Rapport projet SE

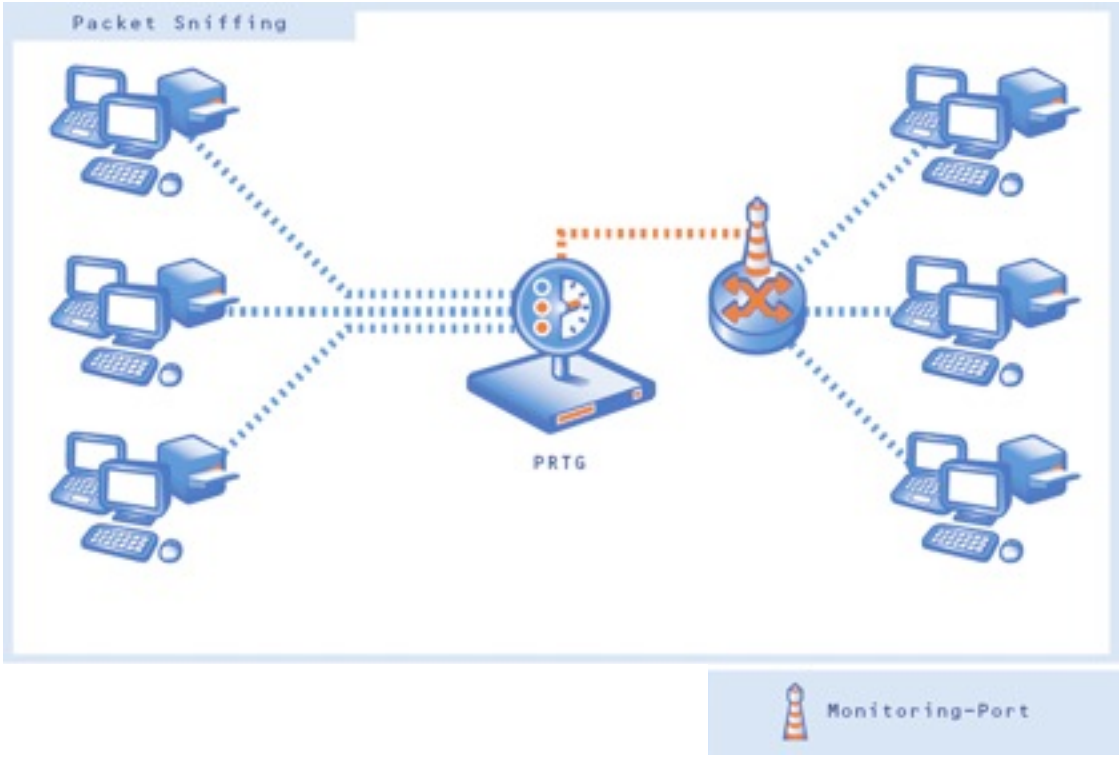
Compilation d'un sniffer sur un routeur wifi

Alexis Aulery, Benjamin Logie, Pierre Martin, Anthony Maupetit
Rapport S9 • ENSEIRB-MATMECA • 19 janvier 2012

Présentation du sujet

Déploiement d'une sonde réseau embarquée sur un routeur IP

CONTEXTE :



Actuellement plusieurs routeurs et bornes sans fil 802.11 peuvent embarquer des OS de type Linux. Les projets Open-WRT et DD-WRT sont deux exemples parmi d'autres, qui permettent de remplacer l'OS original d'un routeur par un OS Linux minimaliste. Avec cette ouverture des systèmes d'exploitation des routeurs, différentes fonctionnalités avancées non disponibles dans les firmwares d'origines peuvent être implantées.

Le but de ce projet est d'identifier, de comparer, et d'installer une ou plusieurs sondes réseaux sur un routeur sans fil Linux afin d'inspecter le trafic qui passe par le routeur. La technique Deep Packet Inspection (DPI) est largement utilisée dans les réseaux d'entreprises et récemment sur Internet en France (depuis la loi Hadopi) pour avoir une visibilité sur le trafic passant par les infrastructures réseaux. Cette technique est capable de reconnaître plusieurs protocoles et attributs protocolaires pour retranscrire le plus fidèlement possible l'activité du réseau.

OBJECTIFS DU PROJET :

L'intérêt de ce projet est de maîtriser les systèmes d'exploitation Linux embarqués sur routeur sans fil et la manipulation de compilation croisée pour les architectures ARM. Les étapes nécessaires à la création des firmwares, la mise à jour des firmwares,l'installation et la suppresion de logiciels sur routeurs, la compilation croisée (Cross-compilation) seront largement abordées durant ce projet.

L'objectif étant d'identifier un outil pertinent pour mettre en place la technique DPI afin d'afficher les statistiques d'utilisation de la bande passante d'un routeur, de remonter des alarmes automatiques lorsqu'un protocole est utilisé (exemple alarme lorsque le protocole P2P est utilisé) ou tout simplement pour surveiller le trafic passant par le routeur. Plusieurs outils simples existent sur un système Linux complet : tcpdump avec scripts, le filtrage L7, Snort, etc. Le travail demandé durant ce projet, est d'identifier l'outil qui permet d'atteindre l'objectif fixé, de le déployer, et de mettre en place quelques scripts et scénarii de tests.

