

Invasion du réseau local via le Web 2.0 et DNS Rebind

Aujourd'hui, les risques liés aux navigateurs Web et les fonctionnalités du Web 2.0 ne sont plus à démontrer. Rappelons néanmoins leur portée : le désormais célèbre Cross Site Scripting, le scan de ports (en JavaScript ou en html), l'identification de site web, le Cross Site Request Forgery, le vol d'historique du navigateur, et enfin, les vers JavaScript/Ajax. Ces techniques avérées, fonctionnent néanmoins uniquement à la condition que l'utilisateur, ou son navigateur soient piégés et accède à un site détourné. Ce détournement peut, aujourd'hui naturellement s'effectuer via le DNS.

Il existe, pour prévenir cela, une restriction apportée à la couche JavaScript : le *same origin policy*. Cette restriction stipule que : « Le *same origin policy* empêche un document ou des scripts chargés d'une origine de récupérer ou d'agir sur des propriétés d'un document d'une origine différente ». En pratique, cela veut dire qu'un Javascript téléchargé depuis www.magsecurs.com ne pourrait agir sur un élément du domaine www.nbs-system.com.

Le principal problème rencontré dans l'application du principe de *same origin policy* provient du fait que le contenu d'un site ne vient pas d'un nom : il vient d'une adresse IP. C'est le DNS, dinosaure de l'internet, qui fournit la relation entre nom et adresse IP et enfin, on assume couramment que la relation entre nom d'hôte et adresse IP restera identique. En pratique, une même adresse IP peut héberger quantité de sites différents via des hôtes virtuels, il existe donc un véritable intérêt à se baser sur le nom du domaine plutôt que sur l'adresse IP.

Le DNS Pinning et ses limitations

Différentes solutions ont été apportées, dont la plus répandue est le DNS Pinning. L'objectif du DNS Pinning est de prévenir l'usurpation de nom DNS : on tente, dans le navigateur client, de verrouiller l'association entre un nom et une adresse IP. Le navigateur met en cache l'adresse IP associée à l'hôte visité, jusqu'à la fermeture de la fenêtre courante. Ce principe simple permet donc de s'assurer que l'adresse IP d'un serveur ne sera pas modifiée pendant la session d'utilisation.

Un certain nombre de techniques pour contourner le DNS Pinning existent déjà. Ces techniques sont communément appelées DNS Rebind. Leur objectif est de modifier la liaison entre nom d'hôte et adresse IP dans le cache du navigateur. La première idée ayant émergé pour contourner cela, consiste à utiliser le cache du navigateur :

1. Le client charge le code malicieux du site, celui-ci se retrouve en cache
2. Le navigateur est fermé (par action utilisateur, ou attaque)
3. A la prochaine utilisation, le code malicieux est pris dans le cache, puisque inchangé
4. Le navigateur Internet effectue une requête DNS, à laquelle une réponse forgée ou modifié peut aboutir à une corruption du cache et rendre le client vulnérable.

Une méthode plus simple consiste à rendre le serveur demandé par le client indisponible (Firewall, déni de service temporaire). Force est de constater que pour supporter une défaillance de serveur via le DNS, les navigateurs modernes supprimeront l'adresse IP associé à un nom de domaine si ce serveur est défaillant, pour utiliser un autre enregistrement disponible dans le DNS. En simulant une

défaillance du serveur, il se crée donc une ouverture sur cette association adresse IP / nom d'hôte cachée par le navigateur, qui effectuera une mise à jour de son cache DNS.

Le véritable problème émergent est posé par les plugins, qui peuvent effectuer des connections par eux mêmes. Les plugins ne partagent pas le cache *pin* avec le navigateur. Il est possible de charger l'objet actif (flash, java ...) d'une adresse, de « rebinder » celle-ci puis d'ouvrir du trafic vers une autre destination. Ces plugins ouvrent donc la porte à une nouvelle génération de possibilités offertes par nos navigateurs Web :

- Le navigateur permet d'effectuer de l'http
- La fonction *XMLHttpRequest* fournit un moyen pour effectuer du « tcp » séquentiellement
- Flash9 permet d'utiliser des sockets TCP
- Java fournit le support de TCP et UDP

Tout ce qui est mis en place pour lier un plugin à son site d'origine ne fonctionne pas très bien : le DNS étant susceptible de changer entre le chargement du plugin et son exécution, il existe un véritable vecteur d'intrusion.

