Reconnaissance réseau à grande échelle : port scan is not dead

Adrien Guinet et Fred Raynal
aguinet@quarkslab.com
fraynal@quarkslab.com

Quarkslab

Résumé Le scan de ports existe depuis que des machines sont connectées à des réseaux. Une nouvelle jeunesse lui est offerte grâce à deux facteurs :
– les réseaux sont de plus en plus rapides et fiables ;
– de nouvelles techniques de scan, principalement asynchrones.
Soulignons la facilité avec laquelle on *scale* une infrastructure, en ajoutant / enlevant des machines, on est alors en mesure de scanner très facilement un grand nombre d'adresses IP pour récupérer l'information ainsi exposée sur Internet. Cet article présente les enjeux pour réaliser un tel projet.

1 Introduction

Les *cyber-attaques* un peu évoluées - pas les Stuxnet, rien que du *spear phishing* déjà - demandent un peu de préparation. Par exemple, il faut collecter des documents qui parleront aux cibles, leurs adresses emails, mettre en place la logistique pour exfiltrer les documents qui seront volés sur les postes des cibles et ainsi de suite.

On fait souvent référence au cycle du renseignement. On rappellera son fonctionnement en étapes, lorsqu'une question est posée pour en obtenir la réponse :

- Questionner : expression du besoin, des éléments recherchés.
- Collecter : récupération des informations susceptibles de donner accès à la réponse.
- Analyser : trier toutes les informations, en évaluer la pertinence, structurer les informations par recoupement, association.
- Disséminer : envoyer les informations aux destinataires, qui pourront alors revenir avec de nouvelles questions.

Une attaque informatique, quant à elle, se modélise très bien par la classique roue de Deming :

- *Plan* : collecter de l'information sur la cible pour élaborer sa stratégie.
- *Do* : préparer ses outils, des exploits, RATs, et autres *rootkits* visant la cible.

- *Check* : tester les outils, par exemple face aux anti-virus, pour en mesurer la performance, les traces.
- *Act* : début de l'opération, qui ne se passe pas nécessairement comme prévu.

De manière assez naturelle, les attaques un peu préméditées reposent sur des éléments acquis par les opérateurs. Concrètement, il peut s'agir d'adresses emails, de noms, de documents internes... Bref, l'étape préalable à toute bonne attaque, c'est la collecte, et la collecte, c'est le cycle du renseignement. Et si cette démarche n'avait pas également un sens pour la défense ?

2 Le reconnaissance réseau pas à pas

La reconnaissance réseau est soit passive, soit active. La première méthode repose sur de la capture de trafic réseau (*sniffing*, *monitoring* ou encore interception) pour acquérir de l'information. Par exemple, certains outils surveillent les informations émises par les navigateurs pour déterminer s'ils sont vulnérables. L'approche passive est majoritairement utilisée sur les réseaux internes car elle évite de déployer des équipements supplémentaires. L'approche active consiste à aller demander aux serveurs des informations. Sur un périmètre exposé sur Internet, une telle approche n'est pas gênante vu le volume de trafic induit par un scan.

Quelle que soit la démarche retenue, la reconnaissance réseau suit le cycle du renseignement dans la démarche :

- Définition des cibles : quels réseaux sont visés ?
- Définition de la collecte : quelles données sont recherchées ?
- Analyse des résultats : quelles informations sont les plus importantes ?
- Dissémination des résultats : comment ré-utiliser les données pour approfondir ses recherches ?

Si ces étapes semblent simples, elles recèlent des pièges, surtout lorsqu'on cherche à faire de la reconnaissance réseau à grande échelle. Les parties suivantes décrivent ces difficultés.

2.1 Acquisition des cibles

L'élément de base manipulé est une adresse IP. Toutefois, scanner des adresses au hasard ne présente pas un grand intérêt si on n'a pas un objectif précis en tête.

Pour éviter cela, on met en place des filtres qui prennent en entrées des données variées et les transforment en ensembles d'adresses IP. Ces

ensembles auront alors un sens propre à celui qui émet la requête, et les données collectées seront interprétables dans ce contexte.

Si la reconnaissance en elle-même peut se faire de manière automatique, il n'en est hélas rien de cette étape de définition des cibles.

Pays

Comment définir un *pays* sur Internet ?

