Le protocole de découverte de sécurisation SCOUT

Antoine Varet, Nicolas Larrieu

Le protocole de découverte de sécurisation SCOUT

Antoine Varet, Nicolas Larrieu
Ecole Nationale de l'Aviation Civile (ENAC)
Groupe ResCo, laboratoire TELECOM, département SINA
Toulouse, France
antoine.varet@recherche.enac.fr, nicolas.larrieu@enac.fr

Abstract—**La sécurisation des réseaux de communication engendre aujourd'hui une charge importante de configuration. C'est pourquoi nous proposons un nouveau protocole de découverte automatique des capacités de sécurisation, afin de réduire cette charge. Notre protocole « Security Capabilities Over Unsecured Topologies » a été conçu pour rechercher les mécanismes de sécurité supportés par les nœuds distants du réseau et déclencher le mécanisme d'établissement de canal sécurisé adéquat. Si l'hôte contacté ne peut prendre en charge la sécurisation, le protocole SCOUT tente alors d'établir un canal sécurisé avec le routeur voisin de l'hôte. Ce protocole permet à l'administrateur de ne pas devoir configurer à l'avance un tunnel par couple de nœuds communiquant de manière sécurisée, ce qui réduit sa charge de travail et améliore la capacité de passage à l'échelle et de déploiement des solutions de sécurisation des réseaux. Après avoir introduit le protocole SCOUT et son fonctionnement, nous présenterons une évaluation des performances de son implémentation sur un réseau IPv6 natif suivie d'une analyse des vulnérabilités de ce protocole. Enfin, nous concluons cet article par une étude qualitative du gain apporté par SCOUT en terme de configuration pour les administrateurs réseaux.**

Mots-clefs: protocole, sécurité des réseaux, découverte, IPv6

I. INTRODUCTION

Les mécanismes de sécurité sont de plus en plus utilisés dans les réseaux informatiques. La première et principale raison vient des évolutions techniques : les nouveaux systèmes disposent de ressources de calcul et de capacités mémoires en constante augmentation et ainsi les réseaux permettent des échanges à des taux de transfert de plus en plus élevés. La seconde raison vient de l'impact croissant de la sécurité informatique dans les politiques des systèmes d'information (SI) : les médias donnent aujourd'hui aux exploits des pirates informatiques une visibilité mondiale alors que la connaissance des attaques des réseaux restait dans des cercles très privés il y a encore quelques années.

Simultanément, les protocoles de sécurisation ont crû en sûreté, en sécurité et en efficacité. Ainsi, le protocole SSH [1], créé en 1995, a évolué jusqu'à sa version actuelle et supporte désormais de nombreux nouveaux algorithmes de sécurité. Un autre exemple concerne la suite Internet Protocol Security (IPsec) [2]. D'après la spécification du protocole IPv6 [3], les systèmes d'exploitation compatibles avec IPv6 doivent supporter la suite IPsec. Cette suite protocolaire définit deux modes de fonctionnement, les modes transport et tunnel. Les canaux sécurisés peuvent être établis à l'aide de différents protocoles d'établissement de canal sécurisé, tels que les protocoles Internet Key Exchange (IKE et IKEv2 [4]) et le protocole Kerberized Internet Negotiation of Keys (KINK [5]). La suite IPsec repose alors sur les protocoles Authentication Header (AH [6]) et Encapsulating Security Payload (ESP [7]) pour sécuriser les données transitant par les canaux négociés à l'aide des protocoles IKE, IKEv2 et KINK.

Toutefois, ces protocoles nécessitent une charge importante de configuration afin de pouvoir établir et utiliser les canaux sécurisés. Il est en effet nécessaire de préciser à l'avance sur chaque extrémité de chaque canal les informations nécessaires à la négociation d'ouverture des canaux. Cette charge est un frein au déploiement des solutions de sécurité de niveau réseau, d'autant plus problématique que les topologies réseaux sont complexes. Cela peut consommer une part importante de temps et d'argent pour les administrateurs réseaux et les sociétés et administrations.

C'est pourquoi nous proposons un nouveau protocole afin de réduire cette charge. Notre protocole de découverte automatique des capacités de sécurité sur des topologies réseau non sécurisées a été nommé SCOUT : « Security Capabilities discovery protocol Over Unsecured Topologies ». Il a été conçu afin de rechercher les possibilités d'établissement de canaux sécurisés sur les nœuds du réseau et d'appeler les mécanismes adéquats pour établir ces canaux et ainsi sécuriser les échanges de données. Le protocole SCOUT permet aux routeurs d'extrémité de sécuriser les données à la place des hôtes d'extrémité lorsque ces hôtes ne peuvent sécuriser eux-mêmes les données (par exemple en cas d'incompatibilités protocolaires ou de puissance de calcul trop faible). Le protocole SCOUT réduit la charge de l'administration de la sécurité du réseau en évitant à l'administrateur d'avoir à configurer à l'avance l'ensemble des canaux potentiels, c'est-à-dire ceux qui pourraient être utilisés pour sécuriser les données.

A. Quelques protocoles d'établissement de canaux sécurisés

Les protocoles de sécurisation des données nécessitent, entre les entités communicantes, une négociation préalable des éléments de sécurité tels que les clefs de chiffrement ou l'ensemble des algorithmes à appliquer. Ces éléments peuvent être intrinsèquement liés aux protocoles employés. Ainsi, le Point-To-Point Tunneling Protocol (PPTP [8]) utilise une connexion TCP pour gérer et contrôler le tunnel Generic Routing Encapsulation (GRE [9]) qui encapsule les données dans des paquets PPP. Le protocole Extensible Authentication Protocol (EAP [10]) est généralement utilisé dans les réseaux sans fils (Wireless Network).

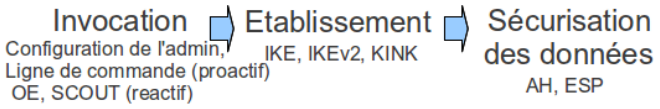


Figure 1. Déclenchement, établissement et sécurisation

Le protocole EAP permet de fournir des méthodes de négociation des canaux sécurisés. L'extension EAP-IKEv2 [11] est une extension de ce protocole EAP avec la négociation faite par le protocole IKEv2 présenté plus bas.

La suite IPsec permet de sécuriser les données au niveau de la couche 3 réseau. Elle repose sur le protocole Internet Security Association and Key Management (ISAKMP [12]) pour gérer localement l'ensemble des canaux sécurisés. La négociation des paramètres de sécurité peut être réalisée à l'aide d'un protocole dédié à l'établissement de canal sécurisé. Différents protocoles permettent de négocier ces paramètres. Les canaux peuvent être négociés à l'aide du protocole Internet Key Exchange (IKE [3]) ou de son successeur IKE version 2 (IKEv2 [13,14]). Le protocole Kerberized Internet Negotiation of Keys (KINK [5]) est une alternative basée sur des tickets gérés d'une manière centralisée autour des serveurs Kerberos [15]. Tous ces protocoles d'établissement nécessitent d'être invoqués par un élément externe, comme présenté figure 1.

Les protocoles d'établissement IKE, IKEv2 et KINK supposent que l'administrateur réseau a préalablement configuré les canaux sécurisés potentiels. Le protocole SCOUT que nous proposons permet d'automatiser ces étapes de configuration, réduisant ainsi la charge de travail de l'administrateur. Le protocole Opportunistic Encryption (OE [16]) étend les implémentations d'IPsec OpenSwan [17] et StrongSwan [18] en proposant d'établir à la volée les canaux de communication, comme le protocole SCOUT se propose de le faire. Cependant, le protocole OE n'a été conçu que pour les réseaux IPv4, il dépend d'un tiers de confiance devant sécuriser un serveur DNS avec le protocole DNSSEC [19], il ne permet pas aux routeurs de se substituer aux nœuds terminaux pour fournir la sécurité des flux de données et il n'est compatible qu'avec les protocoles IKE et IKEv2. SCOUT contribue ainsi à pallier ces limitations comme nous le détaillons dans la section suivante.

