Tunnel Reform Design

Daniel L. McDonald

Solaris Security

Revision 1.0 – 21 October 2005

Overview

The Tunnel Reform project will address several shortcomings in the Solaris IPsec and IKE implementations. All of these shortcomings are related to *tunnel mode* packets. The current Solaris implementation of IPsec tunneling is barely interoperable with other implementations if manually-keyed, and is not at all interoperable with other implementations when IKE manages IPsec Security Associations.

Tunnel Reform includes a series of changes that span the IPsec implementation: configuration commands, the IKE daemon, the IPsec Security Policy Database (SPD), the IPsec Security Association Database (SADB), the PF_KEY key management socket, and the IP Tunnelling Module. This project also interacts, and has at least a soft dependency, on the IP Network Interface Unification (Clearview) project.

Problem Statement

In Solaris 8, Secure IPv4 tunnels were implemented as network interfaces. This decision made for easier network construction, as well as dovetailing with IPv6 tunnels. The downside to this decision, however, was that configuring IPsec to tunnel packets was not solely in the domain of IPsec configuration. Many IPsec administrators are used to dealing with integrated firewalls, where packet protection policy and the use of tunnels have been combined in a single solution. These users were confused by Solaris 8.

Also, because tunneling was a forwarding decision, not an IPsec decision, some possible uses of IPsec tunnel mode could not be performed in Solaris 8. The best example was the case where the outer packet's destination IP address was equal to the inner packet's destination IP address. In Solaris 8 (and later) one could not configure tunnelling to perform such an encapsulation - it would black-hole to prevent a never-ending forwarding loop.

In Solaris 9 and Solaris 10, the changes made for IKE and other IPsec policy improvements did nothing to change how tunnels were implemented. Also, the IKE daemon would express Phase II identities in a non-standard way. According to RFC 2409 the Phase II identities must be set as follows:

> If ISAKMP is acting as a client negotiator on behalf of another party, the identities of the parties MUST be passed as IDci and then IDcr.

This is a fancy way of saying "use some form of the inner IP addresses for Tunnel Mode". Currently Solaris sets both identities for packets protected on a tunnel interface as proto=4 (IP-in-IP), 0.0.0.0/0 for IPv4-in-* tunnels, and proto=41 (IPv6-in-IP), 0::0/0 for IPv6-in-* tunnels. This decision led to many interoperability problems with other IKE implementations.

Furthermore, the upcoming revisions to the IETF IPsec standards (known among the IPsec community as *2401bis*) will also require that tunnel mode packets have inner-

packet selectors. The next revision of IKE, IKEv2, also still specifies the equivalent of Phase II identities, now calling them traffic selectors. IKEv2 also clarifies when it is appropriate to send precise inner-packet information, and when to send inner-packet patterns that matched an SPD entry.

Goals and Non-Goals

The goals of Tunnel Reform are straightforward:

- For packets that originate or are destined for a Solaris node (where a tunnelling interface counts as an originator or destination), implement the full RFC 2401 set of selectors, while attempting to not impede changes that 2401bis will introduce.
- To severely reduce or eliminate interoperability problems. Testing should include the following platforms:
  - Solaris 9 and/or Solaris 10 FCS (for backward compatibility in S9's non-standard tunnel handling in IKE).
  - cisco IOS
  - cisco PIX (uses different IPsec implementation)
  - Checkpoint products
  - Microsoft Windows (2000, XP, and if available, Vista)
  - MacOS X
  - Linux distributions based on the 2.6 kernel, optionally with the “ipsec-tools” project.
  - If not covered by MacOS X and/or Linux testing, a *BSD with the racoon IKE daemon.
  - OpenBSD isakmpd.
- PF_KEY improvements for expressing 2401 selectors, preferably as input into a publicly available PF_KEYv3 specification.
- The ability to tunnel to multiple peers behind a single NAT-ted IP address.

The following items are specifically NOT part of the Tunnel Reform project:

- Policy-directed tunnelling – the ability to use ipsecconf(1m) to configure a tunnel (as opposed to merely setting a tunnel's security properties). While it would allow the use of a policy that may include a packet with the inner destination address is equal to the outer destination address, implementing it now may sabotage other routing or tunnel related projects.
- Performing IPsec protection on forwarded packets that are not explicitly routed through a tunnelling interface. Such work would be done in conjunction with an integrated IPfilter module.
- Cleanup of the main IP processing path that deals with IPsec packets. Such work constitutes a separate project.
- RFC 2401bis work will be delayed until the full RFC 2401bis project.