Table des matières

Choix du routeur WIFI

Linksys WRT54GL Vs Asus WL-500G Premium V1

Pour réaliser la mise en place de la sonde, il nous été offert plusieurs possibilités quant au choix du routeur. En effet, nous avions à notre disposition deux routeurs de marques différentes : le Linksys WRT54GL et le Asus WL-500G Premium V1. Dans cette première partie, nous allons donc regarder les avantages et les inconvénients de chacun.

LINKSYS WRT54GL

Le WRT54GL est un routeur produit par Linksys (société Cisco). Il permet de partager une connexion Internet (modem non intégré) vers des ordinateurs via 4 ports ethernet et une liaison Wi-Fi.

Son ancêtre étant le WRT54G, le WRT54GL est connu pour son firmware sous licence GNU GPL. Cela a permis à de nombreux développeurs de la communauté du logiciel libre de proposer des firmwares alternatifs, comme DD-WRT ou OpenWRT, proposant plus de fonctionnalités et plus de stabilité.

Routeur Linksys WRT54GL

À partir de la version 5, le noyau Linux du WRT54G fut remplacé par un noyau VxWorks propriétaire non modifiable, moins gourmand en mémoire et en ressources, permettant de diviser par 2 la mémoire vive et la mémoire cache utilisées, et donc de réduire les coûts de production.

Linksys n'a pas pour autant oublié les fans de Linux. La société propose donc le WRT54GL en sus du WRT54G, estampillé d'un "L" pour Linux, qui devrait perpétuer la descendance du WRT54G.

Notre objectif étant en parti d'implémenter une distribution Linux afin d'y rajouter des outils, se premier routeur semble particulièrement approprié.

Spécification du produit :

On sait déjà que les deux distributions Linux les plus répandues dans les domaines des routeurs sont OpenWRT et DD-WRT. On regarde donc les spécifications techniques de chacun des produits avec en plus un critère pour la compatibilité de l'OS.

Routeur	Linksys WRT54GL
Chipset	Broadcom 5352
Taille Mémoire Flash	4 MB
Taille Mémoire RAM	16 MB
Fréquence CPU	200 MHz
Consommation	5,9 W
Prise USB (1)	Non
Compatibilité OpenWRT	Oui
Compatibilité DD-WRT	Oui
Prix (2)	≈50€

(1) : La prise USB à une importance car elle permet une extension considérable de la taille mémoire disponible (via une clé par exemple).

(2) : Le prix n'a qu'une valeur indicative et ne fait que refléter les tendances d'internet

ASUS WL-500G PREMIUM V1

Asus est le concurrent direct de Cisco dans le domaine des routeurs. Généralement les routeurs Asus sont développés autour d'un Chipset Broadcom et ajoute la possibilité d'utiliser un ou plusieurs port USB. Cependant, le firmware de base est très souvent critiqué pour son manque de stabilité. Néanmoins la majeur partie des OS Linux sont compatibles avec les routeurs Asus permettant alors d'allier performances du software et du hardware.

Le routeur Asus WL-500G Premium ne déroge pas à la règle, il permet de partager une connexion Internet vers des ordinateurs via 4 ports ethernet et une liaison Wi-Fi.Il dispose de nombreux autres ports tels que le miniPCI, ou encore deux baies Wi-Fi.

Une nouvelle version du routeur a pris la relève, c'est le Asus WL-500G Premium V2.

Connectiques routeur Asus WL-500G Premium V1

Spécification du produit

On peut alors se pencher sur la spécification technique du produit :

Routeur	Asus WL-500G Premium V1
Chipset	Broadcom 4704
Taille Mémoire Flash	8 MB
Taille Mémoire RAM	32 MB
Fréquence CPU	264 MHz
Consommation	6,9W
Prise USB (1)	Oui (x2)
Compatibilité OpenWRT	Oui
Compatibilité DD-WRT	Oui
Prix (2)	≈60€

(1) : La prise USB à une importance car elle permet une extension considérable de la taille mémoire disponible (via une clé par exemple).

(2) : Le prix n'a qu'une valeur indicative et ne fait que refléter les tendances d'internet

CONCLUSION

Au vu des caractéristiques techniques de chacun des routeurs et comme les comptabilités avec les OS sont les mêmes on peut donc directement choisir. En effet, le routeurs Asus semble non seulement plus puissant, mais dispose en plus d'avantage de mémoire et propose même la possibilité d'utiliser des périphériques USB. Le Linksys semble pourtant avoir comme avantage d'être moins couteux et d'avoir une puissance requise plus faible que celle de l'Asus. Dans notre cadre, comme nous disposons déjà des deux routeurs (pas besoin de les racheter) et comme la consommation n'est pas notre problème principal, nous avons donc choisi de réaliser la suite de notre projet avec le routeur WL-500G Premium V1 de chez Asus.

Routeur Asus WL-500G Premium V1

Choix de l'OS Linux

OpenWRT Vs DD-WRT

Les routeurs du commerce possèdent souvent leur propre OS afin de répondre aux attentes du client, mais comme pour bon nombre de systèmes embarqués, rapidement des aficionados ont voulu créer une distribution Linux à destination des routeurs. Plusieurs projets ont été lancés et il sont très évolués. A tel point, que l'on peut considérer les plus connus comme étant plus stables et plus performants que la plupart des distributions propriétaires du marché.

Dans le petit nombre des plus connus : FreeWRT, Tomato Firmware, DebWRT, OpenWRT et DD-WRT. Dans cette partie, nous nous intéresserons plus particulièrement à ces deux derniers qui semblent à l'heure actuelle les plus répandus et les plus utilisés par les Linuxiens.

