# LINUX IPSEC OFFLOAD W/ VIRTUALIZATION

Don Skidmore, ND
Josh  Hay, ND
Anjali Singhai, ND
Oct 2017

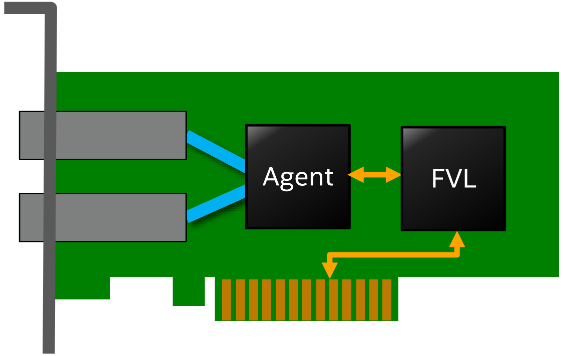
Network Division

# Agenda

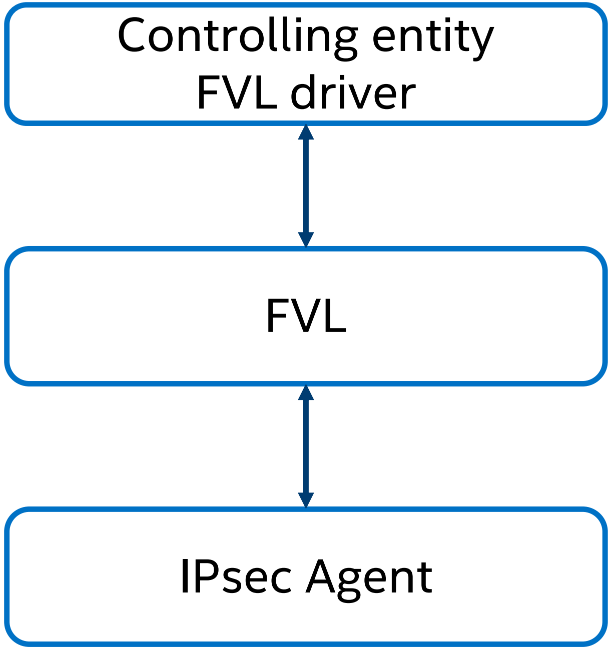
- Background of the environment we are operating in.
- Cover some of challenges we ran into and how we addressed them.
- How we are planning on supporting virtualization (SR-IOV) with IPsec offload.

# Connectivity with our IPsec Agent

- No separate control plane for Configuration and Metadata
- All control data has to go threw the MAC to get to the Agent
- Use one L2 tag to denote Configuration packets
- Different L2 tag to insert Metadata into a packet



