Recent strongSwan Developments

IPsec Workshop 2025

Tobias Brunner

17.07.2025

strongSwan

Overview

- Narrowing with Trap Policies
- Per-CPU SAs
- AGGFRAG Mode
- Regular Expressions
- EAP-Identity Matching

Policies and Reqids on Linux

- Reqids are arbitrary 32-bit numbers tying SAs to policies
- Policy Match $\rightarrow$ Template (reqid, Proto, Mode, IPs, SPI) $\rightarrow$ SA Lookup
- Generated in strongSwan based on traffic selectors
- Trap policies only have a minimal template (primarily reqid)
- If kernel finds no SA:
  - **Acquire** sent to IKE daemon (reqid, selector from matching packet)
  - **Temporary state** created from template (further packets are **blocked**)
- Temporary state is removed when final SA is installed (match on reqid)

Narrowing

- First traffic selector sent is derived from matching packet
- Allows peer to select configs and/or narrow TS:

```
Initiator                          Responder

                                   net-1 {
net-net {                           local_ts   = 10.1.0.0/28
  local_ts  = 10.2.0.0/16          }
  remote_ts = 10.1.0.0/16          net-2 {
}                                   local_ts   = 10.1.0.16/28
                                   }
```

- Packet from 10.2.0.10 to 10.1.0.20 $\rightarrow$ net-2 $\rightarrow$ narrowed TS:

  $$10.2.0.0/16 == 10.1.0.16/28$$

Handling of reqid With Narrowing

- Maintain reqid from trap policy (strongSwan $<$ 6.0.2):
  - Temporary state is removed when SA is installed (reqid matches) but...
  - ...all traffic goes through established SA (trap policy $\rightarrow$ reqid $\rightarrow$ SA)
    $\rightarrow$ **Dropped by responder** based on narrowed policy
- Generate new reqid based on narrowed TS:
  - Temporary state is **not** removed (reqid doesn't match)
  - Timeout only after 165 seconds by default (same as allocated SPIs)
    $\rightarrow$ **No acquires** for other traffic (e.g. to 10.1.0.10) during that time

Changes in 6.0.2

- Use **sequence number** sent by kernel with acquire
- **Change reqid** and pass along sequence number when installing SA
- Temporary state gets removed even if reqid is different
- Acquire for other traffic gets triggered after SA installation

- Other necessary change: Use narrowed and not configured traffic selectors during rekeying/recreation/reauthentication (no packet TS available to perform narrowing again)

- Possible issue: Responder ignores packet TS and always narrows to the same TS $\rightarrow$ Loop and duplicate SAs

Per-CPU SAs

- Support for per-CPU SAs (RFC 9611) added with 6.0.2
- Supported by the Linux kernel since 6.13
- Enabled with `per_cpu_sas` setting
- Requires trap policies: `start_action = trap`

- **Just enabling the feature is not enough for best performance!**
- Requires RSS (or e.g. *XDP cpumap redirect*) for inbound traffic
- Potentially requires process pinning for outbound traffic

Per-CPU SAs Behavior (Outbound)

1. IKE daemon installs trap policies with `XFRM_POLICY_CPU_ACQUIRE`
2. For first match, kernel triggers acquire without CPU ID
   - Matching packet is lost
3. IKE daemon establishes fallback SA that has no CPU assigned
4. Further acquires contain CPU ID
   - Packets are sent via fallback SA
5. IKE daemon establishes CPU-specific SAs (CPU ID on installed SAs)
6. Packets take SA assigned to the originating CPU

Per-CPU SAs Behavior (Inbound)

- RSS uses hash of SPI to distribute packets to NIC queues/CPUs

```
ethtool -N <nic> rx-flow-hash esp4
```

- Alternatively, UDP encapsulation can be used:
  - `encap = yes` – forces UDP-encap without NAT
  - `per_cpu_sas = `**`encap`** – random source ports for outbound per-CPU SAs

```
ethtool -N <nic> rx-flow-hash udp4 sdf
```

- NAT-mapping events are suppressed for inbound per-CPU SAs

AGGFRAG Mode

- AGGFRAG mode is part of IP-TFS (RFC 9347), supported since 6.0.2
- Like tunnel mode, but...
  - ...can aggregate small IP packets into single ESP packets
  - ...can fragment large IP packets into multiple ESP packets
- Supported by the Linux kernel since 6.14
  - **Core of IP-TFS** (fixed-size ESP packets sent at a constant rate) **is not supported yet!**
- Enabled with `<child>.mode = iptfs`
- Several global settings under `charon.iptfs`

Remote Identity Matching

- Matching remote IKE identities (IDr) supports simple wildcards, e.g.
  - `*@strongswan.org`
  - `*.strongswan.org`
  - `C=CH, O=strongSwan, CN=*`
- Simple suffix or complete RDN match, no partial matching
- **Not supported:**
  - `vpn*.strongswan.org`
  - `admin@*.strongswan.org`
  - `C=CH, O=strongSwan, CN=*.strongswan.org`

POSIX Regular Expressions

- POSIX regular expressions supported since 6.0.2
- Configured as anchored pattern with type prefix:
  ```
  <type>:^<regex>$
  ```
- Matched case insensitive against string representation of other identities
- Type must match! (e.g. `fqdn:` only matches `ID_FQDN` identities)

Regular Expression Examples

- **`email:^(moon|sun)@strongswan\.org$`**
  - matches `moon@strongswan.org` or `sun@strongswan.org`
- **`fqdn:^vpn[0-9]+\.strongswan\.org$`**
  - matches `vpn1.strongswan.org` or `vpn42.strongswan.org` but not `vpn.strongswan.org`
- **`"asn1dn:^.*CN=.+\\.strongswan\\.org$"`**
  - matches e.g. `"C=CH, O=strongSwan, CN=moon.strongswan.org"`
  - but due to `.+` also `"C=CH, O=strongSwan, CN=sun, E=sun@internal.strongswan.org"`
- **Be as specific as possible and test them!**

EAP-Identity Matching

- Previously, EAP-Identity exchange only via `remote.eap_id=%any`
- Other values were just statically set and locally used during EAP
- Matching EAP-Identities required a workaround via dummy config

```
conn-eap-init : connections.conn-eap-sun {
  remote {
    groups = group-auth-fake
    eap_id = %any
  }
}
conn-eap-sun {
  ...
  remote {
    eap_id = sun@strongswan.org
    auth = eap-md5
  }
}
...
```

- Awkward and the late switch meant no selection of different EAP methods

EAP-Identity Matching

- Since 6.0.2, **any** value in `remote.eap_id` triggers an EAP-Identity exchange
- If it doesn't match returned value, an alternative config is searched
- Wildcards and regular expressions are supported
- However, no "best" match – configs are evaluated in order
- More specific configs should be defined before potential fallbacks

```
rw-eap-tls {
  ...
  remote {
    eap_id = "C=CH, O=strongSwan Project, OU=Accounting, CN=*"
    auth = eap-tls
  }
}
rw-eap-md5 : connections.rw-eap-tls {
  remote {
    eap_id = %any
    auth = eap-md5
  }
}
```

References I

- strongSwan.
  `https://www.strongswan.org`.
- strongSwan Documentation.
  `https://docs.strongswan.org`.