OPENWRT

Logo OpenWRT

OpenWrt est une distribution GNU/Linux minimaliste pour matériel embarqué (Routeurs, Tablettes, Téléphones ...).

Historiquement développée pour remplacer le firmware des routeurs basés sur des System-on-Chip Broadcom, par exemple les routeurs WLAN de Asus, Belkin, Dell, Linksys, US Robotics, Viewsonic. OpenWRT est une distribution mais possède aussi tous les outils de Buildroot permettant une configuration très aisée de la distribution. De plus, elle permet de gérer les packages (IPKG/OPKG) a postériori et donc de ne pas avoir à re-compiler tout le noyau à chaque fois que l'on souhaite rajouter une fonctionnalité.

OpenWRT est un système d'exploitation qui peut être configuré de façon assez poussé mais qui nécessite tout de même quelques compétences dans la compilation de noyau et dans les distributions Linux en générale.

DD-WRT

Logo DD-WRT

DD-WRT est un système d'exploitation OpenSource basé sur Linux. Il est approprié pour une grande variété de routeurs WLAN et autres systèmes embarqués. Le principe fondamental du projet est de fournir un outil le plus simple possible à utiliser. Cependant bien que très facile à prendre en main (interface graphique) et un grand nombre de fonctionnalités disponibles, il faut nuancer la grande quantité de fonctionnalités par le fait que celles-ci ne peuvent pas être ajoutées après avoir compilé le noyau ou sur un noyau pré-existant.

Cet handicape est relativement important pour nous car nous ne voulions pas perdre trop de temps dans de multiples compilations.

Accessoirement, il est apparu en 2009 une faille dans le système (mise en avant par un Hongrois). Si un patch correctif a été mis en place très rapidement, il n'en reste pas moins que cela crée un mauvais apriori.

AUTRE OS (PACKETPROTECTOR) :

Il existe de nombreux autres système d'exploitation qui sont développés autour des routeurs. On peut citer FreeWRT, Tomato Firmware, DebWRT ou encore PacketProtector. En générale ils restent très proches les uns des autres, on peut donc sans trop de regrets ne pas les aborder plus en profondeur.

En revanche, on peut regarder de plus près PacketProtector. Comme évoqué précédemment PacketProtector n'est pas réellement un système original. En effet celui-ci repose avant tout sur OpenWRT en gardant le principaux atouts de celui-ci.

Il est important de l'évoquer car il est entièrement tourné vers la sécurité réseaux et possède donc de base des fonctionnalités très intéressantes dans le cadre de notre projet. En effet par exemple, contrairement à PacketProtetor, OpenWRT ne possède pas (ni de base ni en package) la version snort-inline et il en est de même pour DD-WRT.

Cependant un désagrément de PacketProtector est qu'il est nécessaire de posséder une grande mémoire (ajout d'une clé USB de 120MB obligatoire).

CONCLUSION

Afin de tester chacun des OS (DD-WRT, OpenWRT et PacketProtector), nous les avons tous implémentés sur un routeur pour les tester. DD-WRT s'est avéré trop léger : pas de possibilité d'évolution du noyau avec d'autres fonctionnalités et une obligation de le re-compiler à chaque fois.

PacketProtector a pu être implémenté sur l'Asus (grâce au port USB) et s'est révélé être plus compliqué à prendre en main. En effet malgré le nombre impressionnant de fonctionnalités pour la gestion des réseaux, les outils de base sont plus compliqués à utiliser. Par exemple snort (version classique) est automatiquement lancé au démarrage ce qui rend la gestion plus difficile. La mémoire est également remplie plus rapidement.

Aussi pour toutes les raisons évoquées ci dessus mais aussi du fait que la documentation et les tutoriels le rendent très facile à prendre en main, nous avons choisi d'utiliser par la suite la distribution OpenWRT.

Il nous a alors fallu choisir entre trois versions disponibles pour celle-ci (kamikaze, whiterussian, backfire). Backfire a été exclue de base car elle ne possède pas de package snort compatible. Ensuite nous avons décidé de nous tourner vers kamikaze car, étant plus ancienne, nous sommes avons pensé qu'elle était plus stable que whiterussian.

Choix du sniffer

Tcpdump Vs Snort

Un sniffer, est un outil capable de récupérer les données qui transitent sur un réseau. Il existe de nombreux programmes capables de réaliser cette fonctionnalité, Ettercap, etherdetect, etc... Parmi tous ces programmes, deux en particulier attirent l'attention, tcpdump et snort. Tcpdump étant l'outil classique d'analyse réseau et snort qui offre des possibilités de configuration plus poussées.

TCPDUMP

Tcpdump est un Packet sniffer, c'est-à-dire un logiciel qui permet de lire les données qui transitent sur le réseau qu'il scrute. Il s'utilise en ligne de commande et permet l'affichage en temps réel des informations qui transitent sur le réseau ou la sauvegarde de ces informations pour une analyse ultérieure. Tcpdump est un logiciel de bas niveau qui permet grâce à la bibliothèque libpcap de connaître précisément ce qui se passe sur le réseau et de prendre en conséquence les décisions qu'il faut pour maintenir un fonctionnement stable. Libpcap est une bibliothèque qui fournit des outils de capture des paquets, ceci même s'ils ne sont pas destinés à l'ordinateur hôte.

