Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

S . G . D . S . N
Agence nationale de la sécurité des systèmes d'information
CERT-FR

Paris, le 15 juin 2015
N° CERTFR-2015-INF-001

Affaire suivie par :
CERT-FR

NOTE D'INFORMATION DU CERT-FR

Objet : DNS Rebinding

Gestion du document

Référence	CERTFR-2015-INF-001
Titre	DNS Rebinding
Date de la première version	15 juin 2015
Date de la dernière version	–
Source(s)	
Pièce(s) jointe(s)	Aucune

TAB. 1 – *Gestion du document*

Une gestion de version détaillée se trouve à la fin de ce document.

1 - Description

Le *DNS Rebinding* vise à permettre à un attaquant situé dans un réseau d'accéder à une application web située dans un autre réseau. Le cas typique est la tentative d'accès à une application web du réseau interne d'un organisme par un attaquant situé sur Internet. Pour ce faire, l'attaquant contourne certains mécanismes de cloisonnement des navigateurs web en employant des réponses DNS fluctuantes.

La communication entre l'attaquant et l'application web interne s'effectue à travers le navigateur connecté aux deux réseaux et employé comme plateforme de rebond.

Pour cela, l'attaquant piège l'utilisateur de façon à lui faire charger un code malveillant, généralement écrit en JavaScript, dans son navigateur. Ce code a alors pour objectif de récupérer les données du réseau inaccessible directement par l'attaquant pour ensuite les retransmettre sur le réseau où se situe ce dernier.

Les navigateurs protègent les utilisateurs contre des attaques venant de pages malveillantes, en isolant chaque site les uns des autres. Pour cela, une politique de restriction d'accès entre sites, appelée *Same Origin Policy (SOP)*, est appliquée.

La SOP assure que chaque site est isolé en fonction de son « origine ». Cette notion d'origine est définie par le triplet protocole, domaine, et port. Par exemple, le triplet pour `http://ssi.gouv.fr` est le protocole http, le nom de domaine `ssi.gouv.fr` et le port 80.

Une page chargée depuis `http://attacker.com/` n'aura donc pas le même triplet que `http://intranet.local`. Un script malveillant chargé depuis `http://attacker.com` ne pourra donc pas accéder aux applications web internes hébergées sous `http://intranet.local/`.

Le *DNS Rebinding* a pour objectif de contourner cette protection en faisant croire au navigateur que l'adresse IP à contacter pour accéder au site `http://attacker.com/` n'est plus celle du site de l'attaquant, mais celle de l'application web interne ciblée. Ainsi, le triplet SOP reste identique mais les requêtes HTTP effectuées par le code malveillant sur la page `http://attacker.com/` seront directement envoyées à l'application web interne.

Pour réaliser ce changement, l'attaquant fait varier l'adresse IP renvoyée par le service de résolution de noms de domaine DNS afin que le nom de domaine `attacker.com` pointe alternativement sur une machine contrôlée par l'attaquant, puis sur une machine à laquelle ce dernier n'a pas accès directement.

Il convient de noter que même si la navigateur utilisé comme plateforme de rebond dispose d'un cookie permettant l'accès à une session préalablement authentifiée auprès de l'application hébergée sur le réseau interne, l'attaquant effectuant un *DNS Rebinding* ne peut en tirer parti. En effet, le cookie de session est associé au nom de domaine légitime de l'application web ciblée et non à `attacker.com`, et n'est donc pas automatiquement adjoint par le navigateur.

2 - Impact

L'attaquant obtient, par l'intermédiaire du navigateur de sa victime, la capacité d'interroger un serveur web hébergé sur un réseau auquel il ne peut pas accéder directement. Le navigateur joue alors le rôle de mandataire (*proxy*).

3 - Historique des attaques

1996 - Utilisation des *applets* Java

À cette époque, le JavaScript ne disposait pas des API nécessaires pour mettre en oeuvre l'attaque par *DNS Rebinding*.

Une version simplifiée de cette dernière était cependant déjà possible, grâce aux greffons (*add-ons*) des navigateurs. En pratique, les applets Java pouvaient être exploités en renvoyant simultanément, pour un nom de domaine donné, deux adresses IP : l'adresse IP du site de l'attaquant et l'adresse IP du serveur web attaqué. L'applet pouvait alors accéder indifféremment au serveur web attaqué et au site de l'attaquant. Sun, l'éditeur de Java à cette époque, répondit à cette vulnérabilité en modifiant la politique de sécurité de la machine virtuelle du langage : après son chargement, une applet ne peut accéder qu'à l'adresse IP depuis laquelle elle a été chargée, et ce pendant toute sa durée de vie.

