Using FOSS to Protect Embedded Systems Data in Transit

Bassem Nomany
bassem.nomany@exodo.engineering

About me

- Engineer and product builder.
- Worked extensively with open source and commercial embedded platforms.
- Spent 13 years working on projects:
  Automotive, E-mobility and smart connected devices.

The Shift in embedded systems

- High complexity
- Modern embedded products = distributed systems
  - MCU/SoC + cloud backends + Mobile companion App + Web portal
- Devices now are deeply connected:
  - Identity (user accounts, credentials)
  - Remote control & telemetry
  - Mobile + API
- Increased attack surface

Software-defined products

- Customers can expect now to always receive updates post-sale, quarter or yearly feature updates.

- We moved from: Build then sell.
- To new reality:
  “Ship early, improve later”

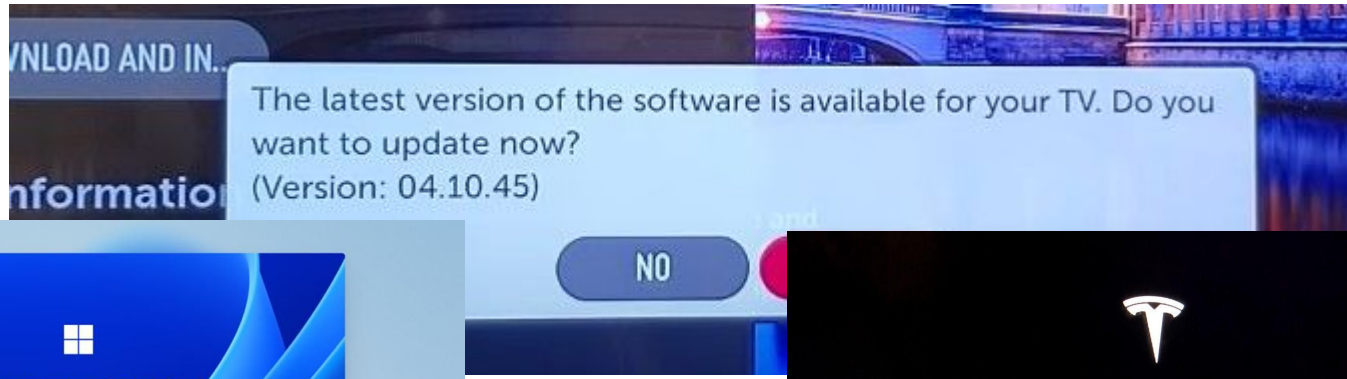
- And when you shift like this … security shifts too.

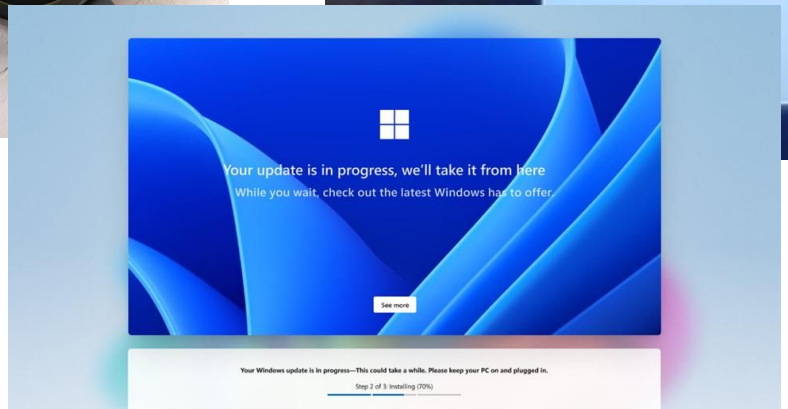
Why speed is important

- Missing a launch window can:
  - Kill a product
  - Kill a company
- OEM priorities:
  - Ship on time
  - Stay within budget
  - Reach market fast

Who’s here have seen these before?

- Out of box experience - Day 1 system Update
- Almost every connected device you buy today has a day-1 software update.





Projects reality

Engineering & Security

- Full TARA coverage
- Robust, secure system

Business Reality

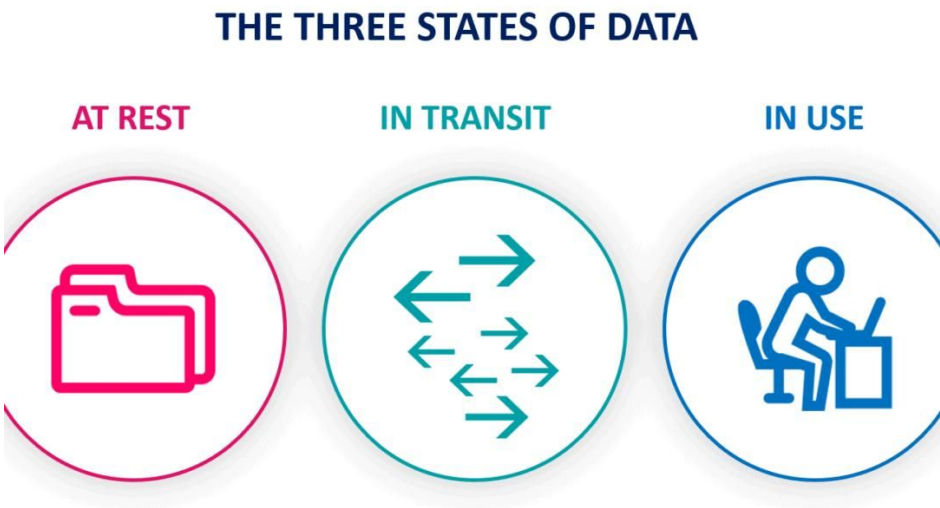
- Feature pressure
- Deadline
- Budget

Security becomes a marathon, not a sprint.

- Security doesn’t end at release anymore.
- Reducing risks to acceptable level before shipping.
- Foundations to get right before launching a product.

What should we prioritize?

A series talk



- Protecting Data in transit
- Protecting Data at rest
- Key Management and Key Infrastructure

- Today's' talk will focus on **Data in transit**.

Data in transit Cybersecurity

Journey into designing network security for embedded products

- Discovery
- Attack surface thinking
- Right mechanism for the right data
- Example use case

The target today:

Think about your data, endpoints, and exposed surfaces in your projects.

Discovery - TARA

- What’s the data coming in and coming out from the device? How sensitive is the data?
- What’s the endpoints? What are their limits (CPU, Network, Memory)?
- What are the endpoint APIs? What do they Read, Write or Control?
- What’s the device mission critical resources?
- What’s the damage scenario? What’s the impact and severity? Is the attack feasible?
- What’s the mitigation per identified item?