# Configuration Packets



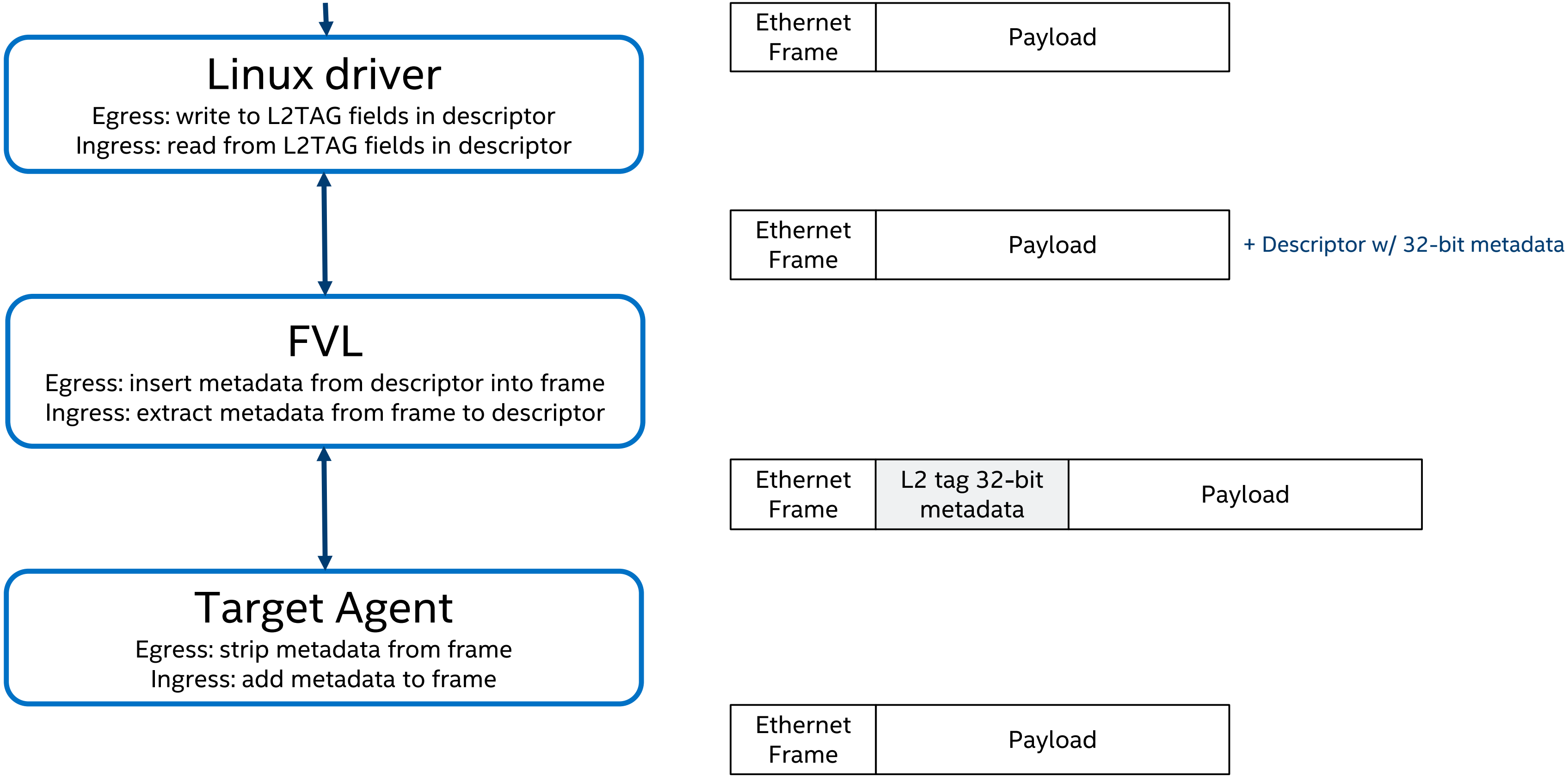
| Ethernet Frame | Special L2 Config tag | Agent specific Config Info |
|---|---|---|

| Ethernet Frame | Special L2 Config tag | Agent specific Config Info |
|---|---|---|

# Example Configuration packets

- Add SA
- Remove SA
- Remove all SA’s on a port

# Adding Metadata to a packet



## Linux driver

Egress: write to L2TAG fields in descriptor
Ingress: read from L2TAG fields in descriptor

## FVL

Egress: insert metadata from descriptor into frame
Ingress: extract metadata from frame to descriptor

## Target Agent

Egress: strip metadata from frame
Ingress: add metadata to frame

| Ethernet Frame | Payload |
|---|---|

| Ethernet Frame | Payload |
|---|---|

+ Descriptor w/ 32-bit metadata

| Ethernet Frame | L2 tag 32-bit metadata | Payload |
|---|---|---|

| Ethernet Frame | Payload |
|---|---|

# Some Metadata fields

- Offload packet bit (Tx)
- Next header field (Tx)
- Possible offsets to fields in the packet for the agent (Tx)
- Error return - i.e. did the decrypt work and if not why (Rx)
- Index to SA used to decrypt (Rx)

