Core testing infrastructure - Bug #4576

ttcn3-hlr-tests need to route multicast traffic between ttcn3-hlr-test and osmo-hlr-master containers

06/02/2020 10:29 PM - neels

Status:	Stalled	**Start date:**	06/02/2020
Priority:	High	**Due date:**	
Assignee:	neels	**% Done:**	40%
Category:			
Target version:			
Spec Reference:			

Description

The MSLookup tests in the ttcn3-hlr-test send out multicast-DNS queries,
and these should be answered by osmo-hlr running in the osmo-hlr-master container.

Running wireshark on the docker-hosting machine on 'any', the DNS queries are visible:

```
No.     Time           Arrival Time Source                Source Port Destination           Destin
ation Port Info
 102721 4919.259341    Jun  3, 2020 00:16:05.287612000 CEST 172.18.10.103         4266        239.
192.23.42         4266             Standard query 0x5ca3 ANY sip.voice.491618327224.msisdn.mdns.os
mocom.org
 102722 4919.259341    Jun  3, 2020 00:16:05.287612000 CEST 172.18.10.103         4266        239.
192.23.42         4266             Standard query 0x5ca3 ANY sip.voice.491618327224.msisdn.mdns.os
mocom.org
```

Running tcpdump in the osmo-hlr-master container shows no such activity.

When running the osmo-mslookup-client commandline tool with the same query, osmo-hlr does receive it and responds in the way that the test expects it:

```
# osmo-mslookup-client sip.voice.262422543586245.imsi
query    result    last    age    v4_ip    v4_port    v6_ip    v6_port
sip.voice.262422543586245.imsi    result    last    257    66.66.66.66    5060
```

But also, the tester does not receive that response
(as I found out by "infinitely" looping the MSLookup send and receive in HLR_Tests.ttcn).

I am not sure how osmith managed to test the MSLookup tests successfully,
there must be some way to get the multicast traffic to pass through the containers.

History

#1 - 06/02/2020 10:32 PM - neels

Looking on the docker-hosting machine, I see some restrictive rules in "DOCKER-ISOLATION-STAGE-1",
maybe some custom iptables could make the multicast pass through?
OTOH, do we really need to resort to manual iptables rules?

```
sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
DOCKER-USER  all  --  anywhere             anywhere
DOCKER-ISOLATION-STAGE-1  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
DOCKER     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
```

```
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain DOCKER (1 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target     prot opt source               destination
DROP       all  -- !172.18.10.0/24       anywhere
DROP       all  --  anywhere            !172.18.10.0/24
DROP       all  -- !172.18.2.0/24        anywhere
DROP       all  --  anywhere            !172.18.2.0/24
DOCKER-ISOLATION-STAGE-2  all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-USER (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

(BTW, during this iptables, ttcn3-bsc-tests was also active on 172.18.2.0/24. ttcn3-hlr-tests is using 172.18.10.0/24)

#2 - 06/02/2020 10:59 PM - neels

On the docker-hosting machine, when I run

```
iptables -I DOCKER-ISOLATION-STAGE-1 -s 172.18.10.0/24 -d 239.192.23.42 -j ACCEPT
```

the traffic reaches the osmo-hlr-master container and osmo-hlr responds as expected.
However, the response still doesn't trigger the ttcn3 tester.

