vue d'applications réalistes, il nous faut maintenant revoir notre approche vers trois directions :

- La première direction consiste à obtenir, de manière automatique, un ensemble de traces d'exécution sans interagir avec le monde extérieur.
- La deuxième direction consiste à retrouver le code source, à partir des traces obtenues. Plus précisément, il s'agit de développer un désassembleur (x86) capable de reconstruire le graphe de contrôle de flot d'un code malveillant.
- La troisième direction est d'adapter nos outils d'analyse à la masse de codes malveillants produits. On parle de plusieurs millions de fichiers à analyser régulièrement.

Le contexte

Les compagnies d'Anti-Virus sont assez discrètes sur les méthodes de détection employées. Ceci dit, la technique classique de détection de code malveillant est d'attribuer à chaque code malveillant une signature qui caractérise le malware en question. Une signature est une expression régulière, souvent réduite à une simple suite d'octets, qui identifie un malware. Tout fichier/binaire ou code exécuté en mémoire contenant cette signature est alors considéré infecté. Etant donné une base de données de signatures, le moteur de détection est alors la partie du logiciel chargée de rechercher une de ces signatures à l'intérieur des fichiers et des programmes. L'avantage de cette approche est sa rapidité et le faible taux de faux-positif. Les inconvénients tiennent essentiellement dans le fait que cette technique n'est pas capable de détecter des variantes ou des mutations d'un code malveillant connu, et donc elle est a fortiori incapable d'identifier une nouvelle attaque. A ce défaut originel, il s'ajoute dorénavant un défaut supplémentaire majeur. Aujourd'hui, *l'industrie du malware* est bien organisée et elle est capable de générer massivement des variantes de codes malveillants par

différents procédés d'obfuscation qui échappent aux protections classiques. Ainsi, il faut faire face à des dizaines de millions d'échantillons à analyser régulièrement. Il est clair qu'une analyse manuelle est devenue impossible et qu'il est nécessaire d'avoir des outils automatiques d'analyse et de classification. La première difficulté importante est donc d'avoir des outils suffisamment robustes et efficaces pour traiter de large corpus de programmes à analyser.

Les codes malveillants se protègent pour deux raisons. D'une part, il s'agit de rendre leur analyse difficile pour ne pas divulguer d'informations sur les auteurs et leurs objectifs. D'autre part, il s'agit d'éviter d'être détecté. Il y a de nombreuses méthodes d'obfuscation dont un grand nombre est présenté dans le livre de Collberg et Nagra. Certaines méthodes consistent à modifier le graphe de controle de flot du programme de départ. Pour cela, on peut insérer des prédicats opaques ou encore « écraser » le graphe de contrôle de flot en une seule boucle (flattening). D'autres méthodes ont pour objectif de ne pas rendre disponible le code exécuté ou seulement partiellement. La seconde difficulté fondamentale est de pouvoir percer ces protections et donc être capable de dé-obfusquer un code malveilllant.

Porgramme de la thèse

Notre approche

Nous concentrons, par faute de force, sur l'analyse des codes binaires x86. Une des raisons est que les codes sources sont rarement disponibles. Les outils d'analyse du code binaire ne sont pas aussi développés que pour les langages de plus haut niveau. Les analyses des codes binaires sont faites le plus souvent à la main avec typiquement IDA, un débugger, ou encore en instrumentant le code dans une machine virtuelle.

Notre travaille s'appuie sur le laboratoire de haute sécurité (LHS). Le LHS nous fournit une base de données importantes de codes malveillants. Nous disposons actuellement de 6 millions de codes malveillants.