II. LE PROTOCOLE SCOUT

A. *Le protocole générique SCOUT*

Lorsqu'un paquet sort par une interface réseau préalablement marquée par l'administrateur comme étant à sécuriser avec le protocole SCOUT, le protocole négocie un canal sécurisé. Il peut fonctionner de 4 manières différentes : en fonction du support ou non de SCOUT par les nœuds du réseau. Ainsi, la figure 2 présente une topologie basique avec deux hôtes d'extrémité H1 et H2, liés respectivement aux deux routeurs R1 et R2.

Si les deux hôtes H1 et H2 supportent SCOUT, alors le canal sécurisé peut être établi entre H1 et H2. C'est le mode privilégié **SCOUT Host-Host** : quand H1 envoie son premier paquet vers H2, le dæmon SCOUT sur H1 intercepte ce paquet et négocie le choix d'un protocole d'établissement avec H2.

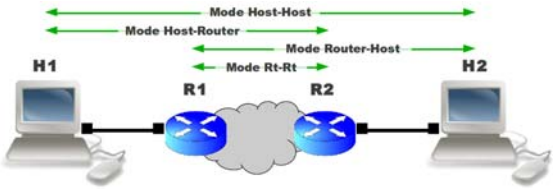


Figure 2. Modes de SCOUT sur une topologie réseau basique

Un canal sécurisé est alors établi entre H1 et H2 et des règles de routage indiquent d'utiliser ce canal pour les flux H1-H2. Ainsi, après la phase de découverte puis celle d'établissement, tout paquet allant de H1 vers H2 ou en sens inverse est sécurisé au travers de ce canal.

Dans le cas où le routeur R2 supporte SCOUT mais l'hôte H2 ne le supporte pas, le protocole SCOUT permet l'établissement d'un canal sécurisé entre H1 et R2, c'est le mode **SCOUT Host-Router**. Quand H1 envoie son premier paquet à H2, le dæmon SCOUT de H1 l'intercepte et envoie une requête à H2 pour connaître les possibilités d'établissement d'un canal sécurisé. H2 n'exécutant aucun dæmon SCOUT, le système ignore les requêtes SCOUT qu'il reçoit et refuse la communication. H1 contacte alors le routeur R2 à proximité de H2. R2 répond positivement à la requête SCOUT de H1 et le canal sécurisé s'établit entre H1 et R2, afin de sécuriser les paquets transitant entre H1 et H2.

Si au contraire, c'est l'hôte d'origine H1 qui ne supporte pas SCOUT mais que le routeur R1 à proximité de H1 et la destination H2 gèrent SCOUT, alors dès que H1 transmet son premier paquet à H2 via R1, le dæmon SCOUT de R1 intercepte ce paquet et négocie l'établissement d'un canal sécurisé entre R1 et H2 ; c'est le mode **SCOUT Router-Host**. Enfin, dans le cas où aucun des deux hôtes H1 et H2 ne gèrent SCOUT mais où les deux routeurs R1 et R2 gèrent SCOUT, le protocole permet l'établissement d'un canal sécurisé entre R1 et R2 pour sécuriser les données de/vers H1 vers/de H2 ; c'est le mode **SCOUT Router-Router**.

Dans les 4 modes de fonctionnement représentés figure 2, les données sont sécurisées sur le tronçon « longue distance » entre les routeurs R1 et R2. Le protocole SCOUT a été élaboré afin de s'adapter de manière optimale aux topologies réseaux complexes, tout en réduisant la charge de son déploiement. Il nécessite en effet d'installer et de configurer le dæmon SCOUT sur certains nœuds du réseau : limiter le déploiement aux routeurs uniquement fournit un niveau de sécurité suffisant pour certains réseaux locaux et ne nécessite pas de modification des hôtes d'extrémité.

Ainsi, le protocole SCOUT rend possible la sécurisation des communications entre des hôtes ne pouvant sécuriser eux-mêmes leurs échanges. Ce cas survient lorsque les hôtes ont des ressources matérielles faibles (mémoire, processeur), fonctionnent avec des systèmes d'exploitation pour lesquels il n'existe pas d'implémentation de protocole de sécurité compatibles avec ceux des autres équipements du réseau... Lorsqu'un seul nœud du réseau supporte SCOUT, la découverte ne peut aboutir positivement et les données ne peuvent donc pas être sécurisées automatiquement. Dans ce cas, le comportement par défaut du nœud consiste à rejeter les paquets ne pouvant être sécurisés.

TABLE I. MODES RESULTANT DE LA DECOUVERTE EN FONCTION DE LA PRISE EN CHARGE DE SCOUT

				H1 supporte SCOUT ?			
				Oui		*Non*	
				R1 supporte SCOUT ?			
				Oui	*Non*	*Oui*	*Non*
H2 supporte SCOUT ?	*Oui*	**R2 supporte SCOUT ?**	*Oui*	Host-Host (H1 et H2)		Router-Host (R1 et H2)	Rejet des paquets
			Non				
	Non		*Oui*	Host-Router (H1 et R2)		Router-Router (R1 et R2)	
			Non	Rejet des paquets			

La table 1 présente les différents cas de figure pouvant se présenter avec la topologie de la figure 2, selon que chaque nœud prend en charge ou non le protocole SCOUT.

Pour résumer ce tableau, les données sont toujours sécurisées sur le réseau longue distance entre les routeurs d'extrémités ou alors elles sont rejetées si la découverte n'a pu aboutir à une compatibilité des solutions de sécurisation. Elles ne transitent jamais de manière non sécurisée sur le réseau.

Si les hôtes d'extrémité supportent le protocole SCOUT, alors ils prennent en charge la sécurisation des données. Dans le cas contraire, les routeurs sécurisent les données s'ils en ont la capacité, sinon les données sont rejetées.

B. Prérequis de fonctionnement de SCOUT

Certaines hypothèses doivent être réunies pour un fonctionnement optimal et sûr avec le protocole SCOUT. Premièrement, la liaison entre les hôtes d'extrémité et leur routeur voisin est supposée sûre : le mode SCOUT Router-Router ne sécurise les données qu'entre les routeurs d'extrémité et non entre les routeurs et leurs hôtes, ce mode n'est donc acceptable que si les réseaux locaux sont sûrs, c'est-à-dire que la sécurité périmétrique est assurée (ou que SCOUT est limité par l'administrateur au mode Host-Host exclusivement). L'étude de la sécurité du protocole SCOUT, présentée section IV.D dans ce document, contient un exemple d'attaque où les données sont découvertes sur le tronçon non sécurisé entre le routeur et l'hôte.

La deuxième hypothèse concerne les protocoles d'établissement des canaux sécurisés. Pour fonctionner, le protocole SCOUT requiert au moins un protocole d'établissement de canal sécurisé. Dans le cas où plusieurs protocoles d'établissements sont candidats, nous appellerons « protocole PE » celui qui est sélectionné pour établir le tunnel. PE peut alors désigner indifféremment IKE, IKEv2 ou KINK. Le protocole SCOUT prend en charge la découverte des protocoles susceptibles d'assurer l'établissement d'un canal, choisit le meilleur « PE » suivant le contexte et ensuite invoque ce protocole PE pour créer le canal sécurisé et reconfigurer en conséquence les tables de routage des systèmes d'exploitation concernés.

La troisième hypothèse concerne aussi le protocole d'établissement « PE ». Le protocole SCOUT fait confiance dans la sécurité de ce protocole PE pour l'authentification des extrémités du tunnel ; SCOUT identifie les extrémités entre lesquelles établir un canal sécurisé et transmet cette information au protocole PE qui authentifie les extrémités. Cela permet de réduire la complexité du protocole SCOUT, la taille des messages échangés et le temps de traitement tout en évitant d'effectuer plusieurs fois la phase d'authentification.