Secure network architecture design principles

Good Secure Network Design =

Control of communication +

Visibility of behavior +

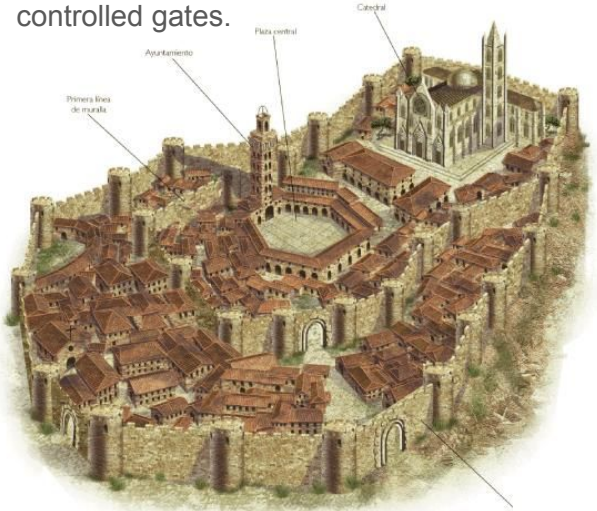
Containment of failure

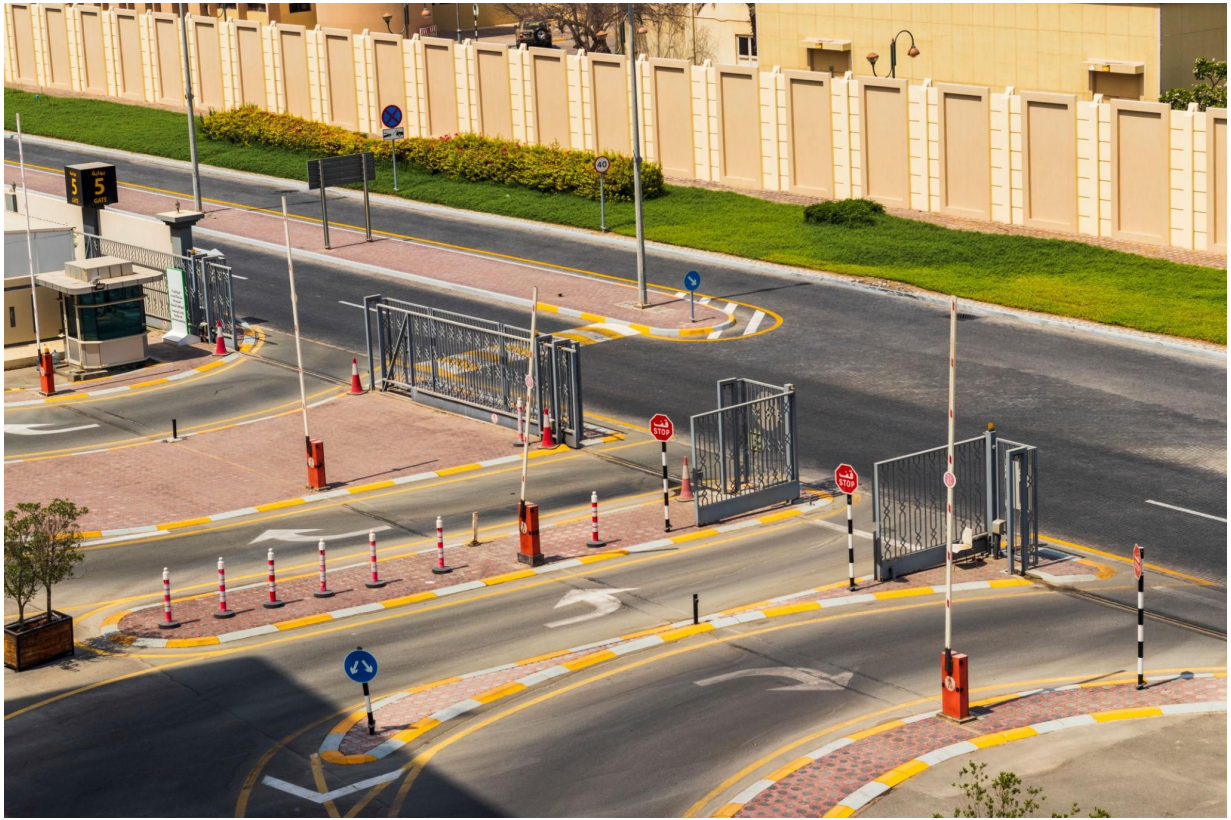
Network Segmentation

Think of the product network as a piece of land with valuable assets.

We don’t protect it with one big wall — we divide it into secured zones.

Each zone is isolated, and movement between zones is only possible through controlled gates.

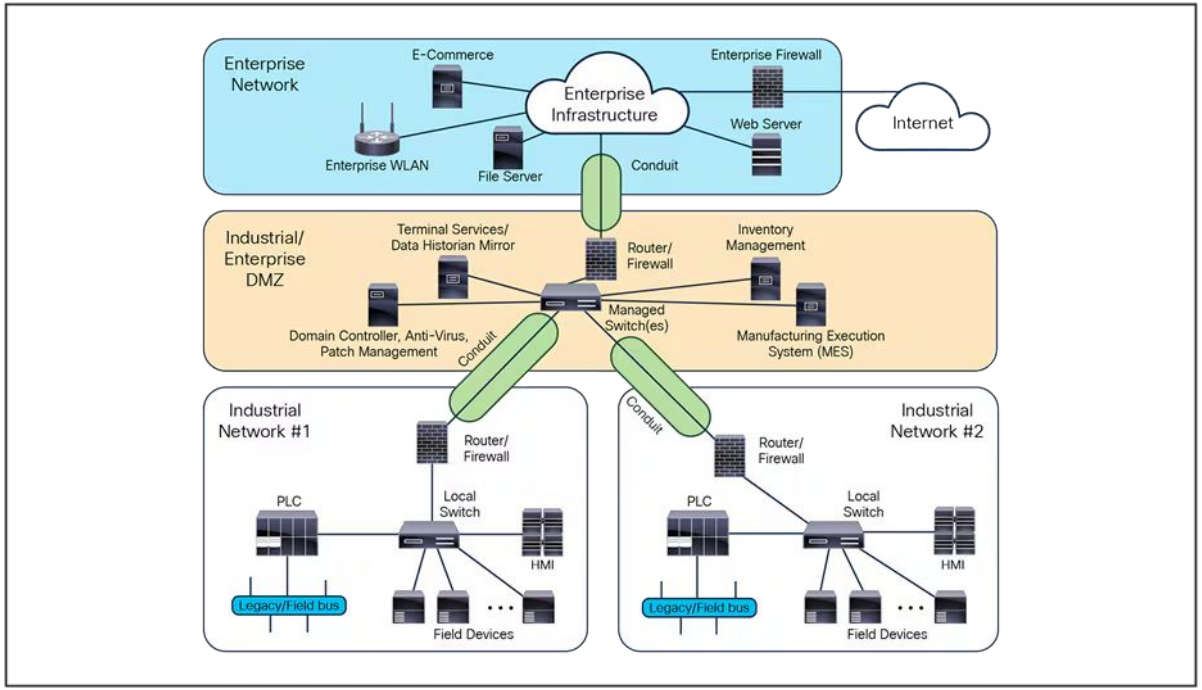




Network Segmentation - Zone Isolation

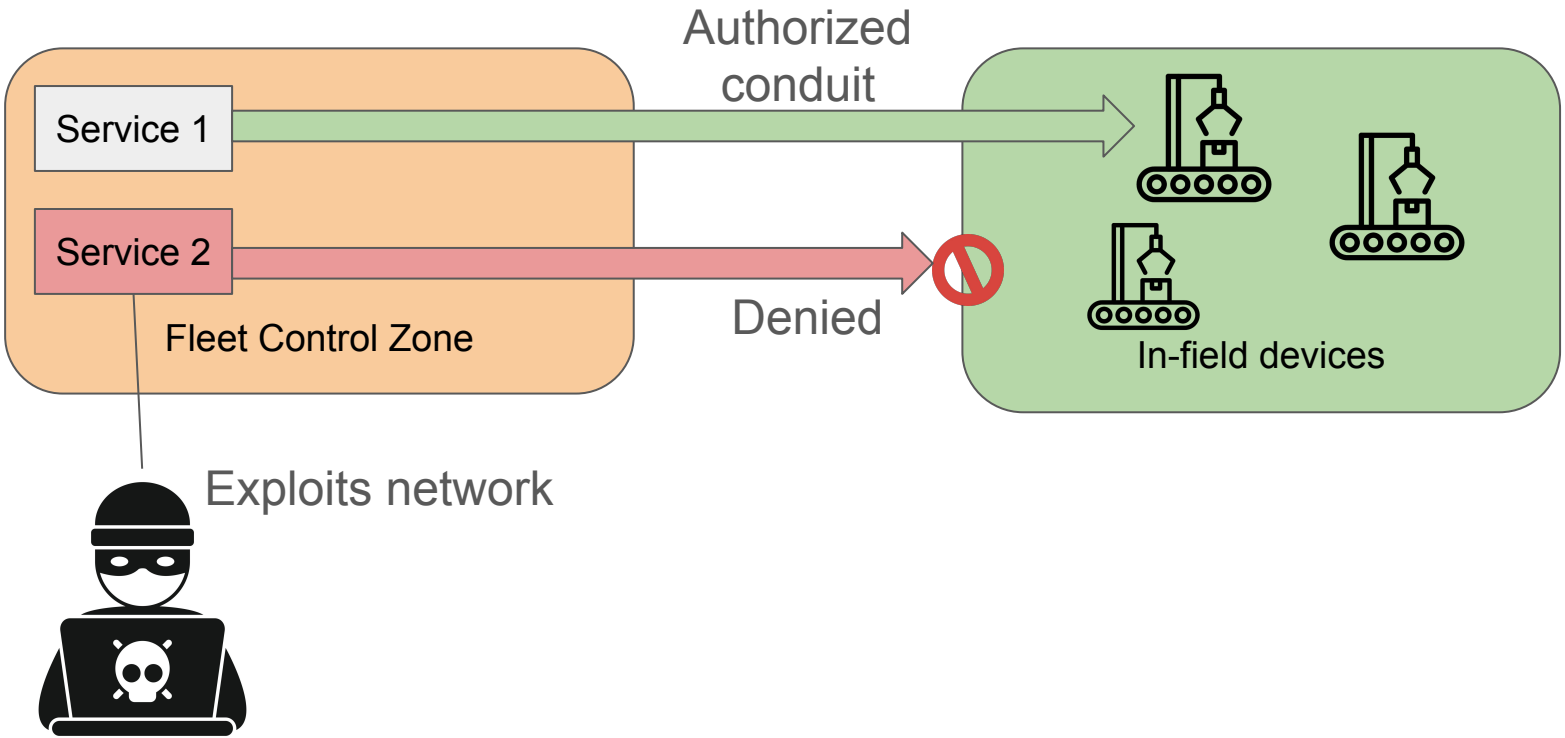
→ Example of industrial network model of isolated zones.

