JAB, une backdoor pour réseau Win32 inconnu

Nicolas Grégoire

Exaprobe
ngregoire@exaprobe.com,
WWW home page : http ://www.exaprobe.com

1 Introduction

Le but de cet article est de montrer les possibilités offertes par Internet Explorer et les objets OLE rattachés dans le cadre de la prise de contrôle à distance d'une machine Windows située dans un réseau totalement inconnu. Nous en profiterons pour présenter JAB, une backdoor utilisant Internet Explorer pour établir un canal de commandes et de données entre la machine compromise et l'attaquant.

Nous verrons tout d'abord les moyens employés jusqu'à présent pour piloter une backdoor Win32, ainsi que les différents outils exploitant la technique de manipulation d'objets OLE utilisée par JAB. Après avoir rapidement examiné le code Perl nécessaire à une requête HTTP minimaliste (GET statique) et donc survolé les fonctions intéressantes du paquetage `Win32 : :OLE`, nous aborderons le cycle de vie de la backdoor, depuis le choix des cibles et des vecteurs d'attaques jusqu'à la gestion de multiples instances via le serveur JABd.

Nous détaillerons ensuite le fonctionnement interne de JAB, dont principalement l'API de communication client/serveur et les techniques de transfert bidirectionnel de données binaires. Nous finirons par une présentation du mode *"command-line"* de la backdoor (utile pour le débugage et l'exfiltration de documents) et des limitations inhérentes à ce type de backdoor.

2 Techniques de contrôle distant de backdoors pour Windows

2.1 Historiques des moyens de contrôle

Le code-source de Windows n'étant pas publiquement disponible, l'évolution des moyens permettant de contrôler à distance une machine compromise est sensiblement différente de celle rencontrée sous Unix. En effet, très peu de *"magic flags"* comme ceux utilisés pour `/bin/login` sous Unix (variables d'environnement, nom d'utilisateur) existent, l'attention s'étant portée sur la fourniture d'un shell (`cmd.exe`, `command.com`) à l'attaquant. La première technique employée est la connexion d'un shell (ou du canal de commande pour *BO2K*) sur un port haut de la machine compromise, mais cela impose que cette machine soit accessible directement par l'attaquant (pas de "masquerading") et qu'il n'y ait pas filtrage INBOUND sur le port employé.

Ces conditions étant rarement réunies dans le cadre d'un réseau interne d'entreprise, la technique suivante consista à initier un reverse-shell directement vers une machine contrôlée par l'attaquant. Le seul pre-requis étant l'existence d'au moins un flux autorisé entre le LAN et l'Internet, cette technique est utile si nous ne sommes pas confrontés à un réseau "strict", c'est-à-dire où les connexions vers l'extérieur doivent passer par des proxys, soit à des fins de lutte anti-virale, soit pour comptabiliser/authentifier les accès sortants.

Dans le cadre de ces réseaux "stricts" employant des proxys, l'attaquant cherche souvent à extraire depuis la base de registre les informations de connexion, afin de les utiliser pour établir son propre canal sortant. Mais plusieurs limitations apparaissent : le proxy peut nécessiter une authentification interactive, ou un firewall personnel peut détecter la tentative d'établissement de connexion via le proxy.

Pour contourner ces limitations, le moyen le plus intuitif est d'arriver à accéder au réseau via un processus autorisé aussi bien au niveau du firewall personnel que du proxy. C'est ainsi que nous avons vu apparaître des attaques portant sur l'injection de code dans l'espace mémoire de processus autorisés, ceci permettant de contourner les firewalls personnels. Mais ce type d'attaque reste assez technique, et n'est pas une solution simple (en terme de code) pour établir un véritable canal de communication entre la machine compromise et l'attaquant.

2.2 Outils utilisant IE/OLE

La première évocation publique de l'utilisation de Internet Explorer et des contrôles OLE pour piloter à distance une machine Windows a été le document "Hacking Guide" de SensePost (p.80) [1].

Depuis, plusieurs outils ont été développés dans ce sens, dont Setiri (par SensePost) [2] et Deep-Pentest (par HSC) [3]. Le premier a été présenté à Defcon 10, et le deuxième semble réservé à l'usage interne de HSC, et n'a donc pas été publié. Il apparaît donc que l'emploi d'Internet Explorer comme vecteur de communication pour une backdoor se développe, bien que nous n'ayons pas pour l'instant de trace d'outil "black hat" employant cette technique.

Le fonctionnement de JAB repose sur un client (situé du côté de la victime) et un serveur (situé côté pen-tester). Ces deux éléments sont codés en Perl, ceci permettant d'utiliser `pack()` pour la conversion de données et `eval()` pour l'interprétation à la volée de code fourni par le serveur. Ceci impose donc la transformation du script contenant le code "client" en exécutable (dans notre cas via *Perl2EXE*), le binaire obtenu étant d'une taille relativement conséquente (environ 900 Ko).

