Security capability discovery protocol over unsecured IP-based topologies

Antoine Varet, Nicolas Larrieu

Security capability discovery protocol over unsecured IP-based topologies

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

***Abstract*—Network security protocols need high configuration workload. We propose a new protocol to reduce this issue: our "Security Capabilities Over Unsecured Topology" (SCOUT) protocol has been designed to search at each IP-based node for the security supported mechanisms on the remote nodes and then to invoke an adequate security channel establishment mechanism. If the remote node does not support any compatible security features, the SCOUT protocol will try to secure the flow at least up the router in the neighborhood of the node. This protocol avoids the administrator manually managing a tunnel for each couple of nodes. This reduces administrator workload and increases network security deployment scalability. We complete the SCOUT protocol presentation by an evaluation of experimental performance and an analysis of vulnerability.**

Keywords: network security, discovery, protocol, IPv6

I. INTRODUCTION

Security mechanisms are growing in usage in computer networks. The first reason comes from technical resource improvements: new processors are faster, memory capacities are growing everyday, powerful materials enable network throughput to be higher and higher… Security breaches are the second reason: whereas several years ago, network attacks were only broadcast in closed network security communities, nowadays they are often publicized in the media.

Nevertheless, security protocols are increasing in safety, security and efficiency. For instance, since its first version in 1995, the Secure Shell protocol (SSH [1]) has evolved until its current version and currently supports most new security algorithms. Another example is the Internet Protocol Security (IPsec [2]). The IPv6 standard requires this framework for security of IPv6 compliant systems. IPsec defines two modes of working, the tunnel and the transport mode. It is linked to dedicated protocols for establishing secure communication channels, such as the Internet Key Exchange (IKE [3]) protocol. Other protocols secure the data, such as the Encapsulating Security Payload (ESP [4]) and Authentication Header (AH [5]) protocols.

These protocols need high configuration workload and this is a restraint for deployment of security solutions. IPsec security requires configuring both end-systems with most information for each communication channel to secure. This can be complex for some network topologies. This creates a greater workload for network administrators.

We propose a new protocol to reduce this issue: our "Security Capabilities Over Unsecured Topology" (SCOUT) protocol has been designed to search at each IP-based node for the security supported mechanisms on the remote nodes and then to invoke an adequate security channel establishment mechanism. If the remote node does not support any compatible security features, the SCOUT protocol will try to secure the flow at least up the router in the neighborhood of the node. This protocol avoids the administrator manually managing a tunnel for each couple of nodes. This reduces administrator workload and increases network security deployment scalability.

A. *Several security establishment protocols*

Before securing the data, security mechanisms require the secure channel to be established. This establishment may be a part of the mechanism, such as for the Point-to-Point Tunneling Protocol (PPTP [6]) which uses a control channel over TCP to manage a Generic Routing Encapsulation (GRE [7]) tunnel encapsulating PPP packets. The Extensible Authentication Protocol (EAP [8]) is an authentication framework frequently used in wireless networks. It provides some methods to negotiate secure channels. The EAP-IKEv2 [9] is an extension from EAP based on the model of IKEv2, presented below.

Other frameworks are composed of a set of protocols with a subset dedicated to secure channel establishment. IPsec is an example of such frameworks. It uses Internet Security Association and Key Management Protocol [10] (ISAKMP) to manage the different channels on the network node. These channels can be established through the Internet Key Exchange (IKE) protocol or its successor the IKEv2 [11, 12] The ticket-based Kerberized Internet Negotiation of Keys (KINK [13]) is an alternative to the IKE protocols, centralized around Kerberos [14] servers to manage authenticity tickets. These establishment protocols require an external invocation to establish secure channels, as presented Fig 1.

Figure 1. Invocation, establishment then securing

IKE, IKEv2 and KINK protocols assume that a network administrator has configured the secure channels prior to using

them to securely exchange data. The SCOUT protocol we propose automates the configuration step, reducing network administrator workload. The Opportunistic Encryption (OE [15]) extension of the OpenSwan [16] and StrongSwan [17] IPsec implementations for Linux kernel provides secure communications on the fly, like SCOUT. Contrary to our SCOUT protocol, OE has been defined for IPv4 networks only; OE depends on a third party, which manages a DNS or DNSSEC [18] server with the pre-loaded keys; it does not enable data flows to be secured by intermediary routers and it is not compatible with different establishment protocols to be used on the same node.

II. SCOUT AND SCOUT6 IN DETAIL

A. *The generic SCOUT protocol*

The SCOUT protocol provides four different working modes, depending on SCOUT support on the network nodes. Fig. 2 illustrates a basic topology with two end-systems H1 and H2, linked through two routers R1 and R2.

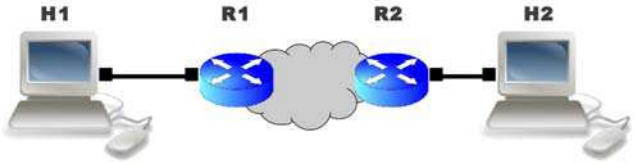


Figure 2. Basic network topology with 2 hosts and 2 routers

If both hosts H1 and H2 support SCOUT, then a secure channel can be established between H1 and H2, this is the Host-Host SCOUT mode. When H1 sends a first packet to H2, the SCOUT daemon of H1 intercepts the packet, communicates with the SCOUT daemon of H2 and a secure channel is established between H1 and H2. After that, all packets from H1 to H2 or in the reverse direction from H2 to H1 are forwarded through this secure channel.