Est-ce l'ensemble de tous les domaines se terminant par le TLD ayant été alloué à ce pays ? Il est évident que cette définition ne colle pas puisqu'il existe des TLD non associés à des pays, et qu'ils sont utilisés partout, le plus célèbre étant le `.com`. On peut en citer d'autres tels que `.ws`, `.net`, `.me` ou encore `.info`.

Une autre approche serait de lister tous les AS associés à un opérateur, mais certains opérateurs travaillent sur plusieurs pays voire continents. Dans le même ordre d'idée, les routes BGP peuvent aider.

On peut ajouter à la liste des ranges IP allouées aux divers acteurs. Cette information est entre autres disponible grâce aux bases whois. De ce point de vue là, le site `http://ipindex.homelinux.net/` [1] offre une aide précieuse : il propose de naviguer dans les plages d'adresses, des classes A jusqu'aux classes C, en indiquant à qui appartiennent les *netblocks* considérés.

Mais l'option la plus naturelle est d'utiliser une base d'adresses géo-localisées. Ces bases ne sont pas toujours précises, et certaines plages d'adresses ne sont pas identifiées. La base gratuite GeoIP de Maxmind [4] en est un bon exemple.

Ainsi, définir un pays, c'est un peu de tous ces aspects à la fois, et pas uniquement un seul. Il s'agit donc d'une union d'ensembles.

Entités

Une entité peut être définie comme un nom de domaine pour lequel on ne trouve pas directement tous les sous domaines et adresses relatifs, ou encore une multinationale avec ses filiales.

Par exemple, si on considère le domaine `.gouv.fr`, il n'existe pas de base publique listant toutes les IP associées et aucun DNS à énumérer. Il est possible de les demander à l'AFNIC, mais il faut justifier d'un sacré motif légitime pour cela.

Si on considère une multinationale, de nombreux sous-domaines sont sous-traités à des tiers, sans parler des filiales. Au final, il est exceptionnel

qu'une DSI *monde* ait réellement une visibilité complète du périmètre dont elle est responsable.

Dans les deux cas, quelques approches, difficilement automatisables, permettent néanmoins de résoudre partiellement le problème :

- La première chose à faire est de rechercher sur `ipindex.homelinux.com` [1] tous les domaines contenant le nom de la cible. Toutefois, pour un domaine comme `.gouv.fr`, cela ne fonctionne pas.
- Immédiatement après, il faut tenter un transfert de zone sur un serveur DNS de la cible. On trouve encore de très nombreux serveurs qui acceptent cela. À noter que ces deux étapes sont à faire pour tous les sous-domaines, filiales ou éléments potentiellement connectés à la cible.
- Vient ensuite la bonne vieille requête Google|Bing|* `site:taget.com`, à affiner avec un `-inurl:www` par exemple, pour filtrer les domaines, comme `www` qui reviennent trop souvent. Cette méthode n'est pas exhaustive, et limite les résultats aux serveurs web, mais cela permet de récupérer de nouveaux noms, qui seront alors résolus en une IP, mais on prendra soin de scanner toute la *range* voisine.
- Utiliser les résultats d'un scan sur les ports 80 et 443, les 2 plus représentés sur Internet, et le bon vieux `grep` dans les certificats ou les bannières des sites pour sortir toutes les IP liées à la cible, avec la même remarque que précédemment : ne pas se restreindre à cette IP, mais considérer la *range* auquel elle appartient.

2.2 Performances

Plusieurs logiciels ont vu le jour dans le domaine du scan à grande échelle. Les deux plus connus sont ZMap [5] et masscan [2]. En soi, ils utilisent tous un modèle asynchrone, où les paquets permettant la découverte sont envoyés par un *thread* et une boucle d'événements (dans un autre *thread*) gère le traitement des réponses. Afin de gagner en vitesse, les stacks TCP et IP noyau sont généralement évités au profit d'implémentations *user-land*.

Les différences entre ces outils sont les suivantes :