Dans ce contexte, Guillaume Bonfante et Jean-Yves Marion ont développé une méthode que nous appelons *analyse morphologique* pour identifier les codes malveillants. Notre approche est d'extraire une sémantique de plus haut niveau d'un code obfusqué x86. L'innovation majeure de notre procédé est d'employer une autre notion de signature qui repose sur les caractéristiques intrinsèques d'un code malveillant. Pour faire simple, il s'agit de prendre en compte la structure d'un programme et de l'annoter par des fonctionnalités comme les types d'instructions exécutées ou les types d'appels systèmes. Une signature est alors un graphe qui synthétise les informations d'un code malveillant. Ensuite et exactement comme dans les anti-virus actuels, étant donnée une base de signatures, le moteur de détection recherche une signature à l'intérieur d'un programme. Le principal avantage de notre approche tient dans la notion de signature. En effet, premièrement une signature peut être calculée automatiquement et donc la mise à jour des bases de signatures devient automatique. Deuxièmement, une signature n'identifie pas seulement un code

malveillant mais une famille de codes partageant les mêmes caractéristiques.

Notre prototype de moteur de détection a fait l'objet d'un dépôt APP[1]. Depuis ce dépôt, il n'a cessé d'évoluer. Dans la version actuelle, les fonctionnalités sont les suivantes. Les programmes cibles sont les codes binaires x86. L'extraction des signatures est soit statique soit dynamique. En mode statique, l'extraction consiste à désassembler un code binaire et à reconstruire un graphe de flot annoté. En mode dynamique, le code binaire est exécuté dans un environnement virtualisé et surveillé dans le LHS pour en récupérer une trace d'exécution. Ensuite, nous reconstruisons un graphe de flot partiel. Dans les deux cas de figure, le graphe-signature obtenu est ensuite réduit et découpé pour former des petites signatures arborescentes comme un ensemble de pièces d'un puzzle, dont la recombinaison partielle permet de retrouver des similarités entre le code analysé et un malware donné. Le code actuel est écrit en langage C, il fait environ 15 000 lignes, et il est déposé sur la forge INRIA. Les bibliothèques que nous employons sont libc, pthreads pour la gestion des processus, Xed qui (Intel) nous sert au désassemblage du code et à l'instrumentation (virtualbox+Pin). Le code du prototype est sous la direction de notre ingénieur Fabrice Sabatier qui aidera au projet de recherche de doctorat.

L'analyse morphologique a été validée de manière qualitative comme en témoigne ces exemples :

- A partir de la signature de Stuxnet, nous sommes capables de détecter Duqu, ce qu'aucun anti-virus actuel n'est capable (à notre connaissance) en surveillant directement les processus en mémoire.
- Nous sommes capables de retrouver le protocole du Botnet Zeus
- Nous sommes capables de retrouver les algorithmes de chiffrement utilisés par le botnet Waledac.
- Nous avons été capable de détecter automatiquement la présence du plugin Qwerty dans Regin, ce qui confirme l'analyse manuelle de Kaspersky.
- Dans chaque cas, nous sommes capables de resynchroniser les parties de codes communs détectées et ainsi nous sommes capables de vérifier l'exactitude de notre procédure.

Descriptions des objectifs de la thèse et retombées

Globalement l'objectif est de reconstruire le code source d'un code obfusqué comme un malware pour pouvoir extraire des informations et définir des signatures dans un contexte d'une masse gigantesque de codes à traiter. Cet objectif est s'articule en trois points :

1. **Récupération traces d'exécution sans interaction avec le monde extérieur.** L'analyse de codes binaires obfusqués est extrêmement difficiles. Comme cela a été mentionné, les codes malveillants sont auto-modifiants (plus de 90%) et une exécution se conçoit comme une suite de vagues (non auto-modifiantes) de code. Chaque vague étant générée par la précédente, ce qui rend pratiquement inaccessible le code source du programme analysé. Une analyse dynamique ne permet que de trouver une trace qui passe au travers des vagues exécutés. Pour aller plus loin, il nous faut combiner une analyse statique et une analyse

[1] MMDEX,2009,IDDN.FR.001.300033.000.R.P.2009.000.10000.

