SCTP

SCTP LSM Support

Security Hooks

For security module support, three SCTP specific hooks have been implemented:

```
security_sctp_assoc_request()
security_sctp_bind_connect()
security_sctp_sk_clone()
security_sctp_assoc_established()
```

The usage of these hooks are described below with the SELinux implementation described in the SCTP SELinux Support chapter.

security_sctp_assoc_request()

Passes the `@asoc` and `@chunk->skb` of the association INIT packet to the security module. Returns 0 on success, error on failure.

```
@asoc - pointer to sctp association structure.
@skb - pointer to skbuff of association packet.
```

security_sctp_bind_connect()

Passes one or more ipv4/ipv6 addresses to the security module for validation based on the `@optname` that will result in either a bind or connect service as shown in the permission check tables below. Returns 0 on success, error on failure.

```
  @sk      - Pointer to sock structure.
  @optname - Name of the option to validate.
  @address - One or more ipv4 / ipv6 addresses.
  @addrlen - The total length of address(s). This is calculated on each
             ipv4 or ipv6 address using sizeof(struct sockaddr_in) or
             sizeof(struct sockaddr_in6).
```

BIND Type Checks	
@optname	@address contains
SCTP_SOCKOPT_BINDX_ADD	One or more ipv4 / ipv6 addresses
SCTP_PRIMARY_ADDR	Single ipv4 or ipv6 address
SCTP_SET_PEER_PRIMARY_ADDR	Single ipv4 or ipv6 address

CONNECT Type Checks	
@optname	@address contains
SCTP_SOCKOPT_CONNECTX	One or more ipv4 / ipv6 addresses
SCTP_PARAM_ADD_IP	One or more ipv4 / ipv6 addresses
SCTP_SENDMSG_CONNECT	Single ipv4 or ipv6 address
SCTP_PARAM_SET_PRIMARY	Single ipv4 or ipv6 address

A summary of the `@optname` entries is as follows:

```
SCTP_SOCKOPT_BINDX_ADD - Allows additional bind addresses to be
                         associated after (optionally) calling
                         bind(3).
                         sctp_bindx(3) adds a set of bind
                         addresses on a socket.

SCTP_SOCKOPT_CONNECTX - Allows the allocation of multiple
                        addresses for reaching a peer
                        (multi-homed).
                        sctp_connectx(3) initiates a connection
                        on an SCTP socket using multiple
                        destination addresses.

SCTP_SENDMSG_CONNECT  - Initiate a connection that is generated by a
                        sendmsg(2) or sctp_sendmsg(3) on a new association.

SCTP_PRIMARY_ADDR     - Set local primary address.

SCTP_SET_PEER_PRIMARY_ADDR - Request peer sets address as
                             association primary.

SCTP_PARAM_ADD_IP          - These are used when Dynamic Address
```

```
SCTP_PARAM_SET_PRIMARY     - Reconfiguration is enabled as explained below.
```

To support Dynamic Address Reconfiguration the following parameters must be enabled on both endpoints (or use the appropriate **setsockopt**(2)):

```
/proc/sys/net/sctp/addip_enable
/proc/sys/net/sctp/addip_noauth_enable
```

then the following *_PARAM_*'s are sent to the peer in an ASCONF chunk when the corresponding `@optname`'s are present:

```
      @optname                     ASCONF Parameter
     ----------                   ------------------
SCTP_SOCKOPT_BINDX_ADD     ->   SCTP_PARAM_ADD_IP
SCTP_SET_PEER_PRIMARY_ADDR ->   SCTP_PARAM_SET_PRIMARY
```

security_sctp_sk_clone()

Called whenever a new socket is created by **accept**(2) (i.e. a TCP style socket) or when a socket is 'peeled off' e.g userspace calls **sctp_peeloff**(3).

```
@asoc - pointer to current sctp association structure.
@sk - pointer to current sock structure.
@newsk - pointer to new sock structure.
```