Tcpdump est donc un outil très puissant compatible avec la majorité des systèmes unix qui permet grâce à de nombreuses options d'analyser de manière plus ou moins fine un flux d'informations.

Il autorise, entre autre, le filtrage des paquets quant à la provenance, leur adresse de destination, le port source, le port de destination, le protocole utilisé, etc...

Il existe plusieurs logiciels tels que wireshark qui offrent une interface graphique et permettent une utilisation plus intuitive de tcpdump.

SNORT

Snort est un système réseau open source de détection et de prévention d'intrusion. Il est capable d'effectuer en temps réel des analyses de trafic et de logger les paquets sur un réseau IP. Il peut effectuer des analyses de protocole, de contenu. Il est également capable de détecter des attaques sur le réseau.

Snort peut effectuer ces analyses en se basant sur des règles qui sont développées par la communauté ou par une personne gérant le réseau ; ceci pour répondre au besoin spécifique à chaque projet.

Pour pouvoir fonctionner sous Linux, Snort requiert l'installation préalable de libpcap, libpcre et de libnet.

On voit donc que Snort est un outil plus performant que tcpdump. Il fournit la même possibilité d'analyse de paquets mais il offre en surplus la génération d'alertes en cas de détection d'intrusion voir même le rejet de paquets jugés malsain si Snort est utilisé en mode IPS (Intrusion Prevention System). Il faut noter toutefois que le projet snort-inline qui correspond au mode IPS a été remplacé par le projet Suricata.

Sachant que nous pouvions implémenter Snort sur les sondes, nous avons décidé de travailler avec cet outil qui nous semblait plus performant tout en restant facile d'utilisation.

Mise en oeuvre

Compilation de Open-WRT avec snort pour le routeur Asus

INSTALLATION DE OPENWRT SUR LE ROUTEUR ASUS

Dans un premier temps, assurons-nous de pouvoir communiquer avec le routeur. L'adresse ip du routeur est 192.168.1.1 par défaut, l'adresse ip du pc hôte doit donc être reconfigurée de la forme 192.168.1.* avec un netmask à 255.255.255.0 :

```
host> ifconfig eth0 192.168.1.100 netmask:255.255.255.0
```

Il faut maintenant télécharger le noyau que l'on souhaite installer à l'adresse http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx où l'on récupère l'image du noyau « openwrt-brcm47xx-squashfs.trx ». Cette version «Kamikaze» a été choisie après avoir consulté le forum d'OpenWRT où il est apparu que les utilisateurs trouvaient cette version plus stable.

Avant de pouvoir communiquer avec le routeur, il faut que celui-ci soit en «diag mode». Pour cela :

- Débrancher l'alimentation du routeur
- Vérifier que l'IP du PC hôte est bien comprise dans la plage 192.168.1.0/24
- Connecter le PC hôte au port LAN1
- Appuyer longuement sur le bouton «RESTORE»
- Tout en continuant d'appuyer sur «RESTORE», rebrancher l'alimentation du routeur
- Si la LED clignote doucement, c'est que le routeur est en «diag mode»

On peut vérifier si le dialogue est possible en lançant un «ping» sur le routeur.

```
host> ping 192.168.1.1
```

Si le ping a réussi, l'installation peut commencer, sinon reprendre les étapes précédentes.

L'installation se fait ensuite en suivant un protocole tftp. Pour cela, il faut entrer les commandes suivantes :

```
host> tftp 192.168.1.1
tftp> binary
tftp> trace
tftp> put openwrt-brcm47xx-squashfs.trx
```

Une fois que l'upload est terminée, attendre au moins 6 minutes que le flash se fasse.

Le routeur devrait ensuite rebooter automatiquement (s'il ne le fait pas, faites le manuellement en débranchant puis rebranchant l'alimentation).

L'installation d'OpenWRT est terminée. Un ping peut être lancé pour vérifier que la communication n'a pas été perdue. Si le ping réussit, Il devrait être possible de lancer un «telnet» pour se connecter au routeur.

```
host> telnet 192.168.1.1
```

Un mot de passe est alors demandé pour «root». Une fois que le mot de passe a été choisi, OpenWRT devrait se lancer normalement.

Il y a désormais 2 méthodes pour accéder au routeur :

1. En utilisant le protocole ssh via un terminal :

```
host> ssh root@192.168.1.1
```

Cela permet d'accéder à l'arborescence du système de fichier d'OpenWRT et de naviguer avec les commandes habituelles via le terminal.

2. En utilisant l'interface web d'OpenWRT depuis votre navigateur web préféré en tapant «http://192.168.1.1/» dans la barre d'adresse. Cette interface propose un grand nombre d'informations et de menus de configurations intéressants (informations sur le systèmes, historique des connexions, gestion des packages, mise à jour du noyau...).

INSTALLATION DE SNORT ET TCPDUMP

Afin de pouvoir installer les packages «Snort» et «tcpdump», il est nécessaire de pouvoir accéder à internet depuis le routeur. Pour cela il suffit de brancher le câble donnant l'accès à internet au port «WAN» du routeur. Dans notre cas il faut effectuer d'autres configurations afin de pouvoir accéder au serveur DNS de notre salle et se servir du réseau de l'école (configuration du proxy). Il a donc fallu entrer les commandes suivantes dans le shell du routeur :

```
asus> export http_proxy=«http://proxyng.enseirb-matmeca.fr:3128»
asus> route add default gw 172.17.0.254   //adresse du serveur DNS
asus> vi /etc/resolv.conf
```

Cette dernière commande nous ouvre un fichier où l'on rentre uniquement «nameserveur 172.17.0.1». Le seul moyen d'éditer un texte depuis le routeur est d'utiliser vi.

