pfSense - Feature #11354

WireGuard should respond from the address used by peer

02/01/2021 12:43 PM - Jim Pingle

Status:	Resolved	**Start date:**	02/01/2021
Priority:	Low	**Due date:**	
Assignee:	Peter Grehan	**% Done:**	100%
Category:	WireGuard	**Estimated time:**	0.00 hour
Target version:	2.5.0		
Release Notes:	Default		

Description

When a WireGuard peer contacts the firewall, the firewall always responds from the address it deems closest to the client, determined by the routing table. If there are multiple addresses on an interface, this means that the kernel may respond to a WireGuard packet from an address the remote peer doesn't expect. In some cases, this can operate OK, but others fail. Ideally responses should be sourced using the same address a client used, and not the closest address. Not sure how viable this is, but some features like HA and certain Multi-WAN use cases require it to function properly.

For example, in an HA configuration, the primary node may have an address like 198.51.100.12, but a CARP VIP of 198.51.100.11. Remote peers should always contact the CARP VIP for proper failover, but when they do, packets come to the firewall on 198.51.100.11 and replies are sourced from 198.51.100.12.

Example: HA node with 198.51.100.12 on interface, VIP of 198.51.100.11, and remote peer of 198.51.100.140. Remote peer is configured for an endpoint of 198.51.100.11:51821, but when the peer sends a packet:

```
12:59:00.174199 IP 198.51.100.140.51783 > 198.51.100.11.51821: UDP, length 148
12:59:00.177233 IP 198.51.100.12.51821 > 198.51.100.140.51783: UDP, length 92
```

The internal roaming feature of WireGuard allows that to work for some cases but it isn't ideal. If the primary node fails the peer will still keep trying to contact it directly since it keeps its last known peer endpoint, rather than using the original CARP VIP endpoint address which should move to a secondary node. The remote peer end may need to manually be restarted in this case to reset its knowledge of the last known peer address.

Also if an administrator wishes to run WireGuard on an alternate WAN that isn't the default route, responses would be sent out using the wrong address and path back to the peer. In some cases that can be worked around, like for static peers using a static route, but that also isn't ideal.

If responses were sourced using the same address the client used, all of this would work automatically.

Additional Notes:

- We can't work around this with outbound NAT when initiating locally because when failover is triggered, the existing state can't match. The source on the existing NAT state is the interface address on the primary node, which won't work when the secondary node has MASTER status on the CARP VIP. The state would need to be manually cleared.
- Can't work around it with port forwards because the responses never match the port forward states as they come from the default WAN address.
- Outbound NAT also fails for cases of remote initiation (e.g. Remote Access clients) as inbound states already exist for the exact ports needed, so NAT fails to create a new state. Using NAT rules without static port ends up back at the first point above. Works at first, but doesn't work when failing over.

Related issues:

Related to Feature #11302: WireGuard XMLRPC sync	**New**	**01/23/2021**

History

#1 - 02/01/2021 12:43 PM - Jim Pingle

- Related to Feature #11302: WireGuard XMLRPC sync added

#2 - 02/01/2021 12:44 PM - Jim Pingle

- Target version set to CE-Next

Not a blocker since, if it is possible, this is likely non-trivial.

#3 - 02/01/2021 02:00 PM - Christian McDonald

Was just about to post this exact issue. As it stands currently, I don't believe there is a way to utilize a CARP VIP (or IP alias) as remote endpoint for remote access clients. I've only had success with hitting the WAN interface address.

For what it's worth, there is a thread on another forum that will remain nameless here, but let's just say they have been talking about the same issue and they suggest it might be a limitation with FreeBSD at this time...the topic ID on that forum is 19327.0 (index.php?topic=19327.0), I leave the rest for the initiated to figure out where to look for this. One solution that was offered is to use a inbound NAT rule to port forward 51820 (or whatever your WG port is) from the CARP VIP to the WAN interface VIP. I haven't tested this myself but worth mentioning.

Edit: Just saw Jim's post on #11302.

#4 - 02/01/2021 02:44 PM - Jim Pingle

I added notes about this limitation in the docs for now: https://docs.netgate.com/pfsense/en/latest/vpn/wireguard/limitations.html

#5 - 02/01/2021 02:47 PM - Jim Pingle

Christian McDonald wrote:

> One solution that was offered is to use a inbound NAT rule to port forward 51820 (or whatever your WG port is) from the CARP VIP to the WAN interface VIP

Port forwards do not help, nor does any NAT really. See my "Additional Notes" section in the description of this issue.

#6 - 02/02/2021 06:02 PM - Peter Grehan

I've had a look at this: it may not be too bad.

The source address for the peer is already recorded to be used in replies, so it's not that much of a stretch to save the destination address as well (if unicast).

On output, the IP_SENDSRCADDR option can be used to force the source address.

I need to check the interaction with interface configuration to make sure that addresses are flushed if e.g. deleted from an interface.

#7 - 02/02/2021 06:13 PM - Peter Grehan

Actually: the code is already doing this - it may not be saving the incoming source addr in all situations. Will check on that.

#8 - 02/04/2021 06:07 AM - Peter Grehan

- File if_wg.ko.carp added

- File pfsense_wg_reflect_addrs.diff added

- Status changed from New to Feedback

- % Done changed from 0 to 80

I believe this is now fixed. The destination address of ingress wg packets wasn't being saved. This is now being done, and the existing code written to override the source address on send worked fine.

