Compte-rendu de réunion

Référence RéSIST/2010-09

30 septembre 2010

Table des matières

1 Présentation de M. Gilles Richard – IRIT

1.1 Introduction

M. Gilles Richard nous a présenté les résultats de travaux de recherche relatifs à la sécurité informatique, basés sur la complexité de Kolmogorov.

Aujourd'hui, concernant la sécurité informatique, on observe
- une grande variété d'attaques,
- une grande variété de techniques,
- une grande variété de méthodes.

Ces trois facteur de variabilité augmentent de façon significative la complexité du domaine. Si l'on se reporte aux techniques les plus utilisées pour filtrer et bloquer les attaques, on constate qu'il s'agit de manière plus ou moins subtile d'analyses « textuelles » de données, et que le comportement est toujours réactif (l'exemple typique étant celui des anti-virus).

Avantages Facile à comprendre, ça marche à peu près correctement, c'est relativement paramétrable.

Rédigé par Pierre-Yves Bonnetain -– pyb@ba-consultants.fr
ReSIST -– http://www.ossir.org/resist
Comptes-rendus -– http://www.ossir.org/resist/supports/index.htm

Inconvénients Il faut tenir le système de détection à jour, il est relativement facile à contourner, il ne peut pas s'adapter seul à de nouveaux périls.

1.2 Complexité de Kolmogorov

Andreï Kolmogorov a travaillé, entre autres, sur la théorie des probabilités et la théorique algorithmique de l'information. Il ressort de certains de ses travaux que :

- plus une suite de données (au sens large) est aléatoire, moins il est facile de l'apprendre,
- plus une suite est régulière, plus elle peut être compressée.

En résumé, lorsque le hasard dans une suite de données augmente,

- la quantité d'informations dans la suite augmente,
- l'« apprenabilité » de la suite diminue,
- la compressibilité de la suite diminue.

Gilles RICHARD nous rappelle ensuite qu'une machine de Turing peut être considérée comme une boîte noire qui, à partir d'un programme P et d'une entrée y, produit un certain résultat x, ce que nous noterons $T(P, y) = x$.

La complexité de Kolmogorov, par rapport à une suite de données x, est notée $K(x)$. Il s'agit de *la taille du plus petit programme s'exécutant sur une certaine machine de Turing et capable de produire la suite x à partir d'une entrée vide.* En reprenant la notation précédente, $K(x) = |P|$ [1], sachant que l'on a $x = T(P, \emptyset)$.

Quelques remarques s'imposent :

1. $K(x)$ est un nombre non calculable. Il est seulement possible de lui trouver des majorants.
2. $K(x) < |x| + c$, où c est une constante dépendant de la machine de Turing. Le plus petit programme produisant x est toujours plus petit que le programme imprimant x sur sa sortie.
3. $K(x \oplus y) \leq K(x) + K(y)$, où $\oplus$ représente la concaténation des chaînes x et y. Le plus petit programme produisant $x \oplus y$ est toujours plus court que la suite des deux programmes produisant x et y respectivement.

Trois exemples d'appromixation de $K(x)$, lorsque x vaut :

la suite `01` répétée 10^9 fois , le programme `for i = 1..10^9 print "01"` génère x. Nous avons donc, ici, $K(x) \ll |x|$.

une suite aléatoire de 10^9 0 ou 1 A l'inverse de l'exemple précédent, $K(x) \simeq |x|$.

1. $|x|$ est la longueur de la chaîne x.

les 10^9 premiers chiffres de π Un programme d'une centaine de lignes de C permet d'obtenir cette suite, on peut estimer que $K(x) \simeq 100 \times 10 \times 10 \simeq 10000$.

$K(x)$ mesure donc la quantité d'information présente dans la chaîne x. Il existe de fait une corrélation entre la capacité de prédire une chaîne, celle d'apprendre la dite chaîne, et celle de produire cette chaîne.

1.3 Comment estimer K(x) ?

En raisonnant à l'inverse, nous prenons une suite de données x et nous la compressons (avec un algorithme quelconque, mais sans perte). Cela nous donne la suite $C(x)$. La machine de Turing T mettant en œuvre l'algorithme de décompression sera donc telle que $T(C(x), \emptyset) = x$.

