XFRM

The sync patches work is based on initial patches from Krisztian <hidden@balabit.hu> and others and additional patches from Jamal <hadi@cyberus.ca>.

The end goal for syncing is to be able to insert attributes + generate events so that the SA can be safely moved from one machine to another for HA purposes. The idea is to synchronize the SA so that the takeover machine can do the processing of the SA as accurate as possible if it has access to it.

We already have the ability to generate SA add/del/upd events. These patches add ability to sync and have accurate lifetime byte (to ensure proper decay of SAs) and replay counters to avoid replay attacks with as minimal loss at failover time. This way a backup stays as closely up-to-date as an active member.

Because the above items change for every packet the SA receives, it is possible for a lot of the events to be generated. For this reason, we also add a nagle-like algorithm to restrict the events. i.e we are going to set thresholds to say "let me know if the replay sequence threshold is reached or 10 secs have passed" These thresholds are set system-wide via sysctls or can be updated per SA.

The identified items that need to be synchronized are: - the lifetime byte counter note that: lifetime time limit is not important if you assume the failover machine is known ahead of time since the decay of the time countdown is not driven by packet arrival. - the replay sequence for both inbound and outbound

1) Message Structure

nlmsghdr:aevent_id:optional-TLVs.

The netlink message types are:

XFRM_MSG_NEWAE and XFRM_MSG_GETAE.

A XFRM_MSG_GETAE does not have TLVs.

A XFRM_MSG_NEWAE will have at least two TLVs (as is discussed further below).

aevent_id structure looks like:

```
struct xfrm_aevent_id {
          struct xfrm_usersa_id           sa_id;
          xfrm_address_t                  saddr;
          __u32                           flags;
          __u32                           reqid;
};
```

The unique SA is identified by the combination of xfrm_usersa_id, reqid and saddr.

flags are used to indicate different things. The possible flags are:

```
XFRM_AE_RTHR=1, /* replay threshold*/
XFRM_AE_RVAL=2, /* replay value */
XFRM_AE_LVAL=4, /* lifetime value */
XFRM_AE_ETHR=8, /* expiry timer threshold */
XFRM_AE_CR=16, /* Event cause is replay update */
XFRM_AE_CE=32, /* Event cause is timer expiry */
XFRM_AE_CU=64, /* Event cause is policy update */
```

How these flags are used is dependent on the direction of the message (kernel<->user) as well the cause (config, query or event). This is described below in the different messages.

The pid will be set appropriately in netlink to recognize direction (0 to the kernel and pid = processid that created the event when going from kernel to user space)

A program needs to subscribe to multicast group XFRMNLGRP_AEVENTS to get notified of these events.

2) TLVS reflect the different parameters:

a. byte value (XFRMA_LTIME_VAL)

This TLV carries the running/current counter for byte lifetime since last event.

b)replay value (XFRMA_REPLAY_VAL)

This TLV carries the running/current counter for replay sequence since last event.

c)replay threshold (XFRMA_REPLAY_THRESH)

This TLV carries the threshold being used by the kernel to trigger events when the replay sequence is exceeded.

d. expiry timer (XFRMA_ETIMER_THRESH)

This is a timer value in milliseconds which is used as the nagle value to rate limit the events.

3) Default configurations for the parameters:

By default these events should be turned off unless there is at least one listener registered to listen to the multicast group XFRMNLGRP_AEVENTS.

Programs installing SAs will need to specify the two thresholds, however, in order to not change existing applications such as racoon we also provide default threshold values for these different parameters in case they are not specified.

the two sysctls/proc entries are:

a) /proc/sys/net/core/sysctl_xfrm_aevent_etime used to provide default values for the XFRMA_ETIMER_THRESH in incremental units of time of 100ms. The default is 10 (1 second)

b) /proc/sys/net/core/sysctl_xfrm_aevent_rseqth used to provide default values for XFRMA_REPLAY_THRESH parameter in incremental packet count. The default is two packets.

4) Message types

a. XFRM_MSG_GETAE issued by user-->kernel. XFRM_MSG_GETAE does not carry any TLVs.

The response is a XFRM_MSG_NEWAE which is formatted based on what XFRM_MSG_GETAE queried for.

The response will always have XFRMA_LTIME_VAL and XFRMA_REPLAY_VAL TLVs. * if XFRM_AE_RTHR flag is set, then XFRMA_REPLAY_THRESH is also retrieved * if XFRM_AE_ETHR flag is set, then XFRMA_ETIMER_THRESH is also retrieved

b. XFRM_MSG_NEWAE is issued by either user space to configure or kernel to announce events or respond to a XFRM_MSG_GETAE.

i. user --> kernel to configure a specific SA.

any of the values or threshold parameters can be updated by passing the appropriate TLV.

A response is issued back to the sender in user space to indicate success or failure.

In the case of success, additionally an event with XFRM_MSG_NEWAE is also issued to any listeners as described in iii).

ii. kernel->user direction as a response to XFRM_MSG_GETAE

The response will always have XFRMA_LTIME_VAL and XFRMA_REPLAY_VAL TLVs.

The threshold TLVs will be included if explicitly requested in the XFRM_MSG_GETAE message.

iii. kernel->user to report as event if someone sets any values or thresholds for an SA using XFRM_MSG_NEWAE (as described in #i above). In such a case XFRM_AE_CU flag is set to inform the user that the change happened as a result of an update. The message will always have XFRMA_LTIME_VAL and XFRMA_REPLAY_VAL TLVs.

iv. kernel->user to report event when replay threshold or a timeout is exceeded.

In such a case either XFRM_AE_CR (replay exceeded) or XFRM_AE_CE (timeout happened) is set to inform the user what happened. Note the two flags are mutually exclusive. The message will always have XFRMA_LTIME_VAL and XFRMA_REPLAY_VAL TLVs.

Exceptions to threshold settings

If you have an SA that is getting hit by traffic in bursts such that there is a period where the timer threshold expires with no packets seen, then an odd behavior is seen as follows: The first packet arrival after a timer expiry will trigger a timeout event; i.e we don't wait for a timeout period or a packet threshold to be reached. This is done for simplicity and efficiency reasons.

-JHS