Quelques éléments en matière de sécurité des cartes réseau

Yves-Alexis Perez, Loïc Duflot,
Olivier Levillain, and Guillaume Valadon

ANSSI 51 bd. De la Tour Maubourg 75700 Paris Cedex 07 France

Résumé Les cartes réseau font partie des composants les plus exposés à la menace informatique. En effet, quel que soit le système d'exploitation employé, la carte réseau est en première ligne et voit passer l'ensemble du trafic adressé à une machine. Ainsi, si une carte réseau était piégée ou comportait une faille d'exploitation, un attaquant pourrait prendre le contrôle à distance de celle-ci, indépendamment des mécanismes de sécurité proposés par le système d'exploitation.

Dans cet article, nous étudions les principes d'architecture des cartes réseau modernes et nous présentons les risques associés à leur complexification. En particulier, nous prenons l'exemple d'une fonctionnalité d'administration de machines à distance gérée par la carte réseau et nous montrons qu'une faille d'implémentation sur certains modèles de cartes permet à un attaquant d'en prendre le contrôle à distance. La contribution principale de cet article est de démontrer la faisabilité d'une prise de contrôle à distance d'une carte réseau et de discuter des différentes mesures qui permettent de limiter les conséquences d'une telle compromission.

Mots-clés: Cartes réseau, vulnérabilité.

1 Introduction

Les cartes réseau sont des objets très exposés. En effet, ces éléments critiques reçoivent la totalité des paquets véhiculés sur le segment réseau auquel elles sont attachées. Il est donc particulièrement important que ces composants ne soient pas vulnérables à des attaques. Un attaquant parvenant à exploiter une faille dans une carte réseau pourrait en prendre le contrôle à distance. Ce constat positionne la carte réseau et son contrôleur embarqué comme des composants critiques d'une machine. Ceci devrait donc inciter les développeurs de ces cartes à s'assurer que leurs composants sont exempts de problèmes d'implémentation exploitables par des attaquants.

Malheureusement, force est de constater que la règle générale est à l'heure actuelle plus à la complexification qu'à la simplification des architectures et des services rendus par les cartes réseau. Une carte réseau moderne possède un niveau de complexité proche de celui de la machine sur laquelle est elle est connectée. On rencontre ainsi des cartes comportant plusieurs CPU, différents types de mémoires, et de multiples interfaces capables de gérer des événements asynchrones. D'autre part, elles intègrent de plus en plus de fonctionnalités avancées nécessitant d'effectuer des traitements complexes sur les paquets réseau qu'elles reçoivent. En particulier, il est de plus en plus fréquent que les systèmes informatiques disposent de fonctions d'administration à distance entièrement gérées au niveau matériel (par les contrôleurs réseau ou les chipsets). Ces fonctions d'administration nécessitent le développement de sous-systèmes logiciels particulièrement complexes à l'intérieur même de la carte réseau.

Les exemples d'attaques mettant en œuvre le matériel deviennent de plus en plus nombreuses [8,11,9]. Arrigo Triulzi a présenté, lors de la conférence PacSec 2008 [14], un projet de réalisation d'un rootkit avancé intégré dans une carte réseau. À notre connaissance, notre contribution présente la première preuve concrète de faisabilité d'exploitation d'une vulnérabilité d'une carte réseau à des fins malveillantes.

Dans cet article, nous étudions l'architecture classique des cartes réseau et analysons leur fonctionnement (section 2). Ensuite dans la section 3, nous détaillons les conséquences d'une éventuelle compromission d'une carte réseau pour la sécurité de la machine hôte. Enfin, section 4, nous étudions en détails une fonction d'administration à distance relativement simple et montrons qu'une faille d'implémentation de ce mécanisme sur certaines cartes réseau est exploitable par un attaquant. Celui-ci peut ainsi prendre le contrôle à distance de la carte réseau. Dans la section 5, nous étudions les différentes mesures permettant de réduire le risque présenté par cette faille.

2 Architecture classique d'une carte réseau Ethernet

Dans cette partie, nous présentons l'architecture classique d'une carte réseau et les traitements qu'elle effectue. Nous détaillons en particulier la façon dont sont effectués les échanges entre une carte réseau et son pilote inclus dans le système d'exploitation de la machine.

Il est fréquent qu'une carte réseau se limite à un unique composant utile (le contrôleur réseau) qui regroupe l'ensemble des fonctions importantes de la carte réseau. Cela est particulièrement vrai sur les cartes réseau dont les contrôleurs sont intégrés dans les chipsets. Dans la suite de cet article nous considérerons donc que les termes « carte réseau » et « contrôleur réseau » sont équivalents.

2.1 Fonctionnement général

D'une manière générale, le rôle d'une carte réseau est de manipuler des trames Ethernet. Pour cela, elle dispose d'une interface physique sur laquelle elle émet ou reçoit des signaux (le plus souvent électriques) conformément à la norme Ethernet [12].

La figure 1 présente les opérations effectuées par la carte réseau. Pour la réception, les signaux physiques reçus sur le lien sont traités par un sous système dit « physique » interne au contrôleur qui les transforme en signaux logiques. Ces signaux logiques sont interprétés par le contrôleur comme des trames Ethernet et stockés temporairement dans une mémoire interne. Ces trames sont ensuite échangées avec le système d'exploitation s'exécutant sur la machine via des transferts directs en mémoire (utilisant généralement le mécanisme DMA[1] de « *PCI Bus Mastering* ») vers la mémoire du pilote de la carte réseau. Lors de l'émission, le pilote de la carte copie les trames Ethernet dans un tampon mémoire, et signale leur présence à la carte réseau qui les récupère et les stocke en interne avant de les transformer en signaux physiques et de les émettre.

1. *Direct Memory Access*, accès direct en mémoire.

