Présentation de l'outil Nmap-Stateful

Olivier Courtay

IRISA / Université de Rennes 1
olivier@courtay.org

1 Introduction

Nmap-Stateful est un projet d'extension de Nmap. Nmap possède un module de détection active de système d'exploitation distant (OS Fingerprinting) qui utilise des paquets soigneusement choisis et étudie les réactions des machines testées. Malheureusement, ces tests ne sont pas configurables car inclus dans le source du programme et ces tests ne permettent que l'étude des ports dans l'état TCP LISTEN ou CLOSE. Nmap-Stateful permet de lever ces contraintes et ainsi d'améliorer ou de concevoir d'autres utilisations que l'OS Fingerprinting. Après un rappel rapide du fonctionnement de Nmap nous introduisons les nouvelles fonctionnalités apportées à l'outil puis nous présentons le fonctionnement et l'utilisation de Nmap-Stateful à travers plusieurs expériences enfin nous conclurons sur les évolutions possibles de l'outil.

2 Rappels sur Nmap

Nmap [7] est un outil réseau créé par Fyodor, il contient un ensemble de fonctionnalités incluant : un scan de port, la détection de machines, l'OS Fingerprinting et récemment un module de détection de service. Dans ce papier, nous nous intéressons uniquement au module d'OS Fingerprinting [6] de Nmap, car Nmap-Stateful se base sur ce module.

Nmap dispose de neuf tests pour prendre une empreinte d'une machine distante :

- un test sur un port UDP fermé qui renvoie un message d'erreur ICMP.
- un test permettant à partir d'un échantillon de paquets TCP de faire des analyses statistiques sur la génération des *IP ID* et des *TCP sequence number*.
- quatre tests sur des ports TCP ouverts.
- trois tests sur des ports TCP fermés.

Ces neuf tests permettent de créer une empreinte de la machine en notant la valeur de champs intéressants dans les paquets TCP/IP, comme le montre cet exemple d'empreinte :

```
TSeq(Class=RI%gcd=<6%SI=<269E81A&>62D97%IPID=Z%TS=1000HZ)
T1(DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
```

```
T4(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=D0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134
%DAT=E)
```

est associée par Nmap à l'OS :

```
Fingerprint Linux 2.6.0 (X86)
Class Linux | Linux | 2.6.X | general purpose
```

Chaque ligne correspond à un test. Pour chaque test, un ensemble d'éléments est étudié, par exemple les flags TCP (*Flags=AS*) ou les options TCP (*Ops=MNNTNW*). Ces empreintes sont ensuite comparées à une base d'empreintes connues et associées à un système.

3 Fonctionnement de Nmap-Stateful

Nmap-Stateful se présente comme une extension apportée à Nmap : le code de base utilisé est donc celui de Nmap qui a été modifié pour ajouter les fonctionnalités voulues. Deux éléments montrent que Nmap peut être amélioré : la non-pertinence de certains tests et des retours d'expérience de l'outil Cron-Os [2]. En effet, Nmap positionne dans son premier test sur un port TCP ouvert (qui est l'un des tests les plus pertinents) le flag ECN. Le résultat ne diffère pas que ce flag soit positionné ou non mais les paquets contenant le flag ECN sont arrêtés par certains Firewall (CheckPoint FW-1 par exemple). Ce qui prouve que les tests de Nmap ne sont pas forcément les plus pertinents. Lors de l'utilisation de l'outil Cron-Os, la pile TCP/IP de la machine distante est amenée dans un état voulu avant d'êre analysé (par une analyse temporelle). Bien qu'il appparaît que ces états peuvent être sources de renseignements utiles, ils ne sont pas exploités par Nmap. Ces éléments montrent l'intérêt de concevoir un outil plus souple sur les tests, ayant la couverture des états TCP la plus large et la plus configurable possible.

Nmap-Stateful semble le seul outil qui rassemble toutes ces fonctionnalités. Pour comparaison, on peut citer les autres principaux outils d'OS Fingerprinting :

- Nmap bien sûr (la référence du domaine) mais qui devient un sous-ensemble de Nmap-Stateful.
- Xprobe [3] qui est un outil basé uniquement sur l'analyse des paquets ICMP. Cependant les messages ICMP sont souvent bloqués par les Firewall, ce qui empêche le fonctionnement de l'outil dans des milieux fortement filtrés.
- Cron-Os utilise les délais de retransmission TCP. Cron-OS utilise aussi plusieurs états TCP mais pas les caractéristiques TCP des paquets.

– pOf [4] est un outil d'OS Fingerprinting passif (il ne fait qu'écouter les paquets transitant sur le réseau). Il utilise trois types de paquets pour faire son analyse : les SYN, les SYN-ACK et les RST. Nmap-Stateful peut aussi utiliser ces paquets.

Contrairement à Nmap, les tests utilisés par Nmap-Stateful sont lus dans un fichier de configuration ce qui permet à un utilisateur de pouvoir les modifier et les adapter à ses besoins. Le comportement des cibles est mis sous forme d'empreintes comme dans Nmap, et un fichier de sortie peut être précisé pour stocker l'empreinte. En effet, comme on le verra par la suite, d'autres tests que de l'OS Fingerprinting sont possibles. Comme les tests ne sont plus fixés, on peut construire soi-même son fichier de référence d'empreintes associées aux tests et il est possible aussi de spécifier le fichier de base d'empreintes à utiliser.

