

OPENSIPS + ENVOY

# OpenSIPS + Envoy

A Solution for TLS networking.

Adam Overbeeke, Principal Architect  
Bence Szigeti, Staff Voice Software Engineer  
May 27, 2025



OPENSIPS + ENVOY

# Presenter



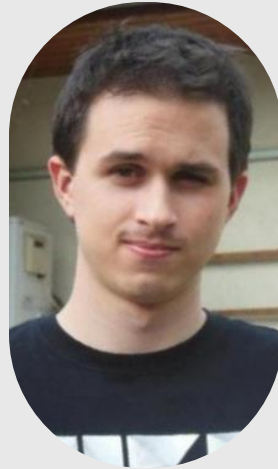
Adam Overbeeke  
PRINCIPAL ARCHITECT, GENESYS

OPENSIPS + ENVOY

# The Genesys Team



**Adam Overbeeke**  
**PRINCIPAL ARCHITECT,**  
**GENESYS**



**Bence Szigeti**  
**STAFF VOICE ENGINEER,**  
**GENESYS**



**Ben Newlin**  
**SENIOR DIRECTOR,**  
**GENESYS**



**David Trihy**  
**LEAD SOFTWARE**  
**ENGINEER, GENESYS**

- Agenda

01 **Introduction**

02 **Envoy**

Basic understanding

03 **System Architecture**

OpenSIPS + Envoy + AWS

04 **Envoy Configuration**

Ingress

Egress

05 **Demo**

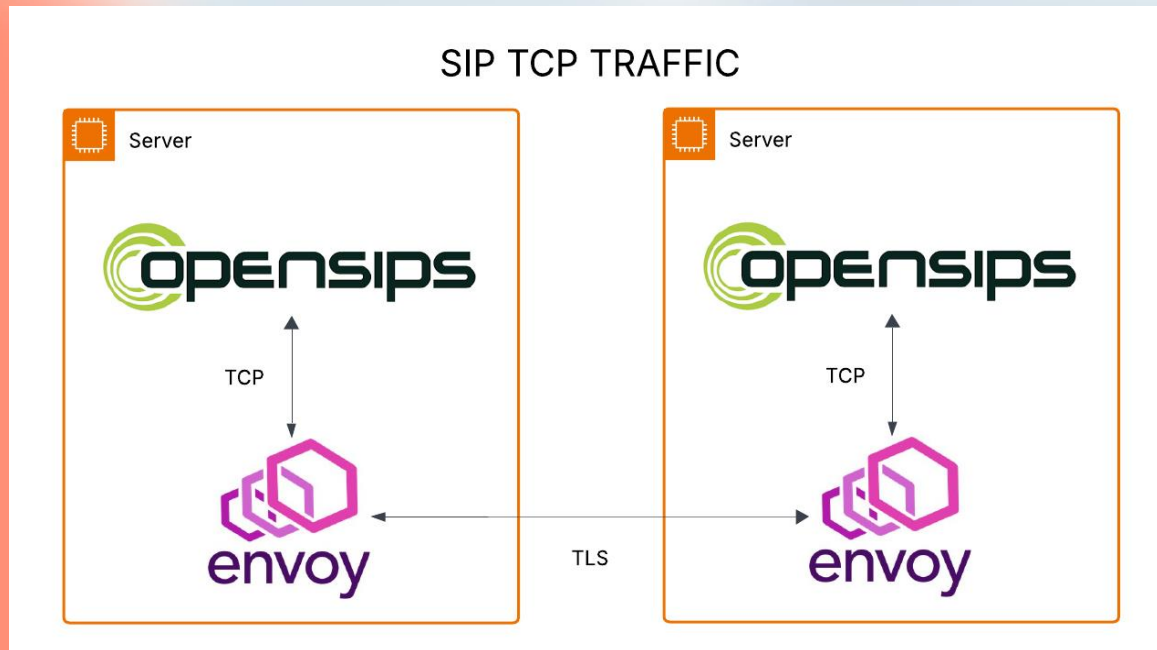
TLS PROXY

06 **What's Next? Questions?**

# Introduction



# Introduction



Using Envoy with TLS for SIP provides secure, flexible control over SIP traffic encryption, allowing custom SSL/TLS configurations.

- Agenda

01 Introduction

02 **Envoy**

Basic understanding

03 System Architecture

OpenSIPS + Envoy + AWS

04 Envoy Configuration

Ingress

Egress

05 Demo

TLS PROXY

06 What's Next? Questions?

# Envoy

## Basic understanding

<https://www.envoyproxy.io/>



Envoy is a modern, high-performance proxy, designed to handle both **Layer 7 (Application Layer)** and **Layer 4 (Transport Layer)** traffic. It is widely adopted for use cases such as **load balancing**, **service discovery**, **traffic routing**, and **observability** in microservices and cloud-native environments.

In our configuration, Envoy serves as a **termination point**, for secure connections, and decodes them for downstream systems like **OpenSIPS**. This utilizes Envoy's **Layer 4 capabilities**, acting as a transparent proxy without needing to understand SIP semantics.

By offloading TLS responsibilities to Envoy:

- We simplify **certificate management**
- Improve **security and compliance ( FIPS )**
- Gain better **traffic control + logging**
- Avoid changes to the underlying applications (remaining unaware of the TLS layer)