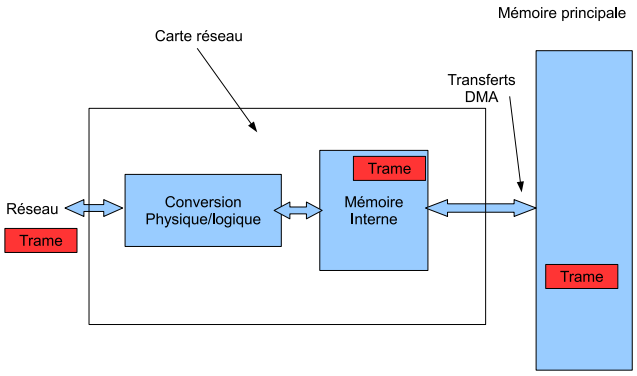

FIGURE 1. Schéma de principe de fonctionnement d'une carte réseau

2.2 Architecture bloc typique

Les constructeurs de carte réseau communiquent rarement sur l'architecture interne de leurs cartes réseau. Certains d'entre eux proposent cependant des documentations destinées aux développeurs de pilotes ouverts. La figure 2 présente une architecture typique décrite dans ces documents. On retrouve sur ce schéma bloc des sous-ensembles « PHY » et « DMA » qui correspondent respectivement aux fonctions de conversion des trames physiques vers des trames logiques (et vice versa) et aux fonctions « DMA » de transfert des trames logiques depuis/vers la mémoire de la machine hôte.

On remarque également un certain nombre d'interfaces vers des mémoires volatiles (SRAM) ou non volatiles (NVRAM). Des interfaces permettent par ailleurs de communiquer avec les LEDs en façade qui permettront de rendre compte de l'activité de la carte réseau.

2.3 Gestion des paquets réseau

Cette section s'intéresse plus particulièrement à la façon dont les transferts sont effectués entre la mémoire interne de la carte réseau

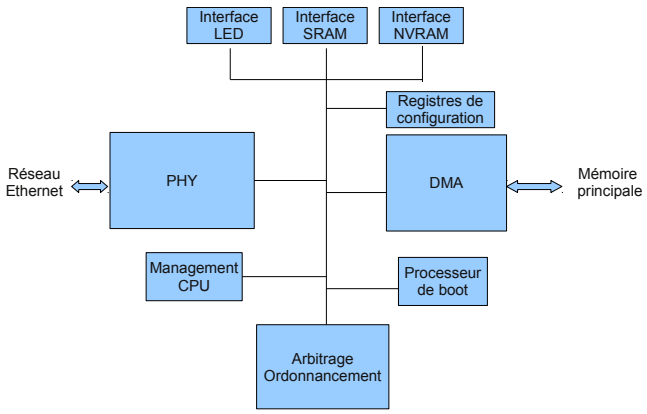

FIGURE 2. Schéma de principe d'architecture d'une carte réseau

et la mémoire de la machine hôte à l'aide des structures appelées anneaux (ou « *rings* » en anglais).

Afin de faciliter les explications, l'architecture présentée ici a été simplifiée. Il existe toutefois des variantes plus complexes avec un nombre de *rings* plus important. Ceci permet notamment d'augmenter la vitesse de traitement ou de gérer différentes priorités. Ainsi, les exemples réels sont plus compliqués[2].

Un *ring* est un tampon circulaire partagé entre la carte réseau et le pilote de périphériques. Il contient des descripteurs de tampons mémoire (BD en anglais, pour « *Buffer Descriptors* »). Ces descripteurs sont essentiellement des pointeurs vers des emplacements de la mémoire principale, alloués par le pilote, pouvant contenir des trames Ethernet. Le principe général des *rings* est présenté dans la figure 3.

Les *rings* fonctionnent selon une logique producteur/consommateur, où la carte joue un des deux rôles, et le pilote l'autre. À chaque *ring* est associé un pointeur « producteur » et un pointeur « consommateur ». Un bloc de contrôle (*Status block*), résidant dans la mé-

2. Le nombre de ring et leur capacité dépendent de la taille des mémoires internes de la carte.

moire de la carte et périodiquement synchronisé avec l'hôte, contient les index des producteurs et consommateurs des différents *rings*.

Il existe deux sortes de *rings* :

1. les « *receive rings* » qui contiennent des descripteurs vers des tampons contenant les trames reçues par la carte réseau. Ces tampons seront ensuite lus par le pilote ;
2. les « *transmit rings* » qui contiennent des descripteurs pointant vers des trames *produites* par le pilote que la carte réseau devra transmettre sur le réseau.

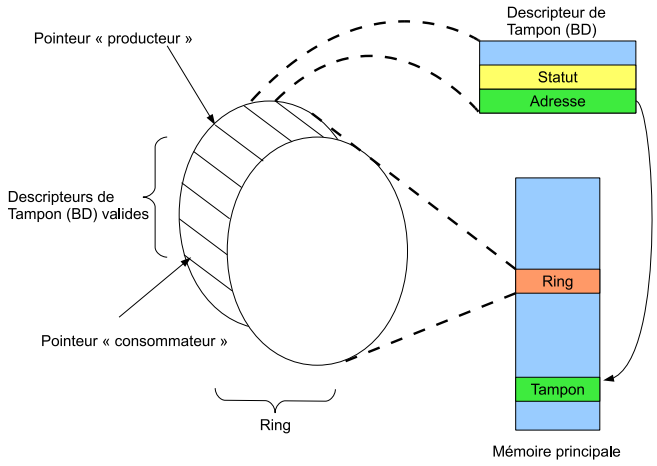

FIGURE 3. Principe des *rings*

Les *rings* résident eux-mêmes dans la mémoire de la machine hôte et sont identifiés par leur adresse de base. Cette adresse de base doit être indiquée par le pilote dans des registres de configuration interne à la carte.

