IPsec in Android
IMS and IWLAN

- IMS supports TLS and IPsec; IR.92 (VoLTE standard) specifies IPsec. (also IR.51 and IR.94)
- IWLAN uses IPsec for the data plane and IKE (with 3GPP extensions) for configuration.
  - TS 23.234 (requirements), TS 24.234 (stage 3 - protocol), TS 33.234 (security)
  - The prefix assigned to the phone is carried between LTE/UMTS and IWLAN within the tunnel.
- IMS requires Transport mode inside Tunnel mode
IMS and IWLAN
Design Goals/Needs

- Initial Focus on the data plane:
  - IPsec transport mode security on a per-socket basis
  - IPsec tunnel mode security to create encrypted **Network**s (e.g., IWLAN)
  - Combined: encrypted socket over an encrypted **Network** (e.g., SIP over IWLAN)
- UDP encapsulation for IPv4
  - Provide IKE/encap socket to userspace with guarantees of safety
- For IMS (transport), keys are manually generated from EAP-AKA.
- For IWLAN (tunnel), keying is performed by IKEv2 with 3GPP extensions.
Goals and Constraints

- Multiple entities on-device concurrently configuring IPsec, such as VPNs and IMS.
  - They have no coordination mechanism
- Global policies are difficult in a multi-app environment
  - Per-socket policies are safe for individual socket owners
  - No acquires sent to userspace
- No dropped packets allowed during keying due to latency limits (esp. for transport mode)
Approach

- SAs are provided by management object - `IpSecTransform`
- VTIs (in the future, XFRMIs) are created along with two pairs of policies (v4/6, in/out) and collected in a management object - `IpSecTunnelInterface`
- Association of an SA with a socket or tunnel is performed by calling an `apply()` method
  - `applyXYZ()` does re-key as follows:
    - SPI in the template selects the SA in the outbound direction by either setting a new socket policy or calling `UPDPOLICY`
    - Input direction “just works”™
- `XFRMA_OUTPUT_MARK` used to bind the tunnel to an underlying interface
Transport Mode Socket Establishment Contd.

- **IPSec Socket Establishment (part 2)**
- **Application Client**
- **Android System**
- **netd**
- **kernel**
- **Application Server**

**Steps in the Process**:
1. **IpSecTransform.Builder**
   - .setUdpEncapsulation(UdpEncapsulationSocket)
   - .setEncryption(...) .setAuthentication(...) .createTransportModeTransform(addr, SPI)
2. **IpSecTransform(...)**
   - IpSecManager
   - .applyTransportModeTransform(socket=fooSocket, dir=in/out, IpSecTransform)
3. **updateSecurityAssociation**
   - (spi, Encryption, Authentication)
4. **applyTransportModeTransform**
   - (fooSocket, dir=in/out, IpSecTransform)
5. **XfrmPolicy(IpSecTransform)**
   - setsockopt
   - (fooSocket, IP(6)_XFRM_POLICY, XfrmPolicy)

**Transform Creation and Application**
- Occurs once in each direction.

**Encrypted Traffic over User Socket**
Tunnel Mode Interface Establishment

IPsec Tunnel Creation

Application Client

Android System

NetD

Kernel

Application Server

NetworkRequest

createIpSecTunnelInterface (saddr, daddr, underlying Network)

createIpSecTunnelInterface (...)

createSecurityPolicy {dir=OUT, mode=TUNNEL, template...}

createSecurityPolicy {dir=IN, mode=TUNNEL, template...}

IpSecTunnelInterface(ifname, ...)

RTM_NEWLINK (type=vt1, addresses, marks)

NEWPOLICY {dir=OUT, mode=TUNNEL, template...}

NEWPOLICY {dir=IN, mode=TUNNEL, template...}

Open Encap Sockets, Allocate SPIs, Perform Negotiations, etc
Tunnel Mode Interface Establishment Contd.

IPSec Tunnel Creation

Android System

NetD

Kernel

Application Server

Application Client

IpSecTransform.Builder
  .setUdpEncapsulation(UdpEncapsulationSocket)
  .setEncryption(...)
  .setAuthentication(...)
  .createTunnelModeTransform
  (addr, GTP, underlyingNetwork)

IpSecTransform(...)

applyTunnelModeTransform
  (IpSecTunnelInterface, dir=in/out, IpSecTransform)

A Transform is built and applied in each direction.

NetworkAgent(ifname)

Update the SAs using negotiated algorithms and keys.

updateSecurityAssociation
  (spi, Encryption, Authentication)

(xfrm) UPDSA(...)

applyTunnelModeTransform
  (...)