- Agenda

01 Introduction

02 Envoy

Basic understanding

03 **System Architecture**

OpenSIPS + Envoy + AWS

04 Envoy Configuration

Ingress

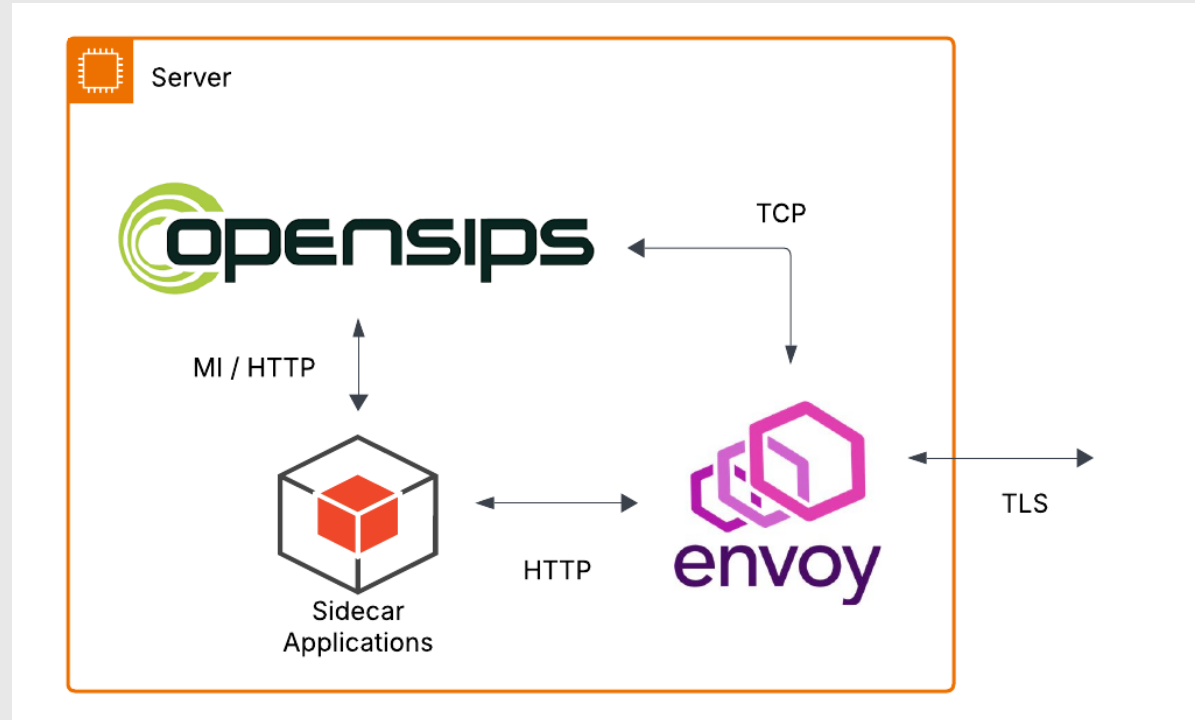
Egress

05 Demo

TLS PROXY

06 What's Next? Questions?

## System Architecture Overview: OpenSIPS + Envoy + AWS



Create a EC2 instance – loaded with OpenSIPS, Envoy, other applications

**Envoy** runs as a sidecar application to OpenSIPS.

**Envoy** handles the ingress and egress traffic for EC2, converting SIP connections from TLS to TCP and TCP to TLS.

Envoy is also used for HTTPS to HTTP traffic for sidecar applications.

- Agenda

01 Introduction

02 Envoy  
Basic understanding

03 Our Architecture  
OpenSIPS + Envoy + AWS

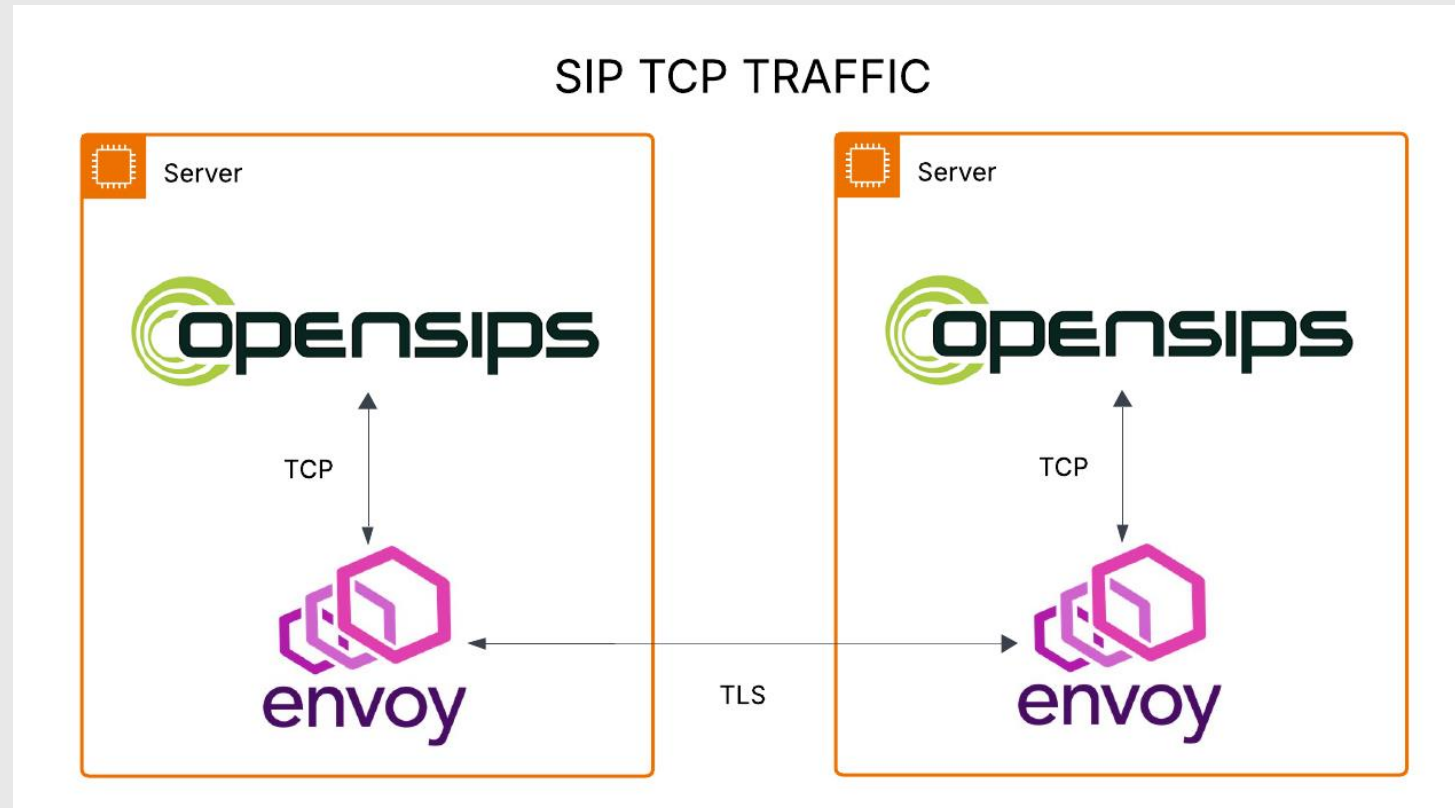
04 Envoy Configuration  
Ingress  
Egress