Traffic flowing to cross boundary from one zone to another
Has to pass through security conduit.



Ref: https://www.cisco.com/c/en/us/td/docs/solutions/Verticals/IoT_Security_Lab/IEC62443_WP.html

Network Segmentation - Zones and conduits



Protecting data flowing in/out a conduit

- **Authentication**: Verify identity of a peer or a user or endpoint. Who?
- **Authorization**: What's the priviliages? What can you do?
- **Confidentiality**: Private data, Private network packets. Only intended user with correct privilege of this data is allowed to view it and access it.
- **Integrity**: Protect data from tampering when it's in transit.
- **Availability**: System is always available .. resources are always available. Backup the data encrypted.
- **Freshness**: Keep crypto keys/Certs fresh and rotate them.

How to design segmentation - Part 1/2

1. Identify what you are protecting

Group assets in Zone by:

- Business function
- Criticality
- Exposure
- Who manages them
- What happens if they are compromised (limit blast radius to next access)

Example zone splits:

- Internet/external-facing
- Diagnostic/control traffic
- Factory / manufacturing
- Engineering / debug / test
- User endpoints
- Server/backend
- Software updates path
- Product/control plane
- Safety-critical domains
- Third-party/vendor access

How to design segmentation well - Part 2/2

2. Draw allowed flows before building anything

Make a matrix:

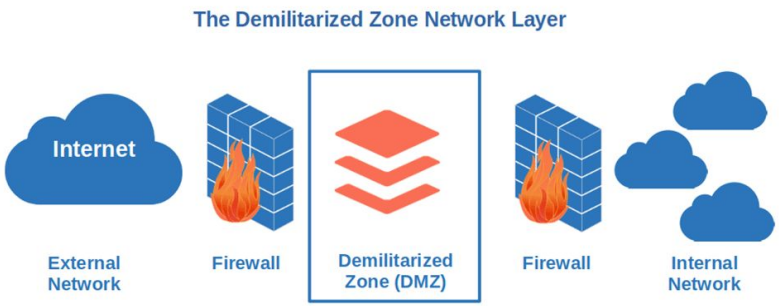
- Source zone
- Destination zone
- Protocol/port
- Direction
- Why it is needed
- Who owns it

3. Design around “default deny between zones”

Inside a zone you may allow normal internal communication.

Crossing boundaries have to go through security policies and secure controls.

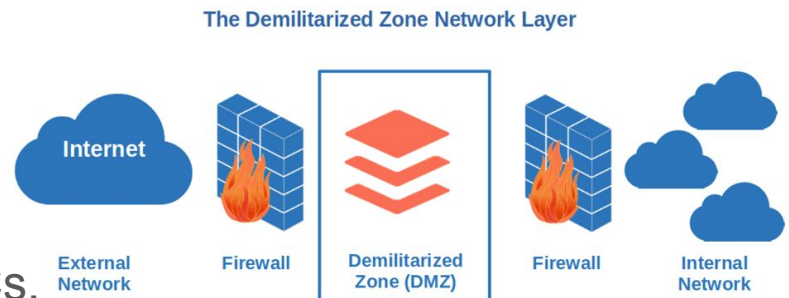
Network Segmentation



Try to not complicate it, Start with:

- 2–4 zones max (Based on product complexity)
- Simple allow rules
- Then tighten gradually, micro-segmentation.
- DMZs / Secure broker zones when dealing with external traffic
  Use strict security controls.

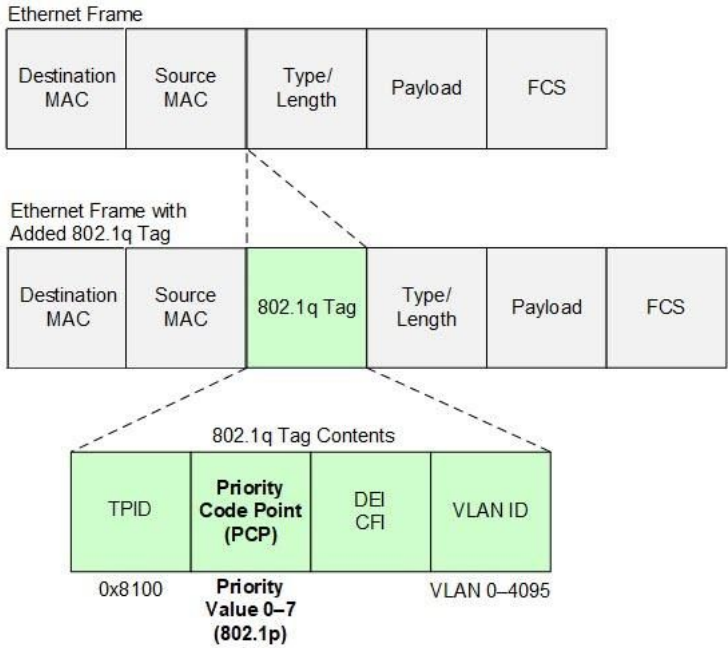
Mechanisms for Isolation



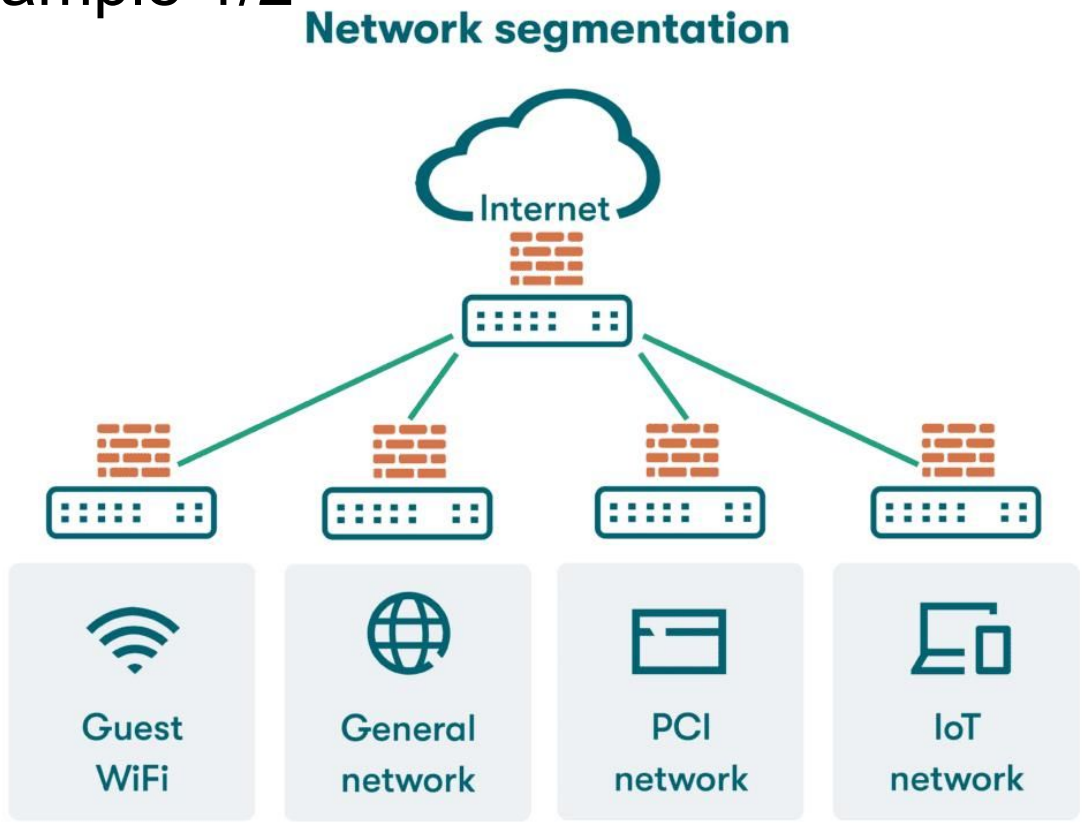
- Physical separation: Separate switches, NICs, or even separate boxes/networks for high-trust and low-trust domains.
- Layer-2 segmentation with VLANs
- Routed boundaries between VLANs - Routing policies
- Secure endpoints APIs using TLS
- Firewall
- Private APN in case of cellular connectivity