4 Apport de Nmap-Stateful

Après avoir décrit les innovations présentes dans Nmap-Stateful, nous allons maintenant nous intéresser aux apports de l'outil. Le premier apport est l'augmentation de l'acuité de l'OS Fingerprinting. En effet, en augmentant les possibilités de tests, la pertinence et la puissance de détection sont augmentées. Nous avions montré avec l'outil Cron-OS que l'étude d'autres états de la pile TCP permettait dans certains cas de voir le véritable système d'exploitation se trouvant derriére un Syn-Relay.

Le deuxième apport est d'en faire un outil de test de pile TCP/IP. En effet grâce à Nmap-Stateful, on peut amener une pile TCP/IP dans un état voulu puis lui envoyer des paquets non-standard. On augmente ainsi la couverture de tests. Cela peut amener à renforcer la prise en compte de certains cas non-standard au sein de la pile et ainsi la rendre moins vulnérable.

Le troisième apport, qui nous semble le plus original, est le test de Firewall Stateful. Si l'on considère un Firewall Stateful (c'est-à-dire un Firewall qui maintient l'état des connexions dont les paquets transitent par ce Firewall), il peut être intéressant de tester et de connaître les mécanismes de transition d'états ou les filtres appliqués aux paquets. On peut bien sur jouer sur le flag TCP : que fait un Firewall lorsqu'il reçoit un paquet avec le flag SYN dans une connexion déjà existante ? Le considère-t-il comme une nouvelle connexion ou rejette-t'il le paquet ? En effet, dans un état ESTABLISHED, un paquet ne doit pas comporter le flag SYN. On peut aussi jouer sur les *sequence number* et *acknowledge number* pour découvrir la fenêtre dans laquelle le Firewall considère que le paquet est valide. C'est la raison pour laquelle ces champs sont modifiables dans la définition des tests de Nmap-Stateful. On consultera le livre de Richard Stevens [8] pour plus de détails sur TCP/IP. Un diagramme d'état TCP simplifié est présenté en figure 1 pour aider à la compréhension.

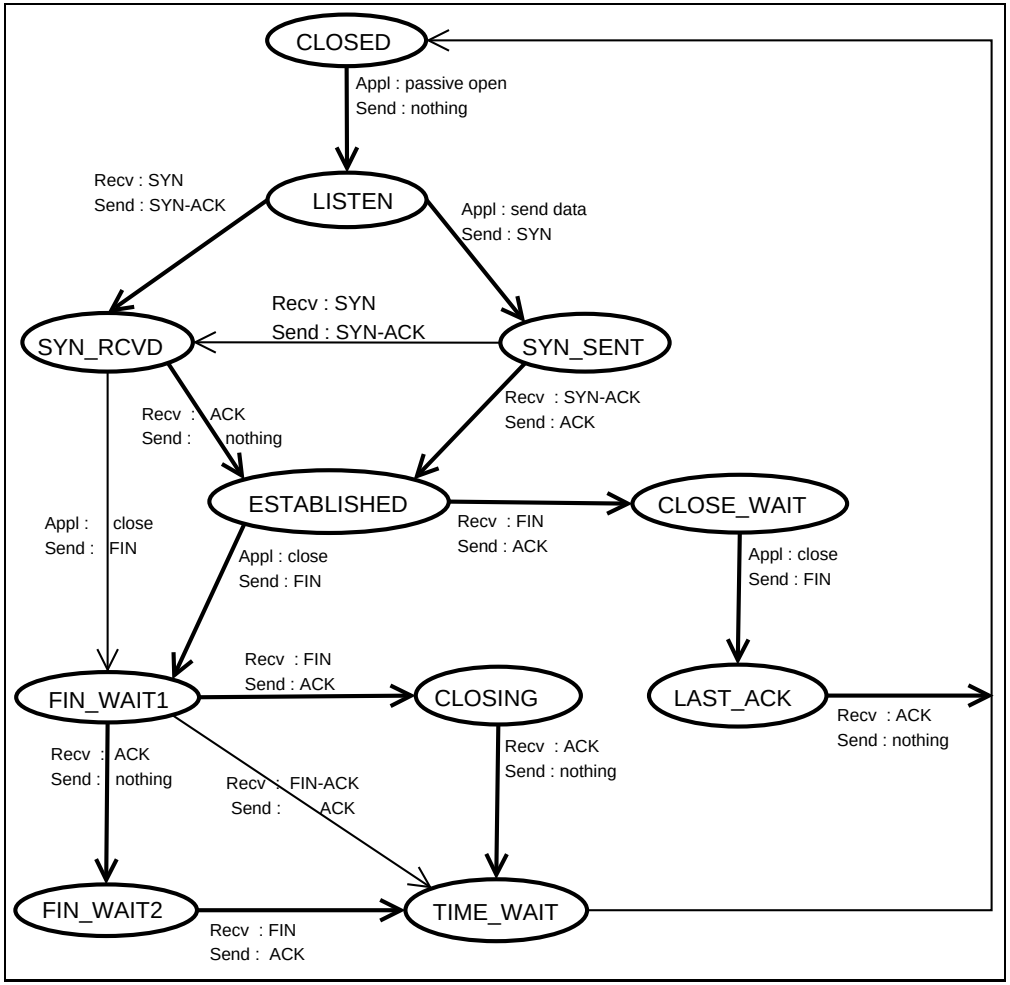

Fig. 1. Udiagramme d'état TCP, Appl : action de l'application possédant la socket, Recv : flags TCP reçus par la socket, Send : flags TCP émis par la socket.