Il devrait désormais être possible de télécharger les packages :

-Snort :

```
asus>wget  http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx/packages/snort_2.4.4-1_mipsel.ipk
```

-tcpdump :

```
asus>wget  http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx/packages/tcpdump-mini_4.0.0-1_mipsel.ipk
```

Ces deux outils nécessitent l'installation d'autres packages en pré-requis :

-Libneto :

```
asus>wget http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx/packages/libneto_1.0.2a-7_mipsel.ipk
```

-Libpcap :

```
asus>wget  http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx/packages/libpcap_0.9.8-1_mipsel.ipk
```

-Libpcre :

asus>wget http://downloads.openwrt.org/kamikaze/8.09.2/brcm47xx/packages/libpcre_7.6-2_mipsel.ipk

Il suffit maintenant d'installer ces packages :

```
asus> opkg install libpcre_7.6-2_mipsel.ipk
asus> opkg install libpcap_0.9.8-1_mipsel.ipk
asus> opkg install libnet0_1.0.2a-7_mipsel.ipk
asus> opkg install snort_2.4.4-1_mipsel.ipk
asus> opkg install tcpdump-mini_4.0.0-1_mipsel.ipk
```

Il est désormais possible d'utiliser Snort et tcpdump.

Remarque : on peut accéder à internet depuis le pc hôte via le routeur asus en entrant la commande suivante depuis un terminal :

```
host>route add default gw 192.168.1.1
```

Les méthodes de flash et d'installation des packages présentées ci-dessus sont celles que nous avons utilisées, mais ce n'est pas l'unique façon de procéder.

INSTALLATION DE LA DISTRIBUTION SUR LE ROUTEUR

La plupart des distributions développées autour des routeurs possèdent deux méthodes principales d'installation : via une interface web ou par un protocole tftp.

- En effet, chacune de ces distributions propose, une fois installée, une interface web qui donne accès à différentes options plus ou moins développées selon le projet. Cette interface web permet notamment d'installer une nouvelle distribution sur le routeur.

  L'avantage de l'utilisation de cette interface web est que l'installation en est grandement simplifiée.

  L'inconvénient majeur est que cela permet seulement d'installer des distributions issues du même projet (mise à jour, ...). Par exemple un routeur équipé d'une version d'OpenWRT ne pourra installer qu'une autre version d'OpenWRT via cette interface web.

  Un second inconvénient est qu'une connexion internet correctement configurée est nécessaire pour récupérer l'image de la distribution que l'on souhaite installer.

- Un exemple d'installation par un protocole tftp est l'installation présentée ci-dessus.

  L'avantage de ce type d'installation est qu'elle peut être effectuée quelque soit l'état actuel du routeur (flashé ou non, par une distribution du même projet ou non) et qu'elle ne nécessite pas de connexion internet, il suffit simplement de posséder l'image de la distribution que l'on souhaite installer, et que le routeur soit prêt à recevoir des paquets.

  L'inconvénient est que la mise en écoute du routeur pour qu'il puisse recevoir des paquets se fait de façons différentes selon le routeur. Ainsi pour chaque routeur il faut apprendre cette méthode, qui peut parfois se révéler complexe à mettre en œuvre.

Prenons l'exemple des routeurs que nous avions à disposition :

-les Lynksys ne passent en mode écoute que quelques secondes durant le boot. Il faut donc arroser le routeur de requêtes (ping) et lorsqu'il y a une réponse, on peut envoyer l'image de la distribution dans le routeur. Cette méthode est quelque peu périlleuse et ne garanti pas le succès.

-l'Asus est plus souple, lors du boot il est aisé de le mettre en "diag mode" comme indiqué ci-dessus. Cette simplicité du flash nous a conforté dans notre choix d'utilisation de l'Asus.

INSTALLATION DES PACKAGES SUR LE ROUTEUR

Encore une fois, il existe 2 méthodes pour installer les packages. Ces deux méthodes nécessitent une connexion internet :

- L'installation de nouveaux packages fait parti des options proposées par l'interface web de la plupart des distributions.

- La seconde méthode est celle présentée dans la partie précédente : récupération du package puis gestion opkg.

Au final l'utilisateur expérimenté préférera certainement se passer de l'interface web pour effectuer ces tâches, puisque le passage par un terminal donne toujours accès à plus d'options et de paramétrages au travers des commandes. Mais l'interface web n'est pas à négliger pour autant, surtout pour l'utilisateur débutant qui y trouvera parfois une multitude d'options et d'informations intéressantes qui lui faciliteront la vie.

Utilisation de Snort

MODE DE FONCTIONNEMENT

La sonde Snort a trois modes de fonctionnement :

-le mode sniffer qui affiche à l'écran du terminal les informations des paquets qui transitent,

-le mode logger qui stock les paquets qui transitent,

-le mode NIDS (Network Intrusion Detection System) qui permet de faire un traitement du trafic.

Le mode SNIFFER:

Ce mode affiche à l'écran en temps réelle les informations des paquets qui sont en cours de transit. Grace aux options on peut choisir le niveau de détails que l'on souhaite.