Les objectifs de JAB sont doubles :

- permettre le contournement de la majorité des protections pouvant exister dans un réseau d'entreprise, ceci justifiant l'emploi de Internet Explorer via les contrôles OLE comme moyen de communication,
- faciliter le travail côté pen-tester, c'est-à-dire principalement fournir une gestion aisée et asynchrone d'instances multiples. Ceci permet par exemple

de contrôler des machines situées dans un fuseau horaire différent de celui du "maître", les ordres pouvant être récupérés sur le serveur sans intervention humaine.

Nous verrons par la suite les choix conceptuels et d'implémentation permettant de respecter ces objectifs.

3 Manipulation d'objets OLE, dont Internet Explorer : détails

Étant donné qu'un court morceau de code est parfois beaucoup plus explicite qu'un long discours, voici un programme minimaliste réalisant une requête statique :

```
use strict;
use Win32::OLE;

my $url = "http://www.exaprobe.org/owned.html?jobs_at_victime.fr"";
my $timeout = 60; my $time = 0;

# Creation de l'objet OLE
my $ie=Win32::OLE->new('InternetExplorer.Application') or die $!;
$ie->{Visible} = 0; $ie->{Offline} = 1; $ie->{Silent} = 1;

# Accès à l'URL
$ie->Navigate2($url,2);
while ($time <= $timeout) {
     Win32::Sleep(1000); $time++;
     last unless $ie->{Busy};
}
```

Ce type de code permet donc d'identifier les adresses email dont les destinataires ont exécuté notre code. Aucun "démon" n'est nécessaire côté serveur, une simple lecture des logs du serveur HTTP permettant de lister les adresses email ainsi remontées.

4 Cycle de vie de la backdoor

Cette partie décrit les différentes phases prenant place entre la décision d'attaquer une entité particulière et la fin de vie des processus constitués par les backdoors déployées.

4.1 Génération des binaires

Deux choix très importants pour la suite des évènements doivent être faits :

- allons-nous tenter de faire exécuter notre code via une faille de navigateur, une faille de client de messagerie ou de l'ingénierie sociale ?
- en cas d'envoi par mail, les destinataires finaux sont-ils clairement identifiés ou écrivons-nous à un alias de type "*tous@victime.fr*" ?

Des réponses apportées à ces questions, nous pourrons déduire l'icône la plus adaptée à la situation (Flash, Zip, Word) ainsi que le schéma de génération des identifiants (aléatoire dans le cadre d'un alias de messagerie global, fixe si très peu de comptes sont visés et que l'on désire une certaine traçabilité de nos binaires).

4.2 Déploiement des binaires

Dans le cas d'une attaque par ingénierie sociale, il suffit de composer un mail incitant son destinataire à exécuter le fichier présent en pièce jointe. Dans le cadre d'une exploitation de faille de navigateur, nous pouvons utiliser un outil comme IEsploit.tcl [4] pour compromettre chaque machine Windows non patchée accédant au site Web ainsi piégé.

4.3 Exécution de notre code offensif

Une fois notre binaire lancé par la victime, les actions d'initialisation suivantes sont entreprises :

- dépôt d'un "flag" permettant de prouver la compromission de la machine
- vérification de la connectivité Internet (*http ://www.google.fr*)
- enregistrement auprès du serveur JABd

On entre ensuite dans la boucle principale du programme, celle-ci consistant à :

- récupérer auprès du serveur JABd une liste d'ordres
- interpréter chaque ligne de la liste (cf. l'API pour plus de détails)
- exécuter l'ordre correspondant

4.4 Gestion côté serveur

La personne en charge du contrôle des backdoors peut définir des actions par défaut, consulter les informations remontées (fichiers, résultat d'exécution de commandes) ou décider de copier sur le poste de la victime des fichiers permettant par exemple d'y augmenter notre niveau de privilèges. La gestion asynchrone de la prise d'ordres et de la remontée d'informations permet au gestionnaire de ne pas faire que ça de ses journées, mais aussi de prendre un repos bien mérité même si les backdoors en cours de déploiement sont situées dans plusieurs fuseaux horaires.