dynamique pour retrouver le programme d'origine et certaines données clés cachées dans les différentes vagues. A ce jour, les codes auto-modifiants sont analysés dynamiquement. Une analyse dynamique consiste à instrumenter un code. Dans le cadre des codes malveillants, nous exécutons les codes au LHS dans un environnement surveillé afin de collecter une trace d'exécution (VirtualBox+PIN). Or une trace est une information partielle et elle n'est pas toujours suffisante pour avoir les informations nécessaires pour reconstruire une signature correcte qui prend en compte tous les comportements (exécutions). De ce fait, nous pouvons passer à côté d'un comportement malveillant ou d'une fonctionnalité interdite par une politique de sécurité.
Nous devons faire des avancées pour pouvoir obtenir un ensemble de traces de manière dynamique qui caractérise le comportement à bas niveau d'un programme auto-modifiant (ou d'un code malveillant). Pour cela, l'objectif est de construire une approximation du graphe de l'ensemble de toutes les vagues possibles pour toutes les exécutions. D'autre part, nous voulons développer algorithme de récupérations des vagues qui n'interagisse pas avec son environnement pour deux raisons : (i) pour rester discret car une connexion réseau dans le cadre d'une étude peut nous rendre suspect au botmaster et (ii) nous voulons une procédure automatique capable d'analyser un grande nombre d'échantillons. Notre direction est d'utiliser une combinaison d'analyse statique et de techniques de test, ce que nous avons partiellement exploré dans XXX.

2. **Désassemblage et reconstruction du graphe de flot.** Dans le cas des codes binaires x86, il est nécessaire d'avoir un désassembleur pour codes malveillants. En effet, les désassembleurs actuels sont incapables de reconstruire le code d'un programme auto-modifiant. Si l'objectif (1) nous permettra d'avoir un ensemble de vagues dont le code est non auto-modifiants et qui collectivement forme le code de départ, le problème du désassemblage et de la reconstruction du graphe de flot d'un code non auto-modifiant reste un verrou. Sans informations du compilateur, les désassembleurs comme IDA pro font des erreurs irréparables dès que il y a des sauts indirects ou dés que le code se chevauche comme dans le petit exemple ci-dessous extrait du packer tElock :

```
01006e7a   fe 04 0b    inc byte [ebx+ecx]
01006e7d   eb ff       jmp +1
01006e7e   ff c9       dec ecx
01006e80   7f e6       jg 01006e68
01006e82   8b c1       mov eax, ecx
```

L'objectif est de développer un désassembleur robuste qui produise également le graphe de contrôle de flot d'un code malveillant fortement obfusqué. Une première architecture a été décrite dans XXX qui combine analyse dynamique et statique, mais il faut aller plus loin. Ainsi, une difficulté importante, et non résolue, est l'approximation des valeurs atteignables du code afin de déterminer les valeurs des registres. Pour cela, nous nous orientons vers une combinaisons d'analyse dynamique, d'évaluation symbolique et d'analyse de valeurs (*value analysis, weakest preconditions calculus,* …).

3. **Analyse de grande masse de codes.** Les outils que nous développons, ont pour vocation à faire une analyse automatique. Cette analyse est absolument nécessaire pour traiter les dizaines de millions d'échantillons reçus régulièrement. Est-ce que nos outils sont susceptibles de traiter ces masses de données ? Il y a une nécessité de passer à une échelle que nous n'avons pas explorée. Or le LHS fournit le cadre expérimental pour le faire. L'objectif est de transformer nos outils de détection et d'analyse en outils de classification d'une quantité volumineuse de codes. Ainsi, nous serons en mesure de classer ces codes en familles. Un code inconnu pourra par la suite être rattaché à une famille, ce qui facilitera sa compréhension et sa détection. Cette partie est exploratoire pour nous. Nous comptons nous appuyer sur les connaissances en fouille de données et en traitement de grande masse de données du Loria.

Les objectifs 1 et 2 sont liés. En générale, une vague est un dump mémoire qui contient, le particulier, le code exécuté (une sous-trace de la trace d'exécution). Dans ce dump mémoire, il faut également retrouver les parties de codes qui sont restées « silencieuses » en désassemblant (objectif 2). Ensuite, cette information peut guider à la recherche de nouvelles traces (objectif 1).
L'objectif 3 est relativement indépendant mais il implique que nos résultats aux deux premiers objectifs passent à l'échelle.

Retombées de la thèse

Cette thèse permettra de renforcer notre avance en définissant des signatures plus précises et permettra également de rendre nos outils d'analyse plus robustes. Plus globalement, notre objectif à court terme est de faire une joint-venture avec la société Tracip, leader européen en forensics, pour valoriser nos outils. Cette thèse contribue à ce projet.