Réception La procédure pour la réception de trame est la suivante :

- la carte stocke les paquets reçus dans sa mémoire interne ;
- s'il existe un BD libre dans le *receive ring*, la carte transfère la trame dans le tampon en question ;

- elle avance ensuite le pointeur « producteur ». Elle peut éventuellement prévenir le pilote de l'arrivée de cette trame à l'aide d'une interruption matérielle ;
- le pilote récupère la trame réseau puis consomme le BD correspondant dans le *receive ring*.

Bien entendu, il est possible que certains *rings* soient saturés. C'est par exemple le cas si la carte réseau reçoit des paquets trop rapidement ou que le pilote ne prend pas le temps de consommer les BD des *receive rings*. Dans ce cas, il est possible que la carte réseau perde des paquets. Elle peut alors avertir ses correspondants sur le réseau que sa mémoire est proche de la saturation à l'aide de trames Ethernet spécifiques.

Transmission La procédure pour la transmission de trame est la suivante :

- le pilote de périphérique produit un BD contenant la trame à émettre dans le *transmit ring* ;
- la carte réseau consomme ce BD depuis le *transmit ring* et analyse l'adresse où trouver la trame réseau correspondante ;
- elle copie la trame réseau dans sa mémoire interne par des transferts DMA ;
- enfin, elle transmet la trame sur le réseau.

2.4 Firmware et administration à distance

Il existe des fonctions rendues par la carte réseau qui ne peuvent complètement être effectuées en matériel. En effet, certaines fonctions nécessitent d'effectuer des traitements sur les trames Ethernet en interprétant successivement plusieurs protocoles (comme IP, TCP/UDP, ou HTTP). Ces fonctions étant trop complexes pour être traitées autrement que par du logiciel, les cartes réseau modernes exécutent un logiciel (« *firmware* »). Celui-ci s'exécute sur des processeurs souvent relativement simples (processeurs RISC [3]). Selon les modèles de carte réseau, ce firmware peut être persistant (stocké dans la NVRAM [4], il persiste même en cas d'arrêt de la machine) ou chargé à la volée systématiquement au démarrage du pilote de la

3. *Reduced Instruction Set Computer*.
4. *Non-Volatile RAM*, mémoire vive non volatile.

carte réseau. Les firmwares chargés à la volée sont souvent *offusqués* ou signés de manière à limiter la capacité d'un attaquant à charger un firmware arbitraire à l'intérieur du contrôleur.

Le firmware embarqué peut en particulier servir à gérer des fonctions d'administration à distance de la carte ou de la machine. En effet, la plupart des machines récentes comportent des fonctions d'administration à distance (AMT [1] ou encore IPMI [4]) permettant de configurer la machine depuis une console d'administration réseau. La console s'authentifie généralement auprès de la machine cible avant toute opération d'administration. Bien que la machine administrée joue au sens strict le rôle de serveur du point de vue de la fonction d'administration considérée et que la console d'administration se comporte comme un client se connectant à ce service, on parle généralement de « machine cliente » ou de « client » pour la machine administrée et de « console » pour la console d'administration. Ce sont ces termes qui ont été retenus dans la suite de ce document.

Les messages d'administration à distance sont échangés sur des protocoles standards (HTTP/HTTPS dans le cas d'AMT) et il est donc nécessaire pour le firmware de traiter tous les protocoles IP pour avoir accès à l'information utile contenue dans un paquet. Le code s'exécutant au sein de la carte réseau a la possibilité d'interagir avec les autres périphériques (et notamment le chipset) soit au travers des bus PCI/PCI-express, soit au travers du « SMBus » (*System Management Bus*), bus 8 bits spécifiquement conçu pour les échanges de message d'administration (alertes, informations de configuration) entre les différents composants d'une machine.

3 Conséquences d'une compromission d'une carte réseau

Présentons à présent les conséquences d'une compromission d'une carte réseau. On suppose ainsi que l'attaquant a la possibilité de modifier de manière arbitraire le code logiciel (firmware) s'exécutant dans la carte réseau. Tout d'abord, nous considérons que l'attaquant profite d'une faille de sécurité de la carte réseau. Dans un second temps, nous considérons que l'attaquant possède un accès local depuis la machine hôte (cas d'un rootkit cherchant à dissimuler des fonctions dans la carte réseau).

3.1 Impact d'une compromission de la carte réseau

Nous considérons ici que l'attaquant ne dispose pas d'autres privilèges sur la machine cible que celui de lui envoyer des paquets. On suppose en outre qu'il existe une faille de sécurité dans le logiciel embarqué dans la carte réseau (firmware) qui permet à cet attaquant d'en prendre le contrôle. Celui-ci est alors capable, sans privilège initial particulier, d'exécuter du code arbitraire dans la carte réseau.

Compromission de la carte réseau elle-même Lorsque l'attaquant parvient à exécuter du code à distance sur la carte réseau, cela ne signifie pas qu'il peut simplement prendre le contrôle complet de la machine cible. Cependant, le processeur embarqué sur lequel s'exécute le firmware a une position généralement centrale au sein de l'architecture du contrôleur. Pour pouvoir effectuer les traitements dans de bonnes conditions, il doit en effet avoir accès à la majeure partie, sinon à la totalité des sous-ensembles du contrôleur. En particulier, il a accès à l'ensemble des mémoires vives (SRAM) et sauvegardées (NVRAM). Ce contrôleur a donc accès à tous les paquets transitant par la carte réseau. La prise de contrôle du firmware est donc en règle générale suffisante pour :

1. intercepter (*sniffer*) l'ensemble du trafic de la machine (en réception et en émission) ;
2. retransmettre le trafic réseau à une machine distante ;
3. effectuer une attaque par le milieu (« *man in the middle* »). Par exemple, il est possible de rediriger tout le trafic de la machine vers des machines sous le contrôle de l'attaquant ;
4. causer un déni de service (pouvant être permanent sur certaines cartes réseau) en effaçant le contenu de la mémoire non volatile. La carte devient dans ce cas définitivement inutilisable ;
5. arrêter ou redémarrer la machine à distance. En effet, le logiciel a bien souvent accès au SMBus qui permet sur la plupart des architectures de demander au chipset l'arrêt ou le redémarrage de la machine ;
6. établir une communication via un canal caché avec une autre interface réseau au moyen du bus SMBus. Ce bus est connecté à chaque interface réseau et chacune d'entre elles peut y être à la

fois maître et esclave. Par conséquent, deux interfaces sous contrôle d'un attaquant peuvent échanger directement des messages (comme des trames réseau) sans passer par la mémoire principale ou le bus PCI. Cela peut ainsi permettre à un attaquant de contourner une politique de filtrage d'un pare-feu, ou encore une politique de chiffrement sur un chiffreur IP.

