IPVS Syncd Strong Authentication extension

Alexandre Cassen
Linux Virtual Server OpenSource Project
Paris, France, March 2004
acassen@linux-vs.org, http://www.LinuxVirtualServer.org/~acassen/

Introduction

The remainder of this document describes the features, design goals and theory of syncd strong authentication extension. Current syncd multicasts IPVS connection entries in a plaintext fashion. These multicasted messages are caught by backup IPVS routers subscribed to syncd multicast group and then appended into local router IPVS connection table. Since IPVS loadbancing decisions are scheduled using this connection table, this document is an attempt to add authentication provisions into syncd to protect against packets injection and other malicious attacks.

Syncd Authentication global design

To support message authentication, we add a new header :

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------------------------+
|                        Sequence Number                        |
+---------------------------------------------------------------+
|                                                               |
|                       HMAC-MD5-96bit ICV                      |
|                                                               |
+---------------------------------------------------------------+
```

We add two new fields. A 32-bit sequence number field to deal with anti-replay attacks and a 96-bit trunc ICV for keyed digest. This design is inspired from the one we found in [IPSEC-AH]. This new header will be inserted between ip_vs_sync_mesg header and first ip_vs_sync_conn. As keyed digest algorithm, we will use [HMAC-MD5]. The main goal is to compute an Integrity Check Value (ICV), using [HMAC-MD5], over the whole syncd message. The sequence number is part of the computation. This brings the following benefit :

- *Anti-replay prevention*: Since sequence number is part of the ICV computation, any attacks based on packet replay will be dealt.
- *Message manipulation*: Any attacks based on manipulating the message field in order to pertube final IPVS scheduling decision will be dealt.

For simplicity reasons, we will not implement any kind of key exchange mechanism. icv-key used for ICV computation will be locally configured on all IPVS routers. So administrator must ensure this icv-key is the same on all IPVS routers. To keep things simple, we will not support different icv-key specification for master and backup state syncd, instead the same icv-key will be used for both state. The last assumption made is to not support asymmetric message authentication handling. In this last point, we mean that an IPVS router can not deal with authenticated messages for master state and unauthenticated messages for backup state (and reciprocity), if authentication is set at the master state, then syncd will assume that incoming messages in backup state MUST use authentication. This is kind of binary authentication selection while configuring syncd, use or not to use the syncd message authentication that is the question.

The Sequence number is monotonically increased by one each time a new syncd message is created. Since syncd is not an election protocol we don't need to deal with kind of anti-cycle mechanism in order to broke a potential dropping loop. Instead, the syncd maintains a local sequence number counter as dropping policy. This mean, while processing incoming syncd message, the sequence number received in the syncd message is compared with a local copy, if sequence number in the syncd message is greater than the local copy then the message is granted otherwise dropped.

Master state extensions

For optimization reasons, the hmac-md5 at master state will be processed using incremental update. The syncd code uses the Kernel [Crypto API]. The ICV computation is done using the following steps:

- *icv_init*: When a new syncd message is allocated we first start with headers initializations. First one is the ip_vs_sync_icv, we set the sequence number to the locally counter increased by one and zero the ICV field. Next we initialize the hmac-md5 tfm and update it using the previously ip_vs_sync_icv initialized :

```
/*
 * MD5 update start with icv  header. We skip the
 * sync_mesg header  since  nr_conns and size are
 * mutable  during  MD5 update  until  curr_sb is
 * fully filled.
 */
```

- icv_update: Next when a new connection is received by the ip_vs_sync_conn function, those connections data are appended to the current sync buffer and the hmac-md5 tfm is updated. This process continues until max buffer sending size is reached.
- icv_final: When syncd message is ready we simply update the hmac-md5 with the ip_vs_sync_mesg data finish the hmac-md5 tfm to generate the final digest. This value is then set to the ip_vs_sync_icv's icv field :

```
/*
 * Final MD5  Update. The  last  MD5  incremental
 * update is done  on the sync_mesg  header since
 * the nr_conns and size fields are now inmutable.
 */
```

Backup state extensions

hmac-md5 updates are not commutative operations. This is why we need to use the same update order as during master production. Only the connections entries buffer can be factorized to a single hmac-md5 update. This is why we divided the incoming message sanity check into three steps :

- *Initialize*: Initialize the backup hmac-md5 tfm. Copy the incoming message ICV field into a temporary place and zero the field. Start with a first update over the ip_vs_sync_icv header.
- *Compute connections*: Update hmac-md5 over the whole connection buffer.
- *Generate ICV*: The last hmac-md5 update is done over the ip_vs_sync_mesg header and the result is generated. The very last step compares both ICV to drive the dropping decision.

For performance reasons and since crypto step are CPU consuming, we optimized the sanity check to first test the sequence number. If sequence number is lower than local copy then no need to compute hmac-md5, we simply drop these packets since it refers a sequence already processed.

Incoming message processing security policy

The last pending point is for the ip_vs_sync_mesg header. Since, syncd messages can be generated using two different policies (with or without authentication), we need to give to the receiving point a clue on which policy must be used on his side.

We have 2 alternatives:

- New 8bit field to store options. Instead of a fully qualified 'auth' field, we prefer 'Options' since we will be able to store other infos than auth ones:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------------------------------+
|  Count Conns  |    SyncID     |             Size              |
+---------------+---------------+-------------------------------+
|    Options    |                   Reserved                    |
+---------------+-----------------------------------------------+
```

- Choose the syncd processing mode based on configuration step.