05 Demo  
TLS PROXY

06 What's Next? Questions?

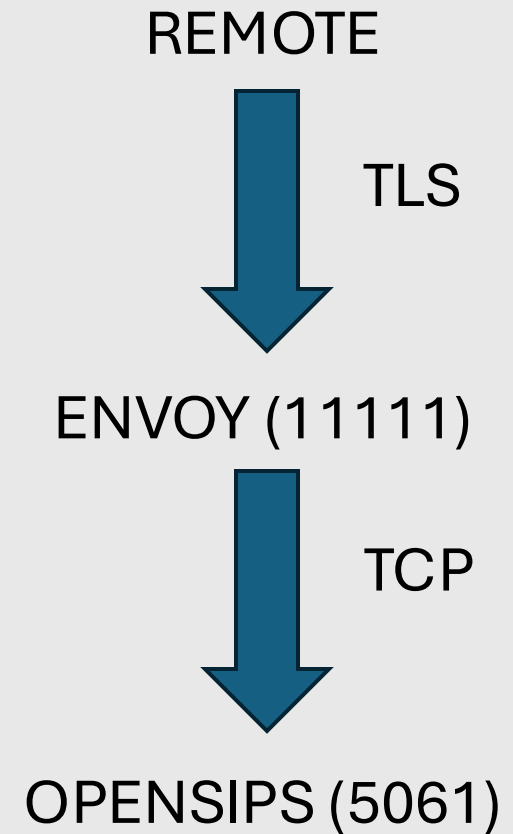
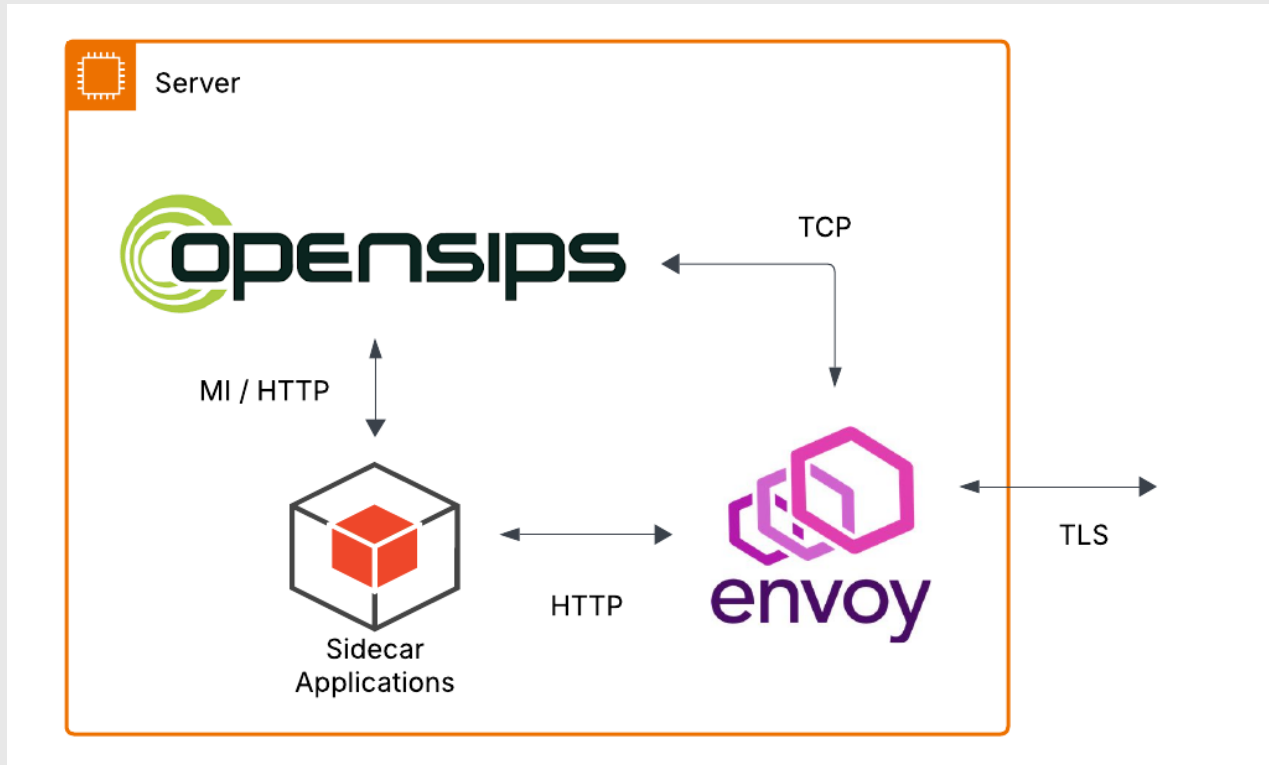
# Trusted Communication

## Flow



# Envoy Configuration

## Ingress Flow



## Ensuring Envoy gets the TLS traffic

### Port Mappings

In our setup we are going to configured OpenSIPS to listen to port 5061 and Envoy to port 11111

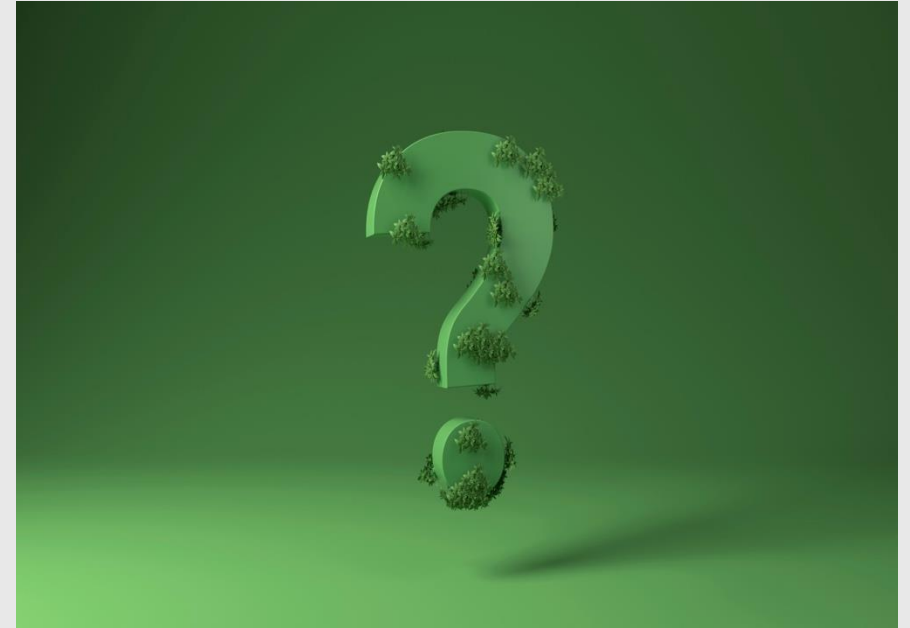
We know Envoy is the gateway for secure traffic on EC2 – How does this work?



Port 5061



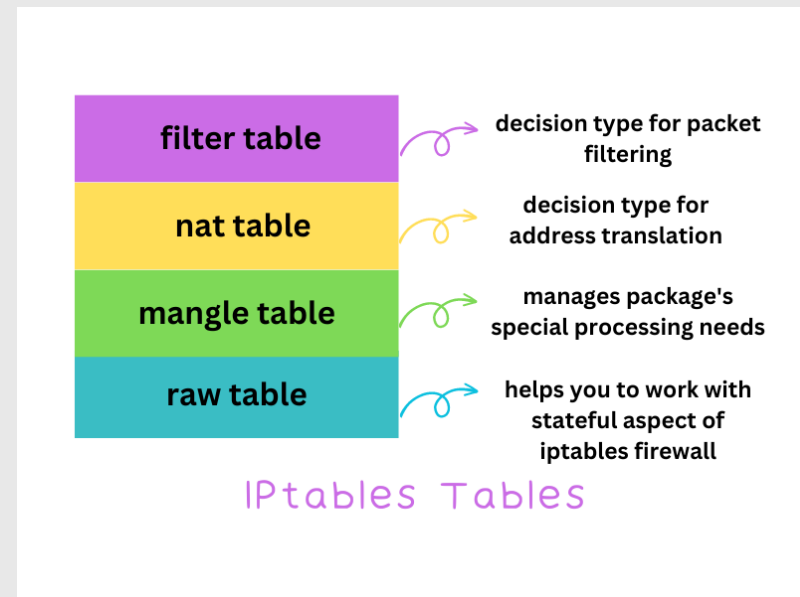
Port 11111



# IPTables

## Redirect Rules

Using the NAT table rules to REDIRECT TCP traffic destined to port 5061 to Envoy on port 11111.