```
316 15.062310      Jun  3, 2020 00:50:04.994885000 CEST 172.18.10.103         4266        239.192.23.42
  4266             Standard query 0x7463 ANY sip.voice.491615433824.msisdn.mdns.osmocom.org
327 15.063073      Jun  3, 2020 00:50:04.995648000 CEST 172.18.10.20          4266        239.192.23.42
  4266             Standard query response 0x7463 TXT A 66.66.66.66 TXT
```

The response also goes from 172.18.10.0/24 to 239.192.23.42, and I can also see it in a tcpdump running in the nonjenkins-ttcn3-hlr-test container.
So there is still some reason why the tester doesn't pick up the response, even though it is apparently reaching inside the container.

Would be nice to get a hint from osmith about how stuff used to work. I'll ping him tomorrow.

#3 - 06/02/2020 11:00 PM - neels

- Status changed from New to Feedback

- Assignee set to osmith

#4 - 06/03/2020 08:10 AM - laforge

On Tue, Jun 02, 2020 at 10:59:56PM +0000, neels [REDMINE] wrote:

> The response also goes from 172.18.10.0/24 to 239.192.23.42, and I can also see it in a tcpdump running in the nonjenkins-ttcn3-hlr-test container.

maybe the multicast TTL (set by the sender) is not high enough and it times out (reaches zero)
in the target (destination, receiver) container and hence is dropped before being delivered?

Unfortunately the pcap of the tcpdump was not attached here, otherwise I could have looked at
it quickly.

There is also the possibility of iptables/ntftables rules inside the destination container,
maybe also set up by docker? There is one separate instance of all the packet filter
tables/chains/hooks/rules in every network namespace, i.e. one on the host and one in each of the
containers.

#5 - 06/03/2020 01:05 PM - neels

- File on_docker_host.pcapng added

- File nonjenkins-ttcn3-hlr-test.pcap added

- File nonjenkins-hlr.pcap added

interesting to note, when I restarted the hlr test docker containers today, the iptables rule to accept the multicast address was still there, but docker inserted new DROP rules before my rule, rendering my rule ineffective.
So I need to re-add the multicast ACCEPT rule as first rule every time the test docker containers restart.

This time I did only -d, no -s

```
iptables -I DOCKER-ISOLATION-STAGE-1 -d 239.192.23.42 -j ACCEPT
```

Took three pcaps, on the docker host, inside the ttcn3 tester container, inside the osmo-hlr container (attached).

The UDP "Time to live" is 1. Seems too low considering a bridge between the containers?
But still, the MSLookup request is actually received by osmo-hlr, only the response (also UDP ttl == 1) is not received by the tester.

#6 - 06/03/2020 01:34 PM - neels

neels wrote:

> The UDP "Time to live" is 1.

of course IP ttl

#7 - 06/03/2020 02:39 PM - neels

- Status changed from Feedback to In Progress

- Assignee changed from osmith to neels

- % Done changed from 0 to 40

about the question why it worked for osmith but not for jenkins nor me:
osmith tells me that he probably ran the tests without docker containers, with ttcn tester and osmo-hlr all running on the host machine directly.
So indeed we need to figure out an all-new way how docker allows the MC UDP traffic across containers.

I'm at the point where I am now seeing the mDNS response on the tcpdump run right next to the ttcn tester program,
but still the tester doesn't seem to pick them up.

In terms of the IP TTL, the tcpdump is on the same stage as the tester, right?
If the tcpdump in that container sees it, the tester should, too?

I still need to examine the iptables within the docker container, haven't done that yet.

#8 - 06/03/2020 06:00 PM - laforge

On Wed, Jun 03, 2020 at 02:39:09PM +0000, neels [REDMINE] wrote:

> In terms of the IP TTL, the tcpdump is on the same stage as the tester, right?

no. tcpdump works (despite its name) on the Ethernet layer, or more precisely at
the boundary between Layer2 and Layer3. This means all of the local IP stack, UDP and socket
code has not yet been traversed. So it may very well be that the IP stack is decrementing
the TTL to 0 and then dropping the packet. There might be some related counters visible in
'lnstat', but I think the easiest thing is to increase the TTL on the sender side (socket option)
and see if it arrives.

The above is also the reason why you see packets in tcpdump which are then subsequenty dropped
by packet filter rules.

> If the tcpdump in that container sees it, the tester should, too?

No, see above.

#9 - 06/03/2020 06:45 PM - laforge

On Wed, Jun 03, 2020 at 07:59:39PM +0200, Harald Welte wrote:

> On Wed, Jun 03, 2020 at 02:39:09PM +0000, neels [REDMINE] wrote:
>
> > In terms of the IP TTL, the tcpdump is on the same stage as the tester, right?

> There might be some related counters visible in 'lnstat', [...]

not in lnstat, but in /proc/net/snmp