If host H1 and router R2 support SCOUT, but H2 does not, then a secure channel can be established between H1 and R2, this is the Host-Router SCOUT mode. When H1 sends its first packet to H2, the H1's SCOUT daemon tries to communicate with H2 which refuses the communication (H2 does not support SCOUT). Then H1's daemon tries to communicate with "the router in the neighborhood of H2" (the first router forwarding the packets sent by H2) and R2 responds to this request. A secure channel between H1 and R2 is established as in the Host-Host SCOUT Mode.

If the first packet sender does not support SCOUT but the router in the neighborhood does, then the SCOUT protocol can provide security on the packet forwarding path. In our example fig. 2, if H1 does not support SCOUT and R1 supports it, then R1 can take the place of the end-system to secure data. In this case, if H2 supports SCOUT, then R1 establishes a secure channel between itself and H2, this is the Router-Host SCOUT mode. If H2 does not support SCOUT but R2 supports it, then the secure tunnel is established between R1 and R2; all the data exchanged between H1 and H2 are then secure between the neighboring routers R1 and R2, this is the Router-Router mode. Table 1 resumes the four secure cases.

TABLE I. SCOUT MODES AND SECURE SECTIONS

SCOUT Mode	Link		
	H1-R1	***R1-R2***	***R2-H2***
Host-Host (H-H)	✓ secure	✓ secure	✓ secure
Host-Router (H-R)	✓ secure	✓ secure	✗ unsecured
Router-Host (R-H)	✗ unsecured	✓ secure	✓ secure
Router-Router	✗ unsecured	✓ secure	✗ unsecured

In the four different modes, the data between routers R1 and R2 are secure. The SCOUT protocol has been elaborated to enable not exclusively the hosts but also the routers to provide the data security, in order to reduce the load of SCOUT deployment. SCOUT needs indeed to install and configure a daemon; limiting the deployment to the routers provides a sufficient level of security for some topologies without requiring deployment of SCOUT on all end-systems. This enables the network administrators to secure data for systems incompatible with SCOUT, for example the systems running exotic operating systems or systems with low hardware resources.

If only one node supports SCOUT, then the data cannot be secure. The default behavior of the SCOUT node is to drop the "secure-unable" packet. Matrix 1 resumes all the cases which can happen for the topology represented in fig. 2. In this matrix, the node names are preceded by an exclamation mark ("!") if they do not support SCOUT, otherwise they support the protocol. For example, the column "H1 and !R1" corresponds to the cases where node H1 supports the SCOUT protocol and router R1 does not support it.

TABLE II. OVERVIEW OF ALL POSSIBLE SECURITY CONFIGURATIONS

	H1 and R1	**H1 and !R1**	**!H1 and R1**	**!H1 and !R1**
H2 and R2	H-H between H1 and H2	H-H between H1 and H2	R-H between R1 and H2	Drop
H2 and !R2	H-H between H1 and H2	H-H between H1 and H2	R-H between R1 and H2	Drop
!H2 and R2	H-R between R2 and H2	H-R between R2 and H2	R-R between R1 and R2	Drop
!H2 and !R2	Drop	Drop	Drop	

To summarize, data are secure at least between the neighboring routers or data are dropped before being sent unsecured on the network. If the end-systems support SCOUT, they secure the data. If not, if the nearest routers support SCOUT, they do the work instead. If no system supports SCOUT, the data cannot be secure and are dropped.

1) *SCOUT prerequisites*

To allow the SCOUT protocol to secure the data, it requires several properties. Firstly, the link between the nodes and their nearest routers must be secure: the Router-Router SCOUT mode is acceptable if and only if the local networks are secure. Without security of the link between the node and the router, the data may be discovered by an attacker.

Secondly, SCOUT requires another protocol "P" to establish a secure communication channel: the SCOUT daemon on the local node finds out the security capabilities of

the remote node, then invokes other programs to create a secure channel and reconfigure the operating system routing tables to forward packets through the secure channel.

Thirdly, SCOUT trusts the other protocol "P" to authenticate the remote node: SCOUT identifies the remote node and transmits the information to the "P" protocol. That enables SCOUT to have a low security overhead and avoids performing multiple times the same authenticity verification.

B. SCOUT in service

1) SCOUT algorithm

The SCOUT protocol algorithm can be summarized as following. We call "initiator" the first packet sender end-system which supports SCOUT or else the first router which forwards the packet and supports SCOUT. We call "destination node" the end-system to which the packet is sent and "destination router" the last router which forwards the packet. An "IP-flow" designates all packets with the same source and destination addresses, in both directions (sending and receiving).

The first packet of any IP-flow triggers the process for the IP-flow. All following packets of the same IP-flow are forwarded through the secure channel (if the channel has been successfully established) or are dropped (if no channel has been created). When the first packet goes through the initiator, it is delayed and the initiator starts the first step, sending a request to the destination node for security capabilities.

If the destination node supports SCOUT, it responds positively with its capabilities to the initiator, then the initiator invokes secure channel establishment, modifies the routing table and SCOUT has completed its task.