Incoming messages processing is critical since connections presents in messages will be appended into IPVS connection table driving the IPVS scheduling decision. It is much more safe setting the daemon processing mode during configuration step than letting daemon determine which mode must be used based on incoming messages options field. If malicious packets are injected, this field value can totally turn off the ICV protection, which will make ICV protection a fake extension. We prefer letting configuration decision to the administrator than auto-select processing mode. This means that administrator MUST configure all the syncds to use the same processing mode.

For example, consider daemon in backup state configured to use authentication mode processing. If malicious packet is injected without ICV header, this will only have for effect to introduce buffer offset while processing message, and icv value computed will not be valid and so packet will be dropped.

Configuration considerations

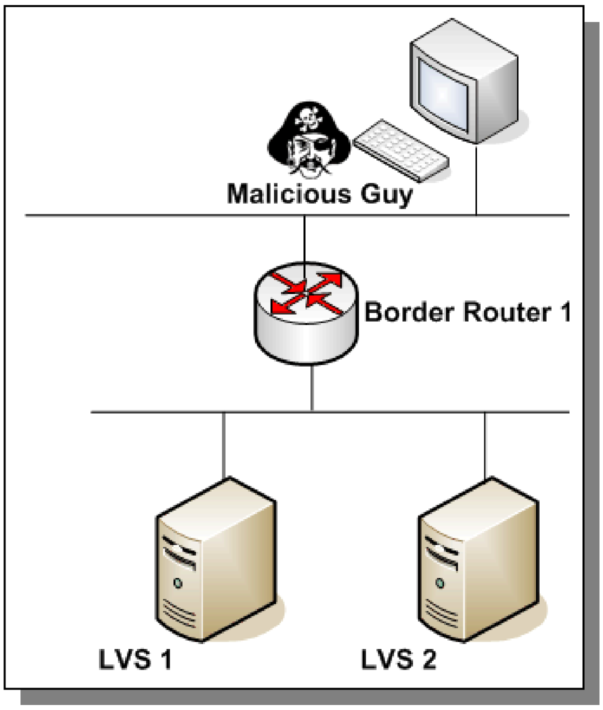
To use authentication mode, a new ipvsadm option have been created. This option is '--icv-key'. If specified on the ipvsadm command line, daemon is configured to use authentication mode with the specified secret key for ICV computation. As previously discussed, syncd will not support mixed mode for both master and backup daemon. If one daemon start with authentication mode, then when starting second one it will automatically use authentication mode (and reciprocity).

For instance:

```
ipvsadm --start-daemon master --syncid 150 --icv-key toto234
ipvsadm --start-daemon backup --syncid 150
```

TTL considerations

The current syncd design is to set IP TTL to 1 so that multicasted messages can not be forwarded by borders routers. We just want to discuss here to potentially switch this design to use TTL=255 instead. We want to introduce security while processing incoming messages. This is much more important securing receiving rather than sending point since incoming messages received will drive IPVS scheduling decision. We want to limit as much as possible packets injections, especially if packet are coming from border router's network. Consider the following topology:



In this topology, LVS 1 and LVS 2 are using syncd in both master & backup state. Consider that Router 1 is miss-configured, or insecure, or local administrator doesn't trust Router 1 administrator. From both LVS 1 & LVS 2 director, receiving a multicast IP message with TTL=1 is ok. But considering IP protocol on the receiving point, this mean that we are the last forwarding router capable for this datagram. This mean that we can receive such packet coming from Malicious guy on a border network. For instance, on our diagram, if Malicious guy forge a packet with TTL=2 (and router 1 can forward this), then LVS router will receive this packet with TTL=1.

If we use TTL=255, then things are much more complicated for malicious guy. because TTL=255 is the maximum IP filed value, and according to IP protocol, TTL is decremented hop-by-hop. So if malicious guy send packet the max value LVS directors will receive is 254. So if we use TTL=255 and syncd receives packet with 255 TTL field value then this intrinsically mean that this message has been generated on the same network segment as the receiving point. This avoids injection from border routers.

Plus, the use of TTL=255 bring optimization benefit since even if packets are valid (good icv, and all sanity check), the first sanity dropping decision is made upon this TTL field. If TTL<>255, then packet is dropped even if valid, which will not monopolize CPU while computing ICV.

Another point, if we want to hide our syncd stream, we just need to ensure that border routers are not using any kind of mcast routing protocol (PIM, DVMRP, ...). This will ensure syncd traffic is only going on a local network segment.

References

[IPSEC-AH]	S. Kent, R. Atkinson, « IP Authentication Header», RFC 2402, November 1998.
[HMAC-MD5]	Madson, C., and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", Work in Progress.
[CryptoAPI]	Linux Kernel source tree: Documentation/crypto/api-intro.txt