C. Algorithme du protocole SCOUT

Le protocole SCOUT utilise 4 termes différents pour désigner les nœuds du réseau en fonction de leur rôle, un nœud pouvant avoir plusieurs rôles.

- Le nœud « **inspirateur** » désigne le nœud générant les paquets de données à sécuriser.
- Le nœud « **destinataire** » désigne le nœud de destination à qui sont adressées les données émises par l'inspirateur.
- Le nœud « **initiateur** » désigne le nœud initiant la découverte SCOUT.
- Le nœud « **répondeur** » désigne le nœud répondant positivement à cette découverte.

Ainsi, l'inspirateur transmet des données non sécurisées au destinataire, et l'initiateur les sécurise jusqu'au répondeur. La table 2 précise quelle fonction est associée à chaque nœud dans les 4 modes de fonctionnement de SCOUT.

TABLE II. INSPIRATEUR, INITIATEUR, REPONDEUR ET DESTINATAIRE, SELON LES MODES

Mode SCOUT	H1	R1	R2	H2
Host-Host	Inspirateur et Initiateur	(aucun rôle)	(aucun rôle)	Répondeur Destinataire
Host-Router	Inspirateur et Initiateur	(aucun rôle)	Répondeur	Destinataire
Router-Host	Inspirateur	Initiateur	(aucun rôle)	Répondeur Destinataire
Router-Router	Inspirateur	Initiateur	Répondeur	Destinataire

Dans ce qui suit, nous parlerons de conversation pour désigner tous les paquets échangés entre deux nœuds du réseau. Une conversation est caractérisée par le couple (adresse nœud 1, adresse nœud 2) et est indépendant du contenu transporté ou même du sens des échanges (nœud 1=source et nœud 2=destination, ou le contraire en sens inverse).

TABLE III. NŒUDS POUVANT ASSURER LES FONCTIONS D'INSPIRATEUR, INITIATEUR, REPONDEUR ET DESTINATAIRE

Fonction	Nœud de la topologie figure 2
Inspirateur	H1 (inconditionnellement)
Initiateur	H1 s'il supporte SCOUT, R1 si R1 supporte SCOUT et H1 ne supporte pas SCOUT
Répondeur	H2 s'il supporte SCOUT, R2 si R2 supporte SCOUT et H2 ne supporte pas SCOUT
Destinataire	H2 (i nconditionnellement)

1) Première phase de découverte de SCOUT : hôte d'extrémité

L'algorithme de SCOUT peut être résumé de la manière suivante, présentée dans la figure 3. Le premier paquet d'une conversation est émis par l'inspirateur, ce qui déclenche la découverte SCOUT. A l'issue de la découverte, si un canal sécurisé a été établi, les autres paquets de cette conversation sont sécurisés au travers de ce canal, sinon ils sont rejetés (afin de ne pas transmettre de données non sécurisées).

Lorsque le premier paquet est émis par l'inspirateur vers le destinataire, le dæmon SCOUT de l'initiateur l'intercepte, le met en attente et émet une requête au destinataire. C'est la première phase de la découverte. Si le destinataire supporte le protocole SCOUT, alors il répond positivement à la requête et a le double rôle de répondeur/destinataire (c'est un des deux modes Host-Host ou Router-Host). La réponse contient la liste des protocoles d'établissement supportés, ce qui permet l'établissement d'un canal sécurisé entre l'initiateur et le répondeur/destinataire.

Si le destinataire ne prend pas en charge le protocole SCOUT, alors il répond de manière négative avec un paquet d'erreur, ou ne répond pas. Dès la réception d'un paquet d'erreur, l'initiateur passe directement à la seconde phase de découverte. Dans le cas où l'initiateur ne reçoit aucune réponse, il émet à nouveau plusieurs fois la requête (arbitrairement 3 fois) avant de considérer que c'est un échec et de passer à la seconde phase de découverte.

2) Seconde phase de découverte de SCOUT : routeur au voisinage de l'hôte d'extrémité

Dans la seconde phase de découverte, l'initiateur contacte le routeur au voisinage du destinataire. Comme précédemment, si le routeur supporte SCOUT, alors il prend le rôle de répondeur, répond positivement à la requête et un canal sécurisé est établi entre l'initiateur et le répondeur. Dans le cas où le routeur ne supporte pas SCOUT il ignore la requête ou alors il retourne une erreur. Après plusieurs retransmissions, l'initiateur considère que la seconde phase de découverte est un échec, il ne peut donc pas sécuriser les données et ajoute alors une règle pour rejeter les données ultérieures de la conversation concernée.

Le protocole SCOUT utilise les adresses source et destination des paquets échangés pour déterminer les données nécessitant de la sécurité. L'administrateur peut ainsi configurer différentes interfaces réseaux avec différentes politiques de sécurité. En utilisant deux adresses IP sur une seule interface, il est possible de faire transiter, via la même interface, des données à sécuriser et des données sans besoin de sécurité, en liant par la configuration l'adresse à sécuriser au dæmon SCOUT.

Le détail des requêtes des premières et secondes phases et du format des réponses est donné dans la prochaine section décrivant comment le protocole générique SCOUT est instancié avec IPv6 pour former le protocole SCOUT6.

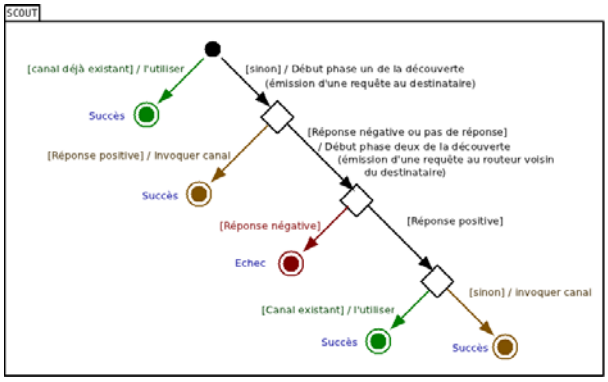


Figure 3. Algorithme de SCOUT

III. SCOUT AVEC IPv6 : LE PROTOCOLE SCOUT6

Le protocole SCOUT désigne les grandes lignes du fonctionnement permettant à des nœuds communicant de sécuriser les données : il définit comment un nœud initiateur contacte un nœud répondeur et découvre ses capacités de sécurisation pour ensuite déclencher l'établissement d'un canal sécurisé. Cette section expose la manière dont les spécifications du protocole SCOUT sont utilisées avec les particularités du protocole réseau Internet Protocol version 6 (IPv6). Cette réunion forme un nouveau protocole appelé SCOUT6. La partie générique de découverte est désignée par le nom « SCOUT », la partie d'adaptation à IPv6 par « SCOUT6 ».

Conformément à la RFC 6434 [20], les nœuds IPv6 devraient supporter IPsec. Le protocole d'établissement actuellement le plus employé pour établir les canaux sécurisés avec IPsec est le protocole IKEv2, d'où le choix de notre implémentation de privilégier l'usage de ce protocole devant les autres protocoles potentiels tels que KINK et IKE. L'administrateur peut cependant configurer le dæmon SCOUT6 afin de sélectionner préférentiellement d'autres protocoles.

A. SCOUT6 dans le détail

Le protocole IPv6 permet d'adjoindre des données supplémentaires aux paquets dans les « extensions headers ». Ces entêtes optionnels ne sont pas destinés aux applications utilisateurs et sont gérés directement par les services du système d'exploitation, afin d'adapter le comportement du système et notamment de sa pile réseau, en fonction des besoins exprimés dans les entêtes. La conception de SCOUT6 utilise deux entêtes d'extensions détaillés plus loin : le hop-by-hop option et le Destination Option. Ces extensions permettent d'étendre IPv6 avec SCOUT6, tout en maintenant une cohabitation avec les nœuds réseaux non compatibles avec SCOUT6.