# Where we are at now

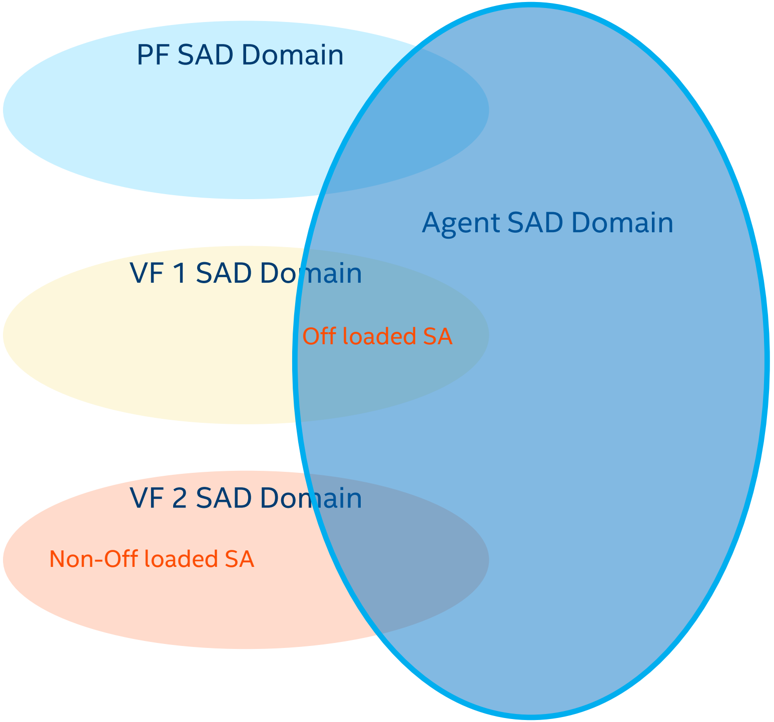
- Implemented all protocols above
- We verified they play nicely with our existing Agent
- Virtualization design is currently ahead of our Agent’s functionality

# Virtualization Challenges

- Multiple SAD domains
- Abandoned SA clean up
- Malicious VMs
- East-West VM traffic

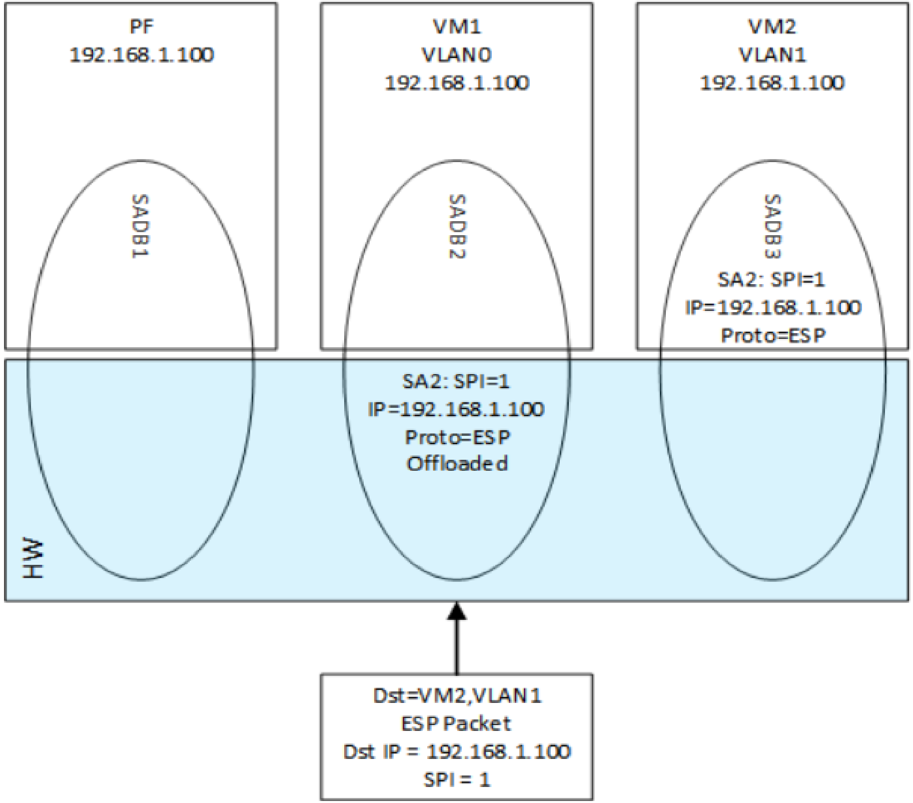
# Multiple SA domains

- Agent SAD unaware of all active SA's
- Any one PF/VF SAD unaware of all SA's being offloaded