Par ailleurs, le code sous le contrôle de l'attaquant est extrêmement discret car l'attaque ne laisse aucune trace dans la mémoire ni sur les supports de stockage de la machine hôte. Une telle attaque est dans une large mesure indépendante du système d'exploitation s'exécutant sur la machine.

Impact potentiel sur la machine hôte La question qui se pose ensuite est la capacité de l'attaquant à utiliser sa position au sein de la carte réseau pour lire ou modifier la mémoire de la machine hôte pour y récupérer, par exemple, des clefs de chiffrement ou en prendre le contrôle. Là encore, il n'existe pas de cas général, tout dépend de la position du processeur embarqué dans l'architecture de la carte réseau. Sur les cartes que nous avons testées, le processeur ne dispose pas d'un accès direct au mécanisme de transferts DMA et le code de l'attaquant ne peut donc pas directement spécifier les adresses mémoire où écrire ou lire des données.

Cependant, le processeur a quand même la possibilité de modifier le contenu des adresses des *rings* utilisés par le système et peut donc effectuer des accès arbitraires en lecture ou en écriture au sein de la mémoire de la machine hôte. L'attaquant peut donc exploiter cette interface pour attaquer la mémoire de la machine hôte. Sur les machines récentes, une fonction appelée I/OMMU [5] chez AMD et VT-d [10] chez Intel permet de restreindre la plage d'adresses accessibles depuis la carte réseau. Si cette fonction est correctement utilisée (ce qui est rarement le cas à l'heure actuelle), il est impossible à la carte réseau d'accéder aux données en mémoire à l'extérieur de la plage d'adresses qui lui a été affectée. Ce mécanisme permet donc de réduire significativement le risque que l'attaquant puisse espionner les informations sensibles en mémoire ou tenter d'attaquer les fonctions vitales du système d'exploitation.

3.2 Utilisation par des rootkits

Le même type d'attaque peut être effectué par un rootkit. Les rootkits sont des codes d'attaques qui disposent d'un contrôle total de la machine cible et cherchent à dissimuler leur présence. Un rootkit pourrait donc camoufler certaines fonctions à l'intérieur de la carte réseau, à des fins de furtivité ou de résilience. Les possibilités offertes au rootkit après modification du firmware embarqué sont similaires à celles décrites dans la section précédente. Les mécanismes de sécurité proposés par les concepteurs de contrôleurs réseau pour protéger leurs firmware embarqués (signature des firmwares, *offuscation*) peuvent potentiellement compliquer la tâche d'un tel rootkit, mais ce dernier dispose généralement de suffisamment d'interfaces bas niveau avec la carte réseau pour charger dynamiquement le firmware directement à l'intérieur de la mémoire embarquée dans le contrôleur. Nous avons pu mettre en évidence au cours de cette étude qu'un tel chargement était relativement trivial sur certains modèles de cartes réseau aux travers d'accès directs aux registres et zones mémoire exposées par la carte à la machine hôte. La présentation d'Arrigo Triulzi [14] lors de la conférence Pacsec 2008 étudie spécifiquement les différents mécanismes de fonctionnement possibles pour un rootkit installé dans le firmware embarqué d'une carte réseau.

4 Un exemple de prise de contrôle à distance

Dans cette partie, nous étudions une fonction d'administration présente dans certaines cartes réseau. Nous montrons tout d'abord que le protocole permettant l'authentification de la console d'administration distante n'est pas conforme à l'état de l'art en la matière. D'autre part, nous mettons en évidence une faille d'implémentation de certains contrôleurs réseau qui permet à un attaquant d'en prendre le contrôle à distance.

4.1 Étude d'une fonction d'administration à distance particulière : ASF 2.0

Certaines cartes réseau (Broadcom ou Intel par exemple) disposent d'une fonction leur permettant d'émettre, périodiquement ou

en fonction d'événements particuliers, des alertes à destination d'une console d'administration distante grâce au protocole ASF (*Alert Standard Format*). Ces alertes, véhiculées par des paquets SNMP, sont routables, et ont en pratique les types suivants :

- des alertes sont émises en cas de problème, notamment en cas d'échec du démarrage de la machine (disque dur non présent ou problème de BIOS par exemple) ;
- des alertes de type « battement de cœur » (« *heatbeat* » en anglais) sont émises périodiquement sur le réseau pour indiquer si la machine est active (la fréquence des battements est configurable).

En principe, le protocole ASF devait seulement permettre d'émettre des paquets d'alerte. Il apparaît cependant que la version 2.0 d'ASF (qui sera la seule considérée dans la suite de ce document) autorise l'envoi de messages d'administration simples à la machine à partir d'une console d'administration distante. Les messages d'administration qui peuvent être envoyés sont les suivants :

- demande de démarrage de la machine : sur le réseau en PXE, sur le disque ou sur support amovible. Cette demande de démarrage passe avant toute autre configuration du BIOS. Il est ainsi possible à la console distante de demander le démarrage sur support amovible même si le BIOS l'interdit explictitement. Le mot de passe BIOS est également ignoré ;
- demande de redémarrage de la machine ;
- demande d'arrêt inconditionnel de la machine. Il s'agit d'un arrêt immédiat de la machine équivalent à une extinction du poste « *Soft Off* » [5] ;
- messages de service (messages de détection « *Presence Ping/Pong* » d'activation des fonctions ASF, message « *GetCapabilities* » permettant de déterminer les fonctions ASF précises activées sur la machine.