Virtual LAN

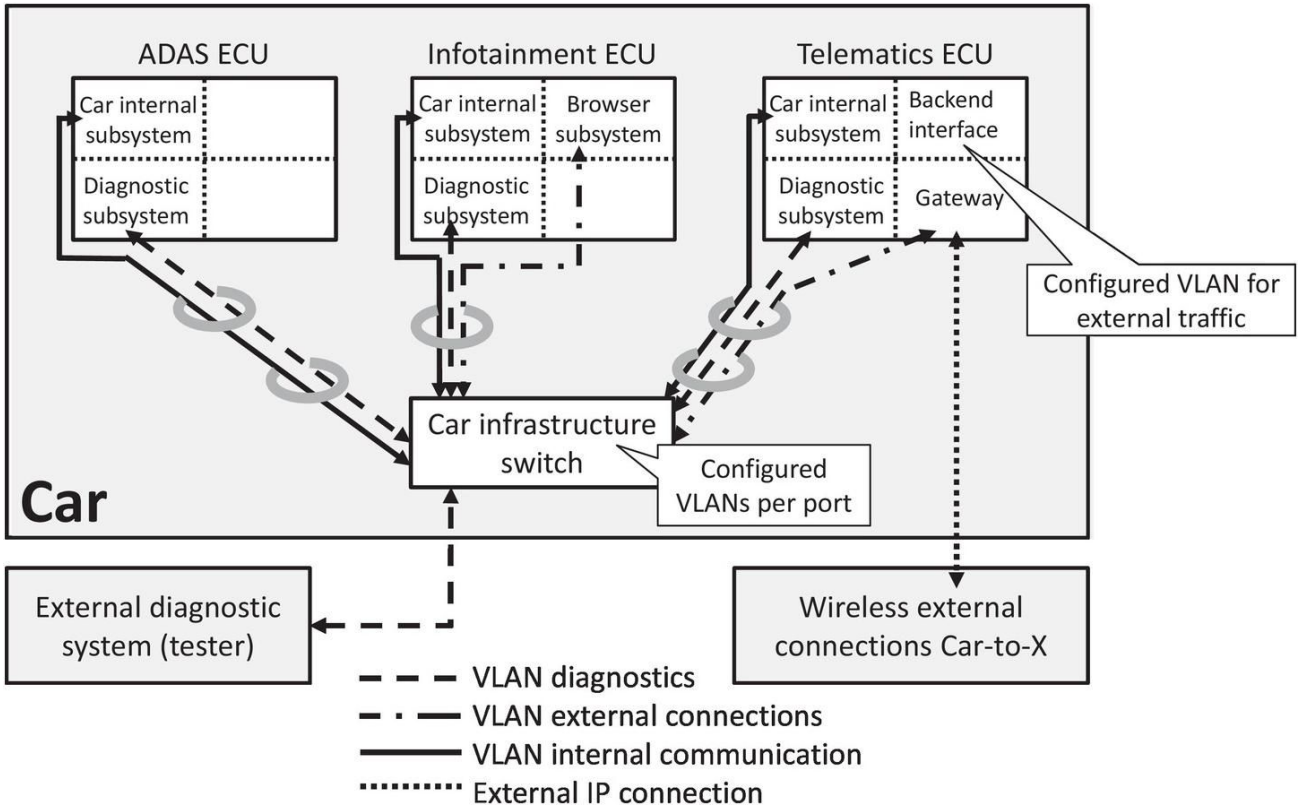
- VLAN splits one physical network into multiple logical networks.
- It can split them on ECU level, Switch level, or Router level.
- Works at Link Layer 2 using 802.1Q tags
- Each frame can carry:
  - VLAN ID (VID) → which network it belongs to
  - Priority (PCP) → QoS level (0–7)
- Devices in different VLANs
  don’t see each other unless explicitly routed.



Virtual LAN - Example 1/2



Virtual LAN - Example 2/2

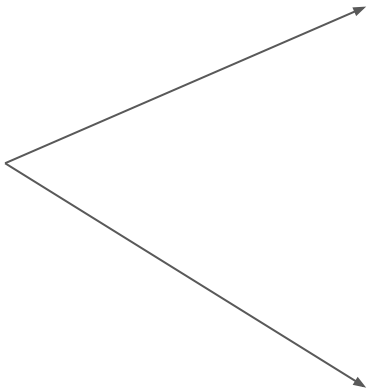


Virtual LAN - Linux

- systemd-networkd

eth0.network:
[Match]
Name=eth0

[Network]
VLAN=eth0.100
VLAN=eth0.200



eth0.100.netdev
[NetDev]
Name=eth0.100
Kind=vlan

[VLAN]
Id=100

eth0.200.netdev
[NetDev]
Name=eth0.200
Kind=vlan

[VLAN]
Id=200

eth0.100.network
[Match]
Name=eth0.100

[Network]
Address=192.168.100.1/24

eth0.200.network
[Match]
Name=eth0.200

[Network]
Address=192.168.200.1/24

Per-Process network isolation in Linux

- Network namespaces will restrict VLAN device or NIC per process
- Cgroups, to constraint access to network as a resource quota
- Systemd isolation using eBPF, per service allowlisting
  IPAddressDeny=any IPAddressAllow=192.168.111.2

Virtual LAN - Zephyr RTOS

- Easily done as simple Kernel config

```
# VLAN support
CONFIG_NET_VLAN=y
CONFIG_NET_VLAN_COUNT=2

# VLAN 100 Control Traffic
CONFIG_NET_CONFIG_MY_IPV4_ADDR="192.168.100.1"

# VLAN 200 Telemetry Traffic
CONFIG_NET_CONFIG_PEER_IPV4_ADDR="192.0.200.1"
```

Ref: https://docs.zephyrproject.org/latest/samples/net/ethernet/vlan/README.html#vlan

Network Segmentation

- Allows for beneficial things to enforce:
  - Data rate constraining
  - Packets priority
  - Protocols allow list.
  - Stateful tracking
    Different security policies to apply, who initiates? How many sessions?

Packet filtering - Stateless

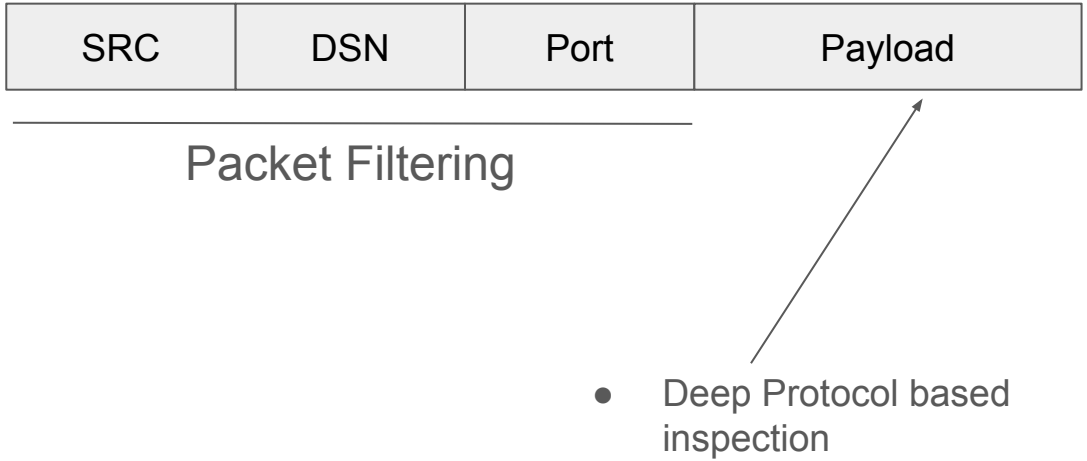
Packet filtering / firewall

Not everyone should access every door.

Segmentation without filtering is just labeling. You need to enforce the segmentation.

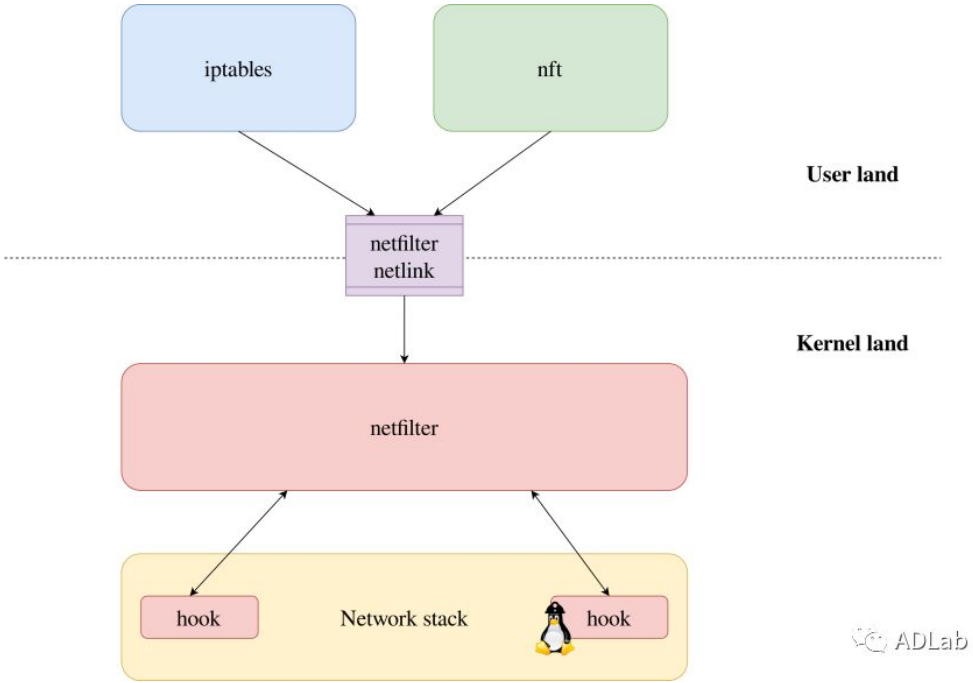
You need filtering when:

- Deny all.
- Explicit allow rules
- Drop / Reject:
  - unexpected ports
  - unexpected IPs
  - malformed traffic

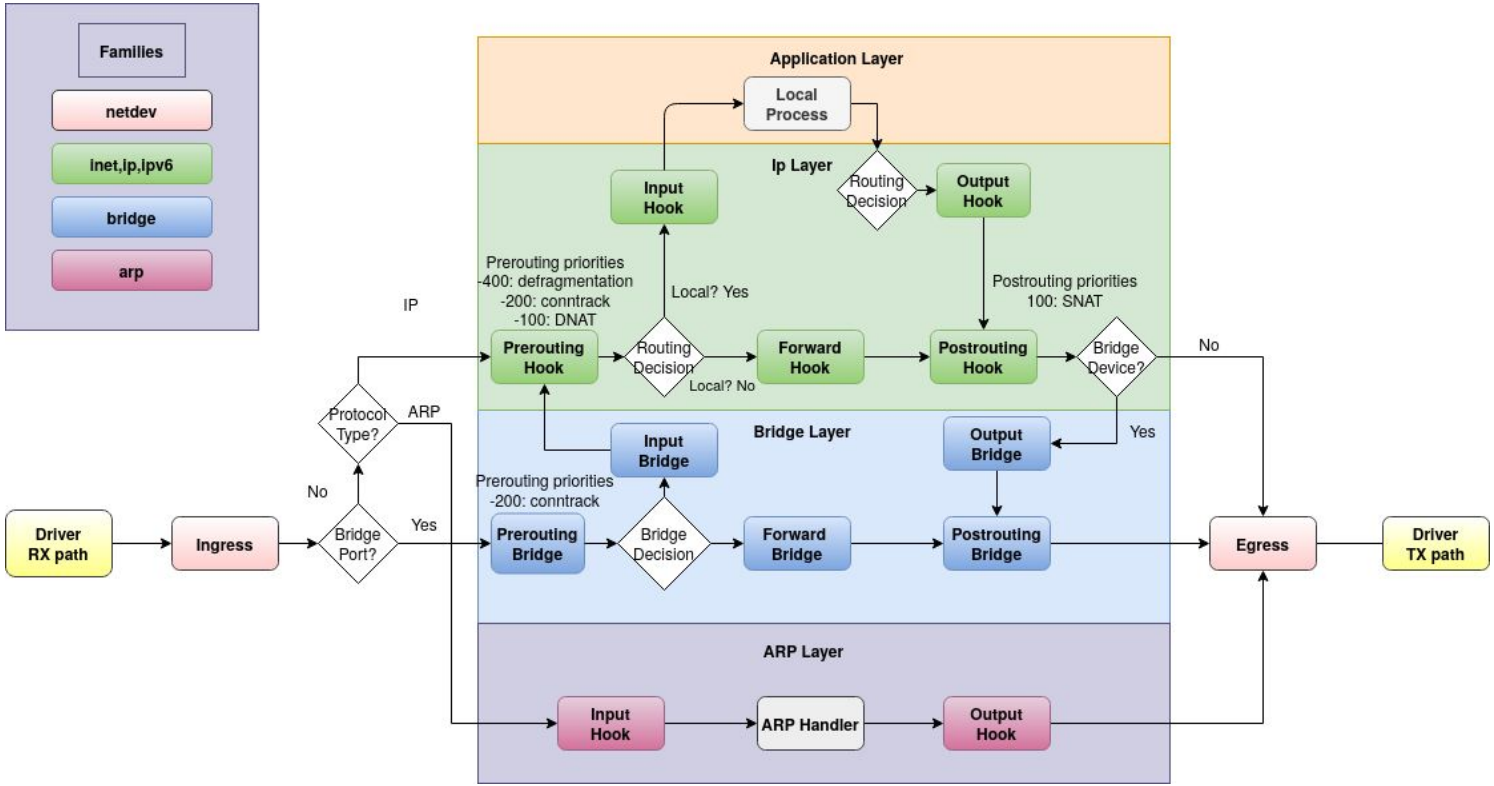


Packet filtering - Linux

- Netfilter: kernel framework
- Nftables: user-space rule engines configuration



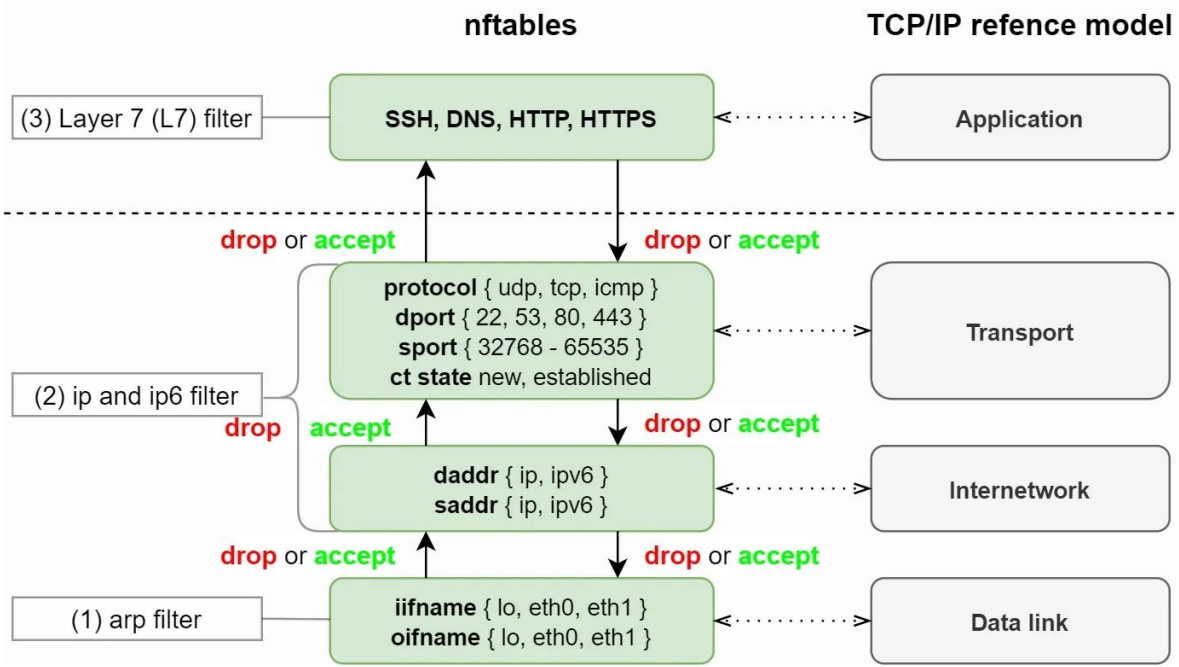
Packet filtering - Linux Kernel Hooks



Ref: https://wyssmann.com/blog/2021/07/packet-filtering-in-linux-iptables-nftables-and-firewalld/

Packet filtering - Linux

- Nftables: user-space rule engines configuration



Ref: https://cryptsus.com/blog/setting-up-nftables-firewall.html

Packet filtering - Stateful

Implemented using Conntrack in nftables:

- First packet → NEW
- Firewall decides → allow or drop
- If allowed → entry created in conntrack table
- Next packets → ESTABLISHED
- Related traffic (ICMP, etc.) → RELATED
- INVALID → broken / unexpected packets

nftables example:

```
ct state invalid drop
ct state established,related accept

iif "eth0" tcp dport 443 ct state new accept
```

Packet filtering - Forensics

- Packets can be logged, counted and marked
- Dropped / Rejected packets
- Unsolicited connections
- New connections
- Wrong protocols
- Can be collected from kernel logs

nftables example:

```
ct state invalid log prefix "INVALID: " drop
log prefix "DROP: " flags all counter drop
tcp dport 22 ct state new log prefix "SSH NEW: "
accept
limit rate 10/second log prefix "FLOOD: " drop
```

Packet filtering - Zephyr

- Rules are defined as C Code, compile-time filtering hooks.
- Zephyr PF is stateless and doesn't track connections for example.

```
/* Define allowed source IP list */
static struct in_addr allowed_ips[] = {
    { { { 192, 168, 1, 10 } } }
};

/* Match allowed source IP */
NPF_IP_SRC_ADDR_ALLOWLIST(src_allow,
    allowed_ips,
    ARRAY_SIZE(allowed_ips),
    NET_AF_INET);

/* Match IPv4 */
NPF_ETH_TYPE_MATCH(ipv4_match, NET_ETH_PTYPE_IP);

/* Rule: allow packets from allowed IP */
NPF_RULE(allow_rule, NET_OK, ipv4_match, src_allow);

/* Install rules */
void setup_filter(void)
{
    npf_append_rule(&npf_recv_rules, &allow_rule);
    npf_append_rule(&npf_recv_rules, &npf_default_drop);
}
```