Pour afficher les entêtes:

```
./snort -v
```

Pour afficher les données des paquets:

```
./snort -d
```

Pour afficher plus de détails (data link layer headers)

```
./snort -e
```

Toutes combinaisons de ces options fonctionnent bien entendu, ex:

```
./snort -vde
```

Le mode LOGGER:

Ce mode recopie les paquets dans des fichiers.

Pour stocker les paquet on utilise l'option: `-l ./log` où ./log est le dossier cible.

Pour stocker toutes les informations relatives aux paquets on utilise:

```
./snort -dev -l ./log
```

Le fait de préciser le dossier cible lance automatiquement le mode logger.

D'autres options existent avec ce mode.

Pour stocker seulement les paquets relatif à notre sous-réseau on utilise l'option -h pour préciser la source des paquets à stocker, par ex:

```
./snort -dev -l ./log -h 192.168.1.0/24
```

Il est possible de stocker les informations au format tcpdump avec l'option -b:

```
./snort -l ./log -b
```

Une fois stocké au format tcpdump le binaire peut être relu par snort avec l'option playback -r et soumit à n'importe quel mode de snort

Pour afficher le contenu d'un fichier de log on peut utiliser:

```
./snort -dv -r packet.log
```

Le mode NIDS:

Ce mode permet un traitement personnalisé du trafic. Il y a énormément d'options de personnalisation proposées et il serait très compliqué et peu pertinent de tout décrire ici. Nous allons nous contenter d'usages classiques du logiciel.

La personnalisation se fait à partir d'un fichier .conf, pour lancer snort dans ce mode on utilise l'option -c, on peut par exemple utiliser:

```
./snort -l ./log  -c snort.conf
```

Si aucun dossier de stockage est donné il se fait par défaut dans /var/log/snort

Il est conseillé de ne pas utiliser les options -v -e et -d si on scrute longtemps le trafic car l'affichage à l'écran est lent et provoquerait la perte de certain paquets.

Le fichier snort.conf:

Le fichier snort.conf contient la configurations de la sonde snort. On peut trouver dans ce fichier, entre autres, la déclaration de toutes les variables d'environnement qui seront réutilisées dans les règles ($LOCAL_HOST, $HTTP_PORT, etc...), mais aussi les chemins vers les fichiers de règles que l'on veut soumettre au trafic.

Voici un exemple de déclaration de variables d'environnement qui seront réutilisées dans la majorité des règles:

```
# Setup the network addresses you are protecting
ipvar HOME_NET any

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET
```

Pour ajouter des fichiers de règles on peut utiliser les lignes de codes suivantes:

```
include $RULE_PATH/local.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/blacklist.rules
```

Principe d'utilisation du mode:

Le filtrage du trafic autorise plusieurs actions dont: - alert- log

L'action alert est l'utilisation principale de ce mode. Elle permet de générer des fichiers qui pourront être relu plus tard ou d'alerter en temps réelle un utilisateur ou un programme qui scrute une socket.

Nous pouvons noter sept façons de traiter les alertes dont sx peuvent être données en ligne de commande lors du lancement de snort. Si un des modes est donné lors du lancement il est prioritaire sur les modes donnés dans les règle. Les six modes sont:

- -A fast : «Fast alert mode. Writes the alert in a simple format with a timestamp, alert message, source and destination IPs/ports.»
- -A full : «Full alert mode. This is the default alert mode and will be used automatically if you do not specify a mode.»
- -A unsock : «Sends alerts to a UNIX socket that another program can listen on.»
- -A none : «Turns off alerting.»
- -A console : «Sends fast-style alerts to the console (screen).»
- -A cmg : «Generates cmg style alerts.»

Il y a également un moyen de stocker en binaire les alertes sur le routeur, ex:

```
./snort -b -A fast -c snort.conf
```

L'alert n'est pas la seule action permise par le mode NIDS mais cela reste sa vocation.

Malheureusement la version de base de snort ne permet pas toutes les actions nommées, en effet le drop, sdrop et reject ne sont pas permis à la base. Il faut utiliser snort en mode INLINE qui nécessite des packages supplémentaires.

ÉCRIRE UNE RÈGLE

Les règles sont toutes les actions qui vont être appliquées à chaque paquet. Les règles s'écrivent selon une syntaxe:

action protocol IP1 PORT1 sens IP2 PORT2 (option1;option2;)

ACTION:

1. alert - generate an alert using the selected alert method, and then log the packet
2. log - log the packet
3. pass - ignore the packet
4. activate - alert and then turn on another dynamic rule
5. dynamic - remain idle until activated by an activate rule , then act as a log rule
6. drop - block and log the packet
7. reject - block the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
8. sdrop - block the packet but do not log it.

PROTOCOL:

TCP, UDP, ICMP, and IP.
Dans le future développement de snort on espère pouvoir détecter aussi:
ARP, IGRP, GRE, OSPF, RIP, IPX, etc.

IP:

L'adresse IP peut être unique mais aussi une liste [IP1,IP2] voir une plage d'adresses IP/24
L'opérateur «!» peut être utilisé pour décrire l'inverse de l'élément.

PORT:

Tout comme pour l'adresse le port peut être unique, une plage fermée 25:685 ou une plage ouverte :6000
L'opérateur «!» peut être utilisé pour décrire l'inverse de l'élément.