Concepts

This document assumes the reader is familiar with TCP/IP and IPsec internals.

Solaris's implementation of packet tunnelling, the IPsec specifications for Tunnel Mode packets, and other proposals from the IETF community have not agreed on the best way to do packet tunnelling and express IKE Quick Mode proposals for such packets.

Solaris 9 (the first Solaris revision to feature IKE) took a halfhearted approach to the problem, in that it specified (for IKE) the broadest set of tunnel mode inner selectors for both internal packet addresses (0.0.0.0/0), but fixed the IP protocol to either IPv4-in-IP or IPv6-in-IP. Furthermore, the Solaris 9 IKE does not check at all on the responder side for transport or tunnel mode -> it merely uses the port and protocol selectors in combination with the addresses in Phase I. While using ports and protocol along with the Phase I addresses is sensible behavior for transport mode negotiations, it is not for tunnel mode.

The IETF specifications for IPsec and IKE specify two details that Solaris 9 did not implement. The first was that the Security Association Database (SADB) allows inner-packet fields to be used in selecting IPsec SAs. Since Solaris does not support inner-packet field SA selection, PF_KEY in Solaris lacks any expression for such selectors. That lack ties into the second detail. The Solaris 9 IKE daemon does not express Quick Mode identities other than the ones mentioned earlier in the Problem Statement section. Tunnel Reform will address this problem in both PF_KEY and IKE, and allow proper tunnel mode Quick Mode negotiations.

Another approach to packet tunnelling is detailed in RFC 3884. RFC 3884 postulates that Tunnel Mode is redundant if tunnels are treated like network interfaces. RFC 3884 suggests that IPsec treat IP-in-IP packets as transport mode, with the protocol selector being set to IP or IPv6, respectively. One suggestion 3884 makes is to also include *just* the inner addresses in SA selection. A drawback to this suggestion is that it cannot be expressed in IKE's quick-mode without heuristics that may violate the IKE specification. This project will not implement 3884 tunnels with inner address selctors, but it will implement 3884 tunnels in order to maintain backward compatibility with Solaris 9. Tunnel Reform will allow IKE to negotiate transport mode in Quick Mode for tunnels using a protocol value of either IPPROTO_ENCAP or IPPROTO_IPV6.

The following chart details the different secure tunnelling semantics:

Specification	***IPsec mode: Tunnel or Transport***	***IKE Quick Mode Identities***	***NOTES***
RFC 2401	Tunnel	Inner packet's addresses or matching prefix, optionally inner-packet's protocol and/or ports.	Modulo case of inner-packet destination == outer-packet destination, can be implemented using tunnelling interfaces.
RFC 3884	Transport	Same as outer packet's addresses, proto=ipip or ipv6.	Can express easily in IKE, but not specified in IKE or in RFC 2401. Also for Solaris 9 or 10 compatibility.

Internal Data Structure Changes

Several internal data structures will need to accommodate the full set of RFC 2401 selectors. The primary internal selector structure is this one:

```
typedef struct ipsec_selkey
{
      uint32_t    ipsl_valid;       /* bitmask of valid entries */
      ipsec_addr_t      ipsl_local;
      ipsec_addr_t      ipsl_remote;
      ipsec_addr_t      ipsl_linner;   /* NEW FIELD */
      ipsec_addr_t      ipsl_rinner;   /* NEW FIELD */
      uint16_t    ipsl_lport; /* NEW SEMANTICS:  Applies to inner */
      uint16_t    ipsl_rport; /* if either inner address is valid */
      /*
       * ICMP type and code selectors. Both have an end value to
       * specify ranges, or * and *_end are equal for a single
       * value
       */
      uint8_t           ipsl_icmp_type;
      uint8_t           ipsl_icmp_type_end;
      uint8_t           ipsl_icmp_code;
      uint8_t           ipsl_icmp_code_end;

      /*
       * NOTE:  These fields apply to the inner packet
       * if either inner address is valid.
       */
      uint8_t           ipsl_proto;       /* ip payload type */
      uint8_t           ipsl_local_pfxlen;      /* #bits of prefix */
      uint8_t           ipsl_remote_pfxlen;     /* #bits of prefix */
      uint8_t           ipsl_mbz;

      uint32_t    ipsl_hval;
} ipsec_selkey_t;

#define     IPSL_REMOTE_ADDR        0x00000001
#define     IPSL_LOCAL_ADDR               0x00000002
#define     IPSL_REMOTE_PORT        0x00000004
#define     IPSL_LOCAL_PORT               0x00000008
#define     IPSL_PROTOCOL                 0x00000010
#define     IPSL_ICMP_TYPE                0x00000020
#define     IPSL_ICMP_CODE                0x00000040
#define     IPSL_LOCAL_INNER        0x00000080  /* NEW */
#define     IPSL_REMOTE_INNER       0x00000100  /* NEW */
#define     IPSL_IPV6               0x00000200  /* CHANGED */
#define     IPSL_IPV4               0x00000400  /* CHANGED */

#define     IPSL_WILDCARD                 0x000001ff /* CHANGED */
```