I've tested this with both v4 and v6 endpoints, and also with a v4 CARP endpoint. I verified that 2 pfsense systems with a CARP endpoint and identical keys was able to fail-over a ping to the CARP address.

(Note: the failover took about 15 seconds, even though the CARP address itself took just a few seconds. This might require some tuning on the time between key updates, or possible future development to sync key state between h/a pairs).

I've attached the diff I used, along with a wireguard module containing this patch for testing. Let me know how this goes.

#9 - 02/04/2021 08:05 AM - Jim Pingle

- Status changed from Feedback to New

It's definitely better with that if_wg.ko. When the peer sends packets, it replies from the correct address.

Testing with Multi-WAN, it responds as expected to a remote peer. If the remote peer sends traffic to WAN1, it responds from WAN1 and works. If the remote peer sends traffic to WAN2, it responds from WAN2 and works.

With CARP is appears to work OK if I reboot a node to trigger failover, but not if I demote it and keep the "failed" node up. When I trigger a failover by demoting, the remote peer behaves inconsistently. Sometimes it won't pass traffic again unless I stop it for a while, other times it starts chatting with the "failed" node directly. Checking packet captures, the remote peer gets a response from the "failed" CARP node's real interface address.

For example:

```
-- Working OK to .11 VIP
08:45:50.199395 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 190: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 148
08:45:50.201240 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 134: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 92
08:45:50.202544 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
08:45:50.202691 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 74: 198.51.100.11.51821
 > 198.51.100.140.52222: UDP, length 32
08:45:50.202776 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 96
08:45:51.220679 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
08:45:51.220893 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 96
08:45:52.238625 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
08:45:52.238879 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 96
08:45:53.250741 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
08:45:53.250944 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 96
08:45:54.273810 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
08:45:54.274026 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 96
08:45:55.290082 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
```

```
08:45:55.290379 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 96
08:45:56.306323 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
08:45:56.306487 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.11.5182
1 > 198.51.100.140.52222: UDP, length 96
08:45:57.329751 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96

-- Initiated CARP failover (Primary into maintenance mode (demoted) and is BACKUP, secondary becomes MASTER)
08:46:02.283152 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96
08:46:07.269692 62:3e:47:93:f8:08 > 00:00:5e:00:01:0b, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.11.51821: UDP, length 96

-- Primary responds from its own "real" address after a few failed pings
08:46:12.190178 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 190: 198.51.100.12.5182
1 > 198.51.100.140.52222: UDP, length 148

-- Remote peer updates its endpoint and now shows the interface address, not the VIP, and continues to communi
cate with the "failed" node.
08:46:12.192043 62:3e:47:93:f8:08 > 00:0c:29:e7:59:af, ethertype IPv4 (0x0800), length 134: 198.51.100.140.522
22 > 198.51.100.12.51821: UDP, length 92
08:46:12.192878 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 74: 198.51.100.12.51821
 > 198.51.100.140.52222: UDP, length 32
08:46:12.272922 62:3e:47:93:f8:08 > 00:0c:29:e7:59:af, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.12.51821: UDP, length 96
08:46:12.273126 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.12.5182
1 > 198.51.100.140.52222: UDP, length 96
08:46:13.286429 62:3e:47:93:f8:08 > 00:0c:29:e7:59:af, ethertype IPv4 (0x0800), length 138: 198.51.100.140.522
22 > 198.51.100.12.51821: UDP, length 96
08:46:13.286611 00:0c:29:e7:59:af > 62:3e:47:93:f8:08, ethertype IPv4 (0x0800), length 138: 198.51.100.12.5182
1 > 198.51.100.140.52222: UDP, length 96
```

I also noticed when rebooting to trigger failover that when it recovers, the secondary node sends packets from its real address as well but since the remote peer is happily talking to the primary node using the VIP at that point it appears to ignore them.

So it's much closer but not quite 100% yet.

#10 - 02/04/2021 08:46 AM - Renato Botelho

- Target version changed from CE-Next to 2.5.0

I'm going to merge this patch before next snapshot

#11 - 02/04/2021 08:49 AM - Renato Botelho

Done

#12 - 02/04/2021 06:54 PM - Peter Grehan

I only tried with reboot failover which simplifies the problem: there are no races where packets can be queued awaiting processing while CARP state is changing.

There may need to be some CARP-specific hooks to drop packets that have been sent to a virtual IP that is no longer available, or to drop packets when forcing the source address, if that address no longer exists.

#13 - 02/05/2021 09:57 AM - Jim Pingle

- Status changed from New to Feedback

- % Done changed from 80 to 100

Latest snapshot has the changes from the patch above, and the responses are sent back from the address used to contact the local peer. That looks good for now.

IMO, any remaining refinements can be pushed to the next (or future) releases. It's much better at this point than it was before and anything else would really be a separate issue, like dealing with CARP-specific items or ways to influence the address used for initiating outbound traffic.

I'll leave this open briefly for some additional feedback in case it doesn't work as expected for others, but to me it is good enough to close for 2.5.0.

#14 - 02/09/2021 12:18 PM - Renato Botelho

- Status changed from Feedback to Resolved

It's working as expected

Files

if_wg.ko.carp	339 KB	02/04/2021	Peter Grehan
pfsense_wg_reflect_addrs.diff	2 KB	02/04/2021	Peter Grehan