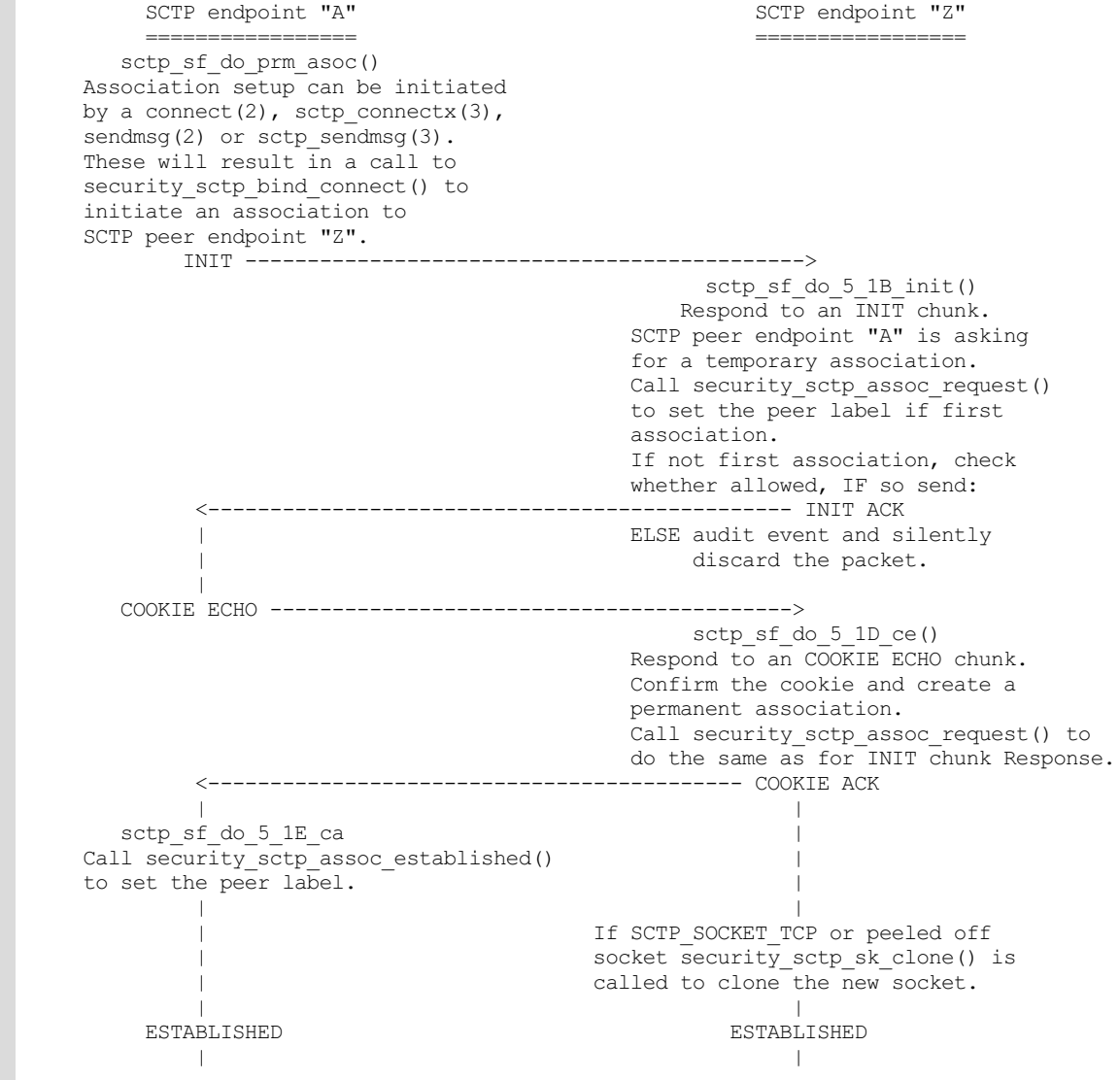
security_sctp_assoc_established()

Called when a COOKIE ACK is received, and the peer secid will be saved into `@asoc->peer_secid` for client:

```
@asoc - pointer to sctp association structure.
@skb - pointer to skbuff of the COOKIE ACK packet.
```

Security Hooks used for Association Establishment

The following diagram shows the use of `security_sctp_bind_connect()`, `security_sctp_assoc_request()`, `security_sctp_assoc_established()` when establishing an association.



```
|                    Association Established                    |
-----------------------------------------------------------------
```

SCTP SELinux Support

Security Hooks

The SCTP LSM Support chapter above describes the following SCTP security hooks with the SELinux specifics expanded below:

```
security_sctp_assoc_request()
security_sctp_bind_connect()
security_sctp_sk_clone()
security_sctp_assoc_established()
```

security_sctp_assoc_request()

Passes the `@asoc` and `@chunk->skb` of the association INIT packet to the security module. Returns 0 on success, error on failure.

```
@asoc - pointer to sctp association structure.
@skb - pointer to skbuff of association packet.
```

The security module performs the following operations:

> IF this is the first association on `@asoc->base.sk`, then set the peer sid to that in `@skb`. This will ensure there is only one peer sid assigned to `@asoc->base.sk` that may support multiple associations.
>
> ELSE validate the `@asoc->base.sk peer_sid` against the `@skb peer sid` to determine whether the association should be allowed or denied.
>
> Set the sctp `@asoc sid` to socket's sid (from `asoc->base.sk`) with MLS portion taken from `@skb peer sid`. This will be used by SCTP TCP style sockets and peeled off connections as they cause a new socket to be generated.
>
> If IP security options are configured (CIPSO/CALIPSO), then the ip options are set on the socket.