The new and changed fields all have to do with the fact that for tunnel mode packet, the inner packet's fields are used for IPsec SA selection, in addition to the outer header's addresses. Selector ranges (as generalized in RFC 2401bis) are outside the scope of this project.

The PF_POLICY API will have to accommodate not only the changes above, but also to tie specify tunnel policy entries to specific tunnel interfaces or (in the case of policy-directed tunnels) an additional IP address for tunnelling.

Tunnel Reform requires a few new extensions, two of them are address extensions, and the third is a new extension type:

- SPD_EXT_LCLINNER
- SPD_EXT_REMINNER
- SPD_EXT_TUNIFNAME

PF_POLICY now needs a way to identify a tunnel interface:

```
typedef struct spd_ifid_s {
      union {
            struct {
                  uint16_t spd_ifid_ulen;
                  uint16_t spd_ifid_uexttype;
                  uint8_t spd_ifid_uid[6];
            } spd_ifid_u;
            uint64_t spd_ifid_alignment;
      } spd_ifid_u;
/* NOTE:  #defines removed for brevity... */
} spd_ifid_t;
```

The interface identity structure select a tunnel for which the specific policy (with inner selectors) is applied. The spd_ifid_id field (the uint8_t array) contains a null-terminated string that names the tunnel interface. This identity may be used by the SPD to either tag an entry for a tunnel interface, or locate a tunnel interface and place the policy entry in an interface-specific place. A policy entry with an interface identifier and no inner addresses means transport-mode will be indicated by PF_KEY ACQUIRE messages.

Given that policy entries can be assigned to tunnel interfaces, it may be helpful to system performance if such entries in the SPD are hashed into their own separate tables. This way, a tunnel-specific entry is not in the way of the main SPD, or other tunnels' entries.

The IPsec Security Association Database (SADB) needs to more properly match the specifications in RFC 2401. While defined, but not used, the “proxy” addresses in the internal SA representation will start being used with Tunnel Reform. The “Proxy Source” and “Proxy Destination” correspond to the inner packet's source and destination addresses. For inner addresses, we will need to add prefix-lengths to them as well, in case the endpoints negotiate based on prefixes instead of actual addresses.

Internally, there will still be no major difference between a Transport Mode SA and a Tunnel Mode SA. For the cases of RFC 3884 and RFC 2401 semantics, both have their protocols set to 4 or 41. RFC 2401 SAs which have inner-header port or protocol selectors (e.g. Quick Mode Identities are a TCP 5-tuple) will have a new byte in the `ipsa_unique_id` field for the inner-packet's protocol.

Internal Packet Processing Changes

PF_POLICY AND SPD CHANGES

If a PF_POLICY message contains a SPD_EXT_TUNIFNAME extension, one of two things will happen. The SPD code will first attempt to locate an active tunnel that matches the interface names specified. If the SPD finds the tunnel it will update its state with the relevant SPD entry. If it does not find the tunnel, then it will link the SPD entry into a “homeless queue” for the named interface. New tunnel instances (see

below) will search for homeless SPD entries.

TUNNEL MODULE CHANGES

The IP tunnelling module will become an active participant in IP policy. Currently, the IP tunnelling module parses an IP_SEC_OPT socket option structure out of the tunnel configuration ioctl, and packages it as a T_OPT_REQ for its IP instance. Packets delivered to a tunnel instance are checked in the initial IP processing for policy. On outbound processing, the tunnel trusts the lower layer of IP to apply the appropriate level of protection.

Under Tunnel Reform, inbound packets will have to arrive in the tunnel module routines with the IPSEC_IN metadata intact. Only the tunnel module, with access to the routing tables and its own inner-packet SPD entries can determine if the inbound packet is acceptable or not. Correspondingly, outbound packets will be tagged by the tunnel module with IPSEC_OUT metadata. This reintroduces an external producer of IPSEC_OUTs and a consumer of IPSEC_INs, but the FireEngine project, while eliminating TCP's direct dependence, does not appear to have eliminate the handling code in IP for external IPSEC_OUT and IPSEC_IN messages.