bash

Copy

Edit

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 5061 -j REDIRECT --to-port 11111
```

## Envoy Configuration

<https://www.envoyproxy.io/docs/envoy/latest/configuration/configuration>

Configuration file - **JSON** or **YAML**

Key parameters:

- **Listeners** – incoming requests
- **Filter Chains** – request processing
- **Clusters** - destination





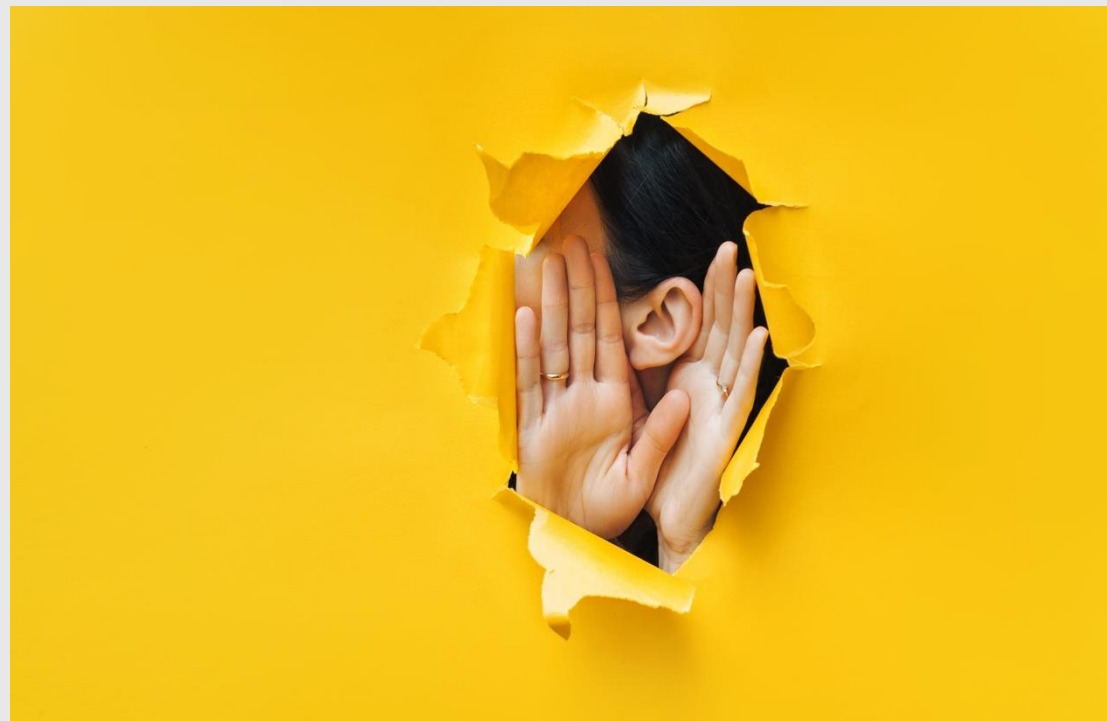
# Envoy Configuration - Ingress

```
1
2 static_resources:
3   listeners:
4     - name: ingress_listener
5       address:
6         socket_address:
7           address: 0.0.0.0
8           port_value: 11111
9       listener_filters:
10      - name: envoy.filters.listener.original_src
11        typed_config:
12          "@type": type.googleapis.com/envoy.extensions.filters.listener.original_src.v3.OriginalSrc
13          mark: 123
14      filter_chains:
15        - filters:
16          - name: envoy.filters.network.tcp_proxy
17            typed_config:
18              "@type": type.googleapis.com/envoy.extensions.filters.network.tcp_proxy.v3.TcpProxy
19              stat_prefix: ingress_tcp
20              cluster: ingress_cluster
21          transport_socket:
22            name: envoy.transport_sockets.tls
23            typed_config:
24              "@type": type.googleapis.com/
25                envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
26              common_tls_context:
27                tls_certificates:
28                  - certificate_chain:
29                    filename: "/etc/envoy/certs/server.crt"
30                  private_key:
31                    filename: "/etc/envoy/certs/server.key"
32      clusters:
33        - name: ingress_cluster
34          type: STATIC
35          lb_policy: ROUND_ROBIN
36          load_assignment:
37            cluster_name: ingress_cluster
38            endpoints:
39              - lb_endpoints:
40                - endpoint:
41                  address:
42                    socket_address:
43                      address: 127.0.0.1
44                      port_value: 5061
```

# Envoy Configuration

## Listeners

```
static_resources:
  listeners:
    - name: ingress_listener
      address:
        socket_address:
          address: 0.0.0.0
          port_value: 11111
      listener_filters:
        - name: original_src
          typed_config:
            "@type": OriginalSrc
          mark: 123
```



# Envoy Configuration

## Filters

```
filter_chains:  
- filters:  
  - name: envoy.filters.network.tcp_proxy  
    type_config:  
      "@type": ...TcpProxy  
      stat_prefix: tcp_proxy  
      cluster: ingress_cluster  
  
transport_socket:  
- name: envoy.transport_sockets.tls  
  typed_config:  
    "@type": ...DownstreamTlsContext  
    common_tls_context:  
      tls_certificates:  
        - certificate_chain: server.crt  
          private_key: server.key
```



# Envoy Configuration

## Clusters