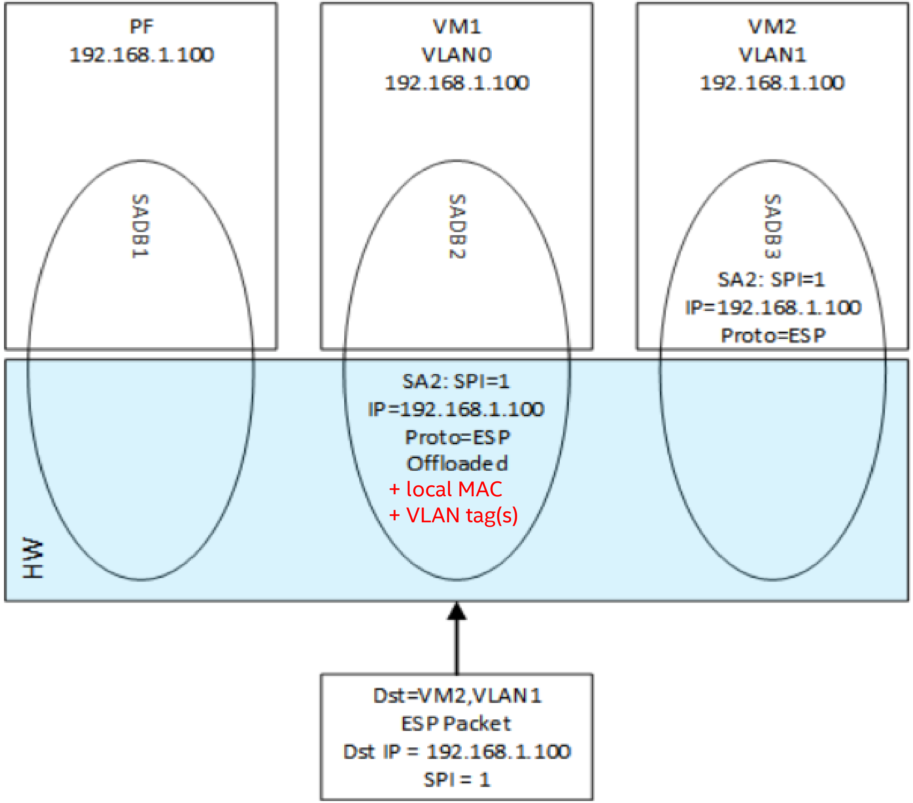
# Where this can be a problem

- Following fields make SA unique in a SAD domain
  - Destination IP address
  - IPsec Protocol
  - SPI
- So using only these fields with multiple SAD domains false matches could occur
- Could lead to offload agent processing packets it shouldn't



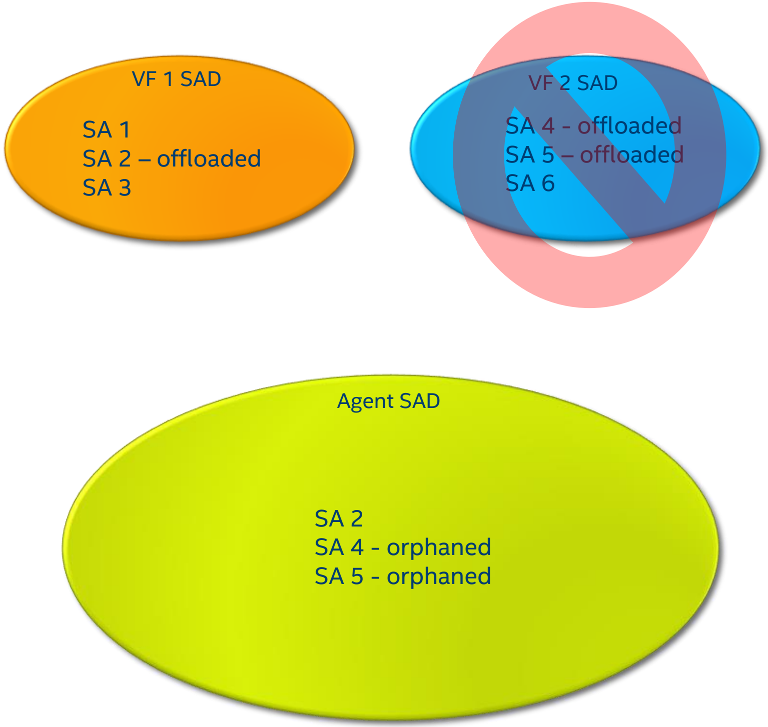
# Extending the key as a solution

- Add to the agent SA's additional fields
  - Local MAC address
  - VLAN (possible multiple for Q-in-Q)
- Any SA lookup would also verify these fields as well.