Les messages ASF d'administration à distance sont des messages UDP encapsulés dans un protocole appelé RMCP (pour *Remote Management and Configuration Protocol*). Pour être conforme à la

5. C'est-à-dire identique à une pression de plus de 4 secondes sur le bouton d'alimentation de la machine.

spécification ASF 2.0, une carte réseau doit être capable d'interpréter les messages RMCP sur deux ports UDP :

- le port 623/UDP encore appelé « *legacy port* » qui permet d'adresser à la machine des messages d'administration RMCP sans authentification ;
- le port 664/UDP appelé « *secure port* » qui impose l'authentification de la console d'administration distante au moyen de clefs symétriques pré-partagées avant toute interprétation de commande RMCP et qui permet d'assurer l'intégrité des messages échangés (ici encore, au moyen de clefs symétriques pré-partagées).

La compatibilité à la spécification ASF 2.0 impose à la carte réseau d'être en écoute sur ces deux ports et de répondre aux éventuels messages de détection d'activation des fonctions ASF (*Presence Ping/Pong*). En revanche, le choix des fonctions d'administration qui sont effectivement supportées sur chacun de ces ports est laissé au concepteur de la machine. Il est donc tout à fait possible de refuser systématiquement tout message d'administration (démarrage, arrêt) sur le port *legacy*. En pratique, toutes les implémentations que nous avons pu regarder n'autorisent l'envoi d'aucun message d'administration sur le port *legacy*. Nous ne nous considérerons donc que le port *secure* par la suite.

Les communications sur le *secure port* utilisent quant à elles le protocole RSP [6] nécessitant l'ouverture d'une session via un protocole nommé RSSP [7]. Celui-ci est constitué de messages RMCP particuliers appelés RAKP [8] qui permettent à la console distante de décrire le type de session qu'elle souhaite ouvrir. En particulier, la spécification ASF permet de définir deux rôles d'administrateurs distants (*Operator* et *Administrator*) disposant de privilèges particuliers. Il est également possible de préciser le nom de l'utilisateur distant (au moyen du champ *Username*). Les messages permettent l'authentification mutuelle du client et de la console à partir de clefs d'authentification symétriques pré-partagées. Une clef sera également dérivée de cet échange pour permettre un contrôle d'intégrité sur les messages d'administration qui seront ensuite échangés.

6. *RMCP Security-extensions Protocol.*
7. *RSP Session Protocol.*
8. *RSSP Authenticated Key-exchange Protocol.*

Les messagés échangés lors de l'ouverture de session et de l'échange RAKP sont décrits dans la figure 4. Le rejeu est prévenu par l'utilisation de nombres aléatoires choisis d'une part par le client (Rc) et d'autre part la console d'administration (Rm).

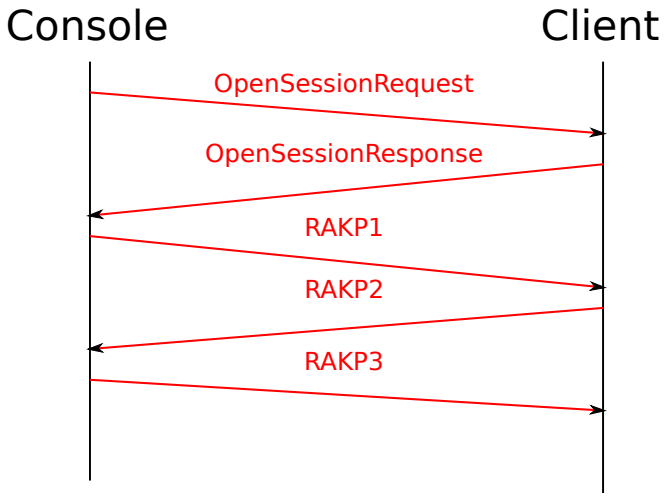

FIGURE 4. Ouverture de session RSP

4.2 Etude de la sécurité du protocole RMCP

Le protocole RMCP souffre d'un certain nombre de faiblesses structurelles qui peuvent théoriquement être exploitées par d'éventuels attaquants :

1. les messages liés à l'ouverture de session ne sont pas authentifiés ;
2. le message RAKP1 n'est pas authentifié ;
3. la clef partagée calculée à partir de l'échange RAKP est un haché d'éléments échangés, mais ne comprend pas l'identifiant du client. En effet, dans le protocole ASF, chaque client dispose d'un identifiant unique *GUID*. L'utilisation directe de cet identifiant dans le calcul des clefs partagées permettrait de garantir cryptograhiquement l'unicité des clefs négociées par deux clients différents avec la console et ce quelles que soient les circonstances. Dans le cas

présent, deux échanges effectués avec la même console d'administration par deux clients différents choisissant le même nombre aléatoire *Rc* aboutiraient au calcul de la même clef partagée si les conditions sont identiques (même rôle, même nom d'utilisateur distant). Un attaquant pourrait alors rejouer a posteriori pour un client les messages émis légitimement par la console pour l'autre client ;

4. les messages RAKP circulant sur le réseau ne comprennent aucun compteur indiquant où en est le protocole. Les messages RAKP ne sont qu'une suite de bits sans code d'authentification de message. Rien ne distingue donc un message RAKP2 émis par un client A vers une console B d'un message RAKP3 émis d'une console D vers un client C (attaque classique basée sur l'utilisation d'un oracle cryptographique).

En raison de la longueur (16 octets) des nombres aléatoires choisis par le client, les attaques qui pourraient être construites à partir de ces différentes faiblesses pour s'authentifier auprès d'un client sans connaissance des secrets pré-partagés ne fonctionneront cependant pas en pratique.