```
clusters:
- name: ingress_cluster
  type: STATIC
  load_assignment:
    cluster_name: ingress_cluster
  endpoints:
  - lb_endpoints:
    - endpoint:
        address:
          socket_address:
            address: 127.0.0.1
            port_value: 5061
```

- Agenda

01 Introduction

02 Envoy

Basic understanding

03 Our Architecture

OpenSIPS + Envoy + AWS

04 Envoy Configuration

Ingress

**Egress**

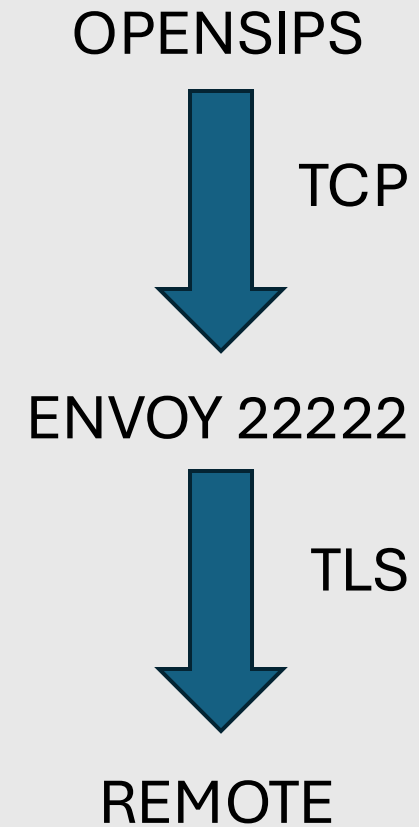
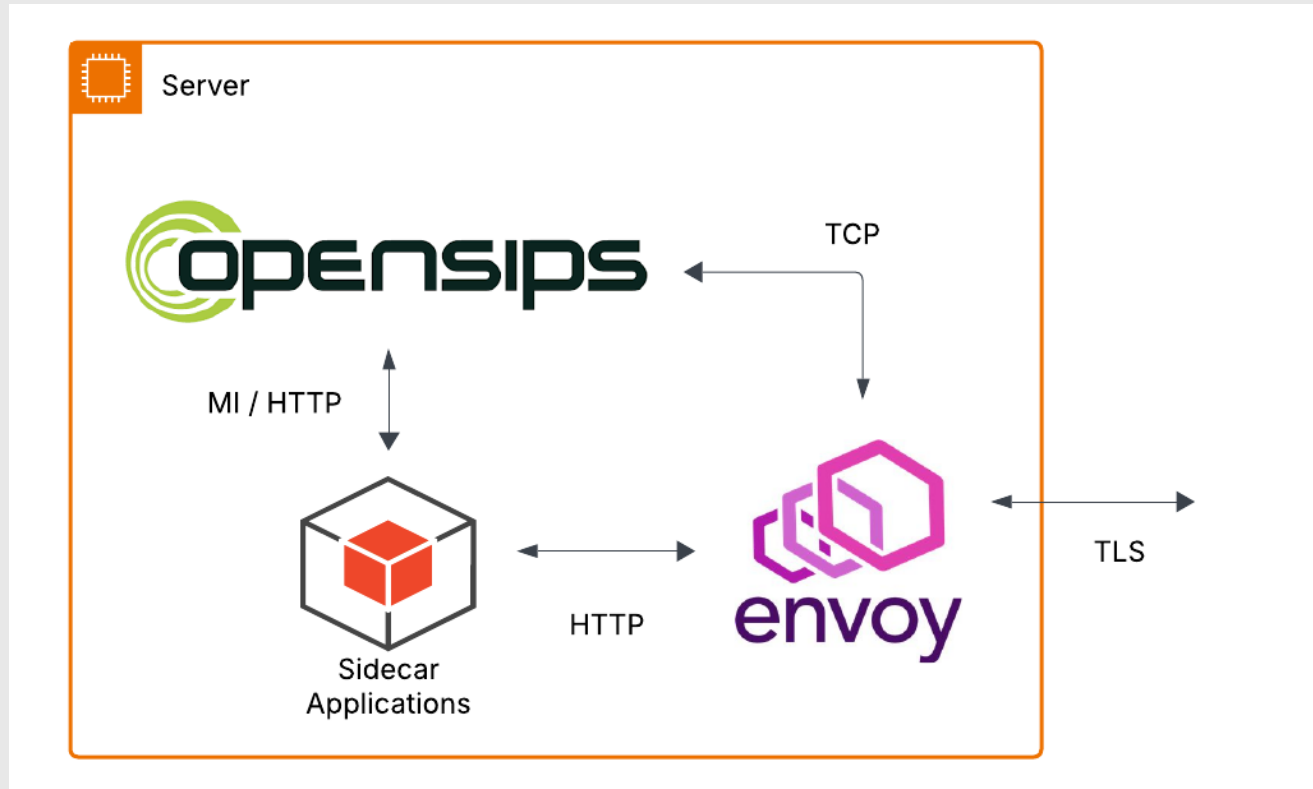
05 Demo

TLS PROXY

06 What's Next? Questions?

# Envoy Configuration

## Egress Flow



# OpenSIPS TLS via TCP

## Packet Marking

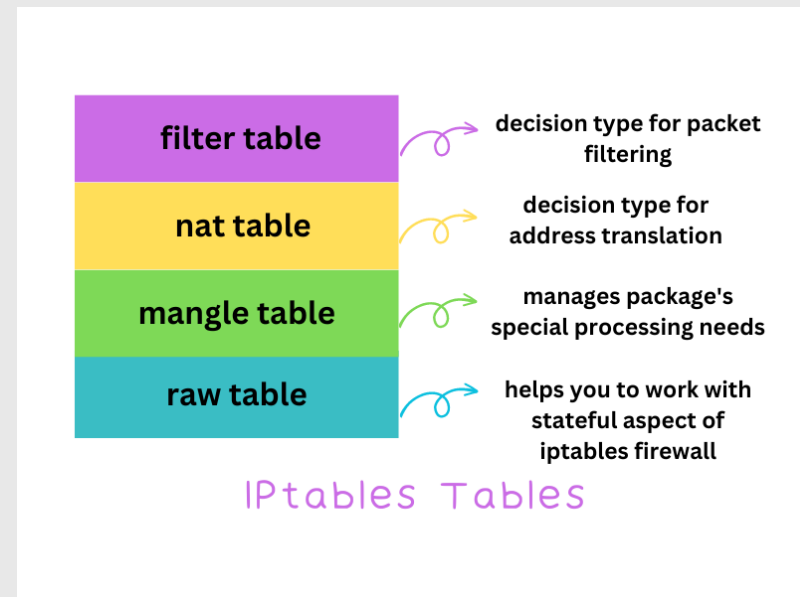
opensips-summit-2025 / config / opensips.cfg

```
1  mpath = "/usr/local/lib64/opensips/modules"
2
3  loadmodule "proto_tcp.so"
4  loadmodule "tm.so"
5
6  socket = tcp:0.0.0.0:5061 mark 42
7
8  log_level=3
9  xlog_level=3
10
11 route {
12     xlog("L_INFO", "[%time(%Y-%m-%d %H:%M:%S)] TCP data from '$si:$sp'.\n");
13     $du = "sip:uas:5061;transport=tcp";
14     t_relay();
15 }
16
```

# IPTables

## Redirect Rules

Using the NAT table rules to REDIRECT marked packets to Envoy on port 22222.



bash

Copy

Edit

```
iptables -t nat -A PREROUTING -p tcp -m mark --mark 42 -j REDIRECT --to-port 22222
```



# Envoy Configuration - Egress

```

1 static_resources:
2   listeners:
3     - name: egress_listener
4       address:
5         socket_address:
6           address: 0.0.0.0
7           port_value: 22222
8       listener_filters:
9         - name: envoy.filters.listener.original_dst
10           typed_config:
11             "@type": type.googleapis.com/envoy.extensions.filters.listener.original_dst.v3.OriginalDst
12       filter_chains:
13         - filters:
14             - name: envoy.filters.network.tcp_proxy
15               typed_config:
16                 "@type": type.googleapis.com/envoy.extensions.filters.network.tcp_proxy.v3.TcpProxy
17                 stat_prefix: tcp_proxy
18                 cluster: egress_cluster
19 clusters:
20   - name: egress_cluster
21     type: ORIGINAL_DST
22     lb_policy: CLUSTER_PROVIDED
23     transport_socket:
24       name: envoy.transport_sockets.tls
25       typed_config:
26         "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.UpstreamTlsContext
27         sni: "*"
28         common_tls_context:
29           validation_context:
30             trusted_ca:
31               filename: /etc/envoy/certs/server.crt