- masscan peut utiliser un driver spécifique Intel pour augmenter la limite possible du nombre de paquets émis par seconde ;
- masscan et zmap n'ont pas d'état par hôte et ne prennent pas en compte de timeout. Ainsi, un paquet SYN envoyé en début de scan peut obtenir une réponse à la fin, et sera pris en compte. D'après

le papier de référence, cela permettrait d'avoir des réponses plus larges ;
- nmap, *a contrario*, gère différents états par hôte, comme le *timeout* ou le nombre de retransmissions de paquets ;
- masscan peut récupérer des bannières après connexion grâce à une connexion TCP *user-land* et gère plus de 10 millions de connexions TCP concurrentes [1]. Le désavantage principal est que ce modèle n'est pas extensible ;
- zmap fournit une bibliothèque qui permet d'implémenter des scanners à base de *sockets* asynchrones, qui reposent sur les résultats d'un scan préalable. Le désavantage est que le nombre de connections concurrentes est limité par les performances du noyau sous-jacent. De plus, il choisit un port source libre à chaque connexion TCP/UDP, et donc en théorie arrivera difficilement à dépasser les 65536 connexions concurrentes. La bibliothèque est de plus en C et ne propose pas de bindings haut niveau pour du prototypage rapide ;
- nmap a un système très évolué de *plugins* fondé sur le langage LUA qui permet de l'étendre à une grande majorité de protocole. Les connexions initiées par ces scripts utilisent des *sockets* asynchrones et la boucle d'événements de nmap. La limite précédente est donc aussi présente.

De plus, aucun de ces scanners ne permet de définir des propriétés par hôte. Par exemple, on peut disposer d'une liste d'identifiants FTP sur certains serveurs et pas sur d'autres. Aucun des scanners précédents ne permet cette distinction. Ainsi, en scannant, soit on passera aucun identifiant, et on manquera donc ceux dont on a les *credentials*, soit les scanners tenteront tous les identifiants sur tous les serveurs...

2.3 Big data, what else ?

Les données récupérées sont de nature extrêmement variables : l'ouverture ou la fermeture d'un port, le whois d'une adresse, une communauté snmp, un certificat SSL... Cela pose deux problèmes dans la manipulation des données.

Tout d'abord, pour la visualisation des données, nous avons imaginé 4 vues :
- Spatiale : c'est une vue linéaire, IP après IP, ou par port, des machines qui ont répondu au moins à un port. Cette vue ne sert pas à grand chose car on ne peut en tirer une information.

1. d'après https ://github.com/robertdavidgraham/masscan#c10-scalability

– Statistique : afin d'enrichir la vue spatiale, on l'agrémente de quelques statistiques, comme les services les plus présents, la localisation des machines, etc.
– Temporelle : on visualise l'évolution soit d'une machine soit d'un paramètre, s'il le permet, dans le temps.
– Différentielle : on mesure la différence entre 2 scans, les nouvelles machines, celles qui ont disparu, et celles modifiées.

Visualiser ces données est indispensable, mais il faut aussi pouvoir y rechercher une aiguille dans une botte de foin. Pour cela, les méthodes classiques d'indexation semblent pour le moment pertinentes.

2.4 Les scannés sont chatouilleux !

À scanner Internet, on chatouille des gens qui n'aiment pas ça. On reçoit de temps en temps des mails indiquant que nous menons une cyber-attaque !

Actuellement, quand on reçoit un mail de ce type, on ajoute le *netblock* dans une *blacklist* qui ne sera plus scannée.

Comme expliqué précédemment, un scan est réalisé sur des ensembles d'adresses IP, elles-mêmes issues de *netblock*. Il est aisé de détecter un tel scan. Pour limiter certaines détections, les scans sont distribués sur plusieurs serveurs, pas tous dans le même réseau. Afin d'assurer un bon mélange des adresses sur les sondes de scan, nous avons élaboré une bibliothèque open source, `libleeloo`, assurant cette distribution aléatoire [3].

Cette bibliothèque est donc responsable de régler deux problèmes inhérents à ce qui a été décrit ci-dessus : gérer une liste de plages d'adresses IP (IPv4 et/ou éventuellement IPv6) et créer une permutation aléatoire des adresses IP représentées.

La cible définie peut donc être sous cette forme :
– la France entière d'après la base GeoIP Maxmind (soit une liste de plages IPv4) ;
– inclure l'AS3215 (clients orange), soit une autre liste de plages IPv4. Ces plages ont de grandes chances d'être déjà inclues dans la liste des IP "françaises", mais il en manque peut être ;
– s'interdire dans tous les cas les plages IPv4 RFC1918 et les autres plages réservées [6].

Représentée autrement, notre cible comporte une liste d'intervalles IPv4 à inclure, et une autre à exclure. La `libleeloo` se charge de traiter ces listes afin de les agréger en une liste finale qui ne contiendra pas de doublons. La représentation finale de la cible est donc une liste d'intervalles d'entiers représentant des adresses IPv4 (en *big-endian*) :

```
[
    [0x02000000, 0x02010000[, // 2.0.0.0/16
    [0x500c0400, 0x500c0500[, // 80.12.4.0/24
    [0x25bb2f46, 0x25bb2f47[, // 37.187.47.70
    ...
]
```

À noter qu'une adresse IPv4 seule est représentée comme un intervalle contenant seulement un élément.