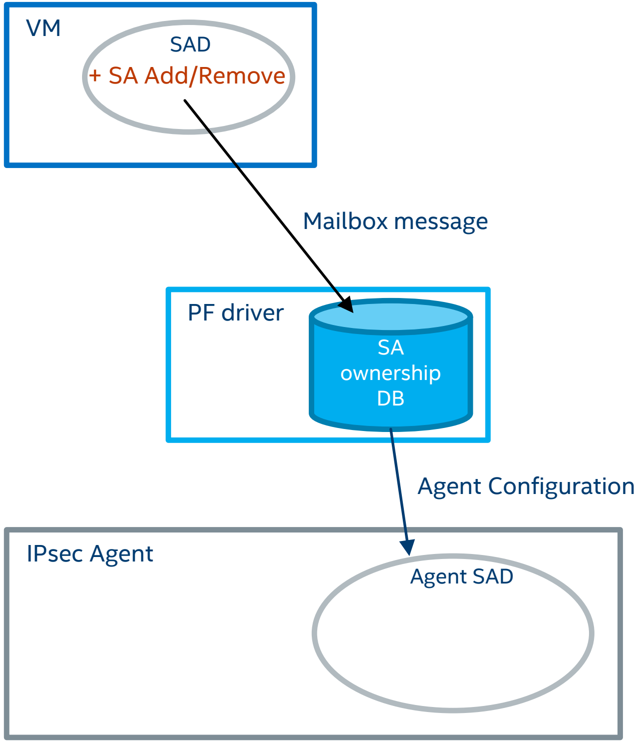
# Abandoned SAs

- What if a VF is removed before it can clear it's SA's from the Agent
  - virsh destroy vm
  - Panics
- Same is true for a PF as well
- Need a method for clearing out these SAD entries in the Agent.
- Our IPsec Agent is only capable of removing induvial SAs or all owned by a given port.



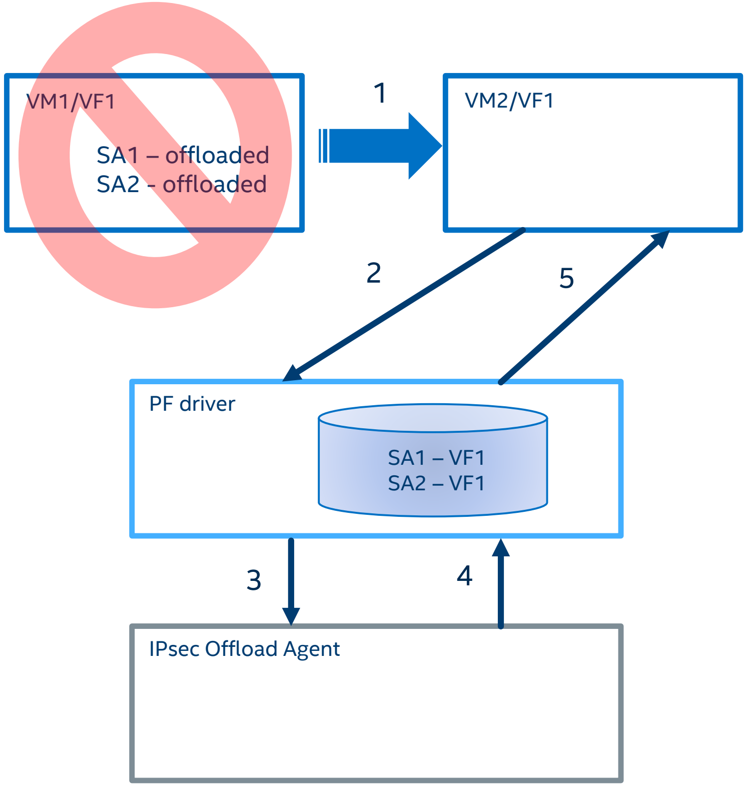
# Proxy all SA add/removals threw PF

- All SA creations and removals proxy threw PF driver.
- PF drive maintains DB of SAs and what SAD domain they are from
- Means the PF has the information needed to clean out obsolete SAs
- Could encrypt keys if concerns about PF being able to view in the clear