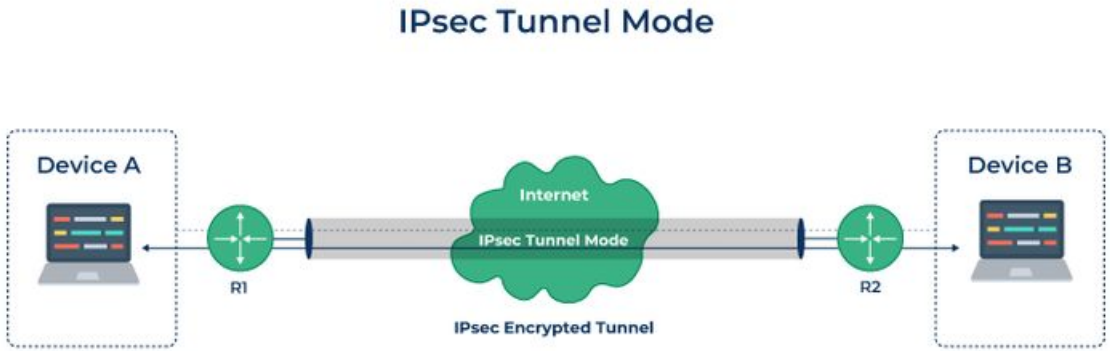
Rule matching result:
NET_OK
NET_DROP
NET_CONTINUE

Ref: https://docs.zephyrproject.org/latest/services/connectivity/networking/api/net_pkt_filter.html

VPN / IPSec

Network Layer Protection: Extend a private network over untrusted networks

- Site to Site
- Device to Device
- Device to Cloud



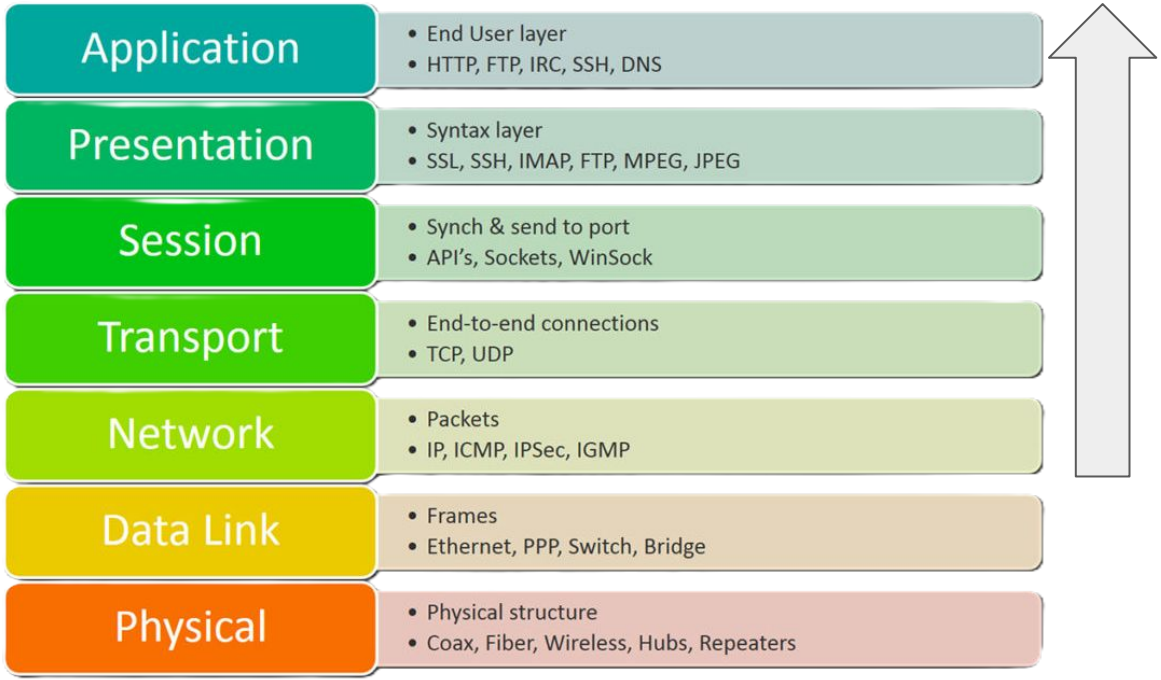
IPSec protects all IP traffic transparently, without apps needing to know

VPN / IPSec

Network Layer Protection: Extend a private network over untrusted networks

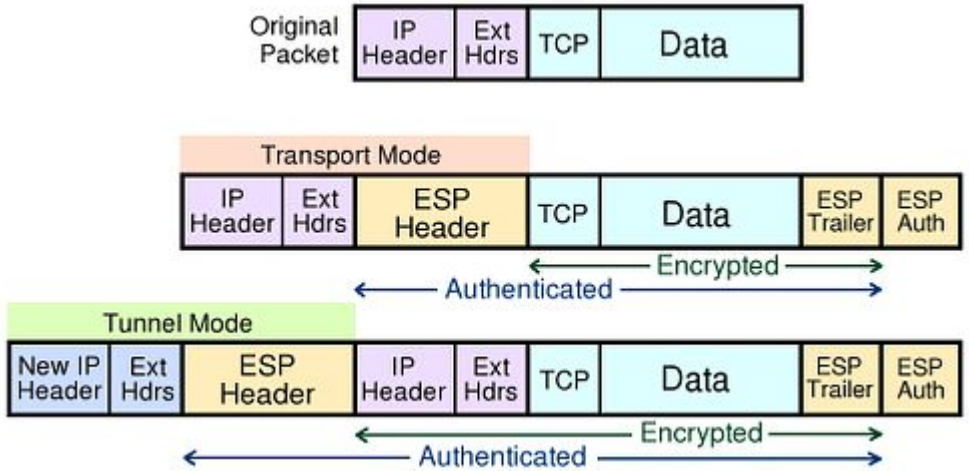
- Site to Site
- Device to Device
- Device to Cloud

VPN / IPSec



VPN / IPSec

- How does it work?
  - Authentication
  - Encryption
  - Integrity
  - Rekeying to keep it fresh

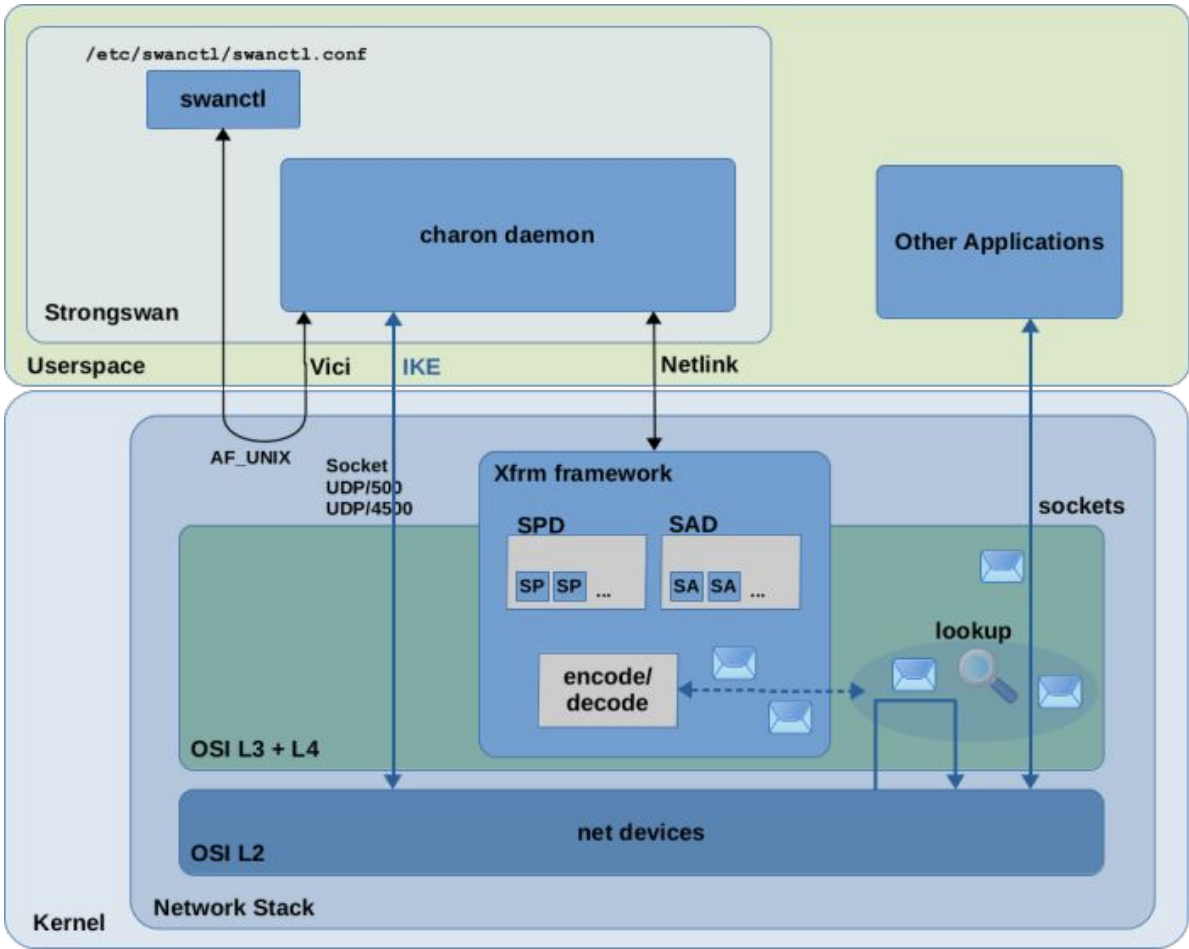


- Support for two modes
  - Transport mode: Payload only. Protect host-to-host (example two ECUs)
  - Tunnel mode: Whole Packet. Protect network to network (example cellular core network to AWS)

