Proxmox and NAT64 - jool seems incompatible with PVE Firewall, Tayga to rescue

Date:	2022-03-03
modified:	2022-03-03
tags:	proxmox, nat64, jool, tayga, pve, firewall
description:	Proxmox PVE Firewall seems incompatible with jool NAT64 module
category:	Linux
slug:	proxmox_and_nat64_jool_seems_incompatible_with_pve_firewall_tayga_to_rescue
Author:	Dominik Wombacher
lang:	en
transid:	proxmox_and_nat64_jool_seems_incompatible_with_pve_firewall_tayga_to_rescue
Status:	published

I'm using IPv6-only whenever possible, but some sites like github.com are still not reachable via IPv6. In such a case NAT64 is the way to go and to simplify the setup as well as to avoid wasting further IPv4 addresses, I wantend to configure it directly on my Proxmox PVE Hosts instead a separate Router VM or similar.

After some research I decided to give Jool a try, it's running as Kernel module, what's promising from a performance perspective and is still under some sort of active development.

Compared to Tayga, which is running in the userland and had the last release in June 2011.

Spoiler: Due to issues in combination with the PVE Firewall, I had to use Tayga at the end, keep reading for the details.

I just wanted to use stateful NAT64, so the setup on my Proxmox Host, which is running Debian 11, was quite easy but it's necessary to install a lot of devel packages and a compiler to build the kernel module, also the module will be rebuild during every kernel upgrade, that might not in all Environments acceptable.

```
apt install -y pve-headers-$(uname -r)
apt install -y jool-dkms jool-tools

mkdir /etc/jool

/etc/jool/jool.conf
```
```
{
      "instance": "default",
      "framework": "netfilter",
      "global": {
              "pool6": "64:ff9b::/96"
      }
}
```
```

systemctl enable --now jool
```

That's it, was working as a charme and I thought this will be my new standard setup, until I enabled the Proxmox Firewall, which immediately dropped any NAT64 traffic. But nothing was blocked by the Firewall also nothing showed up in the logs.

There wasn't much I could find online, doesn't look like a widely used or even evaluated setup, *jool* has 0 search results in the Proxmox Forum for example.

After a few hours, I was able to track it down and from my understanding about the implementation of *jool* as well as the Proxmox PVE Firewall, both just seem to be incompatible.

Debug output of a successful routed package with deactivated firewall:

```
Jool NAT64/8fee3fc0/default: Packet: 140.82.121.4->78.46.88.247
Jool NAT64/8fee3fc0/default: ICMPv4 type:0 code:0 id:63596
Jool NAT64/8fee3fc0/default: Step 1: Determining the Incoming Tuple
Jool NAT64/8fee3fc0/default: Tuple: 140.82.121.4#63596 -> 78.46.88.247#63596 (ICMP)
Jool NAT64/8fee3fc0/default: Done step 1.
Jool NAT64/8fee3fc0/default: Step 2: Filtering and Updating
Jool NAT64/8fee3fc0/default: BIB entry: 2a01:4f8:120:2700:7864:c5ff:fead:4d8c#1547 - 78.46.88.247#63596 (ICMP)
Jool NAT64/8fee3fc0/default: Session entry: 2a01:4f8:120:2700:7864:c5ff:fead:4d8c#1547 - 64:ff9b::8c52:7904#1547 | 78.46.88.247#63596 - 140.82.121.4#63596 (ICMP)
Jool NAT64/8fee3fc0/default: Done: Step 2.
Jool NAT64/8fee3fc0/default: Step 3: Computing the Outgoing Tuple
Jool NAT64/8fee3fc0/default: Tuple: 64:ff9b::8c52:7904#1547 -> 2a01:4f8:120:2700:7864:c5ff:fead:4d8c#1547 (ICMP)
Jool NAT64/8fee3fc0/default: Done step 3.
Jool NAT64/8fee3fc0/default: Step 4: Translating the Packet
Jool NAT64/8fee3fc0/default: Routing: 64:ff9b::8c52:7904->2a01:4f8:120:2700:7864:c5ff:fead:4d8c
Jool NAT64/8fee3fc0/default: Packet routed via device 'vlan10'.
Jool NAT64/8fee3fc0/default: Done step 4.
Jool NAT64/8fee3fc0/default: Sending packet.
Jool NAT64/8fee3fc0/default: Success.
```

Debug output of a dropped package, error *Hairpinning loop*, with enabled firewall:

```
Jool NAT64/8fee3fc0/default: ================================================
Jool NAT64/8fee3fc0/default: Packet: 64:ff9b::8c52:7904->2a01:4f8:120:2700:7864:c5ff:fead:4d8c
Jool NAT64/8fee3fc0/default: ICMPv6 type:129 code:0 id:1551
Jool NAT64/8fee3fc0/default: Step 1: Determining the Incoming Tuple
Jool NAT64/8fee3fc0/default: Tuple: 64:ff9b::8c52:7904#1551 -> 2a01:4f8:120:2700:7864:c5ff:fead:4d8c#1551 (ICMP)
Jool NAT64/8fee3fc0/default: Done step 1.
Jool NAT64/8fee3fc0/default: Step 2: Filtering and Updating
Jool NAT64/8fee3fc0/default: Hairpinning loop. Dropping...
```

First things first, why was it dropped? Because Jool doesn't allow packages, where the source is in the same IPv6 range as the configured pool. Even though the problem is different, the behavior regarding the source IP was explained quite well in this Post on the Jool-list (Archive: [1], [2]).

Well, but why does packages with source IPs from the Jool Pool come up at all, when the Firewall on the NIC of a Virtual Machine is enabled? I learned that three additional interfaces will be created per NIC and PVE VM - Source (Archive: [1], [2]):

```
fwbr<VMID>i<network interface X> is the firewall bridge where the filtering happens.
fwpr<VMID>i<network interface X> is the bridge in device.
fwln<VMID>i<network interface X> is the bridge out device.
```

So the traffic flow changes, packages passing further bridges and devices and packages with NAT64 pool addresses as source start to show up, but are dropped by the Jool Kernel Module before they are further routed or filtered by the PVE Firewall.

I understand why it's happening but not how to solve it, my Google Fu wasn't strong enough and I just couldn't find any useful hints how to tackel this problem.

So I decided to replace Jool, even though it feels like the right solution for NAT64, with Tayga, which is using a own TAP device to route the traffic and perform the actual 6<>4 translation. The Setup was, to be fair, as simple as Jool, with the benefit that I didn't had to install devel and compiler packages at all.

```
apt install -y tayga

/etc/tayga.conf
```
```
tun-device nat64
ipv4-addr 192.168.255.1
ipv6-addr 2a01:4f8:120:225e::64
prefix 64:ff9b::/96
dynamic-pool 192.168.255.0/24
data-dir /var/spool/tayga
```
```

systemctl enable --now tayga
```

VMs with enabled Firewall could reach targets via NAT64 right away after starting the tayga service. Also all rules where applied as expected and for my scenario, the performance was more than sufficient.

At the end I have to admit that Tayga does a great job and is the better fit in a scenario like running on a Hypervisor which also provides Firewalling. This just seem to be something where it can get quite complicated, frustrating and maybe even more or less impossible to use Jool.

I'm still a little concerned about the fact that Tayga doesn't seem to be maintained anymore and might end as abandonware. But as long the package is available in the regular Debian Repository and no critical security issue comes up, I will stick with Tayga for NAT64.