2002 - JavaScript

Deux attaques employant JavaScript comme vecteur ont été publiées [1] en 2002.

La première utilise le « *domain relaxation* ». L'attaquant héberge son code malveillant sur un sous-domaine de `attacker.com`, comme `sub.attacker.com`. Il fait alors pointer `sub.attacker.com` sur l'adresse IP de son serveur HTTP, et `attacker.com`, le domaine parent, sur l'adresse IP du serveur web auquel il n'a pas accès.

Le « *domain relaxation* » est alors effectué en JavaScript, pour permettre à un script chargé dans le contexte d'un sous-domaine d'accéder à des pages hébergées dans un domaine parent.

Ainsi, le code chargé depuis `sub.attacker.com` peut effectuer des actions directement sur le site victime à travers le domaine `attacker.com`.

La contre-mesure déployée a été de demander que le « *domain relaxation* » soit sollicité explicitement, à la fois par un script du domaine parent et par le script du sous-domaine voulant y accéder.

La deuxième, appelée « *Quick-swap DNS* », nécessite que l'attaquant configure son nom de domaine pour que la mise en cache par le résolveur DNS soit de très courte durée. Il configure pour cela un TTL (*Time To Live*) d'une durée de quelques secondes sur l'enregistrement DNS permettant la résolution d'une adresse IP pour le nom `attacker.com`.

L'attaque se déroule alors en deux temps : premièrement, `attacker.com` pointe sur le serveur hébergeant le code malveillant. Une fois ce code malveillant chargé par le navigateur, le code JavaScript attend que le cache DNS expire, puis tente de générer une requête vers le domaine `attacker.com`. Le cache ayant expiré, l'adresse IP n'est plus connue pour ce nom et le navigateur doit faire une nouvelle requête DNS. Lors de cette seconde résolution de noms, le serveur DNS contrôlé par l'attaquant répond que le nom de domaine `attacker.com` pointe vers l'adresse IP du serveur web inaccessible à l'attaquant. La connexion initiée par JavaScript se fait alors vers le serveur hébergeant l'application web objet de l'attaque.

En réponse, Netscape et Mozilla ont implémenté l'épinglement DNS (*DNS pinning*), associant pour le temps de vie d'une page web, un nom de domaine aux adresses IP spécifiées dans la seule première résolution de noms.

Depuis 2006 - Toutes les technologies web sont employées

Depuis 2006, les opportunités d'attaques en DNS Rebinding se multiplient. Elles sont principalement dues à la complexification du modèle de sécurité des navigateurs, équipés de toujours plus de greffons, d'extensions. La prise en charge d'HTML 5 apporte également de nombreuses fonctionnalités qui peuvent être détournées.

À titre d'exemple, certaines vulnérabilités de navigateurs ont permis de contourner des contre-mesures grâce à la mise en cache de pages web grâce à l'en-tête HTTP `Expires` [2], [3].

À l'instar de Java, plusieurs années après, les greffons Flash et Silverlight se sont avérés présenter les mêmes vulnérabilités.

De plus, malgré la protection mise en place sur la machine virtuelle Java, l'utilisation du LiveConnect JavaScript-to-Java a, à nouveau, permis la réalisation des attaques, en abusant des caches DNS distincts entre greffons, et entre les greffons et le navigateur.

Deux nouvelles attaques ont également été trouvées via les *applets* Java : une utilisant différentes instances de la machine virtuelle Java et l'autre exploitant une faille lorsqu'un serveur mandataire est utilisé.

HTML5 apporte de nombreuses nouvelles APIs, dont certaines peuvent être utilisées pour fiabiliser certaines attaques rendues difficiles par les défenses des navigateurs.

En particulier, l'API AppCache, permettant la mise en cache d'applications web pour une utilisation hors-ligne, peut être utilisée pour fiabiliser des attaques par *DNS Rebinding* [4].

4 - Méthodes de protection

Protection dans le navigateur : l'épinglage DNS

La protection principale intégrée dans les navigateurs est l'épinglage des réponses DNS (*DNS pinning*).

Le principe est le suivant : l'association du nom de domaine à son adresse IP est conservée par le navigateur pendant une durée prédéfinie, issue d'une politique propre à chaque navigateur, voire à chaque version de navigateur.

Cette protection présente des inconvénients. En effet, elle viole le protocole DNS et la volonté des administrateurs des noms de domaine légitimes qui configurent des durées de mises en cache courtes, parfois à dessein. C'est notamment le cas pour des techniques de répartition de charge (*DNS round robin*), de reprise à chaud (*failover actif*), ainsi que dans le cadre de contre-mesures aux dénis de service distribués et aux plans de reprise ou de continuité d'activité, suite à un incident.