4.3 Mise en évidence d'une faille d'implémentation

Nous avons étudié l'implémentation effectuée par Broadcom dans ses contrôleurs réseau de la gamme NetXtreme [7]. Sur cette famille de contrôleur réseau, disposant d'une architecture proche de celle présentée section 2.2, le protocole ASF est géré par du code embarqué (firmware) s'exécutant sur un processeur MIPS indépendant qui a accès à la quasi totalité des registres de la carte réseau.

Le firmware ASF n'est en théorie chargé par le pilote que lorsque ASF a été installé et configuré sur la machine. Si tel n'est pas le cas, d'autres firmwares permettant d'effectuer des traitements systématiques sur les paquets peuvent être chargés (comme le firmware TSO, pour *TCP segmentation offloading* permettant le découpage de paquets TCP de taille supérieure à la MTU au niveau de la carte réseau). Lorsque le firmware ASF est chargé, tous les paquets reçus doivent être acquittés par ce firmware. S'il s'agit de paquets ASF, ils sont traités directement au niveau de la carte réseau et ne sont jamais transmis à l'hôte. S'il s'agit d'autres paquets, le firmware ASF

doit d'après nos observations, donner un feu vert pour permettre au sous-système DMA de transmettre les paquets au système hôte.

Notre étude a permis de mettre en évidence une faille d'implémentation du firmware ASF dans le protocole d'authentification RAKP. En effet, selon la spécification ASF, la taille du champ *Username* des messages RAKP1 est limité à 16 caractères. Cette limitation est cependant artificielle car la taille du champ *Username* compris également dans chaque paquet RAKP1 est codée sur 1 octet, ce qui permet d'essayer de renseigner dans les paquets réseau des champs *Username* d'une taille pouvant aller jusqu'à 255 octets.

Le dispositif que nous avons mis en place est le suivant :

- une machine client cible Dell Latitude D530 disposant d'une carte Broadcom NetXtreme et pour laquelle le protocole ASF a été activé et configuré ;
- une machine connectée via le réseau à la machine cible capable d'émettre et de recevoir des paquets réseau à bas niveau grâce à Scapy [6] et connaissant les clefs pré-partagés sur la machine cible. La connaissance de ces clefs est essentielle dans cette phase afin de pouvoir exécuter le protocole d'authentification jusqu'au bout et tester les différents messages. Le support complet du protocole RMCP sur Scapy a notamment été développé pour automatiser l'envoi des messages depuis cette machine.

Les expériences que nous avons menées ont pu été reproduites sur d'autres types de machines cibles (voir 4.6) embarquant le même contrôleur réseau.

Nous envoyons des demandes d'ouvertures de session (messages non authentifiés) au client puis l'envoi de messages RAKP1 (également non authentifiés) avec des champs “Username" de taille et de valeurs arbitraires (fuzzing simple sur le champ “Username"). En fonction des motifs et de la taille choisis, nous obtenons les résultats suivant :

1. l'authentification mutuelle des deux extrémités, et le calcul de la clef partagée sont correctement effectués. Des messages d'administrations sont ensuite échangés au sein de la session ;

2. le client indique que le protocole s'est bien terminé par l'émission d'un message RAKP3 valide. L'envoi des messages d'administra-

tion dans la session négociée échouent car le client considère que les motifs d'intégrité sont invalides ;

3. le protocole échoue (pas de message RAKP3 émis par le client) ;
4. suite à l'envoi du message RAKP1, la carte réseau ne transmet plus aucun paquet (quels qu'ils soient) au poste hôte.

Ce dernier cas est particulièrement intéressant. Il peut être interprété par un crash du processeur embarqué en charge de la gestion du protocole ASF. Les paquets reçus n'étant pas correctement traités par la carte réseau, ils ne sont pas transférés à l'hôte. Nous sommes donc potentiellement en présence d'un débordement de tampon à l'intérieur même de la carte réseau, causé par un champ *Username* trop long dans le message RAKP1. Tous les messages envoyés avant le message RAKP1 sont des messages non authentifiés, dont l'envoi ne demande donc pas la connaissance des clefs pré-partagées.

4.4 Exploitabilité de la vulnérabilité

Afin de vérifier si cette vulnérabilité est exploitable, nous avons utilisé les informations fournies par le constructeur dans sa documentation publique [7]. Cette spécification est tout particulièrement précieuse car elle précise que la mémoire interne de la carte, ainsi qu'un certain nombre de registres utiles, sont exportés en MMIO (*Memory Mapped I/O*) par le chipset dans l'espace de mémoire physique du poste hôte.

```
f6b00000-f6bfffff : PCI Bus 0000:09
f6bf0000-f6bfffff : 0000:09:00.0
f6bf0000-f6bfffff : tg3
```

Listing 1.1. Export des registres de la carte réseau dans la mémoire de l'hôte

De plus, elle décrit précisément tous les éléments accessibles depuis le poste hôte :

- la mémoire interne de la carte (par blocs de 32ko) ;
- le pointeur d'instruction du processeur ;
- les registres de statut du processeur décrivant notamment si le processeur est arrêté et pourquoi ;

– les registres de contrôle et de débogage du processeur permettant notamment d'arrêter le processeur, de le démarrer, de placer des points d'arrêts et de passer le processeur en mode « *step by step* ».

A l'aide de ces éléments, nous pouvons donc observer les instructions exécutées en mémoire pour localiser les structures utiles (pile des appels notamment). Ces informations sont suffisantes pour développer un débogueur externe de cartes réseau.

L'outil développé (voir figure 1.2) nous permet entre autres de :

– arrêter, redémarrer le processeur MIPS ;
– définir des conditions d'arrêt du processeur (arrêt sur une adresse mémoire, type d'instruction exécutée, données lues en mémoire, mode *step by step*) ;
– repérer l'apparition d'un motif dans la pile ;
– tracer l'évolution d'adresses mémoire ou des registres.