5 Le programme Nmap-Stateful

Nmap-Stateful peut être téléchargé sur la page du projet [1]. Comme explicité plus haut, la principale caractéristique de Nmap-Stateful par rapport à Nmap est d'être capable d'amener la machine auditée dans l'état TCP désiré avant de lancer le test. Pour cela, deux éléments sont nécessaires : avoir une pile TCP/IP simplifiée permettant de lancer les paquets TCP correct par rapport aux standards (par exemple renvoyer un ACK au SYN-ACK de la machine distante pour terminer la connexion TCP), et il faut aussi bloquer la réception des paquets de la machine distante pour éviter que la pile TCP/IP de notre propre machine ne réponde. Pour cela, Nmap-Stateful positionne automatiquement des règles sur le Firewall de la machine locale pour bloquer la réception des paquets de la machine distante. Actuellement, Nmap-Stateful ne fonctionne que sous Linux 2.4 ou 2.6 (car il ne connaît que l'outil iptables) mais un portage vers d'autres Unix (FreeBSD, OpenBSD, Solaris) est possible grâce à la libdnet [5]. Nous allons définir quelques éléments nécessaires pour la bonne compréhension du fonctionnement de l'outil, ces éléments seront ensuite utilisés dans un exemple complet dans le prochain chapitre.

5.1 La définition des tests

Dans cette section, nous présentons briévement la grammaire de Nmap-Stateful permettant d'exprimer les tests.

Un test est défini entre les deux mots clé NAME= et END. Avant de lancer le test, on peut définir l'état dans lequel on désire que soit la connexion TCP. Tous les états ne sont pas accessibles car certains sont des états transitoires. À terme les états suivants devraient être disponibles : LISTEN, CLOSE, SYN_RCV, ESTABLISED, FIN_WAIT1, CLOSING, FIN_WAIT2, TIME_WAIT, LAST_ACK. On peut aussi définir les flags TCP à positionner dans le paquet. Les options TCP peuvent être précisées ainsi que les données contenues dans le paquet. Les *sequence number* et *acknowledge number* sont modifiables dans le paquet de test. En effet une connexion TCP possède un *sequence* et *acknowledge number* courant, par défaut le paquet de test les utilise mais on peut altérer ces nombres en ajoutant une valeur précisée dans la description du test.

Exemple de test :

```
NAME=T1
LISTEN
TH_SYN
TH_ECE
OPTIONS=\003\003\012\001\002\004\001\011\010\012\077\077\077\077
OPTIONSLEN=14
SEQ=2
ACK=-3
DATA=example
DATALEN=6
```

```
END
```

Ce test s'appelle T1, il envoit son paquet de test lorsque la cible est dans l'état LISTEN. Le paquet de test contient les flags SYN et ECN et des options TCP. Le paquet transport des données (*example*) et le *sequence number* courant de la connexion sera incrementé de 2 et le *acknowledge number* sera decrementé de 3.

Pour assurer une certaine compatibilité avec Nmap, un fichier de tests correspondant aux tests de Nmap est incorporé dans Nmap-Stateful.

5.2 Le fonctionnement de Nmap-Stateful

Plusieurs options ont été rajoutées à Nmap :
--os_tests_file *file* : spécifie le fichier de tests à utiliser (ou --otf).
--os_result_in_file *file* : spécifie un fichier où écrire les résultats (ou --orf).
--os_fingerprint_file *file* : spécifie le fichier d'empreintes de référence à utiliser (ou --otf).

Il existe d'autres options pour préciser le nombre maximum de tests à effectuer en même temps (pour éviter de stresser la machine) et le nombre de secondes à attendre la réception d'un paquet réponse.

Pour illustrer le fonctionnement de Nmap-Stateful, nous allons commencer par un exemple simple qui consiste à lancer un test lorsque la machine auditée est dans l'état ESTABLISHED. Le résultat du test est seulement affiché car on ne fournit pas de fichier de base d'empreintes.

La machine testée est un linux 2.6 sur un réseau local qui écoute sur le port 22 (SSH). le fichier de tests *test_example* est le suivant :

```
NAME=ESTABLISHED_SUPE
ESTABLISHED
TH_SYN
TH_URG
TH_ECE
TH_PUSH
DATA=example
DATALEN=7
ACK=1
END
```

La trace d'exécution est :

```
#./nmap-stateful --otf test -p 22 192.168.1.1
.....
Send SYN 38557 => 22 for test:ESTABLISHED_SUPE
.....
SYN_SENT seq:1281454224 sp:38557
                --> dp:22 ack:0000000000 SYN_RECV fl:S
```

```
ESTABLI ack:1281454225 dp:38557
                <-- sp:22 seq:1426449042 SYN_RECV fl:SA
Launch test:ESTABLISHED_S with source port:38557
UNKONWN seq:1281454225 sp:38557
                --> dp:22 ack:1426449043 UNKONWN fl:A
UNKONWN seq:1281454225 sp:38557
                --> dp:22 ack:1426449043 UNKONWN fl:SPUB
LISTEN ack:1281454233 dp:38557
                <-- sp:22 seq:0000000000 UNKONWN fl:RA
.....
ESTABLISHED_SUPE(Resp=Y%DF=Y%W=0%ACK=O%Flags=AR%Ops=)
```