SENS:

Le sens est un opérateur qui défini si la règle s'applique pour un paquet qui va d'une certaine source vers sa destination IP_SRC PORT_SRC -> IP_DST PORT_DST ou s'il s'applique

pour les paquets qui transitent dans n'importe quel sens IP1 PORT1 <> IP2 PORT2. A noter que <- n'existe pas.

OPTION:

Les options vont permettre 2 choses:

- préciser les conditions d'adéquation du paquet à la règle par rapport à son contenue.
- préciser l'action à faire au paquet et les caractéristique du paquet détecté.

Les options d'une même règle sont écrite entre ( ) et un ; est nécessaire à la fin de chacune d'elle (même s'il n'y en a qu'une). Le caractère \ peut être utilisé pour aller à la ligne sans terminer la règle ou pour utiliser des caractères spéciaux.

Précision de la règle:

La précision de la règle se base sur le contenu du paquet grâce à l'option content:

```
-content:"<contenue>"
```

Il existe de nombreuses options pour préciser des tailles, des types de contenu, etc ...

Précision de l'action:

La précision de l'action permet de donner des informations supplémentaires lorsqu'un paquet est détecté.

`-msg:"<message>";` envoie le message spécifié pour l'action donnée

`-classtype:attack_type;` précise à la sortie le type d'attaques détectées. Les types existants sont classés dans un tableau par ordre de priorité.

De nombreuses autres options permettent de donner des informations supplémentaires sur le type de paquets détecté.

Voici un exemple de règle:

```
alert tcp any 8080 -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:"mountd access";)
```

Ici la règles se traduit par:

Si :
-le paquet a pour source: toutes IP provenant du port 8080

-le paquet a pour destination: toutes IP de 192.168.1.0 ‡ 192.168.1.255 du port 111
-il contient la chaine en hexadécimal "00 01 86 a5"

Alors créer une alerte avec le message "mountd access"

De manière général les adresses et les ports sont nommés par des variables d'environnement ex: $LOCAL_HOST

EXEMPLE D'UTILISATION

Afin d'illustrer l'utilisation de snort nous avons effectué des tests à partir de règles simple.

```
alert tcp any any <> any 111 (msg:"alert detection mot google";\ content:"google";nocase; )
alert udp any any -> any any (msg:"detection traffic UDP" ;)
log tcp any 22 -> any 22
```

Apres quelques minutes d'utilisation de snort sur le routeur nous l'arrêtons pour observer les résultats. Il affiche alors des statistiques sur ce qu'il a analysé, en voici un exemple

```
===============================================================================

Snort received 40177 packets
    Analyzed: 20407(50.793%)
    Dropped: 19770(49.207%)
===============================================================================
Breakdown by protocol:
    TCP: 20187      (50.245%)
    UDP: 127        (0.316%)
   ICMP: 0          (0.000%)
    ARP: 39         (0.097%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
ETHLOOP: 0          (0.000%)
    IPX: 0          (0.000%)
   FRAG: 0          (0.000%)
  OTHER: 30         (0.075%)
DISCARD: 0          (0.000%)
===============================================================================
Action Stats:
ALERTS: 401
LOGGED: 2806
PASSED: 0
===============================================================================
```

Les alertes et les paquets stockés sont situés dans le dossier /var/log/snort.

```
root@OpenWrt:/tmp/log/snort# ls -l
-rw-------    1 root     root        187968 Jan 26 19:38 alert
-rw-------    1 root     root         90692 Jan 26 15:30 snort.log.1264519805
-rw-------    1 root     root         88764 Jan 26 15:44 snort.log.1264520671
-rw-------    1 root     root         84876 Jan 26 17:47 snort.log.1264528016
-rw-------    1 root     root          7216 Jan 26 17:52 snort.log.1264528301
-rw-------    1 root     root       1866282 Jan 26 19:38 snort.log.1264534459
```

Le fichier alert contient les messages d'alerte générés par les règles ainsi que les entêtes paquets correspondants. Les fichiers de log sont dénommé snort.log. A chaque utilisation de snort un nouveau fichier de log est créé mais la fichier alert lui est mis à jour avec les nouvelles alertes à la suite des anciennes. Voici un exemple de fichier alert et de fichiers de log.

```
[Priority: 0]
01/26-19:38:37.866061 172.17.0.96:39372 -> 147.210.19.249:3128
TCP TTL:63 TOS:0x0 ID:65050 IpLen:20 DgmLen:1071 DF
***AP*** Seq: 0x1CD1D116  Ack: 0x8E084B52  Win: 0x26A5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 577935 1063723518

[**] [1:0:0] alert detection mot google [**]
[Priority: 0]
01/26-19:38:42.143602 192.168.1.127:39372 -> 147.210.19.249:3128
TCP TTL:64 TOS:0x0 ID:65090 IpLen:20 DgmLen:550 DF
***AP*** Seq: 0x1CD22627  Ack: 0x8E089815  Win: 0x26A5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 579005 1063724464

[**] [1:0:0] alert detection mot google [**]
[Priority: 0]
01/26-19:38:42.143909 172.17.0.96:39372 -> 147.210.19.249:3128
TCP TTL:63 TOS:0x0 ID:65090 IpLen:20 DgmLen:550 DF
***AP*** Seq: 0x1CD22627  Ack: 0x8E089815  Win: 0x26A5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 579005 1063724464

[**] [1:0:0] detection traffic udp [**]
[Priority: 0]
01/26-19:38:48.294487 172.17.0.67:57621 -> 172.17.0.255:57621
UDP TTL:128 TOS:0x0 ID:16689 IpLen:20 DgmLen:72
Len: 44
```