```
cat /proc/net/snmp
```

inside the receiving container should give you the following two lines:

```
Ip: Forwarding DefaultTTL InReceives InHdrErrors InAddrErrors ForwDatagrams InUnknownProtos InDiscards InDeliv
ers OutRequests OutDiscards OutNoRoutes ReasmTimeout ReasmReqds ReasmOKs ReasmFails FragOKs FragFails FragCrea
tes
Ip: 1 64 21422602 0 0 1931698 0 0 19101351 11638738 6387 2468 0 0 0 0 0 33 0
```

So the fourth numeric field in the second line is InHdrErrors, which is the value that's incremented every time an IP packet TTL exceeds (or other IP header problems are encountered, but as you can see in the above example, it's still 0 on my laptop with an uptime of 24 days, so you shouldn't see to many other errors incrementing that value.

#10 - 06/08/2020 02:49 PM - neels

I hacked a TTL 2 into osmo-hlr's osmo_mdns_sock_init(), the answer still does not reach the tester.
Now the tester sends out an mDNS query with TTL 1 that reaches osmo-hlr fine,
then osmo-hlr sends back an mDNS response with TTL 2 that does not reach the tester.

(Just to make sure I also tried with TTL 3)

Seems like the TTL is not the problem.

(We should probably anyway consider adding the multicast TTL as config option to osmo-hlr and osmo-mslookup-client, independently from this issue)

#11 - 06/08/2020 03:03 PM - neels

running iptables inside the docker containers seems not to be an option at all.
I installed the 'iptables' package, then get:

```
root@4c22177bd320:/data# iptables -L
iptables v1.6.0: can't initialize iptables table `filter': Permission denied (you must be root)
```

(but obviously am root in the container)

#12 - 06/08/2020 03:10 PM - neels

considering: could it make sense to, just for those tests, configure osmo-hlr to listen and respond to mslookup mDNS in **unicast**, not multicast? Seems ugly, but if it helps with testing.....

#13 - 06/08/2020 03:33 PM - neels

Looking closely in the ttcn3 log output, I see the mDNS port being set up and messages going out on it, and also the same sent messages being received right back on mDNS (as expected in multicast),
but no answer coming from osmo-hlr is received.

...maybe change the HLR tests so that osmo-hlr and the ttcn tester run in the same docker image?

#14 - 10/28/2020 08:13 PM - neels

in some forum i got this hint:

```
Use the '--network host' option, and the container will use same network stack as the host operating system. T
his only works on Linux, though.
```

#15 - 10/28/2020 08:14 PM - neels

erm that '--network host' was actually an answer to the question of how to run a service in a docker container that should be fully visible on a public network interface, not necessarily about multicast between containers with separate networks

#16 - 10/28/2020 08:40 PM - fixeria

Unfortunately, yes: '--network host' is the only way to get multicast traffic working in Docker:

https://stackoverflow.com/questions/51737969/how-to-support-multicast-network-in-docker
https://github.com/moby/libnetwork/issues/552

This is another example why (at least now) Docker is not a good fit for building telecom infrastructure on top of it.

#17 - 10/28/2020 09:48 PM - neels

i'm intuitively thinking about a simplistic "multicast repeater" thing that could forward DGSM multicast requests between two separate networks via unicast UDP...
we'd set that up between two docker containers to get our DGSM tests working.
maybe such could even be useful in some or other real world deployments one day, say two villages separated by some NAT or something?
(another less simple idea is to add the unicast option directly to osmo-hlr and the osmo-mslookup-client used by hook scripts,
i.e. allow configuring a specific unicast target for DGSM instead of, or maybe in addition to, multicast?)
laforge your views on this would be welcome, you invariably have a bigger picture on these things than i do.

#18 - 10/28/2020 09:48 PM - neels

- Status changed from In Progress to Stalled

Files

on_docker_host.pcapng	8.11 KB	06/03/2020	neels
nonjenkins-ttcn3-hlr-test.pcap	2.4 KB	06/03/2020	neels
nonjenkins-hlr.pcap	2.4 KB	06/03/2020	neels