La première ligne correspond à la ligne de commande, on utilise donc le fichier de test *test*, on précise le port (-p 22) et la machine auditée (192.168.1.1). Puisque le test exige que la connexion soit dans l'état ESTABLISHED avant de lancer le paquet de test, le programme envoi automatiquement un paquet de début de connexion (port source 38557 vers le port destination 22). Les lignes suivantes sont des lignes d'informations sur les échanges de paquets, le format est :

- l'état de notre machine (SYN_SENT)
- le *sequence number* du paquet (seq :1281454224)
- le port source (sp :38557)
- la direction du paquet (–¿)
- le port destination (dp :00022)
- l'*acknowledge number* du paquet (ack :0000000000)
- les flags TCP présent dans ce paquet (fl :SA)

Lorsque le paquet de test est lancé, les états des machines dans cette connexion sont positionnés à UNKNOWN car les conséquences de la réception du paquet de test sur la connexion ne sont pas connues. Le programme note la réaction de la machine, la fait passer par le l'analyseur de paquet TCP de Nmap, puis affiche l'empreinte de la réaction de la cible.

5.3 Création de tests pertinents

Nous avons plus haut que le nombre de tests potentiels est très important (la combinaison des flags TCP et des états est important), il faut trouver une méthode pour sélectionner les tests pertinents.

Dans un premier temps il faut se poser la question de la définition d'un test pertinent. Un test pertinent est un test stable, *i.e.*, qui renvoie toujours le même résultat sur le même type de cible et dans le même environnement. Par exemple le *Window size* sera toujours 0xE240 sur FreeBSD 4.9 configuré par défaut. Un test pertinent est aussi un test discriminant ,*i.e.*, qui permet de montrer une différence de comportement entre deux types de cibles ou d'environnements.

Pour cela, un outil est livré avec Nmap-Stateful pour aider les utilisateurs à créer leurs propres tests en générant des ensembles de tests automatiquement

et en les triant suivant leur résultats. Cet outil s'appelle *fingerprinttool* et voici quelques exemples d'utilisations :

La génération automatique d'un ensemble de tests, en utilisant un test de base, par exemple :

```
NAME=example
ESTABLISHED
TH_URG
TH_FIN
END
```

On le met dans un fichier (appelé *test_sample* ici) et on appelle l'outil fingeprinttool :

```
#./fingerprinttool -g test\_sample -o generate\_tests
```

Cet outil prend en argument la commande -g (pour *generate*) suivi du fichier de test (*test_sample*) et écrit le résultat dans le fichier *generate_tests*. Le fichier *generate_tests* contient l'ensemble des tests générés :

```
NAME=ESTABLISHED_______
ESTABLISHED
END
NAME=ESTABLISHED___F______
ESTABLISHED
TH_FIN
END
NAME=ESTABLISHED_____U____
ESTABLISHED
TH_URG
END
NAME=ESTABLISHED___F_U____
ESTABLISHED
TH_FIN
TH_URG
END
```

Si l'état TCP n'est pas précisé dans le fichier de test source, alors un test par état sera aussi généré. On précise dans le fichier source les flags TCP que l'on compte utiliser, l'outil générera toutes les combinaisons possibles (sauf l'option NO_GEN_FLAGS est ajoutée). On peut de même ajouter aussi des variations sur les *sequence number* et les *acknowledge number* en mettant par exemple les deux lignes SEQ=3 et GEN_SEQ. Dans ce cas, le programme fingerprinttool va générer tout les tests suivant SEQ=-3, SEQ=-2,..., SEQ=2 , SEQ=3.

Après avoir généré un grand nombre de tests, il faut trouver des tests stables (donnant toujours le même résultat). Pour cela on lance la suite de tests sur plusieurs cibles dont on attend le même comportement puis on filtre les tests qui donnent le même résultat :

```
\#./fingerprinttool -s -t tests -o stable\_tests resultat1 /
resultat2 ...  resultatN
```

On utilise la commande -s (pour same) avec comme arguments -t pour lui indiquer le fichier de tests utilisés, -o pour préciser le fichier où les tests jugés stables seront stockés, puis ensuite la liste des fichiers de résultats obtenus avec ces tests sur les différentes cibles.

Maintenant que dans l'ensemble des tests générés on a sélectionné les tests stables, il faut y rechercher les tests différentiateurs. Pour cela il faut lancer les tests stables sur des cibles différentes que l'on veut pouvoir différencier grâce à Nmap-Stateful, puis filtrer ceux qui amènent des résultats différents :

```
#./fingerprinttool -d -t tests -o discriminant\_tests /
fingerprintresult1 resultat1 resultat2 ...  resultatN
```

On utilise la commande -d (pour *different*) pour filtrer les tests qui donnent des résultats différents.

Au lieu de -s ou -d, on peut utiliser la commande -c qui trie les tests du plus différentiateur au moins différentiateur car ils n'existe pas toujours des tests qui ont un résultat différent sur toutes les cibles testées.

Au final, on peut ainsi générer des tests pertinents de façon quasi automatique sans avoir besoin d'être un expert TCP/IP.