En résumé, *la taille d'une suite de données compressée selon un algorithme sans perte est un majorant de la complexité de Kolmogorov de cette suite.*

1.4 Utilisation de la complexité de Kolmogorov

Note | Il est rappelé que ces éléments sont encore du domaine de la recherche.

1.4.1 Détection d'attaques

Un système informatique traite des « messages » en entrée. Pour savoir si ces messages correspondent à des attaques/risques ou non, il suffit de les compresser et de comparer la taille de la donnée compressée avec des mesures faites au préalable. Si cette taille tombe dans un certain intervalle, il s'agit de « messages » légitimes. Sinon, il y a probablement un problème quelque part.

Exemples :

Attaques sur un serveur FTP Capture glissante de 2 à 3 Ko de données reçues sur le réseau à destination du serveur. La précision obtenue (détection de situations d'attaque) est de l'ordre de 95%, avec 0,2% de faux positifs et 4,8% de faux négatifs.

`nmap`, scan de ports Plutôt qu'étudier $|C(x)|$, on examine l'ICR, inverse du taux de compression, $\frac{|C(x)|}{|x|}$. Avec une fenêtre de capture glissante d'environ 320 Ko, la détection d'un scan de ports est évidente. En situation normale, l'ICR est voisin de 0,8. Lors d'une attaque, il tombe aux alentours de 0,2. La précision de détection est de 95%, avec 0,3% de faux positifs.

1.4.2 Discrimination entre données

Il est possible d'appliquer la complexité de Kolmogorov à d'autres situations que la détection d'attaques, notamment pour la discrimination entre deux groupes de données

On peut définir une distance (au sens topologique du terme) en calculant par exemple $|K(exe) - K(doc)|^2$ (distance entre un exécutable et un document bureautique). Cette méthode permet alors la classification des fichiers ou bribes de fichiers (par exemple ceux trouvés sur un disque dur lors d'une analyse inforensique).

Si l'on a une suite de « paquets d'informations » [2] p_0, p_1 à p_n , il est possible de comparer $\sum_{i=0}^{n} K(p_i)$ et $K(\prod_{i=0}^{n} p_i)$, où le symbole $\prod$ correspond à la concaténation des paquets. Si la différence entre ces deux valeurs est proche de zéro, les paquets d'informations sont faiblement corrélés entre eux. A l'inverse, plus la différence augmente, plus les paquets sont liés entre eux.

Si l'on définit $m(a,b) = K(a) + K(b) - K(a \oplus b)$ comme étant la *mesure de l'information commune* entre a et b, il est possible de définir la *distance informationnelle de Benett* :

$$d(a,b) = 1 - \frac{m(a,b)}{max(K(a), K(b), K(a \oplus b))}$$

sachant que l'on a

$$\begin{cases} K(a) = K(a-b) + K(a \cap b) \\ K(b) = K(b-a) + K(a \cap b) \\ K(a \oplus b) = K(a-b) + K(b-a) + K(a \cap b) \end{cases}$$

Cette distance informationnelle permet (entre autres) d'estimer les similitudes entre deux documents : $A \equiv B \Leftrightarrow d(A,B) \simeq 0$. En dessous d'un certain seuil, la probabilité d'un plagiat de l'un des documents sur l'autre devient très élevée.

La distance informationnelle de Benett a plusieurs applications. Celle présentée par Gilles RICHARD est la **classification des spams**. Cela part du postulat que, lorsque deux groupes d'éléments sont linéairement séparables, il est possible de calculer la « meilleure droite discriminante » (celle qui est « la plus éloignée » de chacun des points à classifier).

Appliquée à la lutte anti-spam, cela revient à prendre un ensemble de 100 messages correctement classifiés (50 spams, 50 hams). Sur réception d'un nouveau message, on calcule la distance informationnelle de Bennett du nouveau message par rapport à chacun des 100 messages de référence, et on décide de l'état du message en fonction de la majorité de classification des 9 plus proches voisins (distance informationnelle la plus courte). Cette méthode donne une efficacité de détection des spams de 87%.

2. Paquets sur le réseau, blocs sur un support de stockage, etc.

2 Présentation de M. Philippe Bourgeois, CERT-IST

2.1 Bilan du CERT-IST 2009

Les statistiques et le bilan produits par le CERT-IST pour l'année 2009 montrent que, de nos jours, près de 70% des attaques visent le poste de travail. De plus en plus souvent, elles utilisent des vulnérabilités dans les logiciels clients (et non plus dans le système d'exploitation). Les 30% des attaques restantes visent l'infrastructure informatique.