Avec SCOUT6, nous avons besoin d'échanger 3 informations différentes: une pour la requête de la phase initiale (où l'initiateur demande au destinataire les protocoles d'établissement supportés), une pour la requête de la seconde phase (où l'initiateur demande ces informations au routeur voisin du destinataire) et une pour la réponse à ces deux requêtes (où le répondeur renvoie les protocoles supportés à l'initiateur).

1) Destination Option Query (DOq)

Le paquet DOq permet à l'initiateur de demander au destinataire les protocoles d'établissement pris en charge. Il est formé d'un entête IPv6 suivi des données d'extension de la Destination Option, comme présenté figure 4. La spécification IPv6 [20] précise qu'un paquet de ce type doit être examiné par la destination, les équipements intermédiaires n'ayant pas besoin d'effectuer de traitement particulier pour ce paquet.

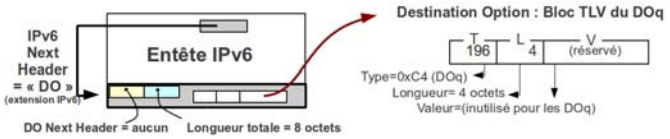


Figure 4. Format d'un paquet IPv6 complété de l'entête Destination Option Query (DOq)

TABLE IV. TRAITEMENT PAR DEFAUT DES « DESTINATION OPTION » EN FONCTION DU CHAMP TYPE

Intervalle pour les types du DO	Comportement par défaut du système d'exploitation
0..63	Ignorer l'extension du paquet
64..127	Détruire le paquet
128..191	Renvoyer un paquet ICMPv6 Parameter Problem à l'expéditeur
192..255	Renvoyer un paquet ICMPv6 Parameter Problem à l'expéditeur si et seulement si le destinataire n'est pas un groupe multicast

Les données d'extension sont au format Type-Length-Value où un octet code le type de paquet, un autre la taille du champ Valeur, suivi éventuellement de données dans le champ Valeur. L'octet de type oriente le traitement par défaut du système d'exploitation, comme expliqué dans la table 4, dans le cas où le paquet n'a pas été reconnu et intercepté par l'un des services du système.

Pour le DOq de SCOUT6, nous utilisons la valeur de type expérimentale 196[1] qui n'est actuellement assignée à aucun service par l'IANA [21], l'organisme de normalisation d'Internet gérant notamment les assignations de constantes pour IPv6. Cette valeur influe sur le traitement du paquet par le système l'ayant reçu : si un dæmon SCOUT6 s'exécute sur la machine, il intercepte le paquet et envoie un DOr comme expliqué plus bas. Si aucun dæmon SCOUT6 n'intercepte le paquet, alors le système d'exploitation envoie un paquet ICMPv6 Parameter Problem à l'initiateur qui sait ainsi que la requête a aboutie négativement : la première phase de découverte est un échec car le destinataire ne supporte pas SCOUT6.

2) Router Alert Query (RAq)

Dans le cas où l'initiateur n'a aucune réponse aux requêtes DOq envoyées ou si la réponse est un ICMP Parameter Problem, alors la première phase est un échec et l'initiateur commence la seconde phase de découverte : il tente de contacter le routeur voisin du destinataire. Pour cela, il émet un paquet de type Router Alert [22] contenant le sous-type « SCOUT6 ». Ce paquet Router Alert est un paquet IPv6 avec l'extension hop-by-hop, ce qui indique à chaque routeur intermédiaire d'analyser les données contenues dans l'extension. Nous utilisons le numéro de sous-type 42 qui n'est actuellement assigné à aucun autre usage par l'IANA. L'initiateur envoie le paquet RAq au destinataire. Les routeurs sur le chemin entre l'initiateur et le destinataire analysent le paquet ; celui-ci ne leur étant pas destiné, le paquet est transmis sans modification par chaque routeur au routeur suivant, qu'il y ait ou non un dæmon SCOUT6 actif sur ces routeurs.

Le cas du dernier routeur (le voisin du destinataire) est différent. Si ce routeur exécute un dæmon SCOUT6, alors il intercepte la requête RAq et y répond avec un DOr comme détaillé plus loin. S'il n'exécute aucun dæmon SCOUT6, le paquet est transmis par le routeur non compatible à l'hôte destinataire qui l'ignore : le paquet RAq ne contient aucune donnée autre que l'extension Router Alert.

[1] Cette valeur pourrait éventuellement évoluer en fonction du contexte d'utilisation de SCOUT6 dans le futur.

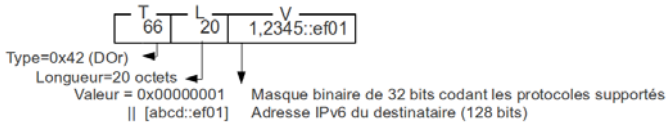


Figure 5. Format de l'entête DOr

L'emploi d'une extension nécessitant d'être analysée par chaque routeur intermédiaire peut sembler lourde. Mais en pratique, la RFC 6434 [20] précise qu'un routeur ne gérant pas spécifiquement les protocoles concernés par les RA n'a pas besoin d'analyser cette extension, les paquets sont alors transmis sans traitement ni délai supplémentaire. De plus, certains fournisseurs d'accès à l'Internet éliminent dès l'entrée dans leurs réseaux les paquets avec l'extension RA. Pour pallier ce problème, nous proposons une solution dans la section V.

3) Destination Option Response (DOr)

Quand le dæmon SCOUT6 reçoit d'un initiateur une requête DOq ou RAq, il répond avec un paquet appelé « Destination Option Response ». Ce paquet IPv6 destiné à l'initiateur contient l'adresse du répondeur et une extension Destination Option avec le type 66. Cette valeur non assignée par l'IANA permet à un système n'exécutant pas SCOUT de rejeter silencieusement ce paquet DOr, contrairement au DOq où un ICMP Parameter Problem est renvoyé.

Le DOr, dont le format est représenté figure 5, contient dans son champ Valeur deux informations. La première est un masque binaire de 32 bits où chaque bit code le support d'un protocole d'établissement. Un bit de poids faible (Least Significant Bit, LSB) à 1 signifie que IKE est supporté, le bit suivant sert pour IKEv2, le suivant pour KINK et ainsi de suite. Le bit ayant le poids le plus fort (Most Significant Bit, MSB) est un bit utilisé pour des usages expérimentaux. Tous les autres bits sont actuellement inutilisés et réservés pour le support futur d'autres protocoles d'établissement.

La deuxième information contenue dans le champ Valeur du DOr émis par le répondeur est constituée des 16 octets de l'adresse IPv6 du destinataire, c'est-à-dire l'adresse de destination du paquet DOq ou RAq. Cette information permet à l'initiateur d'identifier quelle conversation est associée au DOr. L'adresse IPv6 source du paquet DOr est l'adresse du répondeur (H2 ou R2 dans la topologie figure 2). L'adresse IPv6 contenue dans le champ DOr est l'adresse du destinataire (H2 dans la topologie). Cette adresse est soit la même adresse quand le répondeur est aussi le destinataire (H2 en modes Host-Host et Router-Host), soit une adresse différente quand le répondeur est le routeur voisin du destinataire et non le destinataire lui-même (H2 sécurisé par R2 en modes Host-Router et Router-Router). Cela permet ainsi à l'initiateur de savoir si il communique avec le destinataire ou le routeur voisin du destinataire.

A la réception du DOr, si l'initiateur dispose déjà d'un canal sécurisé vers le répondeur, alors l'initiateur peut réutiliser ce canal existant pour économiser des ressources : il n'invoque pas la création d'un nouveau canal mais ajoute uniquement une nouvelle règle de routage.