6 Un exemple réel complet

Pour bien montrer le fonctionnement de l'outil, ainsi que pour montrer quelques résultats, nous proposons de faire un exemple complet d'utilisation de Nmap-Stateful. On ne va pas chercher ici à obtenir les tests les plus pertinents mais plutôt à montrer qu'il est possible d'utiliser l'outil sans aucune analyse détaillée TCP/IP. Le but sera ici de construire un fichier d'empreinte (fingerprint) permettant de reconnaître les principaux OS.

6.1 Génération des tests

Nous choisissions de nous intéresser à l'état ESTABLISHED, en utilisant les flags TCP : SYN, ACK, PUSH, FIN et en transmettant des données dans le paquet TCP. Le test de base permettant de générer l'ensemble des tests est le suivant :

```
NAME=test_sample
ESTABLISHED
TH_SYN
TH_ACK
TH_FIN
TH_PUSH
DATA=example
```

```
DATALEN=7
SEQ=1
END
```

On peut noter la ligne *SEQ=1* car le paquet de test utilise par défaut le *sequence number* et *acknowledge number* du dernier paquet reçut. Dans le cas de notre test, le SYN-ACK de la connexion TCP, puis nous avons envoyé le ACK de fin de connexion, le *sequence number* doit donc être incrémenté de 1.

La génération des tests s'effectue donc ainsi :

```
#./fingerprinttool -g test\_sample -o generate\_tests
```

Dans le fichier *generate_tests*, on a par exemple les tests suivant :

```
NAME=ESTABLI_
ESTABLISHED
DATA=example
DATALEN=7
SEQ=1
END
NAME=ESTABLI_SAFP
ESTABLISHED
TH_SYN
TH_ACK
TH_FIN
TH_PUSH
DATA=example
DATALEN=7
SEQ=1
END
```

6.2 Sélections des tests stables

Pour sélectionner les tests stables il faut vérifier que les tests générés donnent toujours le même résultat sur le même type d'OS. Pour cela, on va lancer les tests sur plusieurs cibles ayant le même OS (on peut aussi lancer le test plusieurs fois sur la même cible pour augmenter la fiabilité). Il est important de choisir des cibles qui ne sont pas protégées par des Firewalls comme nous le verrons par la suite (cela n'a malheureusement pas toujours été possible dans les tests réalisés ci-dessous).

Exemple sur l'OS Linux 2.4 :

Première cible :

```
#./nmap-stateful -n -P0 -sS --ntp 5 --nts 3 --otf generate_tests
--orf generate_tests_Linux2.4-1  premiere_cible
...
ESTABLI_(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
```

```
ESTABLI_S(Resp=Y%DF=Y%W=0%ACK=O%Flags=AR%Ops=)
ESTABLI_A(Resp=Y%DF=Y%W=16A0%ACK=O%Flags=A%Ops=NNTNN)
ESTABLI_SA(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
ESTABLI_F(Resp=Y%DF=Y%W=16A0%ACK=O%Flags=A%Ops=NNTNN)
ESTABLI_SF(Resp=Y%DF=Y%W=16A0%ACK=O%Flags=AS%Ops=MMNW)
ESTABLI_AF(Resp=N)
ESTABLI_SAF(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
ESTABLI_P(Resp=Y%DF=Y%W=16A0%ACK=O%Flags=A%Ops=NNTNN)
ESTABLI_SP(Resp=Y%DF=Y%W=0%ACK=O%Flags=AR%Ops=)
ESTABLI_AP(Resp=Y%DF=Y%W=16A0%ACK=O%Flags=A%Ops=NNTNN)
ESTABLI_SAP(Resp=Y%DF=Y%W=16A0%ACK=O%Flags=AS%Ops=MMNW)
ESTABLI_FP(Resp=N)
ESTABLI_SFP(Resp=Y%DF=Y%W=0%ACK=O%Flags=AR%Ops=)
ESTABLI_AFP(Resp=Y%DF=Y%W=16A0%ACK=O%Flags=A%Ops=NNTNN)
ESTABLI_SAFP(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
```

Les options -n, -P0, -sS sont des options appartenants à l'outil Nmap. Les options de Nmap-Stateful utilisées sont :
--ntp 5 : on lance 5 tests en parallèle.
--nts 3 : la réponse de la cible est attendue trois secondes avant de conclure à l'absence de réponse.
--orf *generate_tests_Linux2.4-1* : le résultat est mis dans le fichier indiqué

Sans rentrer dans les détails, on peut s'apercevoir de plusieurs éléments :
- Certains tests amenaient des réponses et d'autres non.
- Plusieurs groupes de flags TCP sont retournés (A, AR, AS et R).
- Plusieurs groupes d'options sont retournés (NNTNN, MMNVV ou aucune option).

On recommence le test une autre fois sur la même cible et on stocke le résultat dans le fichier *generate_tests_Linux2.4-2*. Puis sur une autre cible ayant le même OS, les tests sont lancés deux fois, ce qui génère deux nouveaux fichiers résultat *generate_tests_Linux2.4-3* et *generate_tests_Linux2.4-4*.

Nous pouvons vérifier si les tests générés sont stables grâce à l'outil fingerprinttool :

```
#./fingerprinttool -s -t generate_tests -o selected_tests /
generate_tests_Linux2.4-1 generate_tests_Linux2.4-2 /
generate_tests_Linux2.4-3 generate_tests_Linux2.4-4 /
```

- L'option -s : ne sélectionne que les tests ayant le même résultat dans tous les fichiers de résultats donnés en paramètre.
- L'option -t : le fichier de test utilisé pour générer les fichiers de résultats.
- L'option -o : le nom du fichier où seront placés les tests sélectionnés.