Identification des rôles et organisation de la thèse

Ce doctorat sera sous la responsabilité de Jean-Yves Marion (Taux d'encadrement en février 2015: 150%) et de Guilaume Bonfante (Taux d'encadrement en février 2015: 0%).

Ce projet fait partie d'un groupe qui est constitué d'un maitre de conférences habilités Guillaume Bonfante, de deux doctorants Hubert Godfroy et de Robin David, et de deux ingénieurs Fabrice Sabatier et Nicolas Shermann (en collaboration avec la PME tracip).

Hubert Godfroy est financé par une bourse de thèse DGA obtenu en 2013. Hubert Godfroy travaille sur la sémantique des programmes auto-modifiants, ce qui est sujet théorique. A la différence, ce sujet est beaucoup plus pratique avec une forte composante de développement nécessaire à la validation des résultats.

Robin David est financé par le CEA depuis 2013, et il est co-encadré avec Sébastien Bardin. Robin David travaille sur des outils d'analyse statique pour déterminer certaines valeurs dans un code binaire.

Ce groupe se réunit toutes les semaines pour faire le point sur les travaux, discuter et préparer les étapes suivantes. A côté des réunions hebdomadaires de ce groupe, le

doctorant aura des réunions hebdomadaires et individuelles avec Jean-Yves Marion et Guillaume Bonfante, avec des points d'étapes tous les trimestres. Un reporting hebdomadaire est demandé.

Ce groupe fait partie du Laboratoire de haute sécurité (LHS). Nous bénéficions des ressources suivantes du LHS :

1. Base de données de codes malveillants (telescope,…)
2. Infrastructure de test et de virtualisation (Eprouvette)
3. Service web de distribution des applications dont le serveur web est en cours de redéfinition (02/15)

Candidat

La virologie informatique nécessite une double compétence. Il faut à la fois aimer la programmation à bas niveau, le système et le réseau et à la fois être capable de créativité pour employer ou développer des méthodes théoriques et formelles et les appliquer. Un exemple simple de cette dualité est la résolution d'un saut indirect en x86. Le candidat devra avoir cette double compétence. Comme l'équipe est petite, nous sommes extrêmement vigilants sur la qualité du candidat.

Courte bibliographie récente et annotée.

1. Philippe Beaucamps, Isabelle Gnaedig, Jean-Yves Marion: Abstraction-Based Malware Analysis Using Rewriting and Model Checking. ESORICS 2012: 806-823
   C'est dans ce travail que nous avons posé les bases de l'utilisation de l'analyse morphologique pour l'analyse comportementale à partir des traces des appels systèmes.

2. Joan Calvet, José M. Fernandez, Jean-Yves Marion: Aligot: cryptographic function identification in obfuscated binary programs. ACM Conference on Computer and Communications Security 2012: 169-182
   Ce travail illustre une avancée, non incluse dans le prototype actuel, pour détecter les fonctions cryptographiques dans un code binaire. La détection de fonction au sein d'un code binaire permet d'une part d'aider à l'analyse du code et d'autre part d'identifier des similarités entre les codes.

3. Guillaume Bonfante, Jean-Yves Marion, Fabrice Sabatier et Aurélien Thierry, :Code synchronization by morphological analysis. Malware 2012. IEEE conference.
   Si cette conférence est moins prestigieuse que les deux précédentes, cela a été l'occasion de confronter de nos travaux sur l'analyse morphologique avec les certains acteurs importants du monde anti-viral aux Etats-Unis. Nous y avons exposé dans les grandes lignes comment comparer deux codes binaires et trouver des similarités. Ce travail a été présenté à différentes conférences de hackers, e.g. Recon, où il a reçu un accueil favorable.

4. P. Ször, The Art of Computer Virus Research and Defense. Addison-Wesley Professional, 2005.

Un classique sur toutes les techniques virales et anti-virales.

5. Collberg, C., & Nagra, J. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection.* (A. Wesley, Ed.)