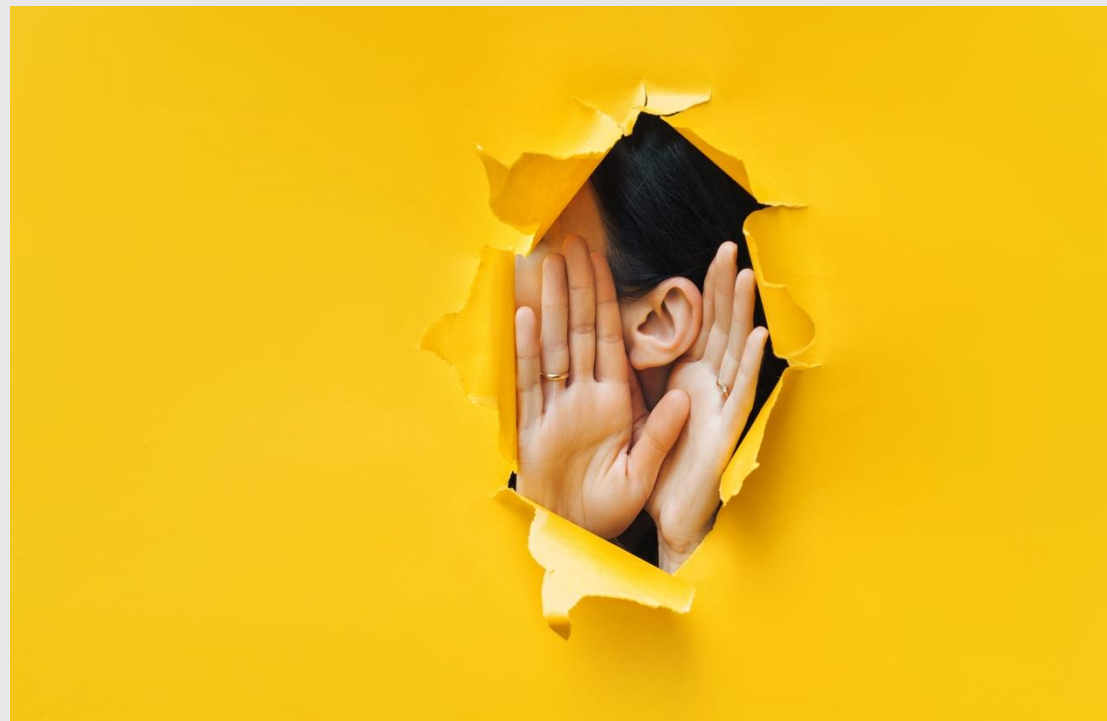
```

# Envoy Configuration

## Listeners

---

```
static_resources:
  listeners:
    - name: example_egress_tls
      address:
        socket_address:
          address: 0.0.0.0
          port_value: 22222
      listener_filters:
        - name: original_dst
          typed_config:
            "@type": OriginalDst
```



# Envoy Configuration

## Filters

```
filter_chains:  
- filters:  
  - name: envoy.filters.network.tcp_proxy  
    type_config:  
      "@type": ...TcpProxy  
      stat_prefix: tcp_proxy  
      cluster: egress_cluster
```



# Envoy Configuration

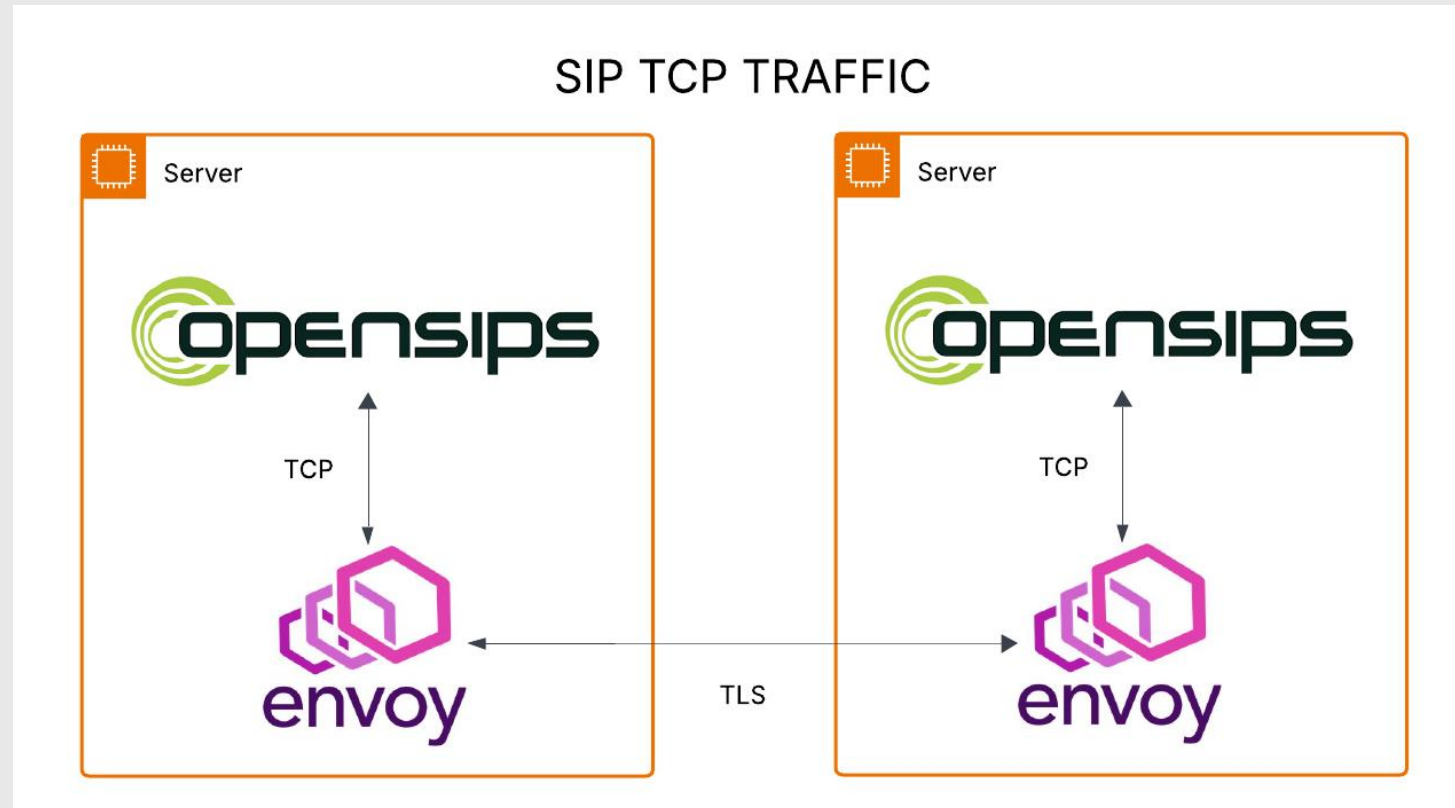
## Clusters