Cet outil comprend également un moteur intégré de marquage *data tainting* automatique. Il permet de marquer une adresse mémoire ou un registre afin de tracer automatiquement l'impact et la propagation de ce marquage. Le moteur permet l'arrêt automatique du débogueur lorsque par exemple, le programme saute vers une adresse marquée. Ce module a permis la localisation d'un certain nombre d'adresses utiles à la démonstration d'exploitabilité. La liste des fonctions supportées à l'heure actuelle est présentée en figure 1.3 de l'annexe A.

```
***********************************************
******** Instruction
Instruction = 3c020001 LUI r2 = 00010000
Last memory access = 00000000
******** CPU Status Registers ********
RXPC   = 00011078 RXHWBRK  = 0000001d
RXMODE = 00009c80 RXSTATE  = 80001400
******** CPU Registers ****************************
$0  = 00000000 $1  = 00010000 $2  = 00000000 $3  = 40000000
$4  = 0001b4b8 $5  = 0001b8e6 $6  = 00000000 $7  = 0001bfc4
$8  = 00000040 $9  = 00000050 $10 = 0001b8bc $11 = 0001bfc0
$12 = 80000000 $13 = 00000001 $14 = 00000000 $15 = ffffffbf
$16 = a4020000 $17 = aaaaaaaa $18 = 00000000 $19 = 0001af48
$20 = 0000ad60 $21 = 018004f1 $22 = 000000fc $23 = 00010000
$24 = ffffffff $25 = 80000000 $26 = 00000b50 $27 = 000110b0
$28 = c0000000 $29 = 0001bfd8 $30 = 0001c000 $31 = 000111f8
******** Stack ***********************************
Stack pointer:  0001bfd8  (max) stack size: 10
Stack bottom:   0001c000
Stack grid: 5 lines, 2 columns, remains 0
```

```
**********************************************
0001bffc:f3ffffff  0001bfe8:00010e00
0001bff8:00010044  0001bfe4:0001a918
0001bff4:0001a000  0001bfe0:0000ad60
0001bff0:0000ad64  0001bfdc:0001a000
0001bfec:0001a80c  0001bfd8:00010b3c
**********************************************
What should I do next (h for help)?
```

Listing 1.2. Débogueur de carte réseau

Le dispositif expérimental pour décider de l'exploitabilité du problème d'implémentation est le même que lors de la mise en évidence de la vulnérabilité, mais cette fois la machine sur laquelle tourne Scapy **ne connait pas** les clés pré-partagées présentes sur la machine cible.

4.5 Exploitation

La vulnérabilité de ce contrôleur réseau est exploitable. Il est ainsi possible de faire exécuter du code arbitraire à la carte réseau à distance sans aucune authentification préalable.

Le débordement du champ “Username" permet indirectement l'écrasement d'une valeur de retour située dans la pile du processeur MIPS par une valeur arbitraire, ce qui permet à l'attaquant de dérouter le programme vers du code qu'il a lui même injecté en mémoire via des paquets réseau envoyés précédemment. Les détails précis de l'exploitation varient d'un type de machines à l'autre.

Cette exploitation se fait au moyen de paquets RMCP (routables) depuis n'importe quelle adresse IP (l'adresse IP de la console d'administration associée à un client est normalement connue du client mais n'est en pratique pas vérifiée).

Notre preuve de concept (voir [3] pour la vidéo) développée pour les machines Dell Latitude D530 permet notamment :

- d'éteindre et d'allumer la machine à distance (en utilisant l'accès privilégié de la carte réseau au bus SMBus) ;
- de faire clignoter à des fréquences variables les LEDs de la carte réseau ;
- de récupérer des données arbitraires en mémoire et de les envoyer sur le réseau ;
- de corrompre le système d'exploitation hôte (pour la démonstration un système Linux à jour).

Les deux derniers points ne sont bien entendus réalisables (voir partie 4.6) que lorsqu'aucun mécanisme de type I/OMMU n'est mis en œuvre sur la machine.

4.6 Impact pratique

Les tests que nous avons effectués montrent que le problème est lié au modèle de carte réseau et qu'il s'exprime de la même façon sur les machines de constructeurs différents. En particulier, le problème a pu être mis en évidence sur certaines machines Dell et Hewlett-Packard (Dell Precision T5540, Dell Latitude D530, HP Compaq DC7600). Les modalités d'exploitation peuvent être spécifiques au modèle de machine (les implémentations des différents constructeurs de machine diffèrent) mais sont en grande partie indépendantes du système d'exploitation mis en œuvre sur la machine cible [9], tant que l'exploitation ne cherche pas à rebondir sur le système hôte. Les modalités de l'attaque depuis la carte réseau du système d'exploitation sont cependant spécifiques à chaque système d'exploitation.

Broadcom a confirmé le problème courant janvier 2010 et réalisé un correctif pour le firmware ASF. Il est distribué par les constructeurs (Dell, Hewlett-Packard par exemple) dans les dernières versions de leurs pilotes.

Seules les machines mettant en œuvre des contrôleurs Broadcom NetXtreme et sur lesquelles la fonction ASF 2.0 est activée et configurée sont a priori vulnérables au problème mentionné dans cet article. Pour ces machines, il est important d'appliquer les correctifs rendus disponibles par les constructeurs en général dans les dernières versions de pilotes disponibles sur leur site internet (voir également les avis du CERTA [2]) ou de désactiver durablement la fonction ASF au moyen de l'utilitaire Broadcom fourni par les constructeurs.

L'impact de ce type d'attaque peut être réduit par l'utilisation et la configuration d'un dispositif d'I/OMMU qui empêchera les accès mémoire arbitraires depuis la carte réseau. Ce type de dispositif n'empêchera cependant pas un attaquant d'intégrer un rootkit à l'intérieur de la carte réseau pour espionner les paquets ou encore éteindre ou allumer la machine à distance.

9. Certains systèmes d'exploitation comme OpenBSD [13] tentent a priori de désactiver ASF sur les cartes réseau lorsque cela est possible.