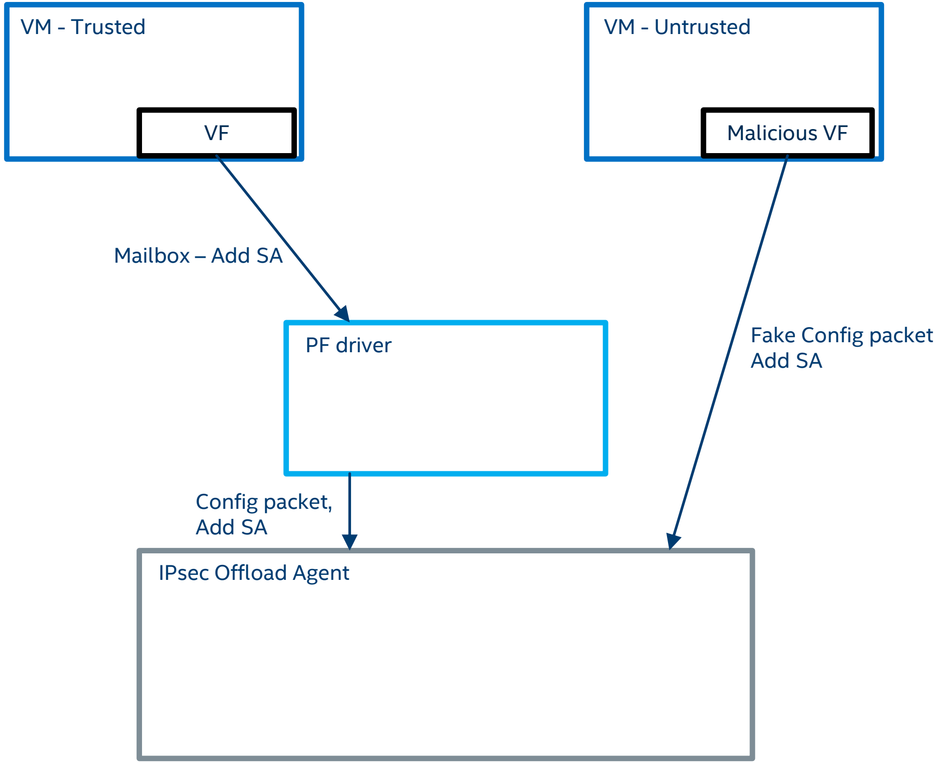
# Example of this in action

1. VM1 is destroyed without releasing it's SAs.
2. Later that VF is brought up in a new VM. It request resources from the PF.
3. Thanks to it's SA to SAD domain mapping the PF knows all the SA that were active in that domain.  It sends multiple remove SA messages to the agent.
4. After the agent removes the SA from it's SAD it replies to the PF driver which can then remove it from it's SAD domain mapping DB.
5. The PF can now reply to the VF request for resources.