B. *Mise en œuvre du dæmon SCOUT6*

Le protocole SCOUT6 présenté ici est complété d'une implémentation gratuite et Open-Source (licence double LGPL 2 et CeCill-B) disponible à l'URL : http://www.recherche.enac.fr/leopart/~avaret/scout6

Cette implémentation a permis de valider la mise en œuvre du protocole dans une configuration réseau réaliste ; les résultats de l'évaluation de performance sont disponibles plus loin dans cet article. Elle est composée de deux parties, l'une étant le cœur de l'implémentation du protocole SCOUT6 écrit en langage C, l'autre étant le code nécessaire à l'intégration du cœur avec le système d'exploitation Debian que nous utilisons pour la validation et constitué de scripts bash.

1) *Accrochage du dæmon SCOUT6 au système d'exploitation*

La figure 6 présente la manière dont le dæmon SCOUT6 interagit avec la pile réseau du système : il capture les paquets entrants et sortants le concernant et les traite en conséquence. Ainsi, en reprenant l'exemple de la topologie de la figure 2 présentant un réseau basique H1 – R1 – R2 – H2, dans le cas de H1 supportant SCOUT, où le nœud est inspirateur et initiateur, le dæmon sur H1 doit intercepter les paquets sortants issus des applications s'exécutant sur H1 afin de déclencher le processus de découverte SCOUT ; l'interception s'effectue au point de capture n°1 de la figure 6. Lorsque H1 ne supporte pas SCOUT contrairement à R1, le dæmon SCOUT sur R1 a le rôle d'initiateur non-inspirateur et doit intercepter les paquets de H1 en transit sur le routeur R1. Dans tous les cas, l'interception n'est effectuée que s'il n'existe pas déjà de canal sécurisé pour le paquet.

Lorsqu'H2 supporte SCOUT, le dæmon sur H2 a les rôles de répondeur et de destinataire, il doit donc réceptionner les requêtes entrantes et y répondre de manière adéquate. Lorsqu'H2 ne supporte pas SCOUT mais que R2 supporte SCOUT, le dæmon sur R2 a le rôle de répondeur non-destinataire, il doit alors intercepter et répondre aux requêtes émises durant la seconde phase de découverte vers les destinataires à sécuriser.

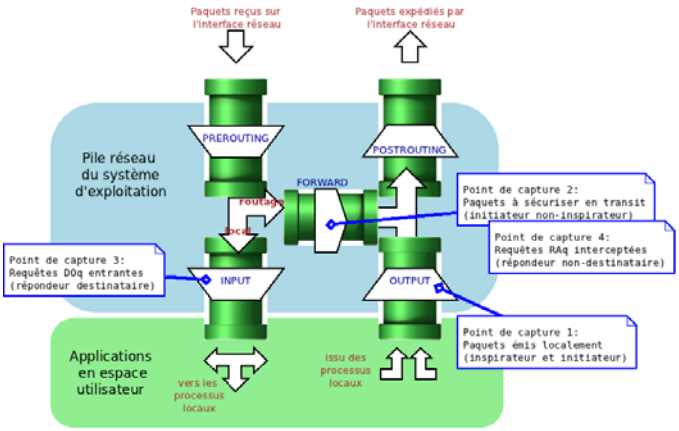


Figure 6. Points d'accrochage du dæmon SCOUT6 sur le système d'exploitation

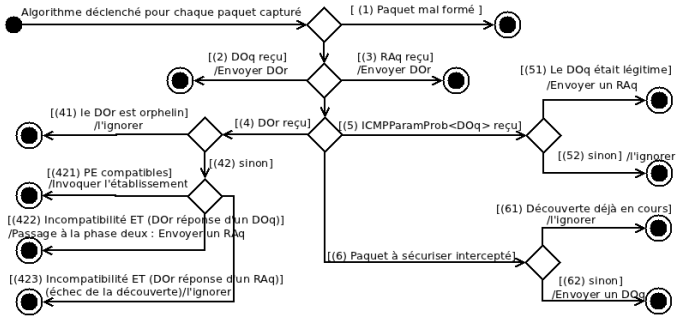


Figure 7. Algorigramme du dæmon SCOUT6

2) *Algorithme central du dæmon SCOUT6*

La figure 7 résume l'algorithme que met en œuvre le dæmon SCOUT6. Elle est une conséquence de l'algorithme de SCOUT présenté figure 3. Après avoir extrait la classe du paquet capturé, elle le traite en conséquence : émission d'un DOr en réponse aux DOq et RAq, émission d'un DOq à la capture de données non sécurisées ou encore traitement du DOr et invocation éventuelle du canal sécurisé.

C. *Mise en œuvre de SCOUT6*

Dans les sections suivantes, nous présentons la séquence pour deux modes de SCOUT : le mode **Host-Host** puis le mode **Router-Router**. Afin d'illustrer ces deux exemples, nous utilisons la topologie présentée figure 8 ci-dessous. Dans cette topologie, les systèmes exécutant un dæmon SCOUT6 sont indiqués à l'aide d'une étoile « * » dans leur nom : H*10, R*1, R*2, H*21. Les autres systèmes sont compatibles avec IPv6 mais non avec le protocole SCOUT6.

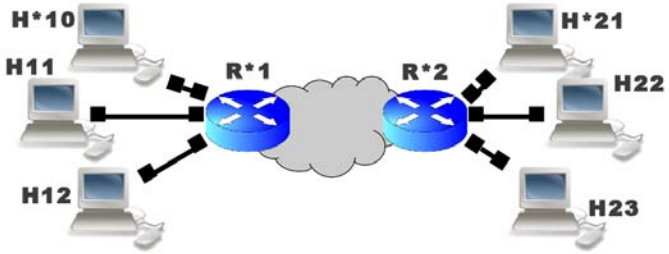


Figure 8. Topologie de démonstration des modes de SCOUT

1) *Le mode SCOUT Host-Host*

La figure 9 illustre un scénario typique de découverte des capacités de sécurisation avec le protocole SCOUT6. Dans ce scénario, les deux hôtes d'extrémité H10 et H21 supportent tous deux SCOUT6 et un protocole d'établissement « PE ». Quand le premier hôte inspirateur/initiateur H10 émet son premier paquet vers le destinataire/répondeur H21, le dæmon intercepte ce paquet et envoie une requête DOq à H21. En réponse, H21 renvoie à H10 un DOr contenant son adresse IPv6 et la liste des protocoles supportés. Le protocole « PE » est alors appelé par H10 pour établir un canal sécurisé de communication entre H10 et H21 et ajouter les règles de routage. Sur l'hôte H21, l'établissement du canal sécurisé entraîne aussi l'ajout de règles de routage duales. Ensuite, tout paquet transitant entre H10 et H21 est sécurisé au travers du canal.

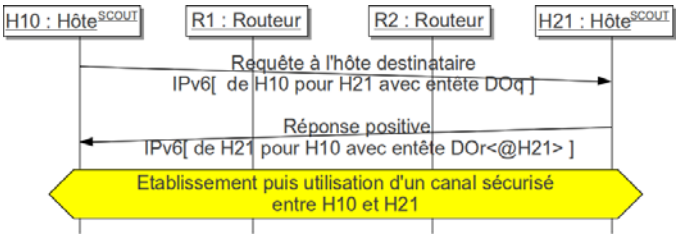


Figure 9. Séquence de découverte aboutissant au mode Host-Host

2) Le mode SCOUT Router-Router

Dans le scénario Host-Host précédent, les routeurs intermédiaires pouvaient ou non supporter le protocole SCOUT6, cela n'influait pas sur la découverte. Dans le scénario suivant présenté figure 10, les hôtes d'extrémité H11 et H22 ne supportent pas SCOUT6 contrairement à leurs routeurs voisins R1 et R2.