IPSec / VPN in Linux

- IKE protocol
- Security Associations
- Security Policies

- Charon
  - Control plane .. Configurations
  - IKE and SA definitions
  - Crypto, Authentication
  - Key exchange
  - Secrets
- Xfrm (transform)
  - Data plane
  - Matching traffic
  - Encrypting+encapsulating
  - Decrypting+decapsulating



VPN in Zephyr

- Zephyr just added support for lightweight VPN (Wireguard) in 4.4
- Zephyr doesn’t support IPSec (IKEv2) protocol.
- Wireguard can setup as Kernel Configs

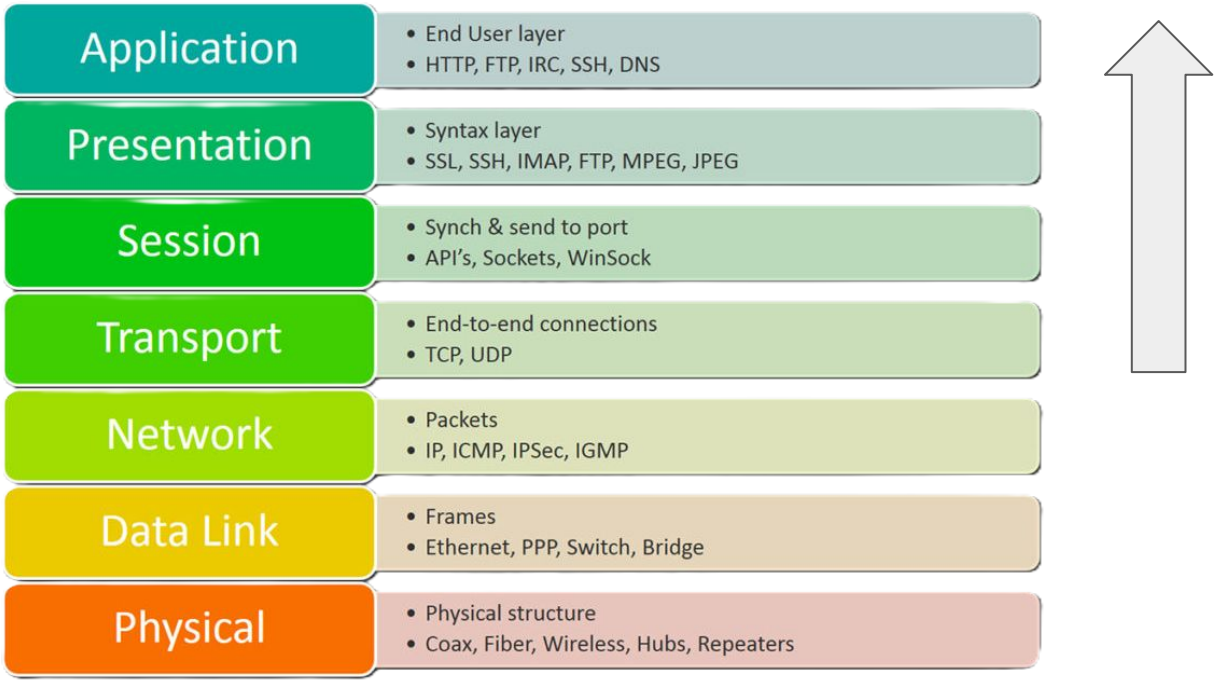
```
CONFIG_NET_SAMPLE_COMMON_VPN_PEER_IP_ADDR=""
CONFIG_NET_SAMPLE_COMMON_VPN_MY_ADDR=""
CONFIG_NET_SAMPLE_COMMON_VPN_ALLOWED_PEER_ADDR=""
CONFIG_NET_SAMPLE_COMMON_VPN_MY_PRIVATE_KEY=""
CONFIG_NET_SAMPLE_COMMON_VPN_PEER_PUBLIC_KEY=""
```

- It can also setup as C code, where keys can be used from secure module.

Ref: https://docs.zephyrproject.org/latest/samples/net/wireguard/README.html

Transport Layer Security - TLS

TLS



TLS - secure the message

Application Layer Protection:

Use TLS when:

- You control the application or endpoint API. You only need to protect specific data flows for one specific endpoint in the app.
- You want identity and authentication per app and endpoint.

IoT Protocols that can support TLS:

- HTTPS
- MQTT over TLS (IoT standard)
- LWM2M
- OTA updates

TLS - Zephyr / Linux

Zephyr

Integrates lightweight libraries like mbedTLS

Kernel Configs for protocols for keys or certificates:

- MQTT
- LWM2M
- HTTPS

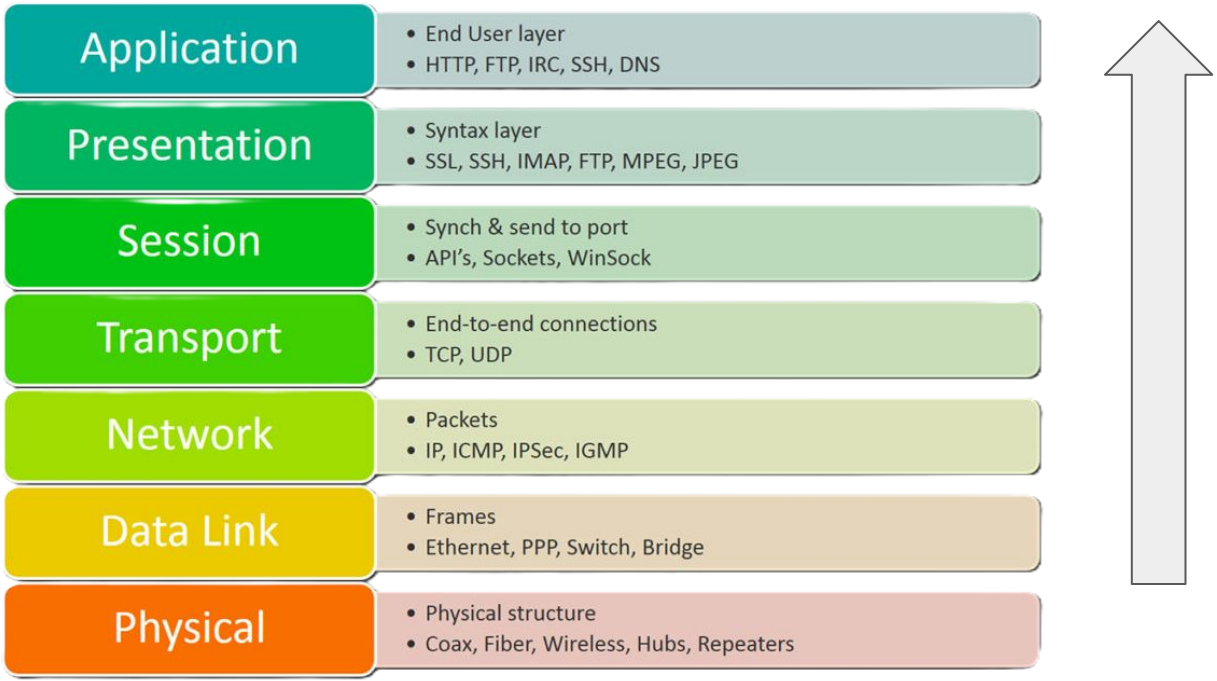
Linux

Most services and protocols rely on:

OpenSSL or GnuTLS

MACSec

MACSec / WPA3



MACSec

L2 Link Layer Protection : Secure the physical cable.

MACsec ensures the data on the wire is encrypted and cannot be read or altered.

Stops attacks like sniffing, spoofing, and unauthorized monitoring.

Inteded for nodes or peers on the same network.