La grande limitation d'une attaque par ce vecteur, consiste en son déroulement très séquencé et son contrôle à partir d'un unique nom de domaine et le fait que la fenêtre doit rester ouverte. Il n'est pas possible de contrôler la destination vers une adresse IP, les plugins du navigateur ne fonctionneront qu'à destination éventuelle de leur hôte d'origine, qu'il faut donc contrôler via le DNS.

Le scénario de démonstration

Nous avons préalablement rappelé les risques liés au Web 2.0 et à la multiplicité des technologies émergentes. En y ajoutant les nombreux plugins, fréquemment autorisés, il est possible d'y ajouter la le rebond TCP pure et simple, via un navigateur client au sein du réseau de l'entreprise. Outre le fait que ceci s'appuie sur des technologies répandues, la création de ce qu'on l'on qualifie généralement d'exploit, pour mettre en œuvre ce schéma complet est générique. Il est donc possible d'avoir un outil permettant de s'introduire au sein du réseau local de l'entreprise via le navigateur Web d'un utilisateur tant qu'il supporte les technologies employées.

Considérons une page Web malicieuse composée de deux IFrames (cadre html présentant les mêmes propriétés qu'une page). L'utilisateur est victime d'une attaque sur son DNS vers le site *example.com*. Il charge la page malicieuse A, qui fait référence, dans ses deux IFrames à deux autres pages. Avec un contrôle du DNS, tel que vu précédemment, il s'ensuit que :

- a. Le DNS rebind fait pointer *example.com* vers l'adresse ip d'un serveur web de l'attaquant
- b. La page A se charge, les deux IFrames se chargent, avec leurs plugins flash (ou autre), programmés pour établir une connexion sur l'hôte d'origine dans un délai en temps suffisant pour opérer l'expiration du cache DNS du navigateur et le modifier
- c. Un DNS rebind est effectué. *example.com* pointe vers une adresse IP interne, par exemple, un pointeur (CNAME en DNS) sur le nom « intranet ».

- d. L'objet invoqué dans la première IFrame se connecte sur example.com (actuellement, intranet)
- e. Un DNS rebind est effectué, example.com pointe désormais vers l'adresse IP d'un poste X contrôlé par l'utilisateur malveillant à l'origine de ces manipulations.
- f. La couche JavaScript de la page parente des deux IFrames ouvertes, en respect du *same origin policy* peut communiquer avec les deux IFrames ouvertes, et donc, lire à partir de l'intranet pour envoyer les données vers le poste X et vice-versa.

Le navigateur doté de JavaScript se transforme au final en routeur. Il sert d'interface entre les différentes IFrames dotées de plugins qui deviennent des sockets à part entière à disposition du code Javascript fournis dans la page d'origine.

Notre cas d'école se limite à deux IFrames, mais un code avancé saura sans conteste gérer quantité d'Iframes offrant une possibilité de connexions réseaux de manière élaborée. Ce subtil mélange de technologies permet néanmoins d'offrir un rebond complet, à partir du navigateur client, pour se connecter sur les ressources internes de l'entreprise : Telnet/ssh, ftp, sites intranet, etc. Le navigateur de la victime offre à l'attaquant les mêmes possibilités qu'à l'utilisateur légitime.

Les solutions

Le dernier rempart pour se protéger de ce problème est le navigateur Web. On peut alors rêver d'une granularité plus évoluée des zones de sécurité Internet Explorer, la mise en place de zones de sécurité pour Firefox et Opéra (qui ne gère aujourd'hui des paramètres de sécurité que site par site).

Le constat est simple, désactiver JavaScript semble aujourd'hui être une tâche difficilement entreprenable pour une entreprise car de trop nombreux sites reposent sur ce langage. Il faut néanmoins ouvrir une véritable réflexion sur les technologies et plugins autorisés : télécharger et exécuter du code provenant d'internet ne sera jamais quelque chose de sûr.

Un certain nombre d'initiatives sont en train de se mettre en place afin de palier au problème, un add-on pour Firefox voit le jour : « localrodeo ». Son objectif est de détecter et bloquer les modifications dans le cache DNS du navigateur. Ceci est certes viable pour JavaScript, mais les plugins ne partagent pas le cache DNS, et les navigateurs derrière un serveur proxy ne bénéficieront pas de ce type de protection. Un autre add-on, NoScript, pour firefox permet d'autoriser l'exécution de JavaScript par liste blanche uniquement.