On the bring-up of a tunnel, once the instance name is known, the homeless SPD entries will be searched for a set that matches the tunnel's name. If homeless SPD entries are found, they will be cached in tunnel state. If a tunnel's name changes, existing SPD entries remain, AND the homeless queue will be searched for new ones matching the new name. Since ipsecconf(1m) input uses tunnel names, it is up to the administrator to ensure that if a tunnel interface name changes, the ipsecconf(1m) input also changes.

IP INBOUND PROCESSING CHANGES

The only major change in IP inbound processing is to deliver IPSEC_IN messages intact to the tunnel module along with the packet, if the packet was protected.

IP OUTBOUND PROCESSING CHANGES

Outbound processing in IP will not be affected as part of this project, except that the tunnel module may inject IPSEC_OUT packets into IP instead of just IP-in-IP datagrams. (An implementation decision about where ESP should be called – IP or tun – will occur.)

SADB LOOKUP and/or ACQUIRE CHANGES

Tunnel reform will add inner-packet fields to both SADB lookup and (upon an outbound SADB miss) PF_KEY ACQUIRE (extended, regular, and inverse) construction. Inverse ACQUIRE handling has to take into account inner-packet selectors, and matching them on per-tunnel SPD entries, possible matching a more general selector for than was expressed with IKE.

An open question is whether the ACQUIRE (which will be translated into an IKE Quick Mode Identity) should have the specifics of the inner packet, or contain the policy entry (or some subset) which marked the packet for protection (e.g. A specific IP address or an IP prefix?), or if the answer should be a configuration option. For the current IPsec specifications, interoperability testing will yield an appropriate answer. For IKEv2 and RFC 2401bis, the answer appears to be “both”, as both the packet specifics and the policy entry which triggered the ACQUIRE (or provided the answer in an INVERSE_ACQUIRE) are used by IKEv2. As with range selectors, IKEv2 and RFC 2401bis changes are beyond the scope of this project.

Configuration Command Changes

Changes to ipsecconf(1m)

A previous section detailed two different semantics: RFC 2401 and RFC 3884. The ifconfig(1m) command should not be bloated any more to allow these different configurations. If the ifconfig(1m) syntax is used, Transport Mode negotiations will be employed. (See the IKE Changes section below for implications.) This project should encourage users to use the new ipsecconf(1m) syntax described in this section.

The ipsecconf(1m) command takes three-tuples of the following form:

{pattern} action {properties}

Tunnel Reform will introduce new pattern keywords to match each of the two semantics. The following table shows examples of each, with the new keywords *tunnel* and *negotiate* (which must appear together).

Specification	***Example ipsecconf(1m) syntax and explanation.***

RFC 2401	`# Like Solaris 9, except for the`
	`# ACQUIRE messages that come up will be different,`
	`# telling IKE or other key management to negotiate`
	`# proper tunnel-mode.`
	`{tunnel ip.tun0 negotiate tunnel} ipsec`
	`  {encr_algs aes encr_auth_algs md5}`
	
	`# If you want to give a specific set of inner`
	`# packets more protection, you may do that.`
	`# laddr == inner-src outbound, inner-dst inbound.`
	`# raddr == inner-dst outbound, inner-src inbound.`
	`# When 2401-tunnel is specified, all pattern specs`
	`# refer to the INNER packet.`
	`{tunnel ip.tun0 negotiate tunnel laddr`
	`  10.0.0.0/24 } ipsec`
	`  {encr_algs aes(256) encr_auth_algs sha1}`

RFC 3884	`# Like Solaris 9, except for the`
	`# ACQUIRE messages that come up will be different,`
	`# telling IKE or other key management to negotiate`
	`# transport mode.`
	`{tunnel ip.tun0 negotiate transport} ipsec`
	`  {encr_algs aes encr_auth_algs md5}`
	
	`# NOTE:  If “negotiate transport” is present, no addresses may be present.`

Like Transport Mode, if addresses are not specified, it is assumed that both IPv4 and IPv6 traffic that matches will be affected. This is a slight semantic change from the ifconfig(1m) method of tunnel security configuration, but it matches the overall feel of ipsecconf(1m) more precisely.

Changes to ifconfig(1m)

This project should consider beginning the EOL process on “*_algs” keywords to ifconfig(1m). As it stands, ifconfig(1m) can only configure Solaris 9 semantics across the whole of the tunnel.

Changes to ipseckey(1m)

Changes in ipseckey(1m) will reflect changes in the PF_KEY messages required for this project. The PF_KEY changes are documented in the next section.

