strongSwan - Feature #2252

libipsec match_packet() does not consider port

15.02.2017 15:11 - Phil Levin

Status:	Closed	**Start date:**	15.02.2017
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	libipsec		
Target version:	5.5.2		
Resolution:	Fixed		

Description

Hello,

I'm not sure if this is intentional, but the 'includes()' method in traffic_selector.c compares the address family and address range for matching on the selector, but does **not** compare protocol and port:

```
METHOD(traffic_selector_t, includes, bool,
               private_traffic_selector_t *this, host_t *host)
{
               chunk_t addr;
               int family = host->get_family(host);

               if ((family == AF_INET && this->type == TS_IPV4_ADDR_RANGE) ||
                                (family == AF_INET6 && this->type == TS_IPV6_ADDR_RANGE))
               {
                                addr = host->get_address(host);

                                return memcmp(this->from, addr.ptr, addr.len) <= 0 &&
                                                                                memcmp(this->to, addr.ptr, addr.len)
>= 0;
               }

               return FALSE;
}
```

Libipsec (ipsec_policy.c) attempts to match an outbound packet to a policy that compares protocol and source/destination address range, but **not** source/destination port range:

```
METHOD(ipsec_policy_t, match_packet, bool,
private_ipsec_policy_t *this, ip_packet_t *packet)
{
               uint8_t proto = packet->get_next_header(packet);
               host_t *src = packet->get_source(packet),
                                *dst = packet->get_destination(packet);

               return (!this->protocol || this->protocol == proto) &&
                                this->src_ts->includes(this->src_ts, src) &&
                                this->dst_ts->includes(this->dst_ts, dst);
}
```

As a result, if you have multiple SP/SAs that have the same /32 source/destination address but differ only in destination port, the above results in all packets being matched by only **one** of those SP/SAs.

Seems to me that 'includes()' is actually 'includes_address()' and that match_packet() should have additional tests for port range matching.

Thanks in advance.

Phil

Associated revisions

Revision d8e12fdb - 02.03.2017 08:27 - Tobias Brunner

libipsec: Match IPsec policies against ports of processed packets

Fixes #2252.

History

#1 - 15.02.2017 15:38 - Tobias Brunner

- Tracker changed from Issue to Feature

- Description updated

- Status changed from New to Feedback

Since *libipsec* is usually used for route-based VPNs (via TUN devices) I'd say it generally makes not much sense to match ports, because doing routing based on them is not that common. But I guess if you install multiple policies that only differ in ports and deliberately only send/route packets with these ports to libipsec that's a possible use case. I pushed a fix to the *2252-libipsec-ports* branch.

#2 - 15.02.2017 23:58 - Phil Levin

Hello Tobias,

Thank you for the rapid fix. Just for your information, due to specific requirements, I use libipsec with native packets punted to userspace via Netfilter rather than routed via TUN devices, thus the port selection requirement.

Regards.

Phil

#3 - 02.03.2017 08:30 - Tobias Brunner

- Status changed from Feedback to Closed

- Assignee changed from Phil Levin to Tobias Brunner

- Target version set to 5.5.2

- Resolution set to Fixed

> I use libipsec with native packets punted to userspace via Netfilter rather than routed via TUN devices

That sounds interesting. Do you have more information on this?

#4 - 03.03.2017 18:04 - Phil Levin

Hello Tobias,

It's straightforward actually. I configure Netfilter to act as a "proxy SP" (for lack of a better phrase), such that it feeds selected outbound traffic to a Netfilter Queue, which is monitored by a new Strongswan plugin. If the destination IP has not been seen before, the plugin dynamically creates initiator config (via the backend) and initiates the connection. This config is a simple host, point-to-point config. I monitor IKE and SA child events to determine if the "connection" is up yet. If not, subsequent packets are dropped (no queuing yet), if so, the packet is sent to libipsec. I happen to configure the Charon IKE socket for IPsec/UDP and tunnel the packets way, but that isn't required.

On the responder, libipsec receives the ESP packets, decrypts and hands them back to the new plugin (registered to libipsec). They are then reinjected via a raw socket and received locally by the target application.

The goal of this contraption is to provide on-demand IPsec tunnels for legacy (cleartext) applications without any modifications to those applications. It avoids modifying the routing plane, adding new interfaces (TUN) and from what I've read, may have better performance than the TUN mechanism. You also get more flexibility in programming packet selection using Netfilter than you do with standard IPsec packet selectors.

I need a user-space solution because I must have FIPS-compliant crypto, which I get with libipsec since it's linked with OpenSSL. I'm using an Ubuntu kernel, which isn't FIPS-compliant.

I hope that answers your question, and again thanks for the quick fix.

Regards.

/Phil

#5 - 06.03.2017 15:54 - Tobias Brunner

Nice, thanks for the detailed description.