```
clusters:
- name: egress_cluster
  type: ORIGINAL_DST
  lb_policy: CLUSTER_PROVIDED
  transport_socket:
    name: envoy.transport_sockets.tls
  typed_config:
    "@type": UpstreamTlsContext
    sni: "*"
    common_tls_context:
      validation_context:
        trusted_ca:
          filename: server.crt
```

# Trusted Communication

## Flow



- Agenda

01 Introduction

02 Envoy

Basic understanding

03 System Architecture

OpenSIPS + Envoy + AWS

04 Envoy Configuration

Ingress

Egress

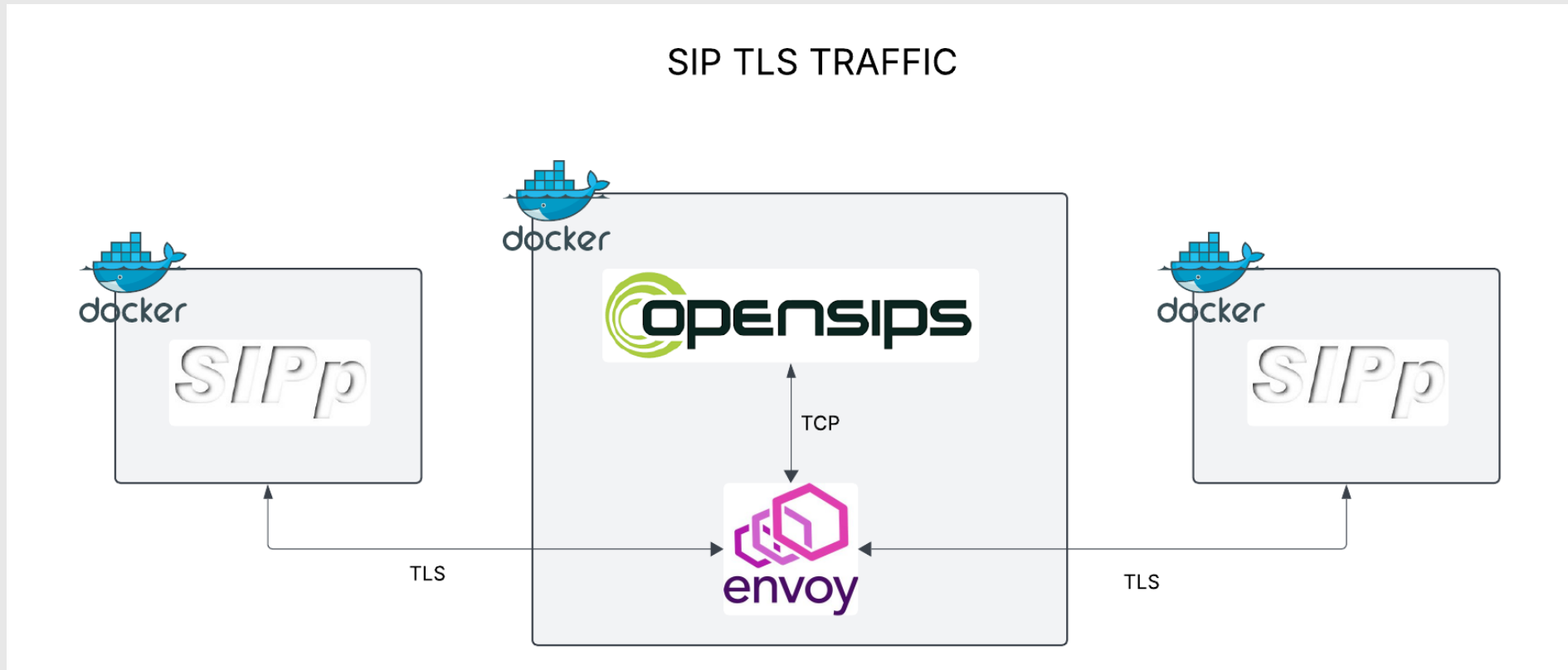
05 **Demo**

TLS PROXY

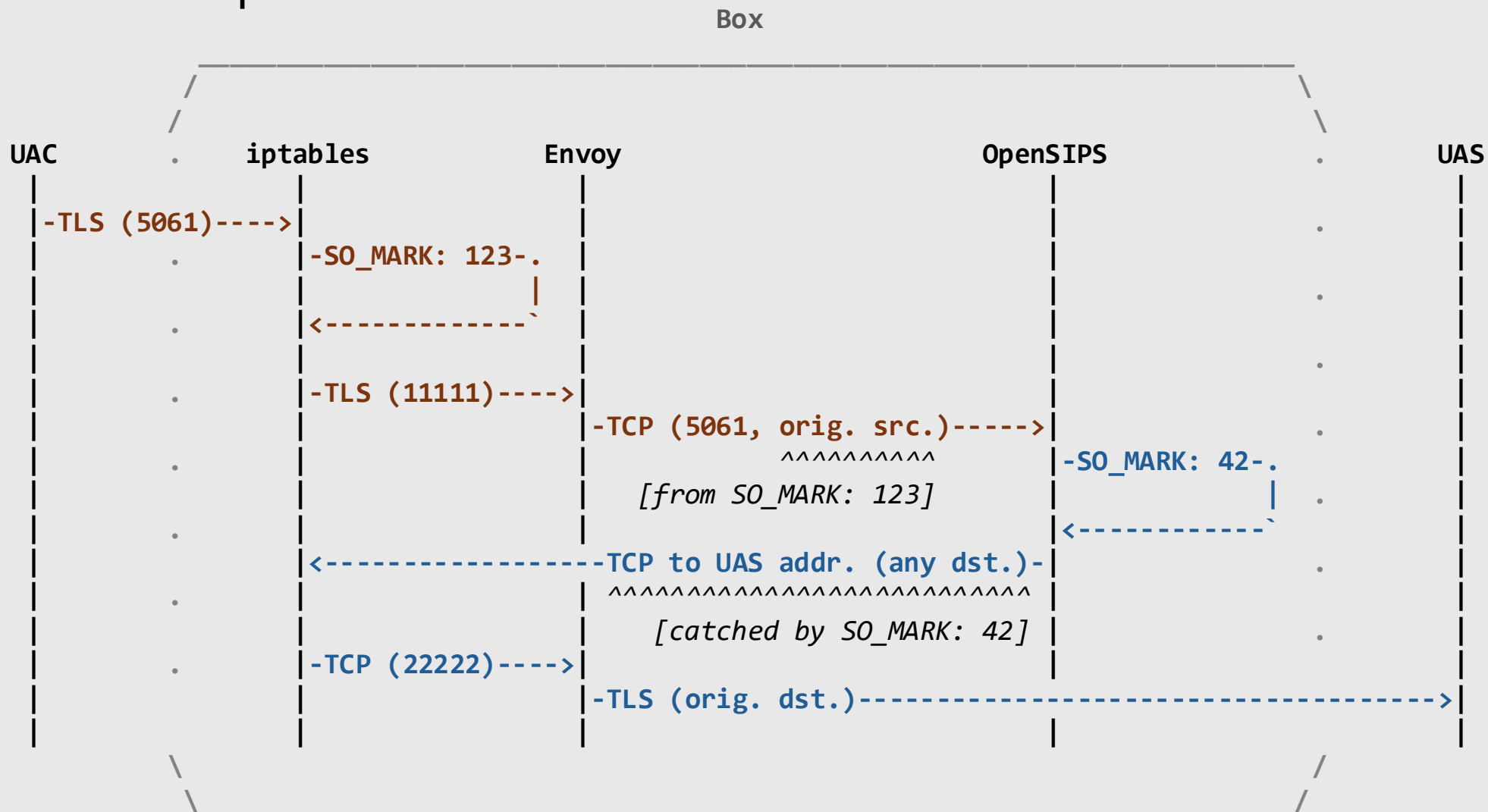
06 What's Next? Questions?

# A STEP FURTHER

## DEMO



# Proof of Concept Demo





# Proof of Concept Demo: UAC Output