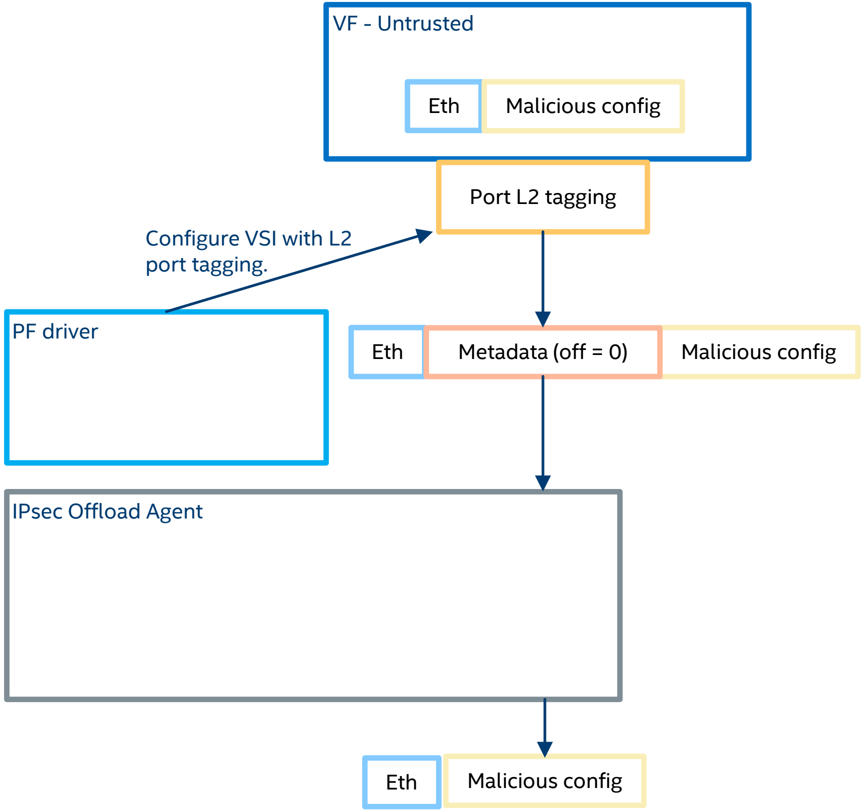
# Malicious VFs

- A concern for us since we don’t have a separate control plane
- A malicious VF driver could craft it’s own configuration packets and add it’s own L2 metadata tag.
- With SR-IOV such traffic by-passes the PF driver going directly to the MAC.



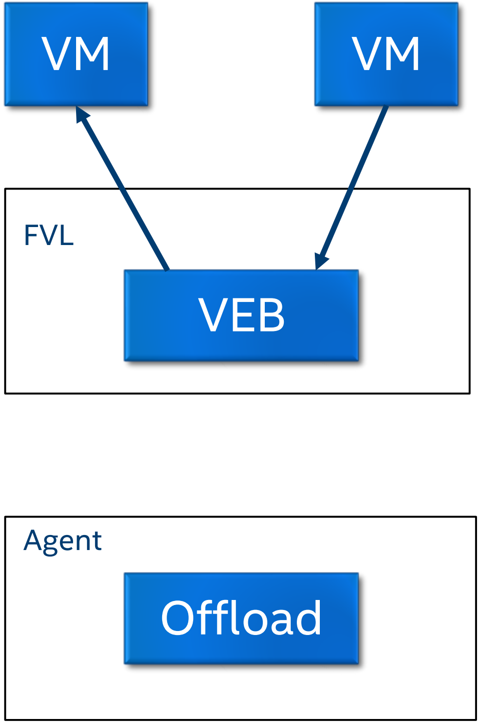
# Identifying untrusted traffic in the agent

- PF sets up Port L2 tagging on untrusted VF
- VF unaware and unable to modify this setting
- Tag add metadata with offload bit cleared
- All the agent will do with this packet is:
  - Strip the metadata header
  - Bypass all IPsec offload.



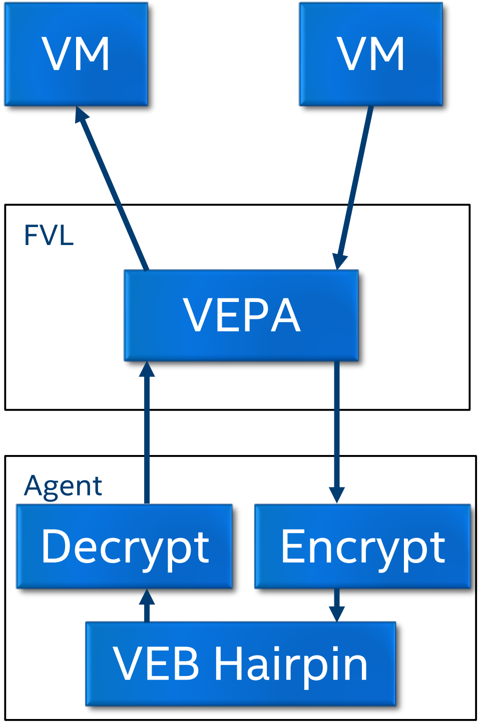
# East-West VM traffic

- The VM doesn’t know if the target for its traffic is local to the same system.
- If it is local packets routed via the VEB will NOT go threw the Agent (i.e. no offload processing)
- Could be solved by using VEPA but not all ToR switch support it.