If the destination node does not support SCOUT, depending on the implementation of the node and on the configuration of firewalls in the network, it can respond by sending an error message or can ignore it and send back no response. In the second case, the initiator tries again to contact the destination node several times (arbitrarily three times for our experiments) before starting the second step. In the first case, the initiator starts the second step as soon as the destination node error response is received.

In the second step, the initiator tries to contact the destination router. If the router response is positive then the initiator invokes the secure channel, as in the first step. If the router response is negative, then the initiator cannot assure data security and indicates to the operating system to drop the IP-flow. If there is no response from the destination router, then the initiator tries again several times before considering the absence of response as negative. Fig. 3 below resumes the SCOUT algorithm.

The SCOUT protocol uses the IP source and destination addresses to decide about the security of the data. If the administrator has nodes with different network interfaces and different security needs and if he wants to have at the same time secure and unsecured data on the same link, it is possible to bind the SCOUT program to certain IP addresses but not all. Operating systems usually accept that administrators define different IP addresses for the same network interface, enabling them to have one address for data requiring security and another one for data without security requirement.

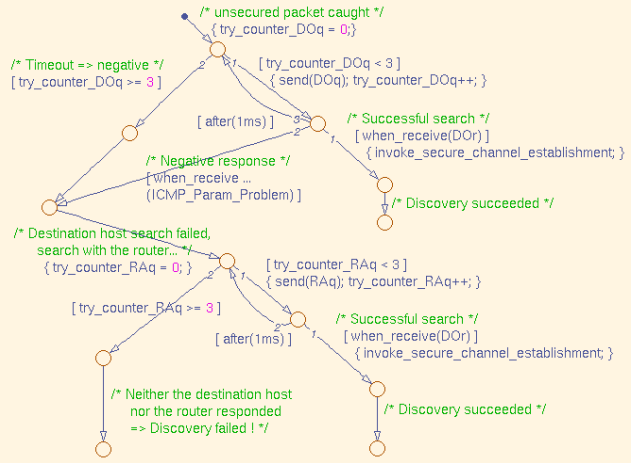


Figure 3. SCOUT algorithm in a diagram

2) Node behaviour for SCOUT packets

There are four different types of nodes interacting with SCOUT packets: initiator end-systems (like H1 in fig. 2), initiator routers (like R1), destination routers (like R2) and destination end-systems (like H2).

On the initiator end-systems, the SCOUT daemon must intercept outgoing packets to trigger the SCOUT algorithm presented in the previous section. On the initiator routers, it must intercept packets from the directly-linked end-system during their forwarding. In both cases, if no secure channel already exists and if security is required, the SCOUT daemon is triggered.

On the destination end-systems, the SCOUT daemon must receive incoming requests and respond in consequence to the sender. On the destination routers, the SCOUT daemon must intercept the packets sent to directly-linked end-systems during their forwarding and respond to the sender.

Details on the contents of the requests and the responses and how the protocol differentiates the packets for destination nodes from the ones for destination routers are explained in the next section describing how the generic SCOUT protocol is instantiated with the IPv6 protocol.

C. SCOUT with IPv6: SCOUT6

The SCOUT protocol designates the main guidelines to enable initiator nodes first to discover destination node security capabilities and second to invoke secure channel creation between the initiator and the destination. This section will give information to concretely merge the SCOUT specifications with the Internet Protocol version 6 (IPv6). The design of the SCOUT protocol instantiated to the IPv6 protocol will be called "SCOUT6". Thus, the generic part is designated by the name "SCOUT" while "SCOUT6" designates the implementation-specific part for IPv6.

Conforming to the RfC 4294[19], IPv6 nodes must support IPsec. The protocol most used to establish an IPsec secure

channel is currently the IKEv2 protocol. This is why SCOUT6 with IKEv2 should be chosen over other protocols, but a SCOUT6 daemon configuration may select another protocol if the administrator so wants.

D. SCOUT6 in detail

The IPv6 protocol enables packets to transport data plus additional information in "extension headers". These optional headers are managed at the operating system level and, depending on the information, may influence the kernel or intermediate network layers for packet management. The SCOUT6 design uses two IPv6 extension headers described in detail later: the Destination Option and the Hop-by-Hop option. These extensions make it possible to extend the IPv6 protocol with SCOUT6 support and to maintain compatibility with the noncompliant SCOUT6 nodes.

With SCOUT6, we need to exchange three kinds of information: one for the initiator node to ask about the destination end-system security capability, one for the initiator to ask about the destination router capability and one for the destination nodes to respond to the initiator.

1) Destination Option Query (DOq)

The IPv6 Destination Option (DO) enables the SCOUT6 protocol to contact the destination end-system SCOUT daemon and ask for security capabilities. IPv6 requirements specify that this option must be examined only by the packet destination system. It contains the type of the next IPv6 packet, the length of the extension header and at least six bytes of specific TLV-coded data. The Type-Length-Value (TLV) contains a byte for the type, orienting the operating system packet management (see table 3 for more details).

SCOUT6 initiator sends to the destination end-system an IPv6 packet query named "DOq". This packet is illustrated by fig. 4. It contains a Destination Option extension header with a type of 196 and a data length of 4 bytes. The DO data value is not significant for DOq, it is some padding data to align the IPv6 total packet length to a multiple of 8.