La bibliothèque est capable d'agréger assez aisément des millions d'intervalles. Par exemple, sur un Xeon E3-1240, un milliard d'intervalles aléatoires sont agrégés en environ 28 secondes. Ce temps est majoritairement dû au tri parallèle effectué sur les intervalles avant de les agréger (de 22 secondes ici). Cet exemple est un cas extrême, et pour un millions d'intervalles (ce qui couvre largement la plupart de définitions de cibles), l'agrégation se fait en 22 ms (dont 16 ms pour le tri).

Ensuite, il faut obtenir une liste aléatoire de ces adresses. Pour cela, une permutation de la liste [0..nombre d'IP représentées] est calculée. Le stockage de cette liste en mémoire pouvant être très coûteuse (par ex. 4Go pour 2^{30} IP (soit environ 1 milliard)), un polynôme générateur assurant l'unicité des nombres calculés est utilisé. Cela est détaillé en détail dans un article de blog disponible ici : http ://blog.quarkslab.com/unique-random-number-set-computation.html. Avec quelques optimisations et en les calculant en parallèle, sur un Core i7-3770, nous sommes capables d'avoir 290 millions de nombres aléatoires par seconde.

Cette liste est ensuite utilisée pour indexer les IP dans la liste des intervalles décrits. Pour cela, un cache est créé permettant de connaître l'index de départ des intervalles. Cet index est stocké (par défaut) seulement tous les 256 intervalles. Cette valeur empirique a été choisie pour le bon compromis utilisateur mémoire/performance qu'elle apporte, et peut être modifiée éventuellement par la suite. À la demande de la N-ième adresse IP, une dichotomie est effectuée sur ce cache afin de trouver l'intervalle le plus proche correspondant. Un parcours linéaire des intervalles est ensuite fait afin de trouver l'IP correspondante.

Les performances totales de la génération d'une permutation aléatoire de la liste d'adresses initiale, pour des intervalles d'une taille moyenne de 1000, avoisine les 7.5 millions d'IP par seconde, ce qui est largement suffisant compte tenu des capacités de scan actuels.

3 Conclusion

Une question qui surgit rapidement dès qu'on parle de reconnaissance réseau et scan de ports est celle de la légalité. Entrons donc dans le vif du sujet.

Tout d'abord, nous ne sommes pas juristes. Nous ne rentrerons donc pas dans un débat juridique. Quand on pose la question à un avocat "le scan de ports est-il légal ?", on a une réponse de normand : ça dépend.

A priori, cela dépend de l'intention derrière le scan. Une personne faisant ce repérage en prévision d'une intrusion, action illégale, commet déjà un acte illégal. Mais si l'intention est propre, correcte et légale, alors, il n'y aurait pas de problème.

L'avocat, prudent, s'empressera alors d'ajouter que tout cela est à l'appréciation du juge, et nous voilà bien avancés.

Donc pour résumer, les méchants ne se privent pas d'aller chercher les faiblesses que nous exposons à longueur de journée sur Internet, pendant que les gentils doivent attendre l'avis d'un juge pour savoir s'ils peuvent imiter les méchants pour collecter des données utiles à l'élaboration de leur défense.

Au-delà du simple plaisir de faire des nœuds au cerveau des juristes, la reconnaissance réseau et autres collectes d'information ont une vraie valeur ajoutée : il suffit de voir combien de personnes ont une idée du périmètre de leur réseau dans une grande entreprise. Cette absence de vision pénalise toutes les initiatives.

Références

1. dihe. Ip index. http ://ipindex.homelinux.net/, 2014.
2. Robert David Graham. Masscan. https ://github.com/robertdavidgraham/masscan, 2013.
3. Adrien Guinet. liblееloo. https ://github.com/quarkslab/liblееloo/, 2014.
4. Maxmind. Base geoip maxmind. http ://dev.maxmind.com/geoip/geoip2/geolite2/, 2014.
5. University of Michigan. Zmap. http ://zmap.io, 2013.
6. Wikipedia. Wikipedia : Reserved ip addresses. http ://en.wikipedia.org/wiki/Reserved_IP_addresses, 2014.