Le résultat ici est que tous les tests sont jugés stables. Il arrive parfois que les tests soient jugés non stables à cause d'erreurs réseau et on conclut faussement à la non stabilité des tests, il faut alors éliminer ces résultats et recommencer de nouveaux tests. Une des possibilités pour que deux tests différent est que le

Window Size ne soient pas le même. En effet le *Window Size* correspond à la place libre du buffer de réception, taille qui peut donc varier lorsque la connexion TCP est établie.
Une nouvelle fonctionnalité de l'outil fingerprinttool qui devrait être incluse dans une prochaine version, pour que par exemple si deux tests ne différent que par le *Window Size*, alors le test soit quand même retenu mais que le *Window Size* du résultat du test ne soit pas comparé au *Window Size* de l'empreinte contenue dans le fichier d'empreintes.

6.3 Sélections des tests discriminants

On suppose maintenant que l'on dispose des fichiers de résultat pour chaque type d'OS que l'on souhaite reconnaître avec le fichier de tests stables construit à l'étape précédente. On va maintenant rechercher les tests les plus discriminants. Pour cela, on va une nouvelle fois faire appel à l'outil fingerprinttool.

```
#./fingerprinttool -d -t generate_tests -o selected_tests /
generate_tests_FreeBSD4.9 generate_tests_FreeBSD5.2 /
generate_tests_Linux2.4 generate_tests_Linux2.6 /
generate_tests_MacOSX generate_tests_Solaris /
generate_tests_Windows2000 generate_tests_Windows2003
```

L'option -d : ne sélectionne que les tests ayant un résultat différent dans tous les fichiers de résultats donnés en paramètre. Les autres options sont identiques à celles vues précédemment.

Malheureusement ici le fichier *selected_tests* généré par l'outil fingerprinttool est vide, aucun test ne permet à lui seul de différencier tous les OS voulus. On va devoir se contenter de sélectionner les tests les plus différentiateurs seulement. L'outil fingerprinttool est capable de trier des moins discriminants au plus discriminants avec l'option -c.

```
#./fingerprinttool -c -t generate_tests -o selected_tests /
generate_tests_FreeBSD4.9 generate_tests_FreeBSD5.2 /
generate_tests_Linux2.4 generate_tests_Linux2.6 /
generate_tests_MacOSX generate_tests_Solaris /
generate_tests_Windows2000 generate_tests_Windows2003

21 occurences of:ESTABLI_P
16 occurences of:ESTABLI_F
16 occurences of:ESTABLI_FP
15 occurences of:ESTABLI_
7 occurences of:ESTABLI_SAFP
5 occurences of:ESTABLI_SFP
4 occurences of:ESTABLI_SA
4 occurences of:ESTABLI_SAF
4 occurences of:ESTABLI_SP
```

```
3 occurences of:ESTABLI_AF
2 occurences of:ESTABLI_S
2 occurences of:ESTABLI_A
2 occurences of:ESTABLI_SF
2 occurences of:ESTABLI_AP
2 occurences of:ESTABLI_AFP
1 occurences of:ESTABLI_SAP
```

Le test qui obtient le plus souvent le même résultat est le test nommé ESTABLI_P (dans l'état ESTABLISHED des données sont envoyés avec le flag PUSH). Les résultats sont les suivants :

```
FreeBSD4.9:ESTABLI_P(Resp=N)
FreeBSD5.2:ESTABLI_P(Resp=N)
Linux2.4:ESTABLI_P(Resp=N)
Linux2.6:ESTABLI_P(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=A%Ops=NNTNN)
MacOSX:ESTABLI_P(Resp=N)
Solaris:ESTABLI_P(Resp=N)
Windows2000:ESTABLI_P(Resp=N)
Windows2003:ESTABLI_P(Resp=N)
```

Seul le Linux 2.6 a répondu à ce test. Ce test est donc peut-être une caractéristique. Inversement le test ESTABLI_SAP (dans l'état ESTABLISHED sont envoyés avec avec les flags SYN, ACK et PUSH) ne renvoit presque jamais les mêmes résultats :

```
FreeBSD4.9:ESTABLI_SAP(Resp=Y%DF=Y%W=E000%ACK=0%Flags=AS%Ops=MNWNNT)
FreeBSD5.2:ESTABLI_SAP(Resp=Y%DF=Y%W=FFFF%ACK=0%Flags=AS%Ops=MNWNNT)
Linux2.4:ESTABLI_SAP(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=AS%Ops=MMNW)
Linux2.6:ESTABLI_SAP(Resp=Y%DF=Y%W=16A0%ACK=0%Flags=AS%Ops=MMNW)
MacOSX:ESTABLI_SAP(Resp=Y%DF=Y%W=8218%ACK=0%Flags=AF%Ops=NNT)
Solaris:ESTABLI_SAP(Resp=Y%DF=Y%W=6028%ACK=S++%Flags=AR%Ops=)
Windows2000:ESTABLI_SAP(Resp=Y%DF=Y%W=FFFF%ACK=0%Flags=AS% /
   Ops=MNWNNTNNM)
Windows2003:ESTABLI_SAP(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
```

Pour le reste de l'exemple, on continuera avec le même fichier de test car tous les tests apportent une part d'information pour différentier les OS, même s'il y a une possibilité d'avoir des tests redondants.