La durée de vie de l'association varie d'un navigateur à l'autre. Une entrée expire après une période d'inactivité avoisinant les deux minutes, et est automatiquement supprimée à la fermeture du navigateur. Il convient également de noter que ces associations sont stockées dans des tables de taille limitée, et qu'il est possible de surcharger la table jusqu'à en faire sortir le nom de domaine dont l'attaquant cherche à faire modifier l'association [5]. À titre d'exemple, dans Chrome 25, la table d'association était limitée à 100 entrées. Depuis Chrome 26 sa taille a été augmentée à 1000 entrées.

Il convient de noter que cette protection est totalement déjouée dès lors que le navigateur est configuré pour employer exclusivement un serveur mandataire. En effet, dans ce cas de figure, le navigateur n'effectue aucune requête DNS, et demande simplement à obtenir une ressource web spécifique. C'est alors le serveur mandataire qui peut être pris pour cible de l'attaque en *DNS Rebinding*. Pour cette raison, les applications web internes ne doivent, en aucun cas, pouvoir être accessibles à partir des serveurs mandataires utilisés pour se connecter à des réseaux de moindre confiance.

Protections au niveau de l'infrastructure réseau

Certains résolveurs DNS tels que OpenDNS, Bind9 ou Unbound peuvent être configurés pour filtrer les réponses DNS suspectes reçues depuis l'Internet et ainsi supprimer celles contenant des adresses IP internes.

Filtrage d'une plage d'IP

BIND9 ainsi que Unbound peuvent être configurés pour filtrer toutes réponses contenant des IP appartenant à une place spécifique. Les instructions spécifiques à chacun de ces logiciels sont détaillées ci-dessous.

Pour BIND9 :

```
[ deny-answer-addresses { address_match_list } [ except-from { namelist } ];]
```

Pour Unbound :

```
private-domain: namelist
```

`private-address: local_subnet/subnet_mask`

L'outil DNSmasq, parfois utilisé en tant que cache DNS local, peut être configuré pour rejeter et journaliser les réponses contenant des adresses IP référencées dans la RFC1918. Cela peut être accompli grâce à l'option `-stop-dns-rebind`.

Considération sur le filtrage des IPv4 RFC1918

OpenDNS [6] et DNSmasq peuvent être configurés pour filtrer toutes les réponses contenant des IP appartenant aux plages RFC1918. Cette protection est cependant limitée car certaines organisations aux réseaux historiques utilisent des IP hors RFC1918 pour des ressources locales. Aussi, cette méthode n'est pas fiable dans le cas des routeurs avec une interface externe (Internet) et une interface interne. En effet, il est parfois possible depuis une IP interne de requêter l'interface externe du routeur et donc de monter le *DNS Rebinding* vers l'interface externe (avec une IP publique) [7].

Il peut être noté que le filtrage des adresse IPv4 RFC1918 peut être également appliqué aux réseaux IPv6 en filtrant les adresses de type *Unique Local Addresses* (`fc00::/7`).

Protections au niveau de la configuration des postes clients

La mise en place d'une stratégie double navigateurs est présentée dans les guides de l'ANSSI [8], [9].

Cette architecture présente de nombreux avantages de sécurité et peut notamment permettre d'éviter des attaques par *DNS Rebinding*.

Ces guides préconisent l'utilisation d'un navigateur pour les accès aux réseaux internes et d'un deuxième navigateur pour la navigation sur Internet.

La mise en oeuvre de cette stratégie implique notamment que le navigateur ayant accès à Internet n'ait aucun moyen de contacter le réseau interne. Cela peut s'effectuer en appliquant une politique forçant le passage par un serveur mandataire, ou par l'emploi d'un pare-feu tenant compte des processus pour appliquer sa politique de filtrage.

Il convient de noter que si un serveur mandataire est employé pour l'accès à l'Internet, il ne doit en aucun cas pouvoir accéder également au réseau interne.

Protections au niveau du serveur applicatif

Authentification

Une première étape est d'obliger une authentification par un mot de passe non prédictible avant d'accéder à l'application. Une authentification basée sur l'IP source est bien entendu inutile.

Protection de l'hôte virtuel par défaut

Une autre défense consiste à s'assurer que l'hôte virtuel (*VirtualHost*) par défaut n'offre aucun service. Cet hôte virtuel est, en effet, le seul pouvant être atteint dans le cadre d'une attaque par *DNS rebinding*. Cette contre-mesure peut être appliquée de deux manières, avec le serveur Apache.