TABLE III. TYPE IDENTIFIER ACTIONS FOR PROCESSING IPV6 NODES, FROM THE RFC 2460[20]

highest-order 2 bits of the DO type identifier	Type range	Operating System default behaviour
00	0..63	Ignore the packet
01	64..127	Drop the packet
10	128..191	Send an ICMP Parameter Problem to the sender
11	192..255	If the sender IPv6 address is not a multicast address, then send an ICMP Parameter Problem to the sender, else drop the packet

The experimental type 196 is not yet assigned by the IANA [21] and the value may change in the future. A value of 196 as type of DO enables the destination end-system operating system to decide what to do with this packet. If the destination node supports SCOUT6, the daemon intercepts the DOq and sends a DOr packet as described below. If it does not support SCOUT6, then the operating system sends an ICMP error to the initiator which consequently knows that the destination end-system does not support SCOUT6.

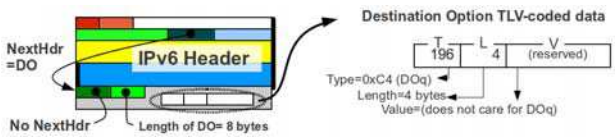


Figure 4. IPv6 packet format with the Destination Option Query

2) Router Alert Query (RAq)

In the case that the initiator does not have any response to several DOq sent or if it receives an ICMP Parameter Problem on its DOq extension, then it sends out, to the destination router, other packets called "Router Alert queries" (RAq). In the same way as with the Destination Option, the IPv6 Hop-by-Hop extension enables a packet sender to ask each intermediate router to analyze the packet data. The Router Alert option[22] uses this extension with a specific TLV-type code; the TLV-value code can specify a subtype. SCOUT6 uses the Router Alert option with an experimental and IANA-unassigned value of 42 to indicate "Router Alert Query". This RAq is for the SCOUT6 daemon on the destination router.

The initiator sends a "RAq" packet, containing this Router Alert extension, to the destination end-system. Each router forwarding the RAq packet will perform an operation. If the operating system does not support SCOUT6, the packet is forwarded and the extension is ignored. If the operating system supports SCOUT6 but the router is an intermediate router and not the destination router, the packet is forwarded. If the router supports SCOUT6 and is the destination router, it intercepts the packet and sends a DOr packet as described below. If the final router does not support SCOUT6, then the packet is received by the destination end-system and ignored (it does not contain other data).

3) Destination Option Response (DOr)

If a SCOUT6 daemon on a destination node (router or end-system) receives a query from the initiator, then the destination node sends back a response. To do so, it sends an IPv6 packet called Destination Option Response (DOr) with a destination option of type 66, as represented in fig. 5. The experimental type 66 informs the receiver to drop the packet if it does not know what to do with it. The DOr value field contains a 32-bit binary mask of supported protocols: a "1" bit means the destination node supports a specific security protocol, depending on the position of the bit in the mask. Currently, the Least Significant Bit is assigned to the IKEv1 protocol, the next bit is for the IKEv2 protocol, the third bit is for the KINK protocol and the Most Significant Bit is for experimental usage; all other bits are reserved for future protocol support. The DOr packet is completed with 16 bytes of data filled with the destination address of the DOq packet.

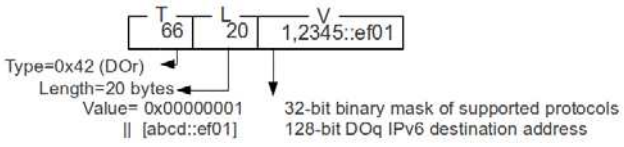


Figure 5. Destination Option Response TLV-data values

This data enables the initiator to identify which IP-flow is allocated to the DOr. The DOr packet source address is the address of the destination node (router or end-system) accepting the security. The DOr packet destination address is the address of the initiator. The DOr packet Destination Option data is the destination end-system address, used to identify the allocated IP-flow and to know which DOq or RAq has generated the DOr.

If a destination end-system sends back a DOr, then the DOr packet source address has the same 16 bytes as the DOr Destination Option data value. If a destination router sends back a DOr, the DOr packet source is not the DOr Destination Option data value. Thus the initiator knows that the security will be assured by the destination router and not the destination end-system. The initiator also knows which destination node to contact in order to establish a secure channel.

In the case that there is already an existing secure channel between both ends, the SCOUT6 daemon does not invoke another channel but reuses the existing one to save resources; the daemon just adds a new route for the IP-flow through the secure channel.

E. Concrete examples

1) Host-Host security

Fig. 6 illustrates a typical usage of the SCOUT6 protocol. In this scenario, both end-systems support SCOUT6. When the first host H10 sends its first packet to H21, the H10 SCOUT6 daemon delays the packet and sends a query (DOq) to H21. In response, the H21 host sends a DOr containing its IPv6 address to H10. Then H10 initiates a secure communication channel with H21, reroutes all packets for H21 through this tunnel and releases the delayed first packet. On the other host H21, the channel establishment is followed by a rerouting of packets for H10 through the secure channel. All following communication between H10 and H21 will be secure.