Lors de l'émission par l'inspirateur H11 du premier paquet non sécurisé vers le destinataire H22, le paquet est intercepté par l'initiateur R1 qui lance la première phase de découverte : une requête DOq est envoyée à H22. Ne supportant pas SCOUT6, H22 retourne à R1 un paquet ICMP Parameter Problem, ainsi l'initiateur R1 sait que la première phase de découverte est un échec.

Il passe alors à la deuxième phase de découverte et envoie un RAq à H22. Les routeurs intermédiaires transmettent le RAq sans modification ni interception : ils ne savent pas gérer les paquets SCOUT6 ou ne sont pas concernés (car pas voisins de H22). R2 intercepte ce RAq et accepte de prendre en charge la sécurisation en renvoyant une réponse DOr à R1.

L'initiateur R1 établit alors un canal sécurisé avec le répondeur R2. Les paquets échangés ensuite entre H11 et H22 transitent de manière sécurisée sur ce canal entre R1 et R2. Ce canal sécurisé peut servir pour d'autres communications transitant entre R1 et R2 : le multiplexage de nombreuses communications au sein du même canal évite l'augmentation incontrôlée du nombre de canaux dans le réseau.

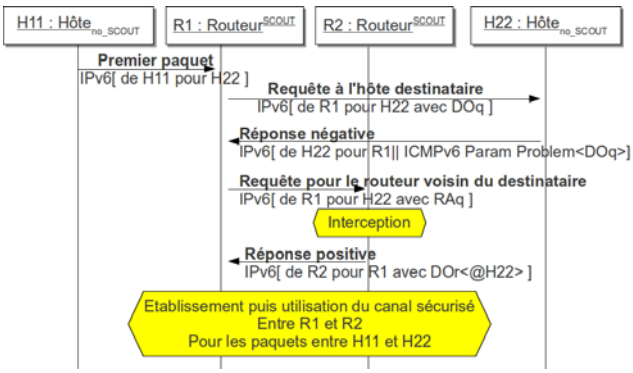


Figure 10. Séquence de découverte aboutissant au mode Router-Router

IV. SCOUT6 EN PRATIQUE

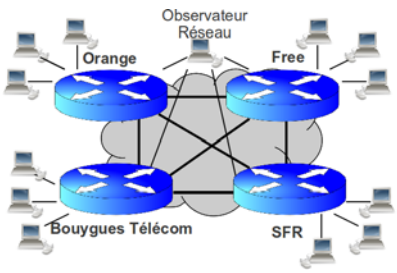
A. Evaluation de l'impact de SCOUT6 en terme de capacité et de délai réseau

Afin de découvrir les différentes possibilités de sécurisation des nœuds, le protocole SCOUT a besoin d'échanger les messages sur le réseau non sécurisé. L'implémentation SCOUT6 est basée sur les DOq, RAq et DOr présentés dans la section précédente. Ces paquets IPv6 ont une taille respectivement de 48, 48 et 64 octets. Dans le meilleur cas, pour une découverte, il y a un paquet DOq et un paquet DOr, soit 112 octets, tandis que dans le pire cas, il y a 3 paquets DOq perdus, 3 RAq et 3 DOr perdus, ce qui représente un total de 480 octets pour les 9 paquets.

Afin d'évaluer les performances de notre implémentation de SCOUT6, nous avons élaboré une topologie d'un réseau IPv6 avec 4 groupes de nœuds représentant 4 fournisseurs d'accès Internet (FAI) français : Orange, Bouygues Télécom, SFR et Free. Chaque communication entre ces fournisseurs a été pénalisée d'un délai de transmission des messages, mesuré le 6 décembre 2011 à 11h lors d'une étape préalable à nos expérimentations à l'aide des commandes traceroute et ping. Pour cela nous sommons les délais entre l'observateur réseau et les domaines d'émission et de réception. Cette méthode de mesure active permet d'obtenir une estimation réaliste du délai présent entre les différents FAI au moment de la mesure. La figure 11 contient le tableau récapitulant les résultats de nos mesures de délai et de gigue.

La topologie expérimentale que nous avons utilisée pour valider l'implémentation est aussi représentée dans cette figure 11. Chaque groupe a été affecté de 256 clients IPv6. Le réseau expérimental a été émulé à l'aide de VirtualBox [23] et de 4 machines virtuelles mono-core dotées de 256 Mo de RAM, d'un processeur cadencé à 2GHz et de cartes Ethernet virtuelles liées entre elles. Le système d'exploitation Linux Debian a été complété du dæmon SCOUT6, lié au système à l'aide du module noyau netfilter et de commandes iptables. De plus, le module Traffic Control Network Emulator (netem [24,25]) a été utilisé pour ajouter les délais et la gigue de transmission.

Enfin, afin de simuler du trafic réseau et de déclencher la découverte SCOUT, chaque machine virtuelle a exécutée un script générant toutes les 20 secondes un ping d'un client d'un quelconque FAI vers un client d'un autre FAI. La table V résume les délais attendus par la commande ping sur le réseau de la figure 11.



Délais moyens et gigue (en ms)

	Orange	Free	SFR
BouygT	41,1 / 1,7	37,6 / 0,5	39 / 0,4
Orange		41,3 /1,6	42,7 / 1,8
Free			39,2 / 0,6

Figure 11. Réseau avec 4 FAI

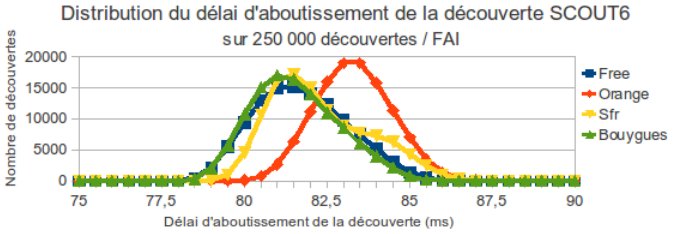


Figure 12. Distribution des délais mesurés pour la découverte SCOUT6

Chaque ping émet un paquet ICMP Echo Request qui est intercepté par le dæmon SCOUT6 local, ce qui déclenche la phase de découverte et aboutit à la création d'un canal sécurisé entre les nœuds. C'est le mode SCOUT6 Host-Host qui est utilisé dans cette phase d'évaluation de performance. Le dæmon a été instrumenté afin de mesurer la durée nécessaire pour chaque découverte.

Le graphique de la figure 12 montre la distribution obtenue avec les résultats de nos mesures de délais sur le dæmon SCOUT6. Le temps de traitement des paquets SCOUT6 est inférieur à quelques millisecondes et donc négligeable lorsqu'il est comparé aux délais d'acheminement des paquets entre les différents FAI, ces délais étant en moyenne autour de 40 ms par transmission de paquet. Nous pouvons donc en conclure que l'implémentation SCOUT6 et le service de découverte automatique qu'il offre ne pénalise pas les performances du réseau de test.

Enfin, au cours de tests de résistance (stress-tests) additionnels, nous avons mesuré le temps de traitement du dæmon pour 1 000 000 de paquets SCOUT6 consécutifs, ce qui nous a permis d'établir que le temps moyen de traitement d'un paquet est de l'ordre de 500 nanosecondes. La différence des ordres de grandeur entre l'écart-type des distributions figure 12 (quelques millisecondes) et le temps de traitement d'un paquet SCOUT6 s'explique par l'imprécision de l'horloge utilisée pour établir les distributions figure 12: la fonction de temps utilisée alors retourne une durée avec une précision de l'ordre de quelques millisecondes. Une implémentation sur une plateforme temps-réel et avec des techniques de métrologie plus perfectionnées nous permettrait de mesurer plus précisément la durée de traitement en environnement réel. Ce travail est actuellement à l'étude mais hors du cadre de cette publication.

