strongSwan - Issue #2870

DNS resolution outside of tunnel if DNS server is in remote TS

24.12.2018 13:52 - Noel Kuntze

Status:	Feedback		
Priority:	Normal		
Assignee:			
Category:	configuration		
Affected version:	5.7.1	**Resolution:**	

Description

Following scenario occurs:

Roadwarrior style VPN with remote peer's address configured as a DNS name
The TS is %dynamic == 0.0.0.0/0
passthrough policy for the LAN exists
DNS server is configured to be 1.1.1.1 and DNS traffic of all applications should go over the configured DNS server.

To initially resolve the peer's name, a DNS request would be made, but it would already trigger an acquire from the kernel for a matching policy. We now have a chicken-egg problem. To bring the tunnel up, the DNS request has to be made successfully and to make the DNS request, the tunnel has to be up.
A solution to this would be a rather complex configuration using dnsmasq and specifying the server for the domain of the VPN server, as well as the source IP to be used, together with a passthrough policy for that DNS server. Assuming that the roadwarrior only has one physical interface, by default, the assigned virtual IPs would be bound to the same interface as the actual LAN IP of it. If the LAN IP is assigned via DHCP, it is unknown in advance and can not be statically configured in dnsmasq's configuration file. It is then required to use the interface to look it up. Considering that it would likely choose the virtual IP, because the routing table advises the usage of it, now we'd have to add a dummy ethernet device and make strongSwan bind the virtual IPs to that device. That's a mighty big solution for such a relatively minor problem.

A simple and worthwhile solution to the problem would be as follows:
I propose applying socket based passthrough policies for all DNS requests that charon makes, like it is already done for the IKE socket. That would effectively solve the problem, because as of the writing of this issue, you can't nest tunnels anyway, so resolving a nested tunnel's peer's DNS name through an existing tunnel wouldn't be a required feature.

Please let me know what you think.

History

#1 - 16.01.2019 17:52 - Tobias Brunner

- Category set to configuration

- Status changed from New to Feedback

> Roadwarrior style VPN with remote peer's address configured as a DNS name
> The TS is %dynamic == 0.0.0.0/0
> passthrough policy for the LAN exists
> DNS server is configured to be 1.1.1.1 and DNS traffic of all applications should go over the configured DNS server.
>
> To initially resolve the peer's name, a DNS request would be made, but it would already trigger an acquire from the kernel for a matching policy. We now have a chicken-egg problem. To bring the tunnel up, the DNS request has to be made successfully and to make the DNS request, the tunnel has to be up.

So also configure a passthrough policy for 1.1.1.1.

> A solution to this would be a rather complex configuration using dnsmasq and specifying the server for the domain of the VPN server, as well as the source IP to be used, together with a passthrough policy for that DNS server.

Why would you need dnsmasq if you also have a passthrough policy? To only resolve the VPN server?

> A simple and worthwhile solution to the problem would be as follows:
> I propose applying socket based passthrough policies for all DNS requests that charon makes, like it is already done for the IKE socket.

That is not easily possible because we use getaddrinfo(3) to resolve host names. So we don't have any control over sockets used there. While we could theoretically do our own DNS queries, I don't really like that approach very much (however, there is the hackish DNS proxy for Android - not sure if it is actually still required there - which does that, see source:src/frontends/android/app/src/main/jni/libandroidbridge/backend/android_dns_proxy.c).

#2 - 16.01.2019 22:53 - Noel Kuntze

Tobias Brunner wrote:

> Roadwarrior style VPN with remote peer's address configured as a DNS name
> The TS is %dynamic == 0.0.0.0/0
> passthrough policy for the LAN exists
> DNS server is configured to be 1.1.1.1 and DNS traffic of all applications should go over the configured DNS server.
>
> To initially resolve the peer's name, a DNS request would be made, but it would already trigger an acquire from the kernel for a matching policy. We now have a chicken-egg problem. To bring the tunnel up, the DNS request has to be made successfully and to make the DNS request, the tunnel has to be up.

So also configure a passthrough policy for 1.1.1.1.

That would counteract the desire to have a VPN to provide privacy. Using a passthrough policy would make all traffic to the DNS server visible to passive attackers. The goal of the VPN is to provide privacy, thus only the necessary information to make a connection must leak. Regarding DNS, that is the request and the response regarding the A/AAAA record of the VPN server's FQDN.

> A solution to this would be a rather complex configuration using dnsmasq and specifying the server for the domain of the VPN server, as well as the source IP to be used, together with a passthrough policy for that DNS server.

Why would you need dnsmasq if you also have a passthrough policy? To only resolve the VPN server?

I now see that this solution won't work either because the local TS installed for an unknown local virtual IP is 0.0.0.0/0 or ::/0, which matches any IP. So dnsmasq wouldn't help either. Thus it *has* to be implemented with a socket bypass OR a dynamic protocol/port based passthrough policy for the LAN's DNS server. Now the only problem left is how to deal with reconnecting to the VPN responder if the VPN goes down. In that case the policies are still in place, but the FQDN of the VPN responder still have to be resolved and the DNS server (that is not the LAN VPN server) that the responder pushed is still installed in resolv.conf. Now we have the chicken-egg problem again. Maybe specifying the source IP here again could work or caching the record.

> A simple and worthwhile solution to the problem would be as follows:
> I propose applying socket based passthrough policies for all DNS requests that charon makes, like it is already done for the IKE socket.

> That is not easily possible because we use getaddrinfo(3) to resolve host names. So we don't have any control over sockets used there. While we could theoretically do our own DNS queries, I don't really like that approach very much (however, there is the hackish DNS proxy for Android - not sure if it is actually still required there - which does that, see source:src/frontends/android/app/src/main/jni/libandroidbridge/backend/android_dns_proxy.c).

I see. The android DNS proxy wouldn't be directly usable, because it only proxies the requests but doesn't create new ones, which is what we want. It is most desired that any functionality is integrated within strongSwan without any outside dependencies.

#3 - 22.01.2019 11:06 - Tobias Brunner

> That would counteract the desire to have a VPN to provide privacy. Using a passthrough policy would make all traffic to the DNS server visible to passive attackers. The goal of the VPN is to provide privacy, thus only the necessary information to make a connection must leak. Regarding DNS, that is the request and the response regarding the A/AAAA record of the VPN server's FQDN.

I guess you could remove the passthrough policy in an updown event and re-add it once the last SA is gone. If privacy is the main concern, DNS over TLS (or HTTPS) provides that as well even without VPN.

> Thus it *has* to be implemented with a socket bypass OR a dynamic protocol/port based passthrough policy for the LAN's DNS server.

Yes, if you use a caching DNS server in your LAN (as many home routers provide by default and which in turn uses whatever upstream DNS server you want) while no VPN is established, you could use *bypass-lan* and let the VPN server assign the external DNS servers you want to use while the VPN is up.

> Now the only problem left is how to deal with reconnecting to the VPN responder if the VPN goes down. In that case the policies are still in place, but the FQDN of the VPN responder still have to be resolved and the DNS server (that is not the LAN VPN server) that the responder pushed is

still installed in resolv.conf

If the DNS resolution fails (e.g. because the DNS server is not reachable) the previous remote IP is just reused. Just like it is the case for reauthentications where hostnames are not actually resolved.