M. Bourgeois se propose de nous décrire un incident réel survenu dans un cadre professionnel. Ce second point signifie notamment que le poste de travail concerné disposait d'un système d'exploitation à jour, d'un anti-virus à jour, et qu'il se situait dans une infrastructure protégée par un firewall.

2.2 Dynamique de l'incident

Après que l'utilisateur a ouvert son navigateur, il a demandé à ce dernier (par le biais d'un groupe de signets) d'ouvrir un ensemble de pages sur Internet. Il est souligné que ces pages concernaient des sites à usage professionnel (presse, informations, etc.)

Suite au chargement de ces pages, de nombreuses fenêtres d'information s'affichent sur l'écran. La première qui apparaît concerne notamment une tentative d'accès à la base de registre.

Note

Cela signifie que des programmes indésirables se sont déjà exécutés, et que la détection ne s'est faite que tardivement, lorsque l'un de ces programmes tente de modifier le registre.

Parmi les autres fenêtres qui s'affichent, l'une signale une corruption mémoire dans Acrobat Reader (et la fin du processus), une autre indique la terminaison anormale de l'explorateur de fichiers Windows, et une troisième demande un mot de passe pour l'export de clés privées.

L'utilisateur interrompt son travail, signale l'incident aux équipes concernées, lesquelles contactent le CERT-IST.

2.3 Analyse

Après une copie du disque dur, le CERT-IST procède à une analyse des dates et heures de modifications de fichiers, à la recherche de fichiers « apparus » au même moment que l'incident signalé par l'utilisateur. Dans l'ordre d'apparition, les analystes constatent :

1. l'arrivée d'une applet Java sur l'ordinateur
2. la création de divers fichiers temporaires

3. la création du répertoire `C:\cleanupsweep.exe\`, répertoire caché par un rootkit (SpyEye)
4. la création de plusieurs fichiers PDF...

Les interprétations vont dans le sens d'une applet malveillante qui a installé SpyEye. Ce dernier a tenté plusieurs opérations de prise de contrôle, élévation de privilèges ou résistance au redémarrage. Ce sont ces actions (certaines d'entre elles) qui ont provoqué les affichages des fenêtres d'informations évoquées initialement.

Les analystes relèvent aussi d'autres éléments :
- le vol de données FileZilla et MSN
- l'envoi de spams, via MAPI et Outlook. Les spams ont été envoyés par Outlook, donc au travers de l'infrastructure de l'entreprise.

L'analyse de l'applet concernée fait apparaître un appel à `MidiSystem.getSoundBank`, fonction vulnérable à un débordement de tampon dans la JRE (versions 1.6_update16 et inférieures).

Le code décompilé se révèle peu complexe, essentiellement lié à l'obfuscation de l'attaque. Cela explique que les échantillons viraux soient très variables (substitution de chaînes de caractères) et donc leur mauvais taux de détection. Sur VirusTotal, le 1er août, un seul anti-virus (sur 42) marquait l'applet comme dangereuse. Le 13 septembre, soit six semaines après, seuls 6 anti-virus la notaient comme dangereuse.

Les fichiers PDF déposés sur le poste de travail sont piégés et contiennent du code malveillant. Il est à noter qu'Acrobat version 7 était utilisé sur le poste de travail.

2.4 En conclusion

Il s'agit très probablement d'une infection via Java-JRE, qui sont des éléments rarement mis à jour.

L'origine de l'attaque n'est pas identifiée, un retour sur les signets ouverts par l'utilisateur ne provoque aucune nouvelle infection ou tentative. Il se peut que cela soit passé par un bandeau publicitaire malveillant qui renverrait vers un kit d'exploitation (de type Phoenix).

Pour lutter contre ce phénomène d'infection lors de la navigation web, le CERT-IST propose 3 approches :
- les postes de travail doivent être à jour de tous leurs composants, et pas uniquement du système d'exploitation
- utiliser des greffons spéciaux de type NoScript et AdBlock
- envisager une virtualisation du navigateur, ou au moins un sandboxing.