Typical in

- Automotive
- Industrial control network, Factory Floor
- Data Center switches

It encrypt everything on the wire, very low latency due to being hardware accelerated, Invisible to IP stack.

Excellent if physical tampering to the wire is a risk identified, or for extra defense in depth

Authentication / Encryption / Integrity dependencies

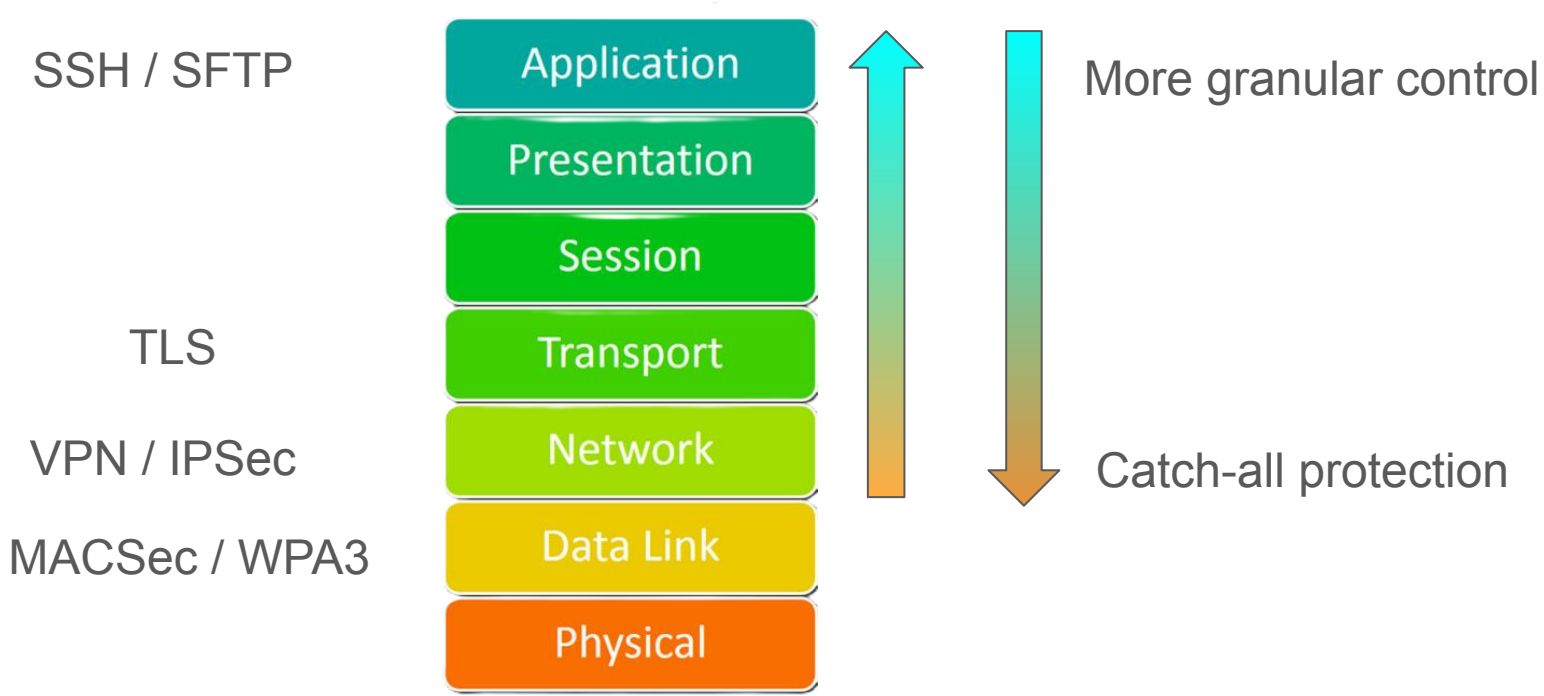
- Public Key Infrastructure
  - Private Key instrumentation
  - Certificate signing issuance and management
  - Pre-Shared key
- Certificate rotation
- Signer Certificate authority (Root Certificates)
- Logistics of provisioning devices at manufacturing with secrets and identity
- Distributing and downloading certificates or secrets
- Storage in Secure elements, hardware security module, Trusted Platform Modules, One-time programmable.
- Crypto-capable processor with cypher suite.
- Correct time on the device (Validation of certificates issue and validity)

The right mechanism for the right purpose

- Use TLS (Layer 7 app) .. Secure the message
  - End-to-End encryption for applications
  - Secure data / APIs endpoints
  - Per-application, Granular security crypto and policies
  - Good example: MQTT/HTTPS APIs
- Use IPSec / VPN (Layer 3) .. Secure the route
  - Secure all traffic exiting the device
  - Secure Site-to-Site or Network to Network or Zone to Zone
  - Encrypt the whole apps/services traffic
  - Good example: Gateway ECU, Backend/Cloud, tunneling over internet.
- Use MACSec (Complimentary to IPSec or TLS)
  - Secure the traffic passing through the cable
  - Secure backbone links
  - Good example: Automotive, Industrial, Servers Rack to Rack

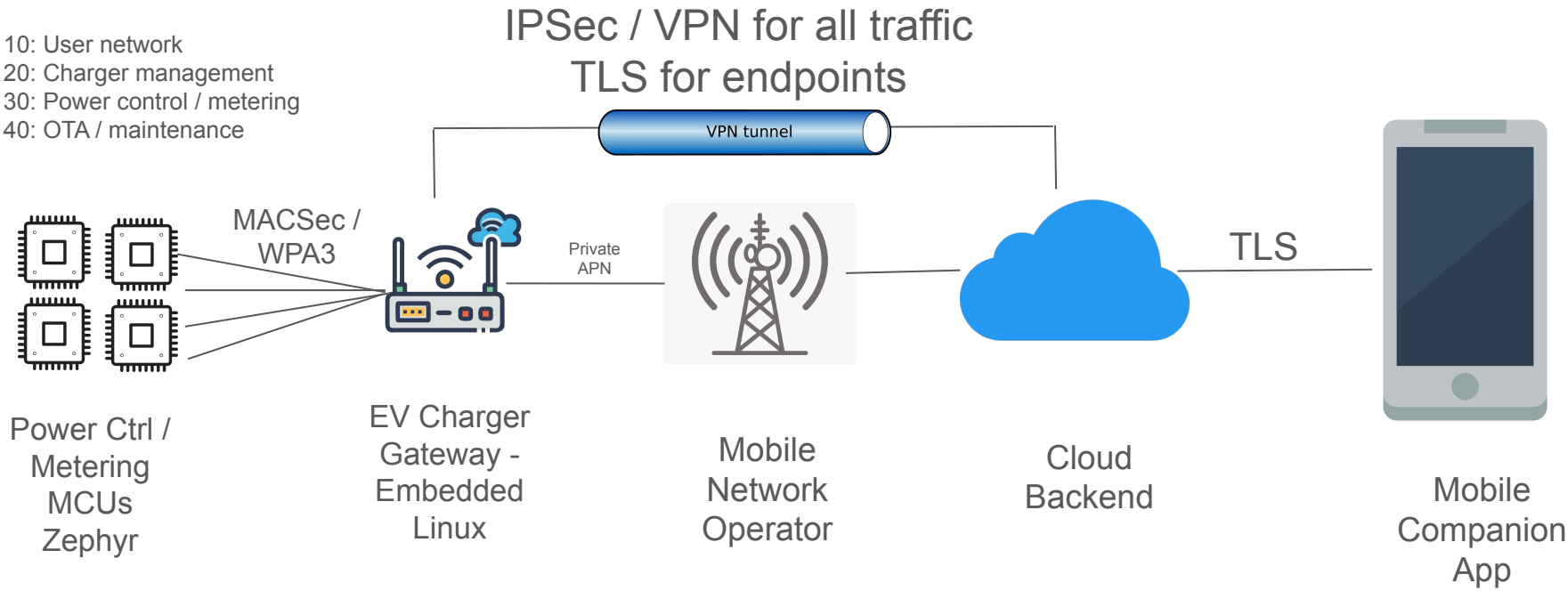
The right mechanism for the right purpose

A security implemented at a lower layer will be inherited by the upper layers



Example case: Smart EV Charging Station

VLAN 10: User network
VLAN 20: Charger management
VLAN 30: Power control / metering
VLAN 40: OTA / maintenance



Summary and conclusion

- Modern products are connected and have higher attack surface than ever.
- Security is a marathon, not a milestone
- You will make trade-offs — be intentional
- Identify your data and the risks
- Set the foundation of the security in the architecture from day-1, so it’s woven inside the product architecture.
- Get the fundamentals and prioritize the right security control.

Summary and conclusion

- FOSS Tools exist to protect data in transit:
  - Segment your network based on security Level
  - Use firewall to lock down the product
  - Protect the data flowing through thesee firewall using trusted channels.

Thanks for listening

Open Questions