Changes to in.iked(1m)

With respect to configuration file changes, in.iked will not change. IKE internal changes are documented in the final section of this design document.

Changes to ikeadm(1m)

Modulo statistics changes in in.iked(1m), there will be no changes to ikeadm(1m).

Changes to ikecert(1m)

There will be no changes to ikecert(1m).

PF_KEY Changes

New PF_KEY extensions and values

Tunnel Reform requires some changes to the PF_KEY key management socket. Most of these changes are semantic ones, but one extension changes name, and this project requires a new extension.

```
#define SADB_X_EXT_ADDRESS_INNER_SRC SADB_EXT_ADDRESS_PROXY
```

This extension, specified in RFC 2367, but not used, represents the source address, or source address prefix, of the inner packet triggering an ACQUIRE message. It can contain an upper-layer protocol and port in the case of RFC 2401 policies. An SA can be created with this extension for precise SA lookup in the case of tunneled packets.

```
#define SADB_X_EXT_ADDRESS_INNER_DST nnn
```

This extension is also of type `sadb_address_t`. It corresponds to the destination side of an ACQUIRE-triggering packet.

An ACQUIRE with inner destination and source addresses set will generate the created of two SAs... one outbound with destination and source matching the ACQUIRE, and an inbound SA with the inner destination and source switched, as is the packet source and destination.

Finally, a bit long-requested now becomes mandatory:

#define SADB_X_SAFLAGS_TUNNEL nnn

If this bit is set on the ACQUIRE message, the key management daemon should negotiate for a Tunnel Mode SA. An SA MAY have this flag set by the key management daemon, to aid in diagnostics.

New PF_KEY message semantics

Tunnel Reform introduces two different tunnelling semantics. These two are distinguished by their requests in an SADB_ACQUIRE message from PF_KEY. These distinctions apply equally to regular ACQUIRE messages or extended ACQUIRE messages (as introduced by IKE, PSARC 1999/166).

The pf_key(7p) manual page currently shows both kinds of ACQUIRE sharing these extensions:

address (SD), (address(P))

where the source and destination addresses (of the outer packet, for a tunnel packet) are required, and the inner address is optional.

Either tunnelling semantic has different requirements for address extensions. The following table will illustrate these.

NOTE: All outer source and destination addresses have protocol set to IPPROTO_IP or IPPROTO_IPV6

Semantic	*Address extensions and values*
RFC 2401	SADB_X_SAFLAGS_TUNNEL set. Inner address extensions are REQUIRED, even if they are wildcards.
RFC 3884	SADB_X_SAFLAGS_TUNNEL is cleared. No inner address extensions. Protocol is set to ipip or IPv6.

Inverse ACQUIRE requests should send down all pertinent information from IKE, including the tunnel-mode flag if need be. It is left up to IKE (see below) to distinguish a Solaris 9 peer with respect to tunnel mode.

QUESTION: v3 or not v3

There is a movement afoot in PF_KEY community to whack RFC 2367 into a better version that all implementations can use. This movement aims to produce what might be considered version 3 of the PF_KEY specification. Tunnel Reform SHOULD attempt to deliver changes into version 3 of the PF_KEY specification.

IKE Changes

The IKE daemon needs to react to the new ACQUIRE message combinations appropriately. All of this work will be concentrated in the in.iked(1m) source. Initiating Quick Mode negotiations need to convert the ACQUIRE to an appropriate set of initiator-side Quick Mode identities. Responding Quick Mode negotiations must use the peer's set of identities to construct an inverse ACQUIRE, and use the results to construct the responder-side Quick Mode identities. Interoperability testing will be needed to see how granular or not the identity sets need to be. (e.g. A prefix or a specific address which reflects the policy, or the packet specifics.)

Negotiating for tunnels protected with transport mode is a bit unusual. Three things can happen:

1. The peer supports 3884 and the negotiation proceeds as normal.
2. The peer does not support 3884 and the negotiation fails.
3. The peer is a Solaris 9 implementation.

Solaris 9 peers do not check the tunnel/transport mode for IKE's Quick Mode as a responder. This allows Tunnel Reform's IKE to only have a two-position switch: tunnel mode or transport mode. If Solaris 9 initiates a negotiation for a tunnel, the Tunnel Reform IKE implementation will detect the tunnel mode negotiation, and the unusual Quick Mode identities (0.0.0.0/0 with protocol set to IPPROTO_ENCAP or IPPROTO_IPV6). Recognizing Solaris 9 peers will require some work, but the recognition code should be isolated, so it can be removed when Solaris 9 is EOLed.