Nos expérimentations n'ont pas révélées de surcharge supplémentaire du routeur mesurable au niveau du CPU engendrées par le traitement des différents paquets SCOUT6 générés lors de la négociation des canaux sécurisés.

TABLE V. PREVISION DES DELAIS SELON LA SECURISATION

Mode	Délai attendu sur notre réseau
Ping non sécurisé	**80 ms** = 40 ms x (1 request+1 reply)
Avec tunnel VPN configuré manuellement	**Quelques minutes** de configuration manuelle + **240 ms** (dont 4 paquets d'initialisation IKEv2)
SCOUT6 en mode Host-Host ou Router-Host	**320 ms** (dont les 80 ms de découverte SCOUT6 présentés figure 12)
SCOUT6 en mode Host-Router ou Router-Router (sans perte)	**400 ms** : >DOq, <ICMPpp, >RAq, <DOr, >IKE_SA_INITi, <INITr, >AUTHi, <AUTHr, >ESP(echo request), <ESP (echo reply)

B. *Complexité des opérations avec et sans SCOUT*

1) *Complexité dans un réseau centralisé*

Afin d'évaluer de manière qualitative les bénéfices d'utilisation de SCOUT6, la table 6 résume les coûts (en nombre de lignes de configuration) pour un réseau centralisé sur le modèle du Client/Serveur. Les coûts sont évalués lors de l'ajout et de la suppression d'un client et d'un serveur. On suppose ici que le client est l'initiateur du canal sécurisé et qu'il initie un canal par serveur contacté. N désigne le nombre de clients, potentiellement élevé.

TABLE VI. EVALUATION DES COUTS DANS LE MODELE CLIENT/SERVEUR

	Sans le protocole SCOUT6	**Avec le protocole SCOUT6**
Ajouter un client (N+1)	Configurer ce (N+1)ème client et configurer le serveur [O(2)]	Installer SCOUT6 sur le client [O(1)]
Supprimer le client N ou modifier les informations de sécurité	Reconfigurer les deux extrémités sur le client N et le serveur [O(2)]	Reconfigurer le client N et éventuellement le serveur (pour accepter les nouveaux certificats…) [O(2)]
Ajouter un nouveau serveur	Sur le serveur ajouté, configurer un canal par client potentiel et sur chaque client, configurer le canal [O(2*N)]	Installer SCOUT6 sur le serveur [O(1)]
Enlever un serveur	Reconfigurer chaque client [O(N)]	Rien à faire [O(0)]

2) *Complexité dans un réseau décentralisé*

Dans un réseau pair-à-pair avec n nœuds interconnectés, chaque nœud peut communiquer avec (n-1) autres nœuds. Dans le pire cas, cela représente un total de n(n-1)/2 canaux bidirectionnels sécurisés à configurer puis établir. Sans le protocole SCOUT, chaque canal devant être configuré aux deux extrémités, cela représente un total de (n^2-n) canaux à sécuriser.

Avec SCOUT, l'administrateur doit installer et configurer le dæmon sur chaque nœud, sans préciser à chaque fois les (n-1) extrémités de canaux potentiels. Ainsi, la configuration doit être effectuée seulement n fois et non plus (n^2-n) fois.

De plus, avec SCOUT, seuls les canaux réellement utilisés sont établis, au moment du premier échange : si deux nœuds ne communiquent jamais ensemble, aucun canal n'est créé, ce qui préserve les ressources réseau et de calcul.

C. *Considérations de sécurité autour du protocole SCOUT6*

Le protocole SCOUT6 a été conçu pour rechercher les possibilités de sécurisation des nœuds et invoquer, si possible, l'établissement de canaux sécurisés entre les nœuds. La sécurité des données communiquées est assurée concrètement par le protocole d'établissement et les algorithmes qu'il négocie.La question à laquelle nous apportons des éléments de réponse dans cette section est : le protocole SCOUT6 introduit-il des vulnérabilités supplémentaires par rapport à une configuration statique ?

Les attaques par injection et rejeu de paquets sont inopérantes par conception : le protocole utilise ces principes afin de fonctionner correctement ; le rejeu est ainsi utilisé pour

compenser l'absence de garantie d'acheminement des paquets du protocole IPv6. Les attaques d'altération et de vol de paquets pourraient empêcher une découverte, mais alors l'attaquant a aussi la capacité de bloquer le protocole d'établissement du canal, l'impact est donc identique pour une configuration statique et pour une découverte par SCOUT6.

Une vulnérabilité du protocole a été détaillée dans une publication [26] : un attaquant bloquant les DOq et laissant passer les RAq pourrait amener les données à être transmises de manière non sécurisées entre les hôtes et leurs routeurs voisins. Cette vulnérabilité peut être réduite en assurant la sécurité du périmètre local des réseaux d'extrémité ou encore en imposant l'usage exclusif du mode Host-Host.

L'authentification dépend des choix de l'administrateur réseau : elle permet de garantir à l'établissement du canal sécurisé l'authenticité de ses extrémités mais nécessite de déployer préalablement du matériel d'authentification. Dans nos expérimentations, nous avons choisi d'employer un secret partagé sur l'ensemble des hôtes à sécuriser avec SCOUT. L'emploi de méthodes basées sur des certificats est aussi possible : bien que plus complexes à déployer, elles sont aussi généralement plus faciles à maintenir. Enfin, l'administrateur peut forcer la configuration des protocoles d'établissement pour ne pas authentifier les nœuds. Dans tous les cas, l'authentification dépend du protocole d'établissement et non du protocole SCOUT, afin d'éviter d'effectuer une double authentification redondante (découverte puis établissement).

Enfin, le protocole SCOUT6 est vulnérable à l'attaque de l'homme du milieu, mais cette vulnérabilité peut être éliminée en configurant le protocole «PE» pour authentifier les extrémités du canal à sécuriser.

V. PERSPECTIVES D'EVOLUTIONS DU PROTOCOLE SCOUT6

Le protocole SCOUT6 présenté dans les sections précédentes a été implémenté et validé avec succès sur des réseaux isolés du monde réel. Cependant, certaines particularités de ce protocole peuvent être incompatibles avec des contraintes imposées par certains fournisseurs d'accès, acteurs incontournables pour les connexions longues distances à travers le réseau Internet. Cette section présente quelques uns de ces problèmes et deux solutions sont proposées pour y remédier.

A. Le problème des Router Alert rejetés par les fournisseurs d'accès

Certains fournisseurs d'accès Internet ajoutent des règles de filtrage afin de détruire dès l'entrée dans leur réseau tous les paquets porteurs de l'extension Router Alert (RA). Cela permet d'éviter de réduire les performances du réseau, les routeurs n'ayant plus à gérer « en profondeur » les paquets IP complétés de cette extension. Ce filtrage est aussi une protection contre les attaques qui exploiteraient l'option RA comme vecteur d'attaque. Mais ce filtrage des RA entraîne la destruction des paquets RAq utilisés pour la découverte de la seconde phase de SCOUT6.

Nous avions pensé initialement à créer un nouveau type de paquet SCOUT6 dérivé du DOq, mais l'optimisation développée dans la section suivante résoud le problème d'une manière plus efficace.

B. Accélération de la seconde phase par interception du résultat de la première phase

Dans le cas où l'hôte destinataire (H2) ne supporte pas SCOUT6 et où le routeur voisin de cet hôte (R2) supporte SCOUT6, la réception du DOq entraîne l'émission d'un paquet ICMPv6 Parameter Problem contenant ce DOq. Le déroulement de la découverte de SCOUT6 présenté plus haut passe d'abord par la réception par l'initiateur de cet ICMP Param Problem<DOq>, du déclenchement de la deuxième phase et de l'émission d'un RAq vers le routeur qui répond alors avec un DOr.