5 Fonctionnement interne du client

5.1 API de communication "serveur vers client"

Nous listons ici les ordres pouvant être transmis au client, avec le type de paramètres attendus et la signification de chacune de ces commandes :

- SLEEP : endort le processus distant
  paramètres : nombre d'unité à attendre et unité de temps employée (1s, 30m, 12H)
- DIE : tue le processus distant
  pas de paramètre
- EXEC : exécute une commande DOS et envoie le résultat au serveur
  paramètres : commande à passer à "`CMD /C`"
- EVAL : télécharge du code Perl, l'exécute et envoie le résultat au serveur
  paramètres : nom du fichier contenant le code côté serveur
- DWLD : copie un fichier sur le disque de la victime
  paramètres : nom du fichier distant, nom du fichier local
- UPLD : copie un fichier vers le serveur JABd
  paramètres : nom du fichier local, nom du fichier distant
- CMD : télécharge un nouveau fichier d'ordres, les interprète puis les exécute
  paramètres : nom du fichier contenant les ordres côté serveur

5.2 Transfert de données

Pour le transfert "serveur vers client", un simple GET est suffisant. Les données à transférer sont UUencodées, certains caractères interprétables par Internet Explorer étant ensuite transformés en leur équivalent HTML ('`&`' converti en '`&`').

Pour le transfert "client vers serveur", les données sont transformées selon le principe vu ci-dessus, mais la remontée d'information est faite via des formulaires utilisant la méthode POST et soumis par un code Javascript `onLoad()`. Le découpage des données en fonction de la taille maximale d'un formulaire est fait automatiquement par le client, tout comme est fait automatiquement le réassemblage côté serveur.

5.3 Mode CLI

Le fait de lancer la backdoor avec en paramètre la chaîne "`--top_secret_option`" permet d'entrer en mode interactif. Ce mode permet de visualiser la configuration de la backdoor (URL du serveur JABd, identifiant, ...) et d'accéder à l'API comme si les ordres étaient communiqués depuis le serveur. Ce mode est principalement utile pour le débogage, ainsi que pour l'exfiltration de documents depuis une machine sur laquelle nous bénéficions d'un accès physique ou interactif (VNC, Terminal Server).

5.4 Limitations de JAB

Nous détaillons ici les différentes limitations actuelles de JAB, qu'elles soient liées à des choix conceptuels (utilisation de IE/OLE) ou d'implémentation.

La principale limitation liée à l'utilisation d'Internet Explorer comme moyen de communication est la nécessité de pouvoir accéder au Web soit :

- de manière directe
- via un proxy sans authentification
- via un proxy avec authentification transparente (par IP source, par NTLM)
- via un proxy avec authentification interactive et sur lequel l'utilisateur ciblé est déjà loggué

Les limitations liées à l'implémentation ou aux moyens mis en oeuvre sont :

- absence de persistance de la backdoor en cas de reboot (la mise en place de ce type de persistance imposerait des modifications en base de registres ou dans les fichiers de démarrage)
- pas de HTTPs (besoin d'un certificat reconnu par Internet Explorer pour éviter une popup demandant d'accepter le certificat proposé)
- authentification faible des clients sur le serveur (basée sur une clé statique transitant en clair sur le réseau)
- absence de chiffrement applicatif des données transférées (un simple UUencodage, légèrement modifié, est réalisé)

6 Conclusion

Il ressort de cet article que contre ce type d'attaque, les défenses traditionnelles sont inefficaces car la backdoor apparaît comme l'usage légitime d'un navigateur Web par un utilisateur autorisé. De plus, l'avenir nous promettant une interaction encore plus grande entre le système d'exploitation et les applications, les moyens d'accéder à Internet via un autre processus se multiplieront. Les backdoors se tourneront donc soit vers un accès de très bas niveau (communication directe avec les drivers de la carte réseau) soit vers un accès par rebond à travers une application de confiance (contrôle externe via une API quelconque, injection en mémoire). Dans les deux cas, les défenses situées sur le poste de travail (anti-virus et firewalls personnels) devront évoluer fortement afin de limiter la menace.

Une des meilleures solutions est donc prévenir l'exécution initiale de la backdoor, c'est-à-dire filtrer les documents exécutables à l'entrée du réseau d'entreprise, patcher les machines (même "end user") malgré la charge de travail induite, et surtout éduquer les utilisateurs pour les sensibiliser quant à leur rôle à jouer dans l'application de la politique globale de sécurité. D'autres solutions, comme l'utilisation de "white lists", permettraient de restreindre les sites Web pouvant être accédés par les utilisateurs. Mais il faut garder à l'esprit que cette solution est très lourde et que l'accès par une backdoor IE/OLE à un site de type Caramail permettrait de toute façon la remontée de données via des messages émis à travers le webmail.

Références

1. Roelof W. Temmingh, SensePost : A guide for breaking into computer networks from the Internet
   http ://packetstormsecurity.org/papers/general/hackingguide3.1.pdf
2. SensePost : Defcon 10 Presentation of Setiri
   http ://packetstormsecurity.org/defcon10/dc10-sensepost-setiri.ppt
3. HSC : Deep-PenTest tool
   http ://www.hsc.fr/ressources/presentations/meleenumerique/img8.html
4. Truff, Projet7 : Upload and code execution on IE
   http ://projet7.tuxfamily.org/factory/exploits/IEsploit.tcl