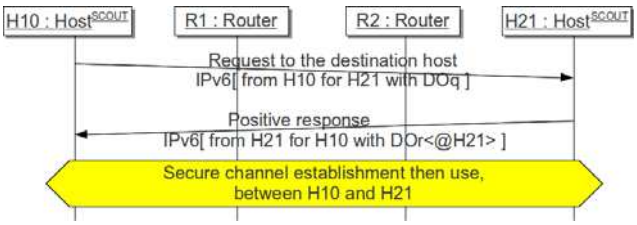


Figure 6. Host-Host SCOUT mode

2) Router-Router security

In the previous scenario, we do not care if the routers support or not the SCOUT6 protocol, because the end-systems do and manage the security. Fig. 7 presents the next scenario and illustrates the opposite case where no end-system supports SCOUT6 but both nearest routers do.

In this scenario, when H11 sends its first packet for H22, its nearest router R1 intercepts and delays the packet during the forwarding and send a query (DOq) instead. H22 does not support the SCOUT6 protocol. Conforming to IPv6 specification, the H22 operating system responds to R1 by sending an ICMP Parameter Problem packet.

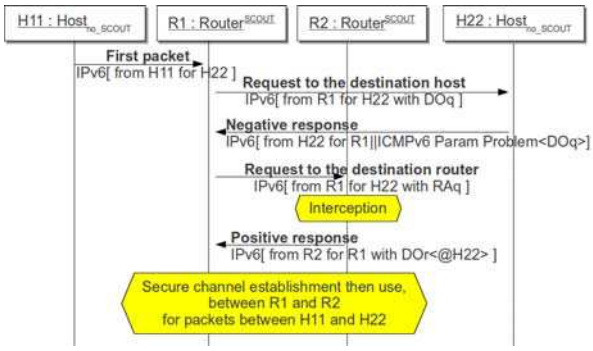


Figure 7. Host-Host SCOUT mode

Then R1 knows H22 cannot assume the security and tries to contact the destination router by sending a query through a router alert message (RAq). Intermediate routers forward this packet, ignoring the RAq information: they do not support SCOUT6 or are not the router nearest H22. The destination router R2 supports SCOUT6 and is the router nearest H22 thus R2 processes this packet RAq: it sends R1 a response (DOr) containing the original destination of the RAq (=H22's IPv6 address). When R1 receives the DOr packet from R2, it invokes a secure channel between R1 and R2 and inserts routes from R1 for R2 and H22 through the secure channel.

After having successfully established a tunnel between R1 and R2, if another host connected to R1 sends unsecured packets to any un-supporting SCOUT6 host connected to R2, the R1 daemon will reuse the secure channel; it just inserts new routes and does not re-invoke a potentially large number of channels between R1 and R2: a single channel multiplexes all unsecured IP-flows between R1 and R2.

3) Scenarios overview

Table 4 below summarizes the different scenarios which can take place with SCOUT6. The topology used to establish the table is shown in fig. 8, where a star "*" means "SCOUT6 supporting node", all other nodes do not support SCOUT6. In the table, the "path" column indicates the security of each segment: a minus ("-") is an unsecured segment and an equal ("=") is secure.

The scenarios of table 4 are in chronological order. In scenario 2, a secure channel has been established from H*10 to R*2 in order to secure the IP-flows between H*10 and H22. This secure channel is reused in the next scenario 3 to secure the IP-flows between H*10 and H23.

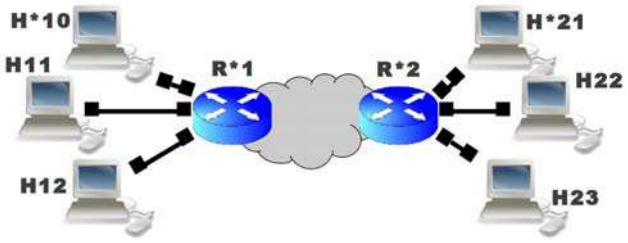


Figure 8. Example of extended topology

TABLE IV. SCOUT6 SCENARIOS OVERVIEW

	Path	SCOUT Mode	used by SCOUT6
1	H*10 = H*21 (H*10 =R1=R2 = H*21)	Host-Host	DOq, DOr
2	H*10 = R*2 – H22 (H*10 =R1=R*2 – H22)	Host-Router	DOq, RAq, DOr
3 (after 2)	H*10 = R*2 – H23 (H*10 =R1=R*2 – H23)	Host-Router, channel reusing	DOq, RAq, DOr
4	H11 – R*1 = H*21 (H11 – R1 = R2 = H21)	Router-Host	DOq, DOr
5 (after 4)	H12 – R*1 = H*21 (H12 – R1 = R2 = H21)	Router-Host, channel reusing	-
6	H12 – R*1 = R*2 - H22 (H12 – R1 = R2 = H22)	Router-Router	DOq, RAq, DOr
7 (after 6)	H12 – R*1 = R*2 - H23 (H12 – R1 = R2 = H23)	Router-Router, channel reusing	DOq, RAq, DOr

III. SCOUT6 IN PRACTICE

A. *Application of SCOUT 6 to a peer-to-peer network*

Peer-to-peer topologies are network distributions without central nodes to control and distribute the tasks; in a peer-to-peer network, all nodes are equally privileged. Internet was initially adherent to the peer-to-peer approach through the Arpanet project but its growing usage of the web converted it year by year into a mainly centralized network around several mainframes and powerful servers. Nevertheless some peer-to-peer protocols continue to support the Internet. For example, the Border Gateway Protocol (BGP[23]) backs the core routing decisions on the Internet; its behavior is based on messages between adjacent peers to exchange mainly IP network prefixes.