La première consiste à s'assurer que le premier hôte virtuel chargé par Apache, lors de son démarrage interdise toute action.

Cela peut être accompli en insérant avant tout autre hôte virtuel, les lignes de configuration détaillées suivantes.

Pour Apache 2.2 :

```
<VirtualHost *:80>
  DocumentRoot /non-existent
  <Directory />
    order allow,deny
    deny from all
    AllowOverride None
  </Directory>
</VirtualHost>

<IfModule ssl.c>
<VirtualHost *:443>
  DocumentRoot /non-existent
  <Directory />
```

```
    Order allow,deny
    Deny from all
    AllowOverride None
  </Directory>
</VirtualHost>
</IfModule>

<IfModule gnutls.c>
<VirtualHost *:443>
  DocumentRoot /non-existent
  <Directory />
    Order allow,deny
    Deny from all
    AllowOverride None
  </Directory>
</VirtualHost>
</IfModule>
```

Pour Apache 2.4 :

```
<VirtualHost *:80>
  DocumentRoot /non-existent
  <Directory />
    Require all denied
    AllowOverride None
  </Directory>
</VirtualHost>

<IfModule ssl_module>
<VirtualHost *:443>
  DocumentRoot /non-existent
  <Directory />
    Require all denied
    AllowOverride None
  </Directory>
</VirtualHost>
</IfModule>

<IfModule gnutls_module>
<VirtualHost *:443>
  DocumentRoot /non-existent
  <Directory />
    Require all denied
    AllowOverride None
  </Directory>
</VirtualHost>
</IfModule>
```

Il est également possible de vérifier l'en-tête HTTP `Host` afin de s'assurer que le domaine utilisé pour accéder à l'hôte virtuel est le bon. En effet, avec le `DNS Rebinding`, l'en-tête `Host` présenté sera `Host: attacker.com` et non le nom de domaine légitime par lequel le serveur web visé aurait dû être contacté.

Jusqu'à Apache 2.2, cette contre-mesure peut être mise en oeuvre grâce aux *Rewrite Rules*, comme suit :

```
ServerName monnomdedomaine.fr
RewriteCond %{HTTP_HOST} !^monnomdedomaine.fr$
RewriteRule ^.*$ - [F,L]
```

Depuis Apache 2.4, on peut écrire directement l'expression :

```
ServerName monnomdedomaine.fr
Require expr %{HTTP_HOST}=="monnomdedomaine.fr"
```

Les règles de réécriture fournies ci-dessus retournent une erreur 403 lorsque le nom de domaine de l'en-tête `Host` est différent de celui renseigné dans la directive `ServerName`.

5 - Conclusion

Bien que certaines contre-mesures aient été mises en place dans les navigateurs, ces dernières s'avèrent insuffisantes pour contrecarrer efficacement les attaques en *DNS Rebinding*.

Lorsque l'application web et l'environnement d'hébergement sont maîtrisés, il convient d'appliquer les recommandations suivantes :

1. Vérifier la cohérence entre la valeur de l'en-tête Host et celle configurée pour l'hôte virtuel courant ;
2. Effectuer une authentification de l'utilisateur accédant aux applications web ; cette recommandation implique, au minimum, l'utilisation d'un mot de passe fort, possiblement aléatoire, et unique à chaque équipement, dans le cas d'un mot de passe « par défaut » ;
3. S'assurer que les services réseaux n'écoutent que sur les interfaces réseaux sur lesquelles ils doivent légitimement rendre le service.

Pour la protection d'équipements non maîtrisés sur un réseau interne, l'adoption de la stratégie double navigateurs avec un serveur proxy configuré de manière adéquate est recommandée.

6 - Bibliographie

- 1 : *Firewall circumvention possible with all browsers*, A. Megacz, http://seclists.org/bugtraq/2002/Jul/0362.html
- 2 : https://bugzilla.mozilla.org/show_bug.cgi?id=689835
- 3 : https://code.google.com/p/chromium/issues/detail?id=98357
- 4 : *Eradicating DNS Rebinding with the Extended Same-Origin Policy*, USENIX, 2013
- 5 : *FireDrill : Interactive DNS Rebinding*, WOOT, 2013
- 6 : http://labs.opendns.com/2013/09/03/what-are-the-suspicious-responses-and-why-you-
- 7 : *How to hack millions of routers*, C. Heffner, BH USA, 2010
- 8 : *Recommandations pour le déploiement sécurisé du navigateur Microsoft Internet Explorer*, ANSSI
- 9

15 juin 2015 version initiale.