```
----- Scenario Screen ----- [1-9]: Change Screen --
Call rate (length)  Port  Total-time  Total-calls  Remote-host
1.0(1000 ms)/1.000s  5061      3.21 s      3  172.18.0.3:5061(TLS)

1 new calls during 1.002 s period      1 ms scheduler resolution
2 calls (limit 3)                      Peak was 2 calls, after 2 s
0 Running, 5 Paused, 5 Woken up
0 dead call msg (discarded)            0 out-of-call msg (discarded)
6 open sockets                        0/0/0 TLS errors (send/rcv/cong)
0 Total RTP pckts sent                 0.000 last period RTP rate (kB/s)

                                     Messages  Retrans  Timeout  Unexpected-Msg
0 :      INVITE  ----->                3         0         0
1 :           100 <-----                3         0         0
2 :           180 <-----                3         0         0
3 :           183 <-----                0         0         0
4 :           200 <----- E-RTD1 3         0         0
5 :           ACK  ----->                3         0
6 :      Pause [  1000ms ]                3         0
7 :           BYE  ----->                1         0
8 :           200 <-----                1         0
```

## Proof of Concept Demo: OpenSIPS + Envoy Output

```

Listening on                                for address 172.18.0.4:32921. Marking with 123
      tcp: 0.0.0.0 [0.0.0.0]:5061
Aliases:                                    for address 172.18.0.4:39255. Marking with 123

Apr 16 20:16:38 [19] NOTICE:core:main: version: opensips 3.4.3 (x86_64/linux)
Apr 16 20:16:38 [19] NOTICE:core:main: using 32 MB of shared memory, allocator: F_
Apr 16 20:16:38 [19] NOTICE:core:main: using 16 MB of private process memory, allo
Apr 16 20:16:38 [19] INFO:core:init_reactor_size: reactor size 1024 (using up to 0 for address 172.18.0.4:39675. Marking with 123
Apr 16 20:16:38 [19] INFO:core:evi_publish_event: Registered event <E_CORE_THRESHO
Apr 16 20:16:38 [19] INFO:core:evi_publish_event: Registered event <E_CORE_SHM_THR
Apr 16 20:16:38 [19] INFO:core:evi_publish_event: Registered event <E_CORE_PKG_THR
Apr 16 20:16:38 [19] INFO:core:evi_publish_event: Registered event <E_CORE_PROC_AUTO_SCALE(4)>
Apr 16 20:16:38 [19] INFO:core:evi_publish_event: Registered event <E_CORE_TCP_DISCONNECT(5)>
Apr 16 20:16:38 [19] INFO:core:mod_init: initializing TCP-plain protocol
Apr 16 20:16:38 [19] INFO:tm:mod_init: TM - initializing ...
Apr 16 20:16:38 [19] ERROR:tm:tm_init_cluster: tm_replication_cluster not set - not engaging!
Apr 16 20:16:38 [19] INFO:core:evi_publish_event: Registered event <E_CORE_LOG(6)>
[2025-04-16 20:16:41.108][48][debug][filter] [source/extensions/filters/listener/original_src/o
[2025-04-16 20:16:41] TCP data from '172.18.0.4:50019'.
[2025-04-16 20:16:41] TCP data from '172.18.0.4:50019'.
[2025-04-16 20:16:42.109][78][debug][filter] [source/extensions/filters/listener/original_src/o
[2025-04-16 20:16:42] TCP data from '172.18.0.4:46329'.
[2025-04-16 20:16:42] TCP data from '172.18.0.4:46329'.
[2025-04-16 20:16:42] TCP data from '172.18.0.4:50019'.
[2025-04-16 20:16:43.107][78][debug][filter] [source/extensions/filters/listener/original_src/o
[2025-04-16 20:16:43] TCP data from '172.18.0.4:59635'.
[2025-04-16 20:16:43] TCP data from '172.18.0.4:59635'.
[2025-04-16 20:16:43] TCP data from '172.18.0.4:46329'.

```

# Proof of Concept Demo: UAS Output

```
----- Scenario Screen ----- [1-9]: Change Screen
Port      Total-time  Total-calls  Transport
5061      5.01 s      2    TLS

1 new calls during 1.002 s period      1 ms scheduler resolution
2 calls                                Peak was 2 calls, after 4 s
0 Running, 3 Paused, 4 Woken up
0 dead call msg (discarded)
4 open sockets                          0/0/0 TLS errors (send/recv/cong)
0 Total RTP pkts sent                    0.000 last period RTP rate (kB/s)

                                Messages  Retrans  Timeout  Unexpected-Msg
0 :  -----> INVITE                2         0         0         0
1 :  <----- 180                    2         0
2 :  <----- 200                    2         0         0
3 :  -----> ACK      E-RTD1 2         0         0         0
4 :  -----> BYE                1         0         0         0
5 :  <----- 200                1         0
6 :  [ 4000ms ] Pause            1
```

# Where to go from here?

## We are going to continue to investigate

Presentation covered a configuration with Envoy to Envoy as TLS tunnel.

- Internal Servers
- Trusted Parties

Demo took it a step further. Using SIP TLS between a single proxy.

- SIPp TLS → OpenSIPS (no TLS module) → SIPp TLS
- Strict configuration

What is not working?

1. Mixing and Matching protocols
2. Must be a proxy

Solutions?

1. Could use a SIP AWARE envoy configuration
2. Adding a new protocol module “**transport\_proxy**” –
  - Challenge: SIP is TRANSPORT AWARE
  - OpenSIPS could think it is doing TLS networking but sends and receive messages through a TCP socket.

OPENSIPS + ENVOY

THANK YOU!