6.4 Génération du fichier d'empreintes

Après avoir construit des fichiers de résultats pour chaque type d'OS que l'on souhaite reconnaître, il faut tous les rassembler dans un seul fichier pour en faire notre base de référence par rapport aux tests sélectionnés durant l'opération précédente.

Pour cela, il faut commencer à éditer le fichier de résultat et à remplir correctement la ligne commençant par Fingerprint. Cette ligne est ajouté automatiquement lors de la génération du fichier, elle comporte des informations sur la cible testée : son adresse IP, le port ouvert utilisé, le port fermé utilisé (le port 0 est indiqué si ce port n'existe pas) et la date (format machine) à laquelle le test a été effectué. Exemple pour le fichier correspondant a l'OS FreeBSD 4.9 :

```
Fingerprint 131.254.200.7 53 0 1081951693
ESTABLI_(Resp=N)
ESTABLI_S(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
ESTABLI_A(Resp=Y%DF=Y%W=E240%ACK=O%Flags=A%Ops=NNT)
ESTABLI_SA(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
ESTABLI_F(Resp=N)
ESTABLI_SF(Resp=Y%DF=Y%W=E000%ACK=O%Flags=AS%Ops=MNWNNT)
ESTABLI_AF(Resp=Y%DF=Y%W=E240%ACK=O%Flags=A%Ops=NNT)
ESTABLI_SAF(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
ESTABLI_P(Resp=N)
ESTABLI_SP(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
ESTABLI_AP(Resp=Y%DF=Y%W=E240%ACK=O%Flags=A%Ops=NNT)
ESTABLI_SAP(Resp=Y%DF=Y%W=E000%ACK=O%Flags=AS%Ops=MNWNNT)
ESTABLI_FP(Resp=N)
ESTABLI_SFP(Resp=Y%DF=N%W=0%ACK=O%Flags=AR%Ops=)
ESTABLI_AFP(Resp=Y%DF=Y%W=E240%ACK=O%Flags=A%Ops=NNT)
ESTABLI_SAFP(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
```

Ici on change donc la première ligne en : Fingerprint FreeBSD 4.9. Puis on rassemble tous les fichiers de résultats en un seul fichier :

```
cat generate_tests_Linux2.4 generate_tests_Linux2.6
generate_tests_FreeBSD4.9 ... generate_tests_Windows2000
 >>  generate_tests_fingerprint
```

6.5 Test de Nmap-Stateful

On a fini la construction des tests et du fichier d'empreintes, nous allons maintenant pouvoir tester la pertinence des tests. La méthodologie consiste à lancer notre suite de tests sur des cibles différentes des machines ayant permis de construire l'empreinte et de vérifier que les OS sont bien découverts.

Sur un autre Linux 2.4 que les deux ayant servis de référence, on trouve :

```
#./nmap-stateful -n -P0 -sS --ntp 5 --nts 3 --otf generate_tests
--off generate_tests_fingerprint -p 80 x.x.x.x
Interesting ports on x.x.x.x:
PORT   STATE SERVICE
80/tcp open  http
OS details: Linux 2.4
```

Toujours sur un Linux 2.4, mais différent de ceux testés précedement, nous obtenons le résultat suivant :

```
80/tcp open  http
Aggressive OS guesses: Linux 2.6 (97%), Linux 2.4 (96%)
No exact OS matches for host (test conditions non-ideal).
```

Nmap-Stateful ne sait pas statuer exactement sur l'OS de la machine testée mais donne les possibilités les plus probables : Linux 2.6 et Linux 2.4. En réalité, l'OS est un Linux 2.4 mais avec une sous version plus récente que les versions ayant permis de faire l'empreinte présente dans le fichier de référence. Il faudrait donc affiner la base pour qu'elle prenne en compte les sous versions du noyau Linux.

Dans la plupart des cas, l'outil va être capable de reconnaître correctement l'OS de la machine cible, directement (comme dans le premier cas) ou indirectement (la plus forte probabilité).

L'outil peut parfois être mis en échec. Prenons le cas où la cible est un Solaris 8 :

```
#./nmap-stateful -n -P0 -sS --ntp 5 --nts 3 --otf generate_tests
--off generate_tests_fingerprint -p 80   x.x.x.x

ESTABLI_(Resp=N)
ESTABLI_S(Resp=N)
ESTABLI_A(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=AF%Ops=)
ESTABLI_SA(Resp=Y%DF=Y%W=0%ACK=0%Flags=R%Ops=)
ESTABLI_F(Resp=N)
ESTABLI_SF(Resp=N)
ESTABLI_AF(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=A%Ops=)
ESTABLI_SAF(Resp=N)
ESTABLI_P(Resp=N)
ESTABLI_SP(Resp=N)
ESTABLI_AP(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=A%Ops=)
ESTABLI_SAP(Resp=Y%DF=N%W=200%ACK=0%Flags=R%Ops=)
ESTABLI_FP(Resp=N)
ESTABLI_SFP(Resp=N)
ESTABLI_AFP(Resp=Y%DF=Y%W=16D0%ACK=0%Flags=A%Ops=)
ESTABLI_SAFP(Resp=N)
```

Cet échec est dû à la présence d'un Firewall qui perturbe les tests. L'empreinte au final dépend de l'OS et du Firewall et ne correspond plus à celle connue par Nmap-Stateful. Nous étudions cette problématique dans le paragraphe suivant.