The BGP protocol was originally designed with little consideration for protection of the information it carries and thus there is no mechanism internal to BGP that protects this protocol. RfC 4272[24] is a BGP security vulnerability analysis enumerating some attacks against BGP. The SCOUT6 protocol we propose may provide automatically established secure channels for a peer-to-peer network, such as the one running BGP. The Secure Border Gateway Protocol (Secure-BGP [25]) uses IPsec features to secure BGP messages and thus improve BGP message security, but lets IPsec configuration information to be defined by the administrators. SCOUT6 can automate configuration item definition.

Without SCOUT, in a peer-to-peer network with n equal and connected nodes, each node may communicate with up to $(n-1)$ adjacent nodes. In the worst case, we have then $n(n-1)/2$ secure duplex channel to establish. Without SCOUT, the administrator overhead is important: they have to configure $n(n-1)$ channel ends.

With SCOUT, administrators must install and configure the daemon on each node. They need not configure it with all remote possible nodes to secure. Therefore, the installation must be repeated only n times and not $n(n-1)$ anymore. Fig. 9 describes the cost in lines of configuration in function of the number of nodes. We estimated we need 9 lines of configuration per channel per node.

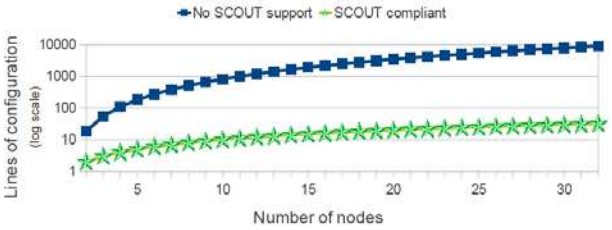


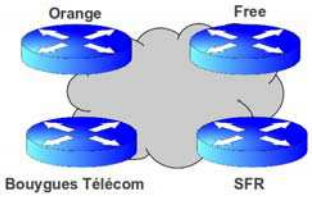
Figure 9. Configuration cost depending on the number of nodes

Moreover, only the useful channels are established: if two nodes communicate together, then SCOUT invokes a channel between them to secure the messages; if two nodes never communicate, then SCOUT never invokes the channel, thus some computer and network resources are saved.

B. *SCOUT6 overhead for network delay and capacity*

To discover the security capabilities of the different nodes, the SCOUT protocol needs to exchange messages on the unsecured network. The SCOUT6 implementation, based on IPv6 packets, uses the DOq, RAq and DOr messages explained in the previous section. Packet lengths are respectively equal to 48, 48 and 64 bytes. In the best case, there is only one DOq and one DOr packet exchanged on the network, thus 96 bytes in two packets. In the worst case, there are three DOq plus three RAq plus three DOr, thus nine packets containing a total of 480 bytes.

To experiment the SCOUT6 protocol, we elaborated a topology with 4 groups of nodes representing major Internet Service Providers (ISP) in France: Orange, Bouygues Telecom, Sfr and Free major companies. Each group has assigned to it an additional delay of transmission, measured with the commands traceroute and ping to their main web server from our laboratory. We are connected through a fifth ISP called RENATER. Fig. 10 describes the average delays and the jitter we have measured on December 2011, the 6th at 11 AM and the experimental topology we use to validate our SCOUT6 implementation. Each group is associated to a set of 256 IPv6 addresses. To emulate this experimental network, we used VirtualBox[26] with 4 Virtual Machines (VM with a mono-core processor and 256 MB of RAM) with Debian and a SCOUT6 daemon, linked with a pseudo-Ethernet and configured with delays and jitter for outgoing packets through the Linux netem[27] kernel module ("Network Emulation") and the program Traffic Control (TC[28]). On each VM, a script tries to ping other VM from random addresses to other ones, the ping simulates data connections.



Average delays and jitters for ISPs
Delays (ms) / jitters (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 10. Average delays and jitters for 4 ISPs

Figure 11. Average delays and jitters for 4 ISPs

The ping request triggers the SCOUT6 daemons and conducts to establish secure tunnels to exchange securely the ping data. The SCOUT6 daemon has been instrumented to measure, for each secure channel, the delays between the time of the first packet to secure and the time of the channel invocation. Fig. 11 resumes the measured delays in function of the load of the VM (the load is quantified in number of established channels).

We can observe an average delay of 85 milliseconds (ms) for the total delay between the packet interception and the channel invocation. During this SCOUT6 discovery stage, the daemon sent a DOq and received the corresponding DOr. Measures are globally constant over the time while an increasing number of tunnels were established. At the end of the experiment, up to 228 tunnels has been established and can be used to transmit securely the data between the peers.

The experimental duration average of the SCOUT6 implementation proceeding is 5 ms (measures are mostly between 1 and 10 ms). Sending one packet for the DOq plus one other for the DOr spends about 40 ms in each case. Several measures are around this Round-Trip Time (RTT) of 40 ms, but most of them are twice. The 80 ms observed are due to an additional RTT for the Neighbour Discovery Protocol (NDP) which resolves the IPv6 global addresses over the link layer of the Internet. In our experiments, the channels need several seconds to be established, about 50 times more than the whole duration of the SCOUT6 discovery stage.