5 Mesures de réduction du risque

Afin de réduire les risques liés à l'exécution de code arbitraire sur une carte réseau, on peut mettre en œuvre les moyens suivants sur les machines clientes :

- désactiver si possible toutes les fonctions d'administration à distance de la machine ou de la carte réseau. Cette désactivation doit être effectuée en chargeant un firmware réseau n'effectuant aucune opération complexe sur les paquets reçus. La méthode de désactivation est spécifique à chaque fonction d'administration. Généralement l'utilitaire utilisé pour activer la fonction peut également être utilisé pour désactiver cette dernière (c'est le cas d'ASF) ;
- réserver les fonctions d'administration à des réseaux séparés physiquement et connectés aux machines via des interfaces spécifiques. Cette mesure est essentiellement destinée à réduire le risque de présence d'un attaquant sur le réseau auquel sont connectés les cartes réseau comportant un firmware lié à des opérations d'administration. Cette mesure n'est bien entendu qu'une mesure de défense en profondeur ;
- maintenir les firmwares (logiciels embarqués) et les pilotes à jour. Cette mesure de bon sens doit s'incrire dans le cadre plus vaste du maintien en condition opérationnelles et de sécurité d'un parc informatique ;
- au plan système, mettre en oeuvre une technologie de virtualisation de l'espace mémoire accessible par la carte réseau (I/OMMU [5] ou Vt-d [10]). Comme on a pu le voir au paragraphe précédent, l'utilisation d'un dispositif de ce type ne permet de prévenir en théorie qu'une attaque du système d'excploitation depuis la carte réseau, mais aucunement la compromission initiale de la carte réseau.

Au niveau de l'infrastructure réseau, on devra :

- placer les machines clientes derrière des pare-feux permettant de bloquer la majorité des flux non sollicités (dans le cas d'ASF, les ports UDP/623 et UDP/664 devront être bloqués) ;
- maintenir en permanence, dans la limite de la maintenabilité du parc des machines concernées, une pratique de diversification technologique à tous les niveaux (logiciel et matériel)

de telle sorte qu'une même vulnérabilité ne puisse pas simultanément s'exprimer sur un pare-feu et sur la machine située derrière un tel pare-feu.

6 Conclusion

Dans cet article, nous avons étudié certains aspects liés à la sécurité des cartes réseau. Nous avons montré que même des fonctions relativement simples pouvaient comporter des failles d'implémentation permettant à un attaquant de prendre le contrôle à distance d'un contrôleur réseau avec quelques paquets routables. Nous avons étudié les conséquences d'une telle compromission et les moyens de s'en prémunir.

Cette étude montre le manque de prise en compte des aspects sécurité en matière de logiciels embarqués (firmware) bas niveau. Cependant, ces logiciels embarqués sont souvent aussi exposés que les logiciels applicatifs s'exécutant sur le système d'exploitation principal de la machine. Il est indispensable que le niveau de sécurité de ces logiciels soit accru pour éviter que certaines fonctions internes aux cartes réseau (fonctions d'administration à distance) ne soient exploitées en pratique par les attaquants. Dans l'intervalle, il est important de privilégier un certain niveau de diversité technologique, de manière à minimiser les risques d'exploitation massive des vulnérabilités sur les composants embarqués.

A Fonctions du débogueur externe de cartes réseau

```
What should I do next (h for help)? h
Usage:   'a' -> Advance n steps
  's' -> Advance 1 step
  't' -> Trace
  'c' -> Continue
  'C' -> Continue (step-by-step)
  'g' -> Break on instruction
  'R' -> Break on pattern in register
  'S' -> Break on pattern in stack
  'H' -> Break on pattern in internal memory
  'M' -> Break on pattern in external memory
  'n' -> Break on next pattern in stack
  'l' -> Break on specific memory access
```

```
        'm' -> Break on any memory access
        'j' -> Break on register write
        'i' -> Break on instruction
        'T' -> Track register
        'L' -> Track memory address
        'Z' -> Track specific memory zone access
        'I' -> Track pattern in memory
        'P' -> Track pattern
        'x' -> Clear tracking
        'f' -> Find pattern in internal memory
        'F' -> Find pattern in external memory
        'A' -> Find all pattern occurrences in external memory
        'd' -> Display memory address
        'D' -> Display memory area
        'w' -> Write a word to memory address
        'r' -> Reset CPU
        'q' -> Quit
```

Listing 1.3. Aide du débogueur

Références

1. Active Management Technology. www.intel.com/technology/platform-technology/intel-amt.
2. Avis du CERTA. www.certa.ssi.gouv.fr/site/CERTA-2010-AVI-121/index.html.
3. Démonstration. www.sstic.org.
4. Intelligent Platform Management Interface. www.intel.com/design/servers/ipmi.
5. AMD. I/O Virtualization Technology (IOMMU) Specification. 2006. http ://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf.
6. P. Biondi. Scapy. www.secdev.org.
7. Broadcom Corp. Host Programmer Interface Specification for the NetXtreme and NetLink Family of Highly Integrated Network Controlers, 2007. www.broadcom.com.
8. C. Devine and G. Vissian. Compromission physique par le bus PCI. In *Symposium sur la Sécurité des Systèmes d'Information et des Télécommunications (SSTIC)*, 2009.
9. L. Duflot. CPU bugs, CPU backdoors and consequences on security. In *ESORICS 2008 : Proceedings of the 13th European Symposium on Research Computer Security*, 2008.
10. D. Grawrock. The Intel safer computing initiative : building blocks for Trusted Computing. In *Intel Press*, 2006.
11. J. Heasman. Implementing and detecting an ACPI BIOS rootkit. In *Blackhat federal 2006*, 2006. www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Heasman.pdf.
12. IEEE. IEEE 802.3 Ethernet. standards.ieee.org.
13. OpenBSD core team. The openbsd project. 2007. http ://www.openbsd.org.
14. A. Triulzi. Project Maux Mk.II : I Own the NIC now I ant a shell. In *Pacsec conference*, 2008.