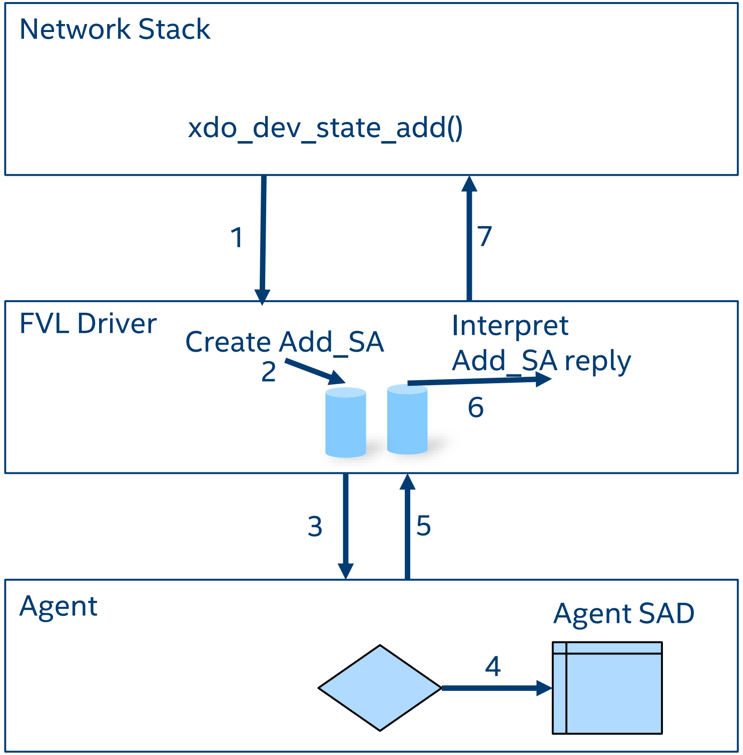
# Hairpin Agent solution

- Place FVL in VEPA mode
- Instead of requiring the first switch to the hairpin have an agent in the Agent do it
- All Agent offloads act as normal
- All routing is still done in FVL the Hairpin just turns around local traffic
- The driver will need to tell the Agent what traffic is local.



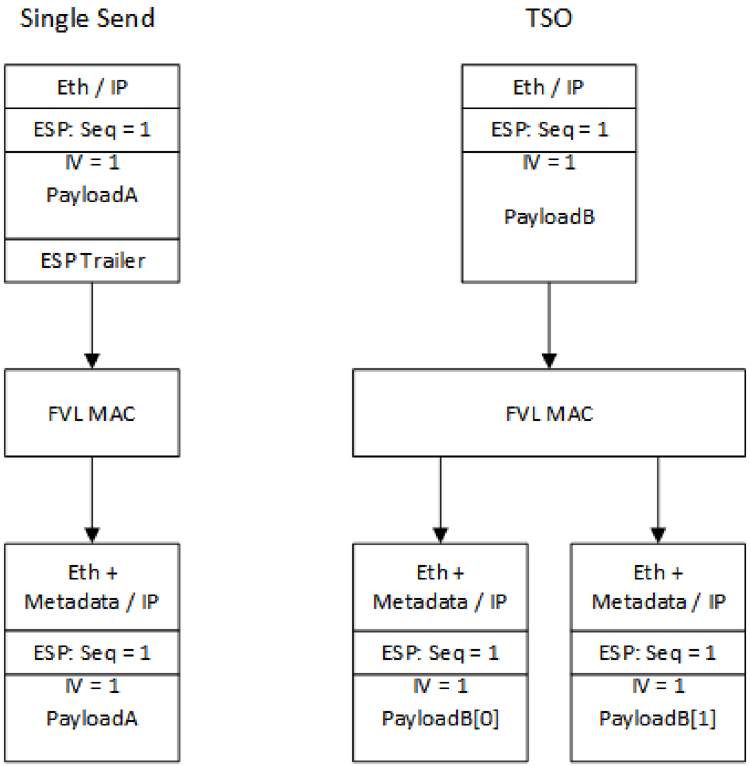
intel®

# IPsec Control Packet flow



1. Stack calls xdo_dev_state_add to add an SA.
2. Driver creates an Add_SA control packet
3. The Add_SA packet is sent to the Agent
4. The Agent adds the SA to its SAD if possible.
5. The Agent sends a Add_SA reply to the driver
6. The driver receives the reply and interprets it.
7. The return value to xdo_dev_state_add reflects what we received in the Add_SA reply

# Simplified Packet Format



# TSO Sequence Number Solution

- Problem: The header is replicated exactly for each segment, but parts of it need to be changed per segment
- Solution: Update RTL to track Sequence Number/IV to the SA entry in the SAD and replace these fields in the packet segments on the fly
  - Also reduces metadata consumption

```
If (IVDB[SA].IV <= packet.IV)
    IVDB[SA].IV = packet.IV + 1
Else if (packet.IV < IVDB[SA].IV)
    packet.IV = IVDB[SA].IV
    IVDB[SA].IV++
```