C. *Complexity of common operations*

To evaluate SCOUT6 benefits, we propose an overview of the costs (in terms of modified lines of configuration) on a centralized network topology (e.g. the traditional use case for network management). We will examine common operations such as adding and removing clients and servers. In a centralized network, we suppose the secure channels are established only between one server and one client. Two clients or two servers cannot be directly connected. Moreover, it is only the clients who initiate channel establishments; the servers do not contact clients to create secure channels. Let's assume a star topology with N clients and 1 server.

TABLE V. OVERVIEW OF COSTS FOR COMMON OPERATIONS

	Without SCOUT	**With SCOUT**
Adding a client (N+1)	Configure this (N+1)th client and configure the server [O(2)]	Install SCOUT6 on the client [O(1)]
Modifying the security data for the client N or removing the client N	Reconfigure both ends: the client N and the server [O(2)]	Reconfigure the client N and, depending on the security protocols, server may invalidate its security caches [O(2)]
Adding a new server	On the new server, configure one tunnel per potential client and on each client, configure a channel with the new server [O(2*N)]	Install SCOUT6 on the server [O(1)]
Removing a server	Reconfigure each client [O(N)]	Nothing to do [O(0)]

IV. CONSIDERATIONS OF SECURITY IN SCOUT6

A. *SCOUT6-related attacks*

SCOUT6 has been designed to search for node security capabilities and to invoke, if possible, secure channel establishment between the nodes. Security is enforced by the establishment protocol invoked by SCOUT6, but does our protocol introduces some new vulnerabilities? We propose to give some elements of response to this question in this section. We would assume here that the "P" establishment protocol invoked by SCOUT6 is the IKEv2 protocol.

1) *Packet injection*

Injection of valid packets into the system by another system not participating in the normal process is a common attack called "packet injection". In the case of SCOUT6, 3 types of packets can be injected: DOq, RAq and DOr packets. These packets can be sent to SCOUT6 compliant systems or else to IPv6 compliant system without SCOUT6 support. Table 6 summarizes the different types of injection and the impacts.

Injecting a DOq or a RAq to a SCOUT6 compliant system leads this system to respond with a DOr packet, thus to add a light overhead on the network (64 bytes are sent). If the nearest router supports SCOUT6 protocol, injection of RAq may lead the attacker to find the nearest router address. Injecting a DOr to a SCOUT6 compliant system does nothing: the receiving system has not sent any associated DOq or RAq in the past and will therefore ignore this orphan DOr packet (a non-orphan DOr is the response to a previous DOq or RAq query, an orphan DOr has no DOq nor RAq associated). Injecting a DOq to a SCOUT6 non-compliant system may be ignored by the receiving system or may generate an ICMPv6 Parameter Problem packet. This case is considered as a normal case and used to trigger the RAq sending step without waiting until the 3 DOq are sent. Injecting a RAq or a DOr packet to an IPv6 compliant system which does not support SCOUT6 has no effect: conforming to the RfCs[20], packets with these kinds of unknown options are dropped by the operating system.

Note that a SCOUT6 non-compliant system which receives a DOr can, in normal conditions, be considered as probably attacked. DOr packets can only be sent in response to a DOq or a RAq, and these packets are normally only sent by SCOUT6 compliant systems!

TABLE VI. PACKET INJECTION AGAINST SCOUT6

Type	Sent to a SCOUT6 compliant system	Sent to a SCOUT6 non-compliant system
DOq	A DOr is sent back in response	An ICMPv6 Parameter Problem is sent back in response
RAq	A DOr is sent back in response, containing the router address	Ignored by the receiver
DOr	Ignored (viewed as an orphan DOr)	Ignored by the receiver

2) Replayed packets

An attacker may listen to the SCOUT6 messages, intercept some of the packets and inject them again into the network to destabilize nodes. But in SCOUT6 design, the replay is already implemented to compensate the risk of packet loss because IPv6 does not guarantee reliability. The DOq and RAq queries, without responses, are sent again another two times before being considered negative. A system may then receive multiple times the same DOq or RAq; it will just send again an associated DOr.

The DOr contains IP source and destination addresses for the IP-flow to secure. The SCOUT6 implementation uses this information that enables the initiator to identify which IP-flow to secure and to know both ends in order to establish a secure channel between them. When the DOr is received, the SCOUT6 implementation checks first that the DOr is not an orphan DOr. Then the SCOUT6 daemon invokes secure channel establishment and updates its cache to ignore future similar DOr: the daemon deletes the associated pending DOq or pending RAq entry from the "list for search in progress". A "list" is required to manage the retries and their timeouts, before considering the absence of response as negative. After updating this list, any other received DOr are considered orphans and therefore ignored: there is no pending DOq or RAq anymore which can be associated to the replayed DOr.

3) Packets stolen, altered and man-in-the-middle

An attacker may intercept the SCOUT6 messages during their forwarding at the middle of the network. It can silently drop these packets, preventing the SCOUT6 daemon from establishing a secure channel. If the attacker has the capacity to drop SCOUT6 packets, then it has the capacity to drop IKEv2 packets too; it may block SCOUT6 security searching but may equally block IKEv2 secure channel establishment. Thus a SCOUT6 steal packet protection is useless.