7 Les Firewalls

L'action des Firewalls sur la prise d'empreinte est connue et Nmap souffre de ce problème aussi. Il y a plusieurs façons d'aborder ce problème, c'est à la

fois une difficulté qu'il va falloir intégrer à la problématique mais c'est aussi un apport car nous allons aussi pouvoir étudier les comportements des Firewalls et faire ainsi non plus de l'OS Fingerprinting mais aussi du Firewall Fingerprinting.
Cette étude débute, mais il existe des techniques qui devraient permettre de réaliser cet objectif. La méthode serait de trouver un grand nombre de cible ayant le même OS et de réaliser une empreinte de ce système pour un grand nombre de tests. Parmi ces cibles, il y aurait des cibles protégées par des Firewalls et d'autres non protégées. La technique serait de sélectionner ensuite les tests qui donnent toujours le même résultat indépendamment de la présence ou non d'un Firewall, ces tests nous permettraient de faire de l'OS Fingerprinting dans toutes les architectures réseau (contrairement à Nmap qui est sensible aux éléments de coupure (Firewall ou Syn-Relay)). Dans un deuxième temps on devrait sélectionner les tests qui donnent des résultats différents suivant le Firewall présent. Ces tests vont donc nous permettre de faire du Firewall Fingerprinting. Cette étude fait partie des travaux restant à réaliser dans ce domaine.

Néanmoins, notre premiére étude montre des résultats encourageants. Lors de la présentation de l'outil fingerprinttool, nous avions fait remarquer que l'on pouvait générer des tests avec des *sequence number* modifiés. Lors de l'essai de ce type de tests sur des cibles nous pouvons observer le phénomène suivant :

Sur un Solaris 8 protégé par un Firewall Stateful :

```
ESTABLI_AP_SEQ-3(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ-2(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ-1(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ0(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ1(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ2(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ3(Resp=T%DF=Y%W=832C%ACK=0%Flags=A%Ops=)
```

Toujours sur un Solaris 8 protégé par un autre type de Firewall Stateful :

```
ESTABLI_AP_SEQ-3(Resp=N)
ESTABLI_AP_SEQ-2(Resp=N)
ESTABLI_AP_SEQ-1(Resp=N)
ESTABLI_AP_SEQ0(Resp=T%DF=Y%W=8325%ACK=0%Flags=A%Ops=)
ESTABLI_AP_SEQ1(Resp=N)
ESTABLI_AP_SEQ2(Resp=N)
ESTABLI_AP_SEQ3(Resp=N)
```

Le test ayant permit de généré tous les tests est :

```
NAME=seq
ESTABLISHED
TH_PUSH
TH_ACK
SEQ=3
```

```
GEN_SEQ
NO_GEN_FLAGS
DATA=example
DATALEN=7
END
```

Le comportement est très différent d'une machine à l'autre, or ce sont les mêmes systèmes d'exploitation, c'est donc bien essentiellement l'empreinte du comportement du Firewall que nous avons. Une remarque est que bien que ce soit deux Firewalls Stateful, un seul d'entre eux vérifie les *sequences number* apparemment et ne laisse passer que les paquets respectant scrupuleusement les standards. Cela montre que l'étude des Firewalls Stateful est possible grâce à Nmap-Stateful et que cette étude peut apporter des éléments de compréhension intéressante dans ce domaine.

8 Conclusion et perspectives

Dans ce papier nous avons montré que les premiers résultats de l'outil sont encourageants et laissent entrevoir que l'outil va tenir ses promesses. L'OS Fingerprinting devrait être le premier domaine où l'outil devrait s'imposer, en effet Nmap-Stateful reprend l'ensemble des méthodes de Nmap et les tests utilisés par pOf peuvent aussi être repris. Une des difficultés qui reste à surmonter est de disposer de suffisamment de plateforme de tests pour pouvoir mettre au point des tests pertinents car le nombre de tests possibles est très grand ainsi que le nombre de combinaisons de cibles possibles (OS et Firewall). La problématique de l'OS Fingerprinting ne relève plus du domaine de TCP/IP mais de la gestion d'un grand nombre d'information.

Références

1. O. Courtay, site de Nmap-Stateful, `http://home.gna.org/nmapstateful/`
2. O. Courtay, O. Heen et F. Veysset, Détection des systèmes d'exploitation avec Cron-OS, *SSTIC 2003*.
3. O. Arkin, F. Yarochkin et M. Kydyraliev, The Present and Future of Xprobe2 - The Next Generation of Active Operating System Fingerprinting, `http://www.sys-security.com/archive/papers/Present\_and\_Future\_Xprobe2-v1.0.pdf`
4. M. Zalewski, L'outil d'OS Fingerprinting passif pOf, `http://lcamtuf.coredump.cx/p0f.shtml`
5. D. Song, Libdnet, `http://libdnet.sourceforge.net/`
6. Fyodor, Remote OS detection via TCP/IP Stack Fingerprinting, Phrack 1998, `http://www.insecure.org/nmap/nmap-fingerprinting-article.txt`
7. Fyodor, Le site de Nmap, `http://www.insecure.org`.
8. R. Stevens, *TCP/IP Illustrated*, Addison-Wesley, 1993.