Voici un exemple de contenue de fichier d'alertes.

```
6D 20 70 72 6F 78 79 6E 67 2E 65 6E 73 65 69 72  m proxyng.enseir
62 2E 66 72 0D 0A 58 2D 43 61 63 68 65 2D 4C 6F  b.fr..X-Cache-Lo
6F 6B 75 70 3A 20 48 49 54 20 66 72 6F 6D 20 70  okup: HIT from p
72 6F 78 79 6E 67 2E 65 6E 73 65 69 72 62 2E 66  roxyng.enseirb.f
72 3A 33 31 32 38 0D 0A 56 69 61 3A 20 31 2E 31  r:3128..Via: 1.1
20 70 72 6F 78 79 6E 67 2E 65 6E 73 65 69 72 62   proxyng.enseirb
2E 66 72 3A 33 31 32 38 20 28 73 71 75 69 64 2F  .fr:3128 (squid/
32 2E 37 2E 53 54 41 42 4C 45 33 29 0D 0A 43 6F  2.7.STABLE3)..Co
6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61  nnection: keep-a
6C 69 76 65 0D 0A 50 72 6F 78 79 2D 43 6F 6E 6E  live..Proxy-Conn
65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69  ection: keep-ali
76 65 0D 0A 0D 0A                                ve....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

01/26-19:38:48.294487 172.17.0.67:57621 -> 172.17.0.255:57621
UDP TTL:128 TOS:0x0 ID:16689 IpLen:20 DgmLen:72
Len: 44
53 70 6F 74 55 64 70 30 61 23 DD 2F 92 68 B8 4C  SpotUdp0a#./.h.L
00 01 00 04 48 95 C2 03 6E F8 BA 32 2D F2 81 B0  ....H...n..2-...
E2 C7 FE 30 FF 0C 58 89 C8 52 0C 0E              ...0..X..R..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

01/26-19:38:48.311875 147.210.19.249:3128 -> 172.17.0.96:39370
TCP TTL:63 TOS:0x0 ID:34470 IpLen:20 DgmLen:675 DF
***AP*** Seq: 0x8D76C95C  Ack: 0x792689E6  Win: 0x7D3  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1063726145 580528
48 54 54 50 2F 31 2E 30 20 33 30 32 20 4D 6F 76  HTTP/1.0 302 Mov
65 64 20 54 65 6D 70 6F 72 61 72 69 6C 79 0D 0A  ed Temporarily..
```

Il est possible, grâces aux entêtes de paquets de retrouver un paquets stocké correspondant à l'alerte déclenchée. Avec l'option replay (-r) de snort on peut relire le contenue des fichiers de log.

Nous pouvons noter que les fichiers de log sont assez lourd sur le disque. 2800 paquets stockés font 1.8Mo. Il est donc nécessaire d'utiliser un moyen de stockage à distance des fichiers. Une solution est l'utilisation d'un client NFS (Network File System) sur le routeur qui stockera les données sur un serveur NFS distant. Avec ce procédé nous avons maintenant la possibilité d'effectuer un traitement complexe sur les paquets par une machine plus puissante.

SNORT-INLINE

Snort-inline est une version améliorée de snort. Elle offre, entre autres, une possibilité de traitement des paquets plus complète. De nouveaux types de règles sont disponibles avec Snort-inline : drop, sdrop et reject. Ces règles permettent respectivement :

- de rejeter et de mémoriser un paquet,
- d'uniquement rejeter un paquet
- de rejeter et de mémoriser un paquet tout en envoyant un message de refus à l'envoyeur.

Il faut noter que cette version de snort n'utilise pas libpcap mais reçoit les paquets via iptables et IPFW (ipfirewall) grâce à bibliothèque libipq, les analyse, puis indique à iptables et IPFW si les paquets doivent être acceptés ou non.

Il est également important de noter que le projet snort-inline a été abandonné au profit du projet Suricata .

Conclusion

Nous avons travaillé dans une optique verticale allant du choix du matériel à implémenter au choix et à l'utilisation de la sonde. Au terme du projet, nous avons réussi à mettre en place une analyse du trafic sur le réseau transitant par le routeur. Pour cela nous avons utilisé le logiciel snort pour lequel il a été créé quelques règles.

Ce projet nous a permis d'approfondir nos compétences dans les domaines des réseaux de communication et de nous pencher plus avant sur la gestion d'une implémentation d'une distribution pour un système embarqué.

L'acquisition en parallèle des connaissances nécessaires au bon déroulement du projet ne nous a permis de résoudre des problèmes que trop tardivement. Cela ayant pour conséquences de ne pas nous permettre de développer autant de fonctionnalités qu'il serait réellement possible de faire (les logiciels ayant un nombre de capacités trop important). On pourra toutefois noter que certaines évolutions seraient aisément implantables avec plus de temps. On peut penser par exemple au contrôle du trafic en ligne grâce au mode "inline" de snort et pourquoi pas, s'en servir comme supplétif à un firewall. De même il serait possible de connaitre l'utilisation du réseau par ses utilisateurs et de pouvoir réaliser des contrôles plus poussés. Cependant, on va rapidement se heurter aux limites de la légalité pour l'utilisation d'un tel logiciel, la liberté de circulation des informations restreignant la possibilité de lire le contenu des données ainsi récupérées.