The attacker may use its capacity of interception to try a man-in-the-middle attack. SCOUT is not designed to secure the connections itself, but is designed to search for connection security solutions. The attacker may intercept and alter SCOUT6 packets, causing both legitimate ends to try to establish a secure channel between them and the attacker. But then the IKEv2 daemon will realize the attempt at impersonation and reject channel establishments.

The attacker could try to alter the DOr data content, i.e. the IPv6 address of the destination host. In this case, the altered DOr no longer contains the valid information for the initiator to associate this DOr to the DOq or RAq query, thus the altered DOr will be considered orphan and therefore ignored.

4) Spoofing of IP addresses

An attacker may try to spoof the source address of the SCOUT6 packets. If the attacker sends a DOq or RAq with a spoofed source address, the packet may be ignored (if the destinations are not SCOUT6 compliant) or the DOr will be sent to the spoofed node which will ignore it (the spoofed node considers this DOr as orphan). If the attacker sends a DOr with a fake IP source address, the packet will be discarded: the receiving system will ignore it in all cases (orphan DOr for a SCOUT6 compliant node, packet to drop for others).

An attacker may try to spoof the destination address. If the packet is a DOq or a RAq, the attacker will receive an ICMP error parameter problem (if the spoofed node does not support SCOUT6) or a DOr (if it supports SCOUT6) from the spoofed destination. If the altered packet is a DOr, this packet will be an orphan for the destination host which will manage it.

5) Denial of Service

An attacker may use SCOUT6 packets to try a denial-of-service attack on the victim. SCOUT6 daemons can serve as mirroring hosts: when you send them a DOq or RAq, they respond with a DOr thus increasing the network load. But SCOUT6 packets are small (48 and 64 bytes). Moreover the attacker can easily generate IPv6 packets with the same DoS effects and with more data to overload a network, just by conforming to the IPv6 specifications: IPv6 compliant nodes should send an ICMPv6 Parameter Problem packet to the emitter when they receive an IPv6 packet with an unknown Destination or Hop-by-hop Option.

A node may be led to accept more secure channels than it can manage: if the attacker has a great number of IPv6 addresses, it can use them to establish 1 channel per address and then overload the destination system. This is not really a vulnerability of the SCOUT6 protocol but a negative consequence of the scalability it provides. This vulnerability can be avoided by limiting the maximum number of channels the system accepts. If this limitation is not acceptable, the vulnerability can be reduced by using efficient algorithms to manage the channels and to delete unused channels.

6) An effective attack against SCOUT6

An attacker may influence SCOUT6 behavior successfully by silently discarding all DOq queries and letting through the RAq and DOr messages. If both the destination router and the destination host are SCOUT6 compliant, then the initiator will establish the secure channel not between itself and the destination host but between itself and the destination router. In this case, the path between the destination router and the destination host is NOT secure despite the fact that it could in fact be secure.

This is why the first hypothesis of SCOUT protocol is that "the local networks are secure". This hypothesis may be ignored if the SCOUT daemon is configured to refuse the Host-Router and Router-Router modes and then it will never send any RAq. But this possibility may significantly decrease security capabilities.

V. CONCLUSION AND FUTURE WORK

We have presented in this paper the new SCOUT protocol which automatically discovers the security capabilities of nodes on an IP-based network topology. Not only can the SCOUT protocol invoke secure channel establishment between end-systems if both support the protocol, but it can also compensate for the absence of SCOUT support on end-systems by enabling the nearest routers to provide the security if they support SCOUT. Moreover the SCOUT protocol reduces administrator configuration workload and increases scalability of network security mechanisms.

The instantiation of the SCOUT protocol with IPv6 completes the generic SCOUT protocol with a concrete protocol named SCOUT6. Our experiments indicate a low overhead in terms of exchanged data and additional delays. Our paper concludes with a study of the vulnerabilities linked to the SCOUT6 protocol.

Nevertheless we have to conduct further research to find solutions to the effective attack against SCOUT6 presented above and, if possible, to validate formally these solutions. Moreover, we are working on completing the set of establishment protocols supported by SCOUT6: currently the SCOUT6 protocol experiments are based only on one secure channel establishment protocol (IKEv2), whereas SCOUT6 is designed to support multiple protocols. Another point we are working on concerns secure multicast channels. Current SCOUT design is clearly unicast-oriented. In the future, we plan to extend the SCOUT protocol to support multicast communications.

ACKNOWLEDGMENT

We would like to greatly thank Rupert Salmon for his help in editing this paper. We express also our gratitude to Julien Marchand for his discussions around this paper.

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] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409 (Proposed Standard), November 1998. Obsoleted by RFC 4306[11], updated by RFC 4109.

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

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

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

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

[8] 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.

[9] 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.

[10] D. Piper. The Internet IP Security Domain of Interpretation for ISAKMP. RFC 2407 (Proposed Standard), November 1998. Obsoleted by RFC 4306[11].

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

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

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

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

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

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

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

[18] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005

[19] J. Loughney. IPv6 Node Requirements. RFC 4294 (Informational), April 2006. Updated by RFC 5095.

[20] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722.

[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] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[24] S. Murphy. BGP Security Vulnerabilities Analysis. RFC 4272 (Informational), January 2006.

[25] Kent, S., Lynn, C., and Seo, K., “Secure Border Gateway Protocol (Secure-BGP)”, IEEE Journal on Selected Areas in Communications, Vol. 18, No. 4, April 2000, pp. 582-592

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

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

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