(xfrm) UPDSA(mark=WTL_I/O KEY)

If direction out, update policy tmpl to match SPI of SA.

(xfrm) UDPolicy(spi)
IPsec data usage and firewalls

- Major headache

- Solution:
  - Dependent on `xt_policy` module
  - `uidBillingDone` bit in packet mark (fwmark)
  - Exempt ESP, and count inner packets only
  - For firewalling, allow ESP packets, and packets that have no socket through

- Bypassing powersaving firewalls
  - ESP traffic blanket-exempted (regular apps can’t send/receive it)
  - Forwarding with UDP-encap-ESP allowed
### IPsec data usage and firewalling - cases

<table>
<thead>
<tr>
<th>Non-IPsec UDFs</th>
<th>INBOUND</th>
<th>OUTBOUND</th>
<th>FORWARD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COUNT</td>
<td>DILL</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Transport Mode**

<table>
<thead>
<tr>
<th>INBOUND</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>BLANKET IGNORE</td>
</tr>
<tr>
<td>COUNT</td>
<td>BLANKET ALLOW</td>
</tr>
<tr>
<td>COUNT</td>
<td>ORIGINAL UDP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTBOUND</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>BLANKET IGNORE</td>
</tr>
<tr>
<td>COUNT</td>
<td>BLANKET ALLOW</td>
</tr>
<tr>
<td>COUNT</td>
<td>ORIGINAL UDP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FORWARD</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>BLANKET IGNORE</td>
</tr>
<tr>
<td>COUNT</td>
<td>BLANKET ALLOW</td>
</tr>
<tr>
<td>COUNT</td>
<td>ORIGINAL UDP</td>
</tr>
</tbody>
</table>

**Tunnel Mode**

<table>
<thead>
<tr>
<th>INBOUND</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>BLANKET IGNORE</td>
</tr>
<tr>
<td>COUNT</td>
<td>BLANKET ALLOW</td>
</tr>
<tr>
<td>COUNT</td>
<td>ORIGINAL UDP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTBOUND</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>BLANKET IGNORE</td>
</tr>
<tr>
<td>COUNT</td>
<td>BLANKET ALLOW</td>
</tr>
<tr>
<td>COUNT</td>
<td>ORIGINAL UDP</td>
</tr>
</tbody>
</table>

**FORWARD**

- Cases, for reference:
IPsec data usage and firewalling - cases

- Cases, for reference:

<table>
<thead>
<tr>
<th>Inbound</th>
<th>Tunnel</th>
<th>Transport</th>
<th>Plaintext</th>
<th>Doze</th>
<th>Outbound</th>
<th>Tunnel</th>
<th>Transport</th>
<th>Plaintext</th>
<th>Doze</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>raw+PRE</td>
<td>filter=INPUT</td>
<td>raw+PRE</td>
<td>filter=INPUT</td>
<td>Plaintext</td>
<td>Tunnel</td>
<td>Filter=OUTPUT</td>
<td>mangle=POST</td>
<td>Tunnel</td>
</tr>
<tr>
<td>IKE-Tunnel</td>
<td>BILL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TUNNEL UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IKE-Transport</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TRANSPORT UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[ESP]</td>
<td>BLANKET IGN</td>
<td>BBC</td>
<td>-</td>
<td>-</td>
<td>BLANKET IGN</td>
<td>IGN based on VTI (&lt; vI*)</td>
<td>BLANKET IGN</td>
<td>IGN based on VTI (&lt; vI*)</td>
<td>BLANKET IGN</td>
</tr>
<tr>
<td>[UDP-encap] [UDP-encap]</td>
<td>BILL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TUNNEL UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[UDP-encap] [ESP]</td>
<td>BILL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TRANSPORT UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[ESP] [UDP-encap]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>ORIGINAL UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OUTBOUND</td>
<td>IKE-Tunnel</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IKE-Transport</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>TRANSPORT UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[ESP]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>ORIGINAL UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[UDP-encap] [UDP-encap]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>ORIGINAL UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[UDP-encap] [ESP]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>ORIGINAL UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>[ESP] [UDP-encap]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>ORIGINAL UID</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Questions

- Policy check isn’t performed for SAs in transport mode with socket policy on input.
  - Minor security issue?
  - Removing socket policies has no effect on inbound direction.
- Will XFRMI match against ‘0’?
  - Can we update the XFRMI on SAs while they are in ACTIVE state? Same question for the mark?
- Can XFRMI support multiple policies?