La figure 13 montre comment raccourcir cette séquence. Le paquet ICMPv6 Parameter Problem<DOq> transite forcément par le routeur destinataire. Celui-ci peut intercepter ce message, le traiter comme s'il avait reçu un RAq et donc renvoyer immédiatement un DOr adéquat. Cela permet de sauter les étapes de réception de la réponse négative de la première phase et d'émission de la requête de la deuxième phase. La figure 13 représente une découverte aboutissant au mode Router-Router avec cette optimisation.

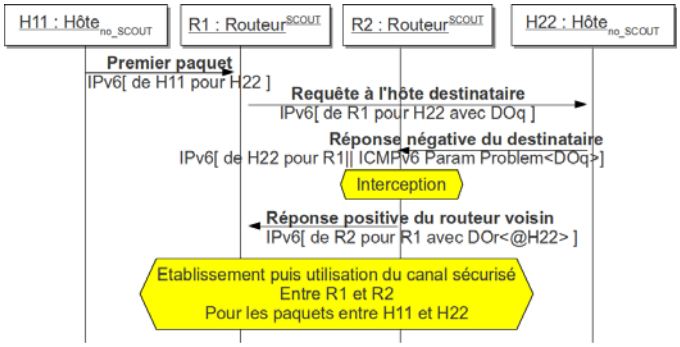


Figure 13. Séquence optimale pour SCOUT6

Les avantages de ce raccourci sont les suivants. Tout d'abord, il n'est plus nécessaire d'envoyer de RAq, ce qui résout le problème de filtrage de certains fournisseurs d'accès, présenté ci-dessus. Cela permet d'économiser l'acheminement de deux paquets (un retour pour la réponse négative puis un aller pour le RAq) sur le réseau. De plus, le temps nécessaire à la découverte est réduit : dès le premier DOq transmis, l'initiateur est notifié soit par un DOr venant du nœud destinataire (H22) ou de son routeur (R2) soit par un ICMP Parameter Problem, ce dernier indiquant que SCOUT n'est pris en charge ni par H22 ni par R2 et donc l'échec de la découverte. Ainsi, un seul aller-retour est nécessaire pour tous les résultats possibles de la découverte.

Cependant, des règles de filtrage spécifiques ou un système d'exploitation partiellement compatible avec la spécification IPv6 pourraient supprimer les paquets contenant les Destination Option (notamment les DOq) et ainsi ne pas renvoyer de paquet ICMP Parameter Problem. Dans ce cas de figure, cette optimisation est inapplicable.

C. *Adaptation de SCOUT6 avec le protocole UDPv6*

Afin de pallier les problèmes de filtrage des paquets contenant des extensions hop-by-hop et destination option, nous avons envisagé d'utiliser une encapsulation de l'option dans un datagramme UDP, comme présenté dans la figure 14.

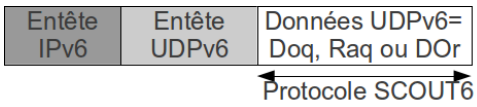


Figure 14. Encapsulation des options pour SCOUT6 dans des datagrammes UDP

L'avantage de cette solution est évidemment de garantir une « apparence normale » aux paquets IP et donc de permettre leur traitement avec les techniques couramment utilisées de filtrage des datagrammes UDP sur les routeurs, pare-feux et autres équipements réseaux. De plus, le protocole UDP étant très proche en IPv4 et IPv6, cette solution ouvre des perspectives d'application du protocole SCOUT avec le protocole IPv4. Cependant, cela pourrait poser des problèmes de mise en œuvre, notamment pour la traversée des Network Address Translation (NAT) qui s'avèrent peu compatibles avec IPsec [27].

VI. CONCLUSIONS ET TRAVAUX FUTURS

Nous avons présenté dans cet article le nouveau protocole SCOUT qui découvre automatiquement les possibilités de sécurisation sur un nœud d'un réseau. Le protocole SCOUT peut non seulement invoquer l'établissement d'un canal sécurisé entre deux hôtes mais aussi invoquer cet établissement avec les routeurs de proximité de ces hôtes si les hôtes ne disposent pas des capacités nécessaires à la sécurisation. De plus, le protocole SCOUT réduit la charge d'administration de la sécurité des réseaux et augmente la mise à l'échelle des solutions de sécurité réseau.

La concrétisation du protocole SCOUT avec IPv6 complète le protocole SCOUT générique avec un protocole concret nommé SCOUT6. Nos expérimentations indiquent une surcharge négligeable pour le réseau en terme de données échangées et de délais additionnels. Notre article se conclut avec des perspectives d'amélioration du protocole.

Cependant, nous poursuivons nos recherches afin de valider le protocole sur des topologies plus complexes et en environnement réel, de réduire encore les vulnérabilités découvertes et leur impact et de procéder à une validation formelle de ces solutions. De plus, nous travaillons sur la prise en charge de nouveaux protocoles d'établissement de sécurisation : les expérimentations actuelles reposent sur un seul protocole d'établissement alors que le protocole a été conçu pour en supporter plusieurs. Un autre axe de travail concerne la sécurisation multicast : le passage à l'échelle apporté par SCOUT pour les communications unicast pourrait être étendue aux communications où plus de deux entités communiquent simultanément.

REFERENCES

[1] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard), January 2006

[2] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005.

[3] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998.

[4] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409 (Proposed Standard), November 1998. Obsoleted by RFC 4306[13].

[5] S. Sakane, K. Kamada, M. Thomas, and J. Vilhuber. Kerberized Internet Negotiation of Keys. RFC 4430 (Proposed Standard), March 2006.

[6] S. Kent. IP Authentication Header. RFC 4302 (Proposed Standard), December 2005.

[7] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard), December 2005.

[8] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-Point Tunneling Protocol (PPTP). RFC 2637 (Informational), July 1999.

[9] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation (GRE). RFC 1701 (Informational), October 1994.

[10] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard), June 2004. Updated by RFC 5247.

[11] H. Tschofenig, D. Kroeselberg, A. Pashalidis, Y. Ohba, and F. Bersani. The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method. RFC 5106 (Experimental), February 2008.

[12] D. Piper. The Internet IP Security Domain of Interpretation for ISAKMP. RFC 2407 (Proposed Standard), November 1998. Obsoleted by RFC 4306[13].

[13] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), December 2005. Updated by RFC 5282.

[14] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen. Internet Key Exchange (IKEv2) Protocol. RFC 5996 (Proposed Standard), September 2010. Updated by RFC 5998.

[15] Website « Kerberos : The Network Authentication Protocol », http://web.mit.edu/kerberos/, 2011/12/13

[16] M. Richardson and D.H. Redelmeier. Opportunistic Encryption using the Internet Key Exchange (IKE). RFC 4322 (Informational), December 2005.

[17] The Openswan project website, https://www.openswan.org/projects/openswan/, 2012/03/09

[18] Website of Strongswan, the Open Source IPsec-based VPN Solution, http://www.strongswan.org/, 2012/03/09

[19] R. Arends, R. Austein and al. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005

[20] E. Jankiewicz, J. Loughney, T. Narten. IPv6 Node Requirements. RFC 6434 (Informational), December 2011.

[21] Internet Assigned Numbers Authority website, www.iana.org, 2011/12/13

[22] C. Partridge and A. Jackson. IPv6 Router Alert Option. RFC 2711 (Proposed Standard), October 1999.

[23] VirtualBox website, www.virtualbox.org, the 2011/12/13

[24] The Network Emulation webpage, http://www.linuxfoundation.org/collaborate/workgroups/networking/netem, the 2011/12/13

[25] Linux Advanced Routing & Traffic Control website, lartc.org, the 2011/12/13

[26] A. Varet and N. Larrieu, Security Capability discovery protocol Over Unsecured IP-based Topologies, Mai 2012

[27] T. Kivinen and B. Swander, Negociation of NAT-Traversal in the IKE, RFC 3947 (Proposed Standard), January 2005.