security_sctp_bind_connect()

Checks permissions required for ipv4/ipv6 addresses based on the `@optname` as follows:

```
-------------------------------------------------------------------
|                   BIND Permission Checks                        |
|       @optname             |         @address contains          |
|----------------------------|------------------------------------|
| SCTP_SOCKOPT_BINDX_ADD     | One or more ipv4 / ipv6 addresses |
| SCTP_PRIMARY_ADDR          | Single ipv4 or ipv6 address       |
| SCTP_SET_PEER_PRIMARY_ADDR | Single ipv4 or ipv6 address       |
-------------------------------------------------------------------

-------------------------------------------------------------------
|                 CONNECT Permission Checks                       |
|       @optname             |         @address contains          |
|----------------------------|------------------------------------|
| SCTP_SOCKOPT_CONNECTX      | One or more ipv4 / ipv6 addresses |
| SCTP_PARAM_ADD_IP          | One or more ipv4 / ipv6 addresses |
| SCTP_SENDMSG_CONNECT       | Single ipv4 or ipv6 address       |
| SCTP_PARAM_SET_PRIMARY     | Single ipv4 or ipv6 address       |
-------------------------------------------------------------------
```

SCTP LSM Support gives a summary of the `@optname` entries and also describes ASCONF chunk processing when Dynamic Address Reconfiguration is enabled.

security_sctp_sk_clone()

Called whenever a new socket is created by **accept**(2) (i.e. a TCP style socket) or when a socket is 'peeled off' e.g userspace calls **sctp_peeloff**(3). `security_sctp_sk_clone()` will set the new sockets sid and peer sid to that contained in the `@asoc sid` and `@asoc peer sid` respectively.

```
@asoc - pointer to current sctp association structure.
@sk - pointer to current sock structure.
@newsk - pointer to new sock structure.
```

security_sctp_assoc_established()

Called when a COOKIE ACK is received where it sets the connection's peer sid to that in `@skb`:

```
@asoc - pointer to sctp association structure.
@skb - pointer to skbuff of the COOKIE ACK packet.
```

Policy Statements

The following class and permissions to support SCTP are available within the kernel:

```
class sctp_socket inherits socket { node_bind }
```

whenever the following policy capability is enabled:

```
policycap extended_socket_class;
```

SELinux SCTP support adds the `name_connect` permission for connecting to a specific port type and the `association` permission that is explained in the section below.

If userspace tools have been updated, SCTP will support the `portcon` statement as shown in the following example:

```
portcon sctp 1024-1036 system_u:object_r:sctp_ports_t:s0
```

SCTP Peer Labeling

An SCTP socket will only have one peer label assigned to it. This will be assigned during the establishment of the first association. Any further associations on this socket will have their packet peer label compared to the sockets peer label, and only if they are different will the `association` permission be validated. This is validated by checking the socket peer sid against the received packets peer sid to determine whether the association should be allowed or denied.

NOTES:

1. If peer labeling is not enabled, then the peer context will always be `SECINITSID_UNLABELED` (`unlabeled_t` in Reference Policy).
2. As SCTP can support more than one transport address per endpoint (multi-homing) on a single socket, it is possible to configure policy and NetLabel to provide different peer labels for each of these. As the socket peer label is determined by the first associations transport address, it is recommended that all peer labels are consistent.
3. **getpeercon**(3) may be used by userspace to retrieve the sockets peer context.
4. While not SCTP specific, be aware when using NetLabel that if a label is assigned to a specific interface, and that interface 'goes down', then the NetLabel service will remove the entry. Therefore ensure that the network startup scripts call **netlabelctl**(8) to set the required label (see **netlabel-config**(8) helper script for details).
5. The NetLabel SCTP peer labeling rules apply as discussed in the following set of posts tagged "netlabel" at: https://www.paul-moore.com/blog/t.
6. CIPSO is only supported for IPv4 addressing: `socket(AF_INET, ...)` CALIPSO is only supported for IPv6 addressing: `socket(AF_INET6, ...)`

   Note the following when testing CIPSO/CALIPSO:
   a. CIPSO will send an ICMP packet if an SCTP packet cannot be delivered because of an invalid label.
   b. CALIPSO does not send an ICMP packet, just silently discards it.
7. IPSEC is not supported as RFC 3554 - sctp/ipsec support has not been implemented in userspace (**racoon**(8) or **ipsec_pluto**